T.C.
İNÖNÜ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

DNA DİZİLERİNİN DE BRUIJN GRAFLARI İLE İNCELENMESİ

İRFAN KILIÇ

YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

HAZİRAN 2016

T.C.
İNÖNÜ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

DNA DİZİLERİNİN DE BRUIJN GRAFLARI İLE İNCELENMESİ

İRFAN KILIÇ

YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

HAZİRAN 2016

Tezin Başlığı: **DNA DİZİLERİNİN DE BRUIJN GRAFLARI İLE İNCELENMESİ**

Tezi Hazırlayan: **İrfan KILIÇ**

Sınav Tarihi: 24/06/2016

Yukarıda adı geçen tez jürimizce değerlendirilerek Bilgisayar Mühendisliği Ana Bilim Dalında Yüksek Lisans Tezi olarak kabul edilmiştir.

Sınav Jüri Üyeleri

Tez Danışmanı:	**Prof. Dr. Ali KARCI** İnönü Üniversitesi	.....
Eş Danışman:	**Doç. Dr. Resul DAŞ** Fırat Üniversitesi	.....
	Yrd. Doç. Dr. Fatih KOCAMAZ İnönü Üniversitesi	.....

Prof. Dr. Alaattin ESEN

Enstitü Müdürü

ONUR SÖZÜ

Yüksek Lisans Tezi olarak sunduğum "DNA DİZİLERİNİN DE BRUIJN GRAFLARI İLE İNCELENMESİ" başlıklı bu çalışmanın bilimsel ahlak ve geleneklere aykırı düşecek bir yardıma başvurmaksızın tarafımdan yazıldığını ve yararlandığım bütün kaynakların, hem metin içinde hem de kaynakça da yöntemine uygun biçimde gösterilenlerden oluştuğunu belirtir, bunu onurumla doğrularım.

İrfan KILIÇ

ÖZET

Yüksek Lisans Tezi

DNA DİZİLERİNİN DE BRUIJN GRAFLARI İLE İNCELENMESİ

İrfan KILIÇ

İnönü Üniversitesi
Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

84 + xi sayfa

2016

Danışman: Prof. Dr. Ali KARCI

Son yıllarda bilgisayar bilimlerindeki hızlı gelişmelere paralel olarak genetik mühendisliğinin önemi artmıştır. Bu gelişmeler biyoinformatik ve biyoistatistik disiplinlerinin ortaya çıkmasına ve ilerlemesine vesile olmuştur. Günümüzde özellikle "İnsan Genom Projesi" ile biyolojik verilerin daha hızlı ve daha güvenilir yöntemlerle incelenmesi hayati bir konu haline gelmiştir. Bu amaçla, bilgisayar bilimlerinin temel konularından graflar ve graf algoritmaları daha sık kullanılmaya başlanmıştır. Son yıllarda genetik verilerin hizalanması, dizilenmesi, sadeleştirilmesi ve analiz edilmesinde graf tabanlı yaklaşımlar çok sık kullanılmaya başlanmıştır.

Bu tez çalışmasında, biyolojik veri olarak DNA dizileri, DNA dizileme ve DNA dizi sadeleştirmede kullanılan De Bruijn grafları incelenmiş ve DNA verileri üzerinden De Bruijn graflarını modelleyen bir yazılım aracı geliştirilmiştir. Bu çalışmanın esasını oluşturan De Bruijn grafları basit DNA dizileri kullanılarak çeşitli örneklerle detaylı olarak incelenmiştir. Örnek DNA kısa-okuma verileri ile referans genom elde etmek için De Bruijn graflarını modelleyerek geliştirilen bir yazılım aracı ile uygulamalar sunularak sonuçlar değerlendirilmiştir.

Sonuç olarak bu tez çalışmasında geliştirilen yazılım ile De Bruijn graflarının DNA dizileme, DNA sadeleştirilmesi işlemlerinde önceki yöntemlere göre daha hızlı ve güvenilir sonuçlar verdiği gösterilmiştir.

ANAHTAR KELİMELER: DNA dizileri, DNA dizileme, De Bruijn dizileri, De Bruijn grafları

ABSTRACT

M.S. Thesis

ANALYSIS OF DNA SEQUENCES WITH DE BRUIJN GRAPHS

İrfan Kılıç

İnönü University

Graduate School of Natural and Applied Sciences

Department of Computer Engineering

84 + xi pages

2016

Supervisor: Prof. Dr. Ali KARCI

Parallel to the rapid advances in computer science in recent years has increased the importance of genetic engineering. These advances have led to the emergence and improvement of bioinformatics and biostatistics disciplines. Nowadays especially the "Human Genome Project" with faster and more reliable method of investigation biological data has become is a crucial topic. For this purpose, the main theme of graphs and graph algorithms in computer science began to be used more often. In recent years, graph-based approaches, alignment of genetic data, sequencing, simplification and analysis have been used very often. In this thesis, especially De Bruijn graphs used in the simplification of genetic data and analysis were described in detail.

In this thesis, De Bruijn graphs which are used in biological data such as DNA sequences, DNA sequencing and DNA simplication were investigated and the implemantion of De Bruijn graphs on DNA data was handled. The De Bruijn graphs, basic-building blocks of this study were investigated by using DNA sequences. In order to obtain reference genome with short-reads data by using De Bruijn graphs a software was implemented.

In conclusion, in this thesis DNA sequences were analyzed by using De Bruijn graphs and graph algorithms. De Bruijn graphs of the developed software, DNA sequencing, DNA simplification of the process has been shown to provide faster and more reliable results than the previous methods.

KEYWORDS: DNA sequences, DNA sequencing, De Bruijn sequences, De Bruijn graphs

ÖNSÖZ

Bu çalışmada DNA dizilemede kullanılan De Bruijn grafları ve De Bruijn graf algoritmaları anlatılmış, bu graf ve algoritmaların DNA dizileri üzerinde simülasyonu için bir yazılım paketi geliştirilmiştir. Ayrıca DNA dizileme, biyoinformatik ve DNA dizilerinin elde edilmesi ve De Bruijn graflarını kullanan yazılım paketleri anlatılmıştır. Bu çalışma ile De Bruijn grafları üzerine yeterli kaynak bulamayan araştırmacılara bir kaynak oluşturulması umulmaktadır.

Bu çalışmada kıymetli zamanını bana ayırarak çalışmamın tamamlanmasında her türlü yardımı yapan danışman hocam Prof. Dr. Ali KARCI'ya, yazılım aracının kodlanmasında bilgisini paylaşan Bilgisayar Mühendisi Uğur DEMİROĞLU'na ve düzeltmelerde yardımcı olan Uzman Orhan YAMAN'a teşekkür ederim.

Ayrıca bu çalışma esnasında ihmal ettiğim aileme, eşime ve çocuklarıma bana verdikleri destekten dolayı çok teşekkür ederim.

İrfan KILIÇ

2016

İÇİNDEKİLER

ŞEKİLLER DİZİNİ

ÇİZELGELER DİZİNİ

KISALTMALAR LİSTESİ

DNA : Deoksiribo Nükleik Asit

RNA : Ribo Nükleik Asit

A : Adenin

T : Timin

G : Guanin

C : Cytosine

NTP : Nucleoside Triphosphate (Nükleozid Trifosfat)

NGS : Next Generation Sequencing (Yeni Nesil Sekanslama)

NCBI : National Centre for Biotecnology Information (Ulusal Biyoteknoloji Bilgi Merkezi)

BGI : Beijing Genomics Institute (Pekin Genom Kurumu)

GB : Giga Bayt

TB : Tera Bayt

NP : Non-deterministic Problem (Deterministik Olmayan Problem)

OLC : Overlap Layer Consensus (Örtüşen Katmanlar Uyumu)

IUPAC : International Union of Pure and Applied Chemistry (Uluslararası Temel ve Uygulamalı Kimya Birliği)

BP : Base Pair (Baz Çifti)

BFS : Breath First Search (Geniş Öncelikli Arama)

1. GİRİŞ

Tüm canlı ve insanların özelliklerini üzerinde tutan DNA molekülünün keşfi (Watson, Crick, 1953) biyolojide büyük bir adımın başlangıcı olup biyoinformatiğin doğuşunun ilk aşaması sayılabilir. Virüs, canlı ve insanlara ait DNA genlerinin elde edilmesi ve tüm insan genomunun elde edilmesine çalışılması ile bilimde yeni bir çığır açılmaya başladı. Bu elde edilen genlerin doğru gen dizilimi, klasik kimyasal yöntemler ve yeni nesil dizileme (NGS) teknikleriyle elde edilebilir ve canlılara ait tahmini genom ortaya çıkarılabilir (Zülal, 2001). Bu genomun elde edilmesi meşakkatli, zaman alan ve maliyetli bir işlemdir. Şubat 2014'te Illumina şirketi $1.000 maliyetle insan genomu elde edebileceğini belirtmiştir (Illumina Inc, 2014). Bu maliyetli ve zorlu işlemlerden sonra elde edilen genom ve gen dizisi, kullanılan yöntemler ve çeşitli sebeplerden dolayı (okuma hataları, çevresel ve kimyasal faktörler vb.) tam doğrulukta değildir. Gen dizisinin ve genel olarak genomun %100'e yakın doğruluğa sahip olması canlıların genetik özelliklerinin yorumlanmasında hayati öneme sahiptir. Bundan dolayı elde edilen genomun doğru yorumlanması için genomun sadeleştirilmesi, hatalardan arındırılması ve doğruya en yakın gen dizisinin yeniden inşa edilmesi büyük önem kazanmaktadır. Günümüzde bilgisayar bilimlerindeki ilerlemeler sayesinde bu çalışmalar gittikçe hız kazanmıştır.

Bu tezde DNA dizi analizinden elde edilen veriler kullanılarak yapılan yeniden dizileme işlemleri ile bu işlemlerde ortaya çıkan problemleri çözmek için kullanılan De Bruijn graflarına ve bu graflar kullanılarak geliştirilen çeşitli yöntemlere açıklık getirilmektedir. Bu yöntemlerin kullanıldığı De Bruijn grafları (Idury, Waterman, 1995), bilgisayar bilimlerinin temel konularından biri olan graf ve graf algoritmalarını temel almaktadır. Ayrıca bu tezde DNA dizilemenin temelini oluşturan biyoinformatik ve biyoistatistik disiplinlerine de değinilmiştir (Bayat, 2002). Bahse konu gen dizisinin insan genomu olduğunu düşünürsek çok büyük veri karşımıza çıkmakta ve NP-zor bir problem çözülmeye çalışılmaktadır (Phillip, Pevzner, Tesler, 2011). Problemin NP-zor bir problem olmasından dolayı graf algoritmaları ve özellikle De Bruijn graflarına gereksinim duyulmuştur. Bu grafların dışında DNA dizileme de string grafları da kullanılmaktadır (Myers, 2005).

Sadeleşmemiş büyük gen dizileri içerisinden herhangi bir enzim, protein vb. yapıların tespiti ve bu yapılardaki bozuklukların bulunması kritik öneme sahiptir. Bu işlemlerin yüksek doğrulukla ve kısa sürede yapılabilmesinde De Bruijn graflarının ciddi katkısı olmuştur. Özellikle son yıllarda DNA dizileme, yeniden dizileme ve benzeri çalışmalarda De Bruijn graflarının kullanımı yaygınlaşmıştır. Yeni nesil dizilemede (NGS) 2 yöntem ön plana çıkmaktadır. Bunlardan biri olan parçalı dizilemede kullanılan OLC (overlap layout consensus –örtüşen katman birliği-) yaklaşımıdır ve bu yaklaşım NP-tam Hamilton yolu problemine

neden olmaktadır. 2. yöntem De Bruijn graflarını kullanan Euler algoritması ile 1980'lerden itibaren çözülemeyen "tekrarlı bölgeler" problemine yeni bir çözüm bulmuştur (Pevzner, Tang, Waterman, 2001). Sonraki yıllarda ise gen dizilerinde var olan okuma hatalarını tespit etmek ve yeniden gen dizileme yapmak için de De Bruijn graf algoritmaları kullanılmıştır. De Bruijn grafları ve bu graflar kullanılarak geliştirilen algoritmalar DNA dizileme ve DNA yeniden dizileme üzerine çalışanların yeni ilgi odağı olmuştur.

De Bruijn graflarını kullanan başlıca yazılım paketleri Velvet Assembler (Zerbino, 2008), ABySS (Simpson, 2009) ve IDBA (Yu Peng, 2010) yazılım paketleridir. Son yıllarda bu yazılım paketlerini daha da iyileştirmek Minia (Chikhi, 2013) adında Bloom Filtre yaklaşımını kullanan bir algoritma geliştirilmiştir. Ayrıca De Bruijn graflarını simule etmek için PHAST (Taylor, 2012) adında web tabanlı bir uygulama geliştirilmiştir.

Velvet genomun yeniden yerleştirilmesi ve kısa-okuma dizileme hizalanmalarını çözmek için tasarlanmış bir algoritma paketidir. Bu algoritma hataların kaldırılması ve tekrarlı bölgelerin basitleştirilmesi yoluyla genom dizisini yerleştirmek için De Bruijn graflarının kullanılmasıyla gerçekleştirilmiştir (Zerbino, 2008). Velvet ile ilgili detaylı bilgilendirme 4. Bölümde yapılmıştır. Velvet'in kullanılmasıyla ilgili bilgilendirme Ek C'de verilmiştir.

ABySS (Assembly By Short Sequencing) insan genomlarını kısa okumalarla dizileyerek çok büyük veri kümelerini bir araya getirmek için geliştirildi. ABySS'in birinci yeniliği bir bilgisayar ağı yoluyla montaj algoritmalarının paralel hesaplanmasına imkân veren De Bruijn graflarını dağıtık olarak temsil edebilmesidir. ABySS algoritması 2 aşamadan oluşur. Birinci aşamada olası tüm k-harfliler okunan DNA dizisinden oluşturulur. k-harfli veri kümesi okuma hatalarını kaldırmak için işlenir ve kontigler inşa edilir. İkinci aşamada DNA sarmalının 2. ipliğindeki baz çifti bilgisi kontig örtüşmelerindeki belirsizlikleri çözerek kontigi genişletmek için kullanılır (Simpson, 2009). ABySS kullanımı ile ilgili detaylar Ek A'da verilmiştir.

IDBA (Iterative De Bruijn Graph Assembler) algoritması De Bruijn grafının temelini oluşan k-harfli sayısını iteratif olarak tüm alabileceği değerler içinde en küçük değerden en büyük değere doğru iterasyonunu yaparak ideal k-harfli değerini bulmaya çalışır. IDBA algoritması gerçek ve simüle veriler ile hem de benzer doğrulukla daha az bellek kullanarak büyük kontigleri inşa etme de var olan algoritmaları geride bırakmaktadır (Yu Peng, 2010). Bu algoritmanın yazılım paketinin kullanımı ile ilgili detaylar Ek B'de sunulmuştur.

String graf yaklaşımı k veya daha fazla örtüşen nükleotid ile iki okumayı bağlayarak kontigleri oluşturur. Edena (Exact De Novo Assembler) string grafları kullanarak aynı uzunlukta çok kısa okumaları içeren veri kümelerinden doğru kontigleri yeniden bir araya getiren bir yazılımdır. Bu uygulama hesaplanmış ve yapılandırılmış bir grafta tüm

örtüşmelerin olduğu klasik bir montajlamayı temel alır. Belli sayıda kilobazı doğru kontigin dizilenmiş genomu en kapsayan biçimde üretir (Hernandez, 2008).

1.1. Tez Çalışmasının Amacı ve Kapsamı

Özellikle gen dizilerini iyi şekilde modelleyen ve yeniden dizilemede kullanılan De Bruijn grafları, bu grafların temelini oluşturan De Bruijn dizileri ve De Bruijn graflarında kullanılan graf algoritmaları bu tezin ana konusunu teşkil etmektedir. Bu grafları ve algoritmalarını DNA dizileri üzerinde modellemek için tarafımızdan geliştirmeye açık bir yazılım yazıldı. Bu yazılım NetBeans IDE platformunda Java programlama dili ile kodlandı. Yazılım oluşturduğu verileri görselleştirmek için Python programlama dili destekli Graphviz kütüphanesinden faydalanıldı. Bu yazılım ile De Bruijn graflarının farklı örneklerle oluşturulması, oluşturulan grafta belli işlemlerin yapılması, tüm bu adımların grafiksel gösterimi için gerekli işlemlerin yapılması ve DNA yeniden dizilemede diğer yöntemlere göre avantajları gösterilmiştir.

Bu tezin De Bruijn dizileri, grafları ve algoritmaları konusunda Türkiye'de eksik kalan kaynağı tamamlaması beklenmektedir. Ayrıca bu konuda çalışma yapacaklara bir yol göstereceği umulmaktadır.

1.2. Tez Çalışmasısın Organizasyonu

Bu tez çalışması genel olarak 5 ana bölümden oluşmaktadır.

Bölüm 1'de, tezin amacı ve kapsamına yönelik genel bilgiler sunularak genel literatür bilgileri verilmiştir.

Bölüm 2'de, temel mikro biyoloji, DNA vb. genel biyoinformatik konuları hakkında genel bilgiler sunulmuş ve bir organizmanın gen dizisinin NCBI veritabanından nasıl elde edilebileceği gösterilmiştir.

Bölüm 3'de, gen dizilerinin nasıl elde edildiği, gen dizilemenin önemi, klasik yöntemler ve Yeni Nesil Dizileme (NGS-Next Generation Sequencing) yöntemleri açıklanmıştır.

Bölüm 4'de, De Bruijn graflarının temelini oluşturan De Bruijn dizilerinin matematiği verilmiş, gen dizilerinin grafını elde etmede kullanılan De Bruijn grafları anlatılmış ve basit gen dizileri üzerinden De Bruijn graf örnekleri ayrıntılarla verilmiştir.

Bölüm 5'de, De Bruijn grafları kullanılarak geliştirilen algoritma ve yazılım aracı detaylıca anlatılmış, örnek DNA dizisi ve farklı DNA kısa-okuma dosyaları ile algoritmaların nasıl uygulandığı sunulmuştur.

Ayrıca, farklı DNA kısa-okuma dosyalarından yazılımımızda kullanılan algoritmaların performansları karşılaştırılarak elde edilen sonuçlar değerlendirilmiş ve sonuçlar verilmiştir.

Bölüm 6'da tezin genel sonuçları irdelenmiş ve gelecek çalışma önerileri tartışılmıştır.

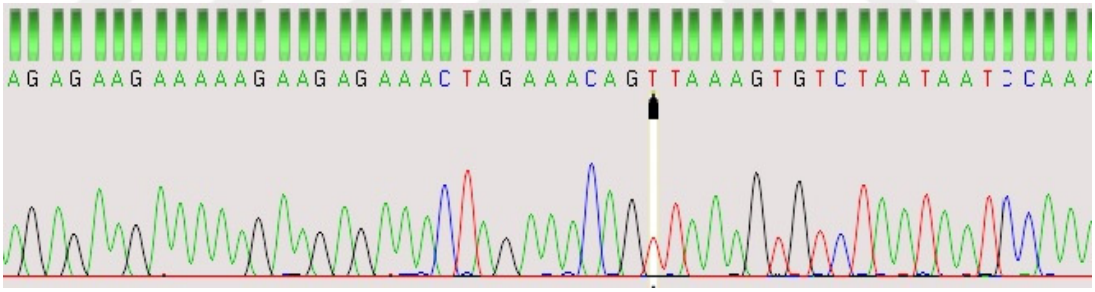
2. MİKROBİYOLOJİ VE BİYOİNFORMATİK

2.1. Giriş

Biyolojik veri sayısal bir şekilde depolanan veya değiştirilen biyolojik kaynaklardan elde edilen verilerdir. Biyolojik veri çoğunlukla dosyalarda veya veritabanlarında tutulur. Biyolojik verilere örnek olarak DNA baz çifti dizileri ve ekolojideki popülasyon verisi örnek gösterilebilir.

1953 yılında James Watson[1] ve Francis Crick[2] isimli 2 araştırmacı bugün DNA'nın kabul görmüş yapısını keşfetmiştir (Watson, 1953) DNA temel olarak hücrenin tüm özelliklerinin kodlanarak saklandığı uzun bir moleküldür. Tüm hücreler, DNA'da kodlanmıştır. DNA, hücrelerin oluşturulması ve işleyişinden sorumlu bir temel taslak sağlar. Bu taslak hangi hücreler büyüyebilir veya hangi hücreler ölebilir veya vücudun çeşitli kısımları oluşurken hücreler nasıl bir yapıya geçer gibi direktifleri barındırmaktadır. Örneğin DNA insan saçının kalitesi, rengi, bolluğu veya eksikliğini belirlemeden sorumludur. Canlı bedenleri DNA rehberliği süreciyle formüle edildiğinden her canlı ebeveynlerine benzer.

DNA, Deoksiribo Nükleik Asittir. Neredeyse tüm organizmalarda bulunan DNA organik bir bedenin inşasında kullanılan uzun terimsel bilgiyi saklar. **DNA dizisi** veya **genetik dizi**, DNA ipliğinin birincil yapısından elde edilen harfler dizisidir.



Şekil 2.1. Bir Otomatik Dizileme Aracı Çıktısı ile Okunabilen DNA Dizisi (dnabaser.com web adresi, 2015)

DNA dizisinde olan harfler *C*, *A*, *T* ve *G* harfleridir ve bu harfler DNA ipliğinde bulunan sitozin, adenin, timin, guanin isimli 4 baz yerine kullanılan harflerdir. A, T, C ve G dışında özel durumlar için farklı harfler bulunabilir. DNA dizisinde belirsizlikleri belirtmek için bu özel harfler kullanılmaktadır. Bu amaçla IUPAC (Uluslararası Temel ve Uygulamalı Kimya Birliği)'nin belirlediği bazı sembollerin anlamları aşağıdaki gibidir (Cornish-Bowden, 1985):

[1] James Dewey Watson (d. 6 Nisan 1928, Chicago), 1954'de araştırmacı Francis Crick ile birlikte DNA'nın ikili sarmal yapısını bulan Nobel ödüllü bilim adamıdır.
[2] Francis Harry Compton Crick (8 Haziran 1916 - ö. 28 Temmuz 2004), Moleküler biyolog, fizikçi ve nörobilimci. James Dewey Watson ve Maurice Wilkins ile birlikte DNA'nın molekül yapısını keşfetmesinden dolayı 1962 yılında Nobel Fizyoloji veya Tıp Ödülü almıştır.

- **C** = **s**itozin (ing. ***c****ytosine*)
- **T** = timin
- **A** = adenin
- **G** = guanin
- **S** = G veya C (kuvvetli bağlılar: ing. ***s****trong bonds*)
- **W** = A veya T (zayıf bağlılar: ing. ***w****eak bonds*)
- **N** = A G C T (4'dün biri)
- **U** = uridin (RNA dizileri için kullanılır)
- **R** = G veya A (pü**r**in)
- **Y** = T veya C (pirimidin: ing. *p****y****rimidine*)

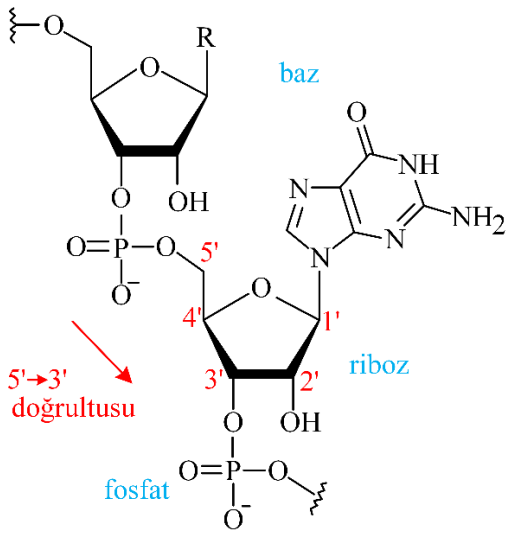
Bazı belirsizlik sembolleri genelde iki durum için kullanılmaktadır:

- DNA dizilemesi(sekanslama) esnasında bir baz teknik nedenlerle tam okunamadığında (örnek olarak, G veya C olursa, S ile belirtilir)
- Benzer DNA veya RNA dizilerinin ortak yönlerini belirtmede kullanılır. Örnek olarak SREBP isimli transkripsiyon faktörünün bağlandığı dizilerin ortak bir özelliği CGAT**N**GGCAC şeklinde yazılabilir. Bu örnekte N harfi ile belirtilen yerde herhangi bir baz bulunabilir.

DNA'daki dört farklı nükleotid birbiri ardınca dizilenerek DNA zincirlerini meydana getirir. İnsanın her DNA'sı on binlerce nükleotidden oluşur. DNA zincirindeki belli uzunluktaki birime **gen**, DNA'nın özel şekilde paketlenmesiyle ortaya çıkan yapıya göre **kromozom denir ve** her kromozomda çokça sayıda gen vardır (Demir, 2011). DNA nükleotidlerinin bazı dizilimleri olay tetikleyici olduğundan bunlara DNA **dizi motifi** denilmektedir (Demir, 2011).

Bir diziyi ifade eden harfler aralarında boşluk olmaksızın yazılır, örnek olarak GTAACGTTAC dizisi aralarında boşluk olmadan yazılmıştır. Bu dizi soldan sağa okunurken **5' → 3'** (5 üssü, 3 üssü) doğrultusu yönüne karşılık gelmektedir.

Burada bahsedilen **doğrultu**dan kastedilen, nükleik asit sarmalını meydana getiren nükleotidlerin uç uca eklenme yönünü ifade etmektedir. Kimyasal adlandırmadaki belli kurallara göre, bir nükleotidin şeker halkasındaki karbon atomları 1', 2', 3', 4' ve 5' şeklinde adlandırılır (3 üssü, 5 üssü olarak okunur). Nükleik asitlerin bir ucundaki şeker grubunun serbest bir 3' hidroksil (-OH) grubu, öbür ucundaki şekerin ise serbest bir 5'-OH grubu bulunmaktadır. Bu iki uca, sırayla 3' ve 5' uçları denilmektedir. Sarmallı DNA veya RNA dizileri yazılırken bazlar 5'-3' doğrultusu şeklinde yazılır.



Şekil 2.2. Bir RNA Zincirinde 5' → 3' Doğrultusu (Lodish, 2000)

DNA ve RNA dizilerinin biyoinformatik programları ile okunabilmesi için standart bazı formatlar oluşturulmuştur. Bunlardan en çok kullanılanı FASTA formatıdır. FASTA formatında birinci satır ">" sembolüyle başlayan bir başlık içerir, sonraki satırlarda DNA dizisi yer almaktadır (zhanglab.ccmb.med.umich.edu web adresi, 2016). Çizelge 2.1'de örnek FASTA formatı incelenebilir.

Çizelge 2.1. FASTA Formatında Dosya (zhanglab.ccmb.med.umich.edu web adresi, 2016)

```
>gi|186681228|ref|YP_001864424.1| phycoerythrobilin:ferredoxin
oxidoreductase

MNSERSDVTLYQPFLDYAIAYMRSRLDLEPYPIPTGFESNSAVVGKGKNQEEVVTTSYAFQT
AKLRQIRAAHVQGGNSLQVLNFVIFPHLNYDLPFFGADLVTLPGGHLIALDMQPLFRDDSAY
QAKYTEPILPIFHAHQ...
```

2.2. Biyoinformatik

Biyoinformatik, bilgisayar teknolojileri ile moleküler biyoloji ve bunlarla ilgili veri hesaplama araçlarını içerisinde bulunduran bilimsel bir çalışma alanıdır. Başka bir ifadeyle kompleks biyolojik verileri derleyip, analiz eden bir bilim dalıdır. Biyoinformatik, biyolojik bilgilerin oluşturulması ve depolanması için veritabanlarının oluşturulmasıdır (Bayat, 2002). Biyoinformatik, bilişim teknolojileri kullanılarak biyolojik problemlerin çözülmesi temeline dayanan ve biyolojik vakaların moleküler seviyede açıklanmasına yardımcı olan bir disiplindir (Luscombe, 2001; Polat, 2009). Biyoinformatik, yaşam ve bilgisayar bilimler ile bir bağlantı kurmakta, verilerin çok etkili bir şekilde elde edilmesi ve aşamaları hızlandırması sebebiyle oldukça önemlidir (Feagan, 2007). Biyoinformatiğin esas gayesi genomu verilen bir canlının tüm fonksiyonlarının anlaşılıp, yaşam kalitesinin artırılmasıdır.

Biyoinformatiğin amaçları veri organizasyonu, sistemlerin geliştirilmesi ve sistemlerin uygulanması şeklinde üç ana başlık altında toplanabilir (Polat, 2009).

Biyoinformatiğin doğuşunda 20. yüzyılın ikinci yarısı itibariyle biyolojik bilginin devasa olarak artması ile bu bilgi organize etmek için güçlü araçlara geresinim duyulması etkili olmuştur. 1960'lar ile bilgisayar uygulamalarının biyolojide kullanılması, bu iki alandaki teknolojik gelişimle hızlı bir şekilde ilerlemiş ve bu şekil ortaya çıkmış olan Biyoinformatik dalı bugün popüler akademik ve endüstriyel sektörlerden biri haline gelmiştir.

Günümüzde biyolojik sistemler ile ilgili ortaya atılan sorular ziyadesiyle karmaşık olabilir ve bu soruların cevapları insanın kapasitesi içerisinde sınırlandığında bunların cevapları bulunamayacaktır (Buttle, 2001). Biyolojik sistemler ile ilgili bilgiler karmaşık olup, transkripsiyon regülasyonları, hücresel aktiviteler, gelişimsel organizasyon ya da hücresel iletişim gibi karmaşık biyolojik sistemlerdeki sinyallerin ve yolların karşılıklı etkileşimi insan beyninin niceliksel açıdan prensip olarak anlayamayacağı kadar karmaşık ve hassastır. Daha fazla bilinmeyen biyolojik olayların açıklanabilmesi ile biyolojik bilgiler ve bu bilgilere ulaşmadaki genel kavramlar değişecektir. Bu bilgileri düzenlemek ve ulaşılmak istenen konu ile ilgili bilgiyi anlaşılabilir bir şekilde sunmak için daha fazla bilişim araçlarına ihtiyaç duyulmaktadır. Böyle kompleks bilgilerin bütünleştirilmesi için modeller ve kavramlar geliştirmek ve anlaşılabilir olması için görselleştirilmesini sağlamak biyoinformatiğin en önemli uğraşı alanlarından biridir (Jain, 2001; Bayat, 2002).

Moleküler biyolojide kullanımı bilgisayarların kullanımı ile moleküler yapıların üç boyutlu grafikler olarak temsil edilmesi, üç boyutlu moleküler yapıların veritabanlarının oluşturulması ve moleküler dizilimlerin oluşturulması ile başlamıştır. Biyoinformatik alanındaki bilgisayar uygulamaları, çok yüksek düzeylerde üretilen biyolojik veri, endüstriyel gen ekspresyonu, aktif molekül araştırmaları, protein-protein ilişkisi, bakteri, maya, hayvan ve insan genom projeleri benzeri biyolojik gelişmelerin ortaya çıkardığı zorunluluklar neticesinde takip edilemez şekilde çok hızlı gelişme göstermiştir. Son yıllarda biyoinformatik dalı disiplinler arası ayrı bir bilim dalı olarak görülmeye başlamıştır.

Genel olarak biyoinformatik bilgisayar teknolojilerinin biyolojik problemlerin incelenmesi ve çözülmesinde kullanılmasıdır. Kısa tanımla gen dizilemede kullanılan biyolojik veritabanlarının meydana getirilmesi ve işletmesinin yapılması, geniş tanımla var olan bütün bilgisayar uygulamaları ve tekniklerinin biyolojik problemlerin incelenmesi ve çözümünde kullanılması şeklinde tanımlanabilir.

Uygulamalı bir bilim dalı olan biyoinformatik, günümüz genetik bilgi arşivlerinden, moleküler biyoloji, tıbbi biyolojiden faydalanılarak geliştirilmiş bilgisayar yazılımları yardımıyla sonuçlar elde edilmekte ve bu şekilde önemli öngörülerde bulunulmaktadır (Polat, 2009). NCBI (Biyoteknoloji Ulusal Bilgi Merkezi), yaşam bilimlerine (Biyokimya, Biyoloji, Tıp), bilişimdeki teori ve teknolojilere, matematik ve istatistiğe dayalı disiplinler arası bir bilim dalı olarak biyoinformatiği tanımlamıştır. Biyolojik veri bankalarının oluşturulmasıyla verilerde muazzam bir artış görülmüştür. Mevcut veriler 1986 yılında 3939 iken, 1999 yılında 80.000'e çıkmış ve kayıtlara göre 2004'te yaklaşık 160.000 veriye ulaşılmıştır. Günümüze kadar artış göstermekte olan veri miktarı 2005 yılı itibariyle yaklaşık 300.000 civarındadır (Kumar, 2005). Biyoinformatik bu verileri hızlı bir şekilde işleyebilecek yeni algoritmaların geliştirilmesini ön ayak olmuştur. Biyoinformatik, elde edilmiş veriler üzerinden işlem yaptığından laboratuvar çalışmalarına kıyasla çok az bir maliyete mal olmaktadır. Bunun sonucu olarak biyoinformatiğe dayanan algoritmaların geliştirilmesi ile belli özel hastalıklar için teorik ilaç keşiflerinin yapılabileceği öngörülmüştür (Doerry, 1997; Gatto, 2003). Dünyada biyoinformatik tabanlı yazılımlarla ilaç geliştirilmesi amacıyla ilaç sanayi bağlantılı birçok şirket kurulmuş olup sayıları da gün geçtikçe artmaktadır. Bu şirketlerin çoğunluğu ABD, İsviçre ve İngiltere'de olmakla birlikte yazılım alanında büyük atılım gerçekleştiren Hindistan'da da birçok biyoinformatik şirketi kurulmuştur (Kumar, 2005).

Modern biyolojinin aşağıda belirtilen iki ana akışı biyoinformatik ile sağlanmaktadır:

1. Deneysel bilgi akışı: Biyolojik olayların gözlenmesi ile elde edilen veriler, açıklayıcı bir şekilde tanımlanır, sonrasında bu şekillerin doğruluğu yeni yapılan deneyler ile denetlenir.
2. Genetik bilgi akışı: Canlının DNA'sı incelenmek suretiyle karakteristik özelliklerinin belirlenmesi, incelenen bu canlı türünün oluşturduğu topluluğun karakteristik özelliklerinin bilgi akışı, elde edilen DNA dizisi yeniden genetik havuzun belirlenmesi için kullanılabilir.

Özellikle son yıllarda esas biyolojik araştırmaların klinik tıp bilgi sistemleri ve klinik tıp uygulamaları üzerine etkisi çok belirgin hale gelmiş ve günümüzde yeni nesil epidemiyolojik, teşhis, tanı ve tedavi amaçlı uygulamaların ortaya çıkmasına ön ayak olmuştur. Biyoinformatik çalışmalar ana bilimsel incelemelere özgü görünmekte fakat bununla beraber önümüzdeki 10 yıl içerisinde klinik bilişim için vazgeçilmez bir hal alacaktır. İleriki yıllarda hastaların medikal formlarında artan bir şekilde artık DNA dizilimleri yer alacaktır. Günümüzde ABD'de bazı sigorta firmaları, risk primlerini tespit ederken kişinin genetik tarama sonuçlarını isteyebilmektedir.

Biyoinformatikte araştırmalar için geliştirilmiş algoritmalar yakında klinik bilişim sistemleri için uygun olarak kodlanması kaçınılmaz olacaktır.

Aşağıda biyoinformatik araçların kullanıldığı genel araştırma konuları metodolojik ve biyolojik çalışmalar şeklinde özetlenebilir (Searls, 2010; wikipedia web adresi 2015).

Metodolojik çalışmalar

1. Protein dizilime ve DNA sıralama araştırmaları
2. DNA dizilime ve DNA sıralama araştırmaları
3. RNA, DNA ve protein gibi yapıların üç boyutlu dizilime araştırmaları
4. Küçük moleküllerin bağlarıyla etkileşiminin incelenmesi
5. Büyük GENOM projelerinden çıkan sonuçların incelenmesi
6. Bilişim teknolojileri ile otomatikleştirilmiş veri incelemesi ve iletimi
7. Gen ürünlerinin bilgi ağlarının meydana getirilmesi
8. Biyolojik heterojen veritabanları
9. Biyolojik bilginin paylaşımı
10. Biyolojik faaliyet süreçlerinin simüle edilmesi

Biyolojik çalışmalar

1. Biyolojik fonksiyonu önleyen veya izin veren küçük moleküllerin oluşturulması
2. Proteinin fonksiyon ve yapısının incelenmesi
3. Genetik faktörlerin etkilerinin ortaya çıkarılması
4. Karışık genetik düzenleme faaliyetlerinin oluşturulması
5. Endüstriyel veya tıbbi amaçlı büyük moleküllerin üretilmesi

2.2.1. Biyoinformatiğin Önemi

İlaç endüstrisi firmaları gen dizileme (sekanslama) projelerini yakından izlemekte olup, bu yaşamsal genetik veri tıbbi tanı ve tedavi amaçlı uygulamalar için gereklidir ve diğer endüstriyel alanlarda da kullanılabilmektedir (Polat, 2009).

Biyoinformatiğin başlıca görevlerinden biri; bütün biyolojik türlerin (insan dahil) genomlarına, proteinlerin üç boyutlu yapılarına, protein sekanslamaya, metabolik yol veritabanlarına ve biyo-çeşitliliğe bağlı bilgilerine ait niceleyici bilgilerin toplanmasını sağlamaktır. Son yıllarda gen dizileme projelerinde yaygın uygulamalarla biyoinformatik oldukça önem kazanmıştır. İnsan genomu projesinin başarı ile tamamlanmasında biyoinformatiğin oldukça önemli katkıları olmuştur. Buna ek olarak biyoteknoloji temelli üretim ve süreç geliştirmede de biyoinformatiğin katkısı büyüktür. İlaç tasarımı, geliştirilmesi masraflı ve vakit alan bir süreçtir. Biyoinformatik bu süreçlerin hem maliyetini

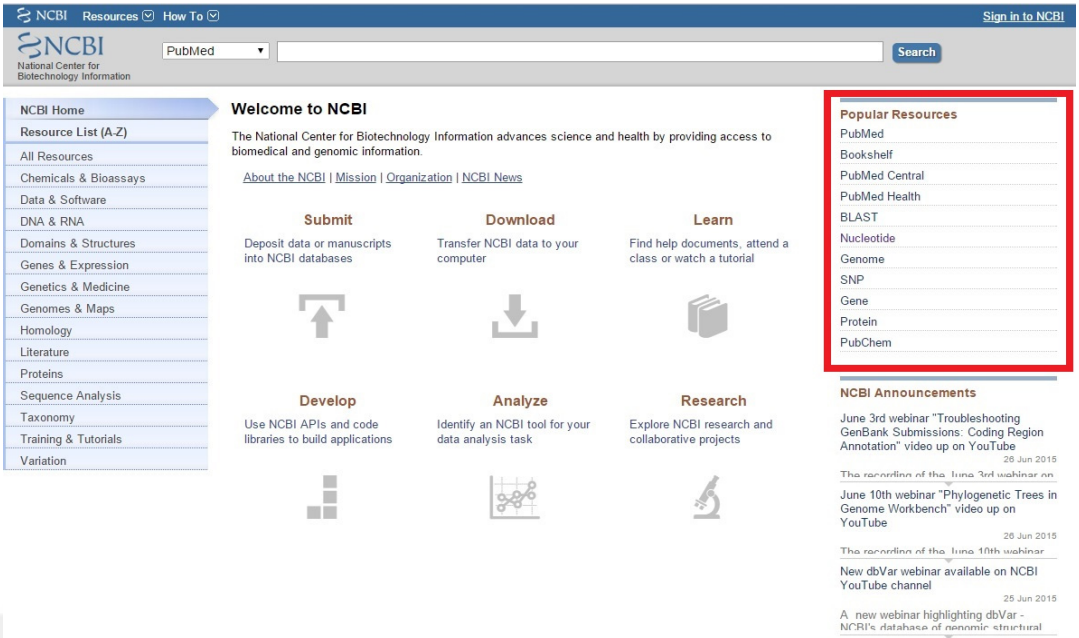
hem de gerçekleşme süresini çok kısaltmaktadır. Büyük biyoteknoloji ilaç şirketlerinin neredeyse hepsinde çok geniş biyoinformatik araştırma-geliştirme grupları kurulmuştur. Eldeki verilere göre biyoteknoloji en hızlı gelişen üretim teknolojilerinden biri olmuştur. Uygulamalarda elde edilecek başarılar biyoinformatiğin önemini ortaya çıkaracaktır. Biyoinformatik, ilaç tasarımıyla beraber tıbbi tanı ve tedavide de ciddi başarılar elde etmiştir (Hogue, 2002; Polat, 2009).

Biyoinformatik esasında DNA/RNA dizi incelemeleri için geliştirilmiştir. Ancak günümüz itibariyle; genomik ve gen ekspresyon çalışmaları, yapısal biyoloji gibi geniş bir alanda hizmet verir hale gelmiştir. Biyoinformatik çalışmalarını hepsini ilke olarak destekleyen iki yaklaşım mevcuttur. Bunlardan birincisi verilerin biyolojik olarak anlamlı benzerliklere göre karşılaştırılması ve gruplanması, ikincisi de belli bir veri çeşidinin incelenerek, başka bir veri çeşidinin anlaşılması ve buna göre değerlendirilmesidir. Bu yaklaşımlar biyoinformatiğin ana hedefleri ile uyumludur (Brooksbank, 2003).

2.3. Biyolojik Verilerin Elde Edilmesi

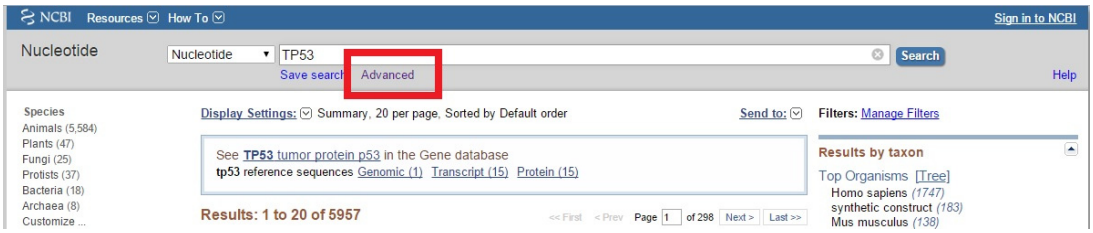
Herhangi bir organizmanın geninin DNA dizisini bulmak; sınıflandırma ve genomik benzerlik çalışmaları, primer seçimi, DNA parmak izi çalışmaları, restriksiyon enzimi uygulamaları için çok önemlidir.

DNA dizisini bulmadan önce, dizisini öğrenmek üzere bir genin seçilmesi gereklidir. Bunun için, kanser genetiği araştırmalarında çok önemli olan **TP53** geni seçilsin. Gen seçildikten sonra, bu genin aranacağı veritabanı belirlenir. Moleküler biyolojide, buna benzer aramaların çoğunda Amerika kökenli Ulusal Biyoteknoloji Bilgi Merkezi (NCBI)'nin veritabanı üzerinden gerçekleştirilmektedir. Gen dizisini bulmak için bu merkezin resmi web sitesi http://www.ncbi.nlm.nih.gov adresi ziyaret edilebilir (ncbi.nlm.nih.gov web adresi, 2015). Web sitesi yüklendiğinde sayfanın sağında "**Popular Resources (Popüler Kaynaklar)**" başlığı altındaki "**Nucleotide (Nükleotit)**" linkinden bunla ilgili veritabanına girilebilir.



Şekil 2.3. NCBI Resmi Sitesi (ncbi.nlm.nih.gov web adresi, 2015)

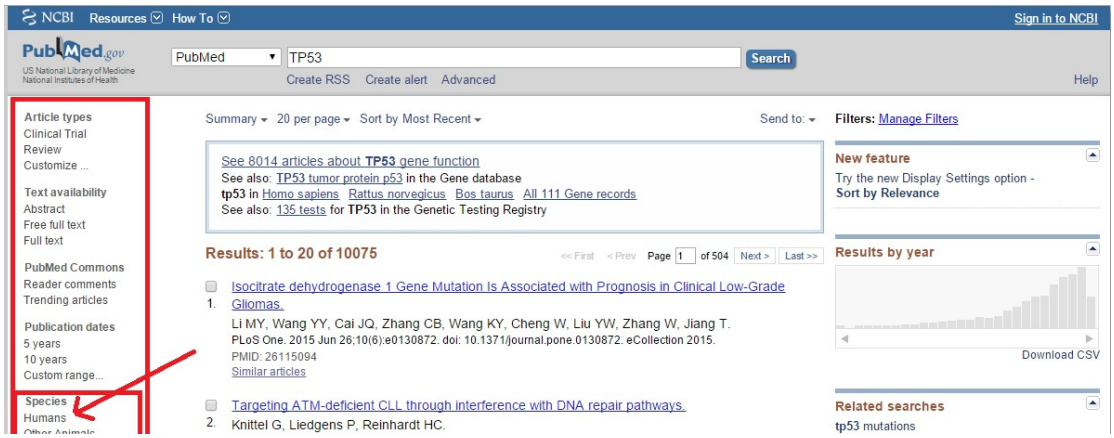
Bu veritabanı kullanılarak, seçilmek istenen genin nükleotid dizisi öğrenilebilir. Bu maksatla sayfanın üst tarafında bir arama kutucuğu bulunmaktadır. Bu alana aranacak gen doğrudan yazılabilir. Ancak, aranmak istenen gen yalnızca bir organizmaya has değilse arama sonuçları çok fazla çıkacaktır. Örnek olarak, **TP53** geni direkt arandığında çok fazla sonuç çıkacaktır. Bu durumda ulaşmak istenen sonuca erişim zorlaşacaktır.



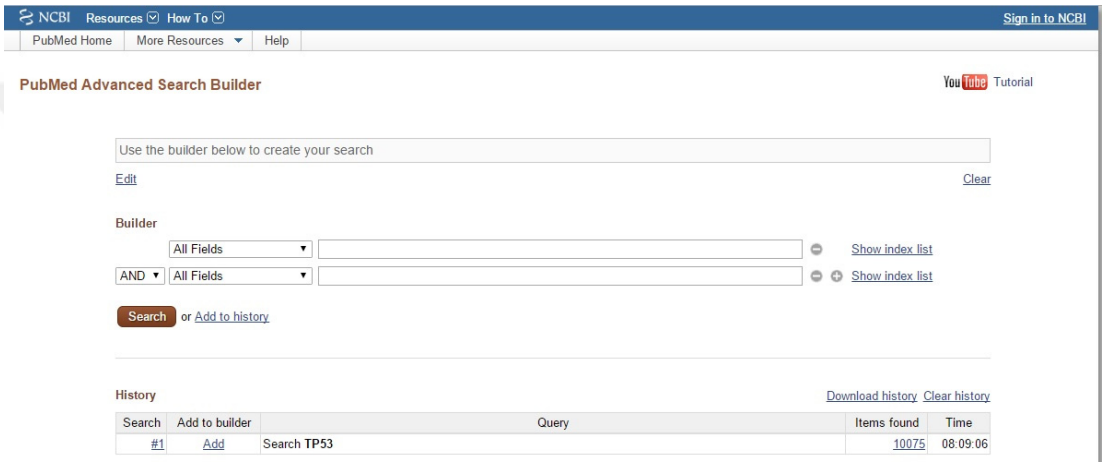
Şekil 2.4. NCBI Arama Sonuç Sayfası (ncbi.nlm.nih.gov web adresi, 2015)

Bu problemi çözmek için aramanın daraltılması gerekmektedir. Bunun için Şekil 2.4'teki arama kutucuğunun yanındaki gelişmiş arama (**Advanced**) bağlantısı kullanılabilir (Şekil 2.4) veya Şekil 2.5'deki Species altındaki "Humans" seçeneği kullanılabilir.

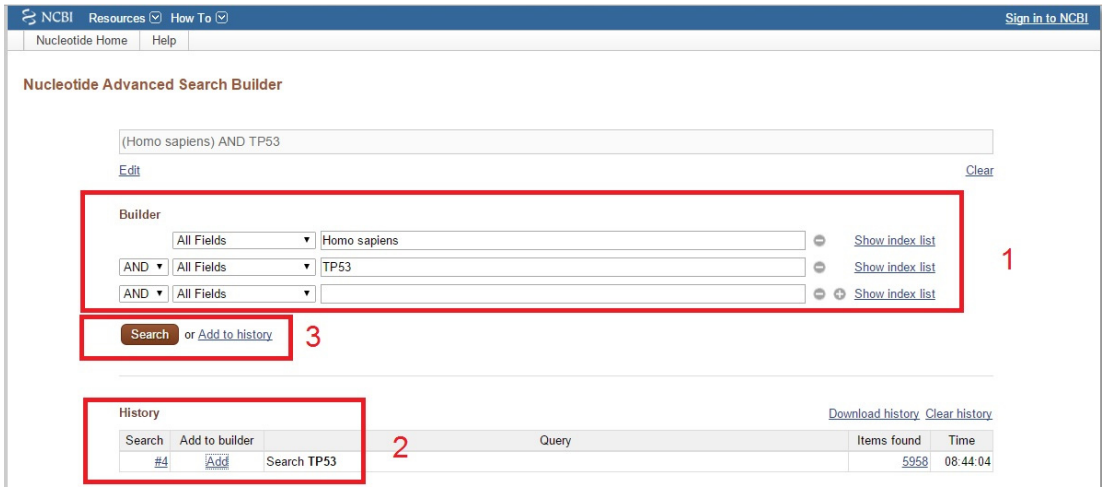
Gelişmiş arama alanında, genin hangi organizmadan olduğunun belirtilmesi gerekmektedir. Bunun sebebi aynı genin, başka organizmalarda farklı dizilime sahip olabilme ihtimalidir. Bu fark organizmaların değişim sürecinde birbirine uzaklığına göre artmaktadır. Bundan dolayı gen dizisini öğrenilirken, ilgilenilen canlının da belirtilmesi gerekmektedir.



Şekil 2.5. NCBI Arama Sayfası Gelişmiş Seçenekler (ncbi.nlm.nih.gov web adresi, 2015)



Şekil 2.6. NCBI Gelişmiş Arama Sayfası (ncbi.nlm.nih.gov web adresi, 2015)

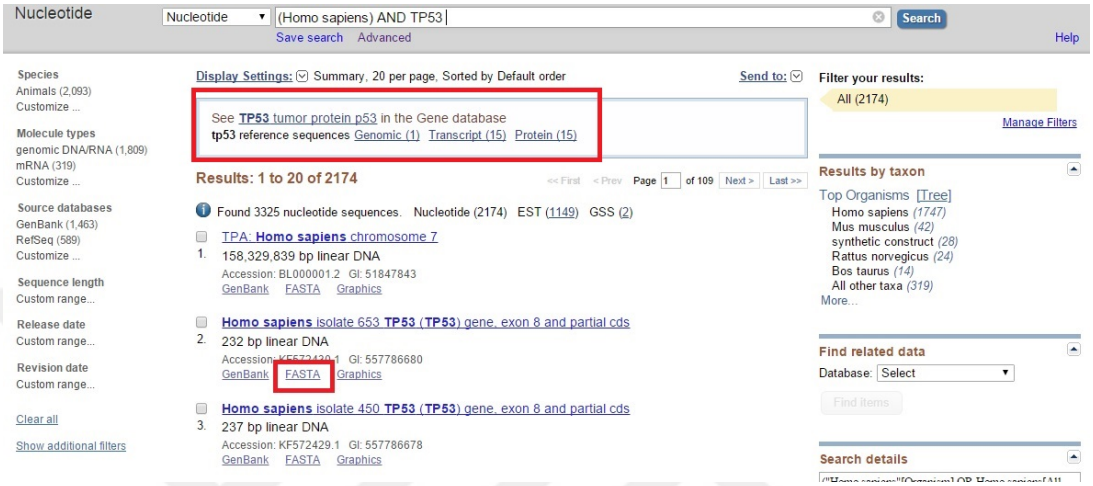


Şekil 2.7. NCBI Gelişmiş Arama Sayfası Seçimler (ncbi.nlm.nih.gov web adresi, 2015)

Gelişmiş aramada denemek amacıyla ***Homo sapiens*** yazılabilir. Örneğin ev ***Mus musculus***(ev faresi)'da yazılabilir. Organizma belirtildikten sonra "History" deki önceden

yapılan arama “Add” e tıklanarak aramaya eklenebilir ve üstteki **Search** butonuna tıklanarak arama gerçekleştirilebilir (Şekil 2.7).

Arama sonucu adında **TP53** geni geçen türü ***Homo sapiens*** olan tüm DNA dizileri listelenecektir. Bu listede ilk çıkan sonuç aranan genin Homo sapiens tumor protein p53 yani TP53 olduğu ortaya çıkmaktadır.



Şekil 2.8. NCBI Gelişmiş Arama Sonuç Sayfası (ncbi.nlm.nih.gov web adresi, 2015)

Listedeki her bir arama sonucunun aşağısında **FASTA**, **GenBank**, **Graphics** ve **Related Sequences** linkleri bulunmaktadır. Aranan genin nükleotid dizisini almak için, buradaki **FASTA** linkinden ulaşılmaktadır.

FASTA bağlantısı kullanıldığında (Şekil 2.9), ilgili genin **FASTA** formatında nükleotid dizisi gözükecektir. Dikkatlice bakılırsa görüntülenen dizi formatı “>” karakteri ile başlamaktadır ve sonrasında, gen id numarası, genin adı ve genin kromozom numarası bilgileri gözükmektedir. Bu bilgilerinden sonra ise nükleotid dizisi görülmektedir.

Şekil 2.9. NCBI Gelişmiş Arama FASTA Sonuç Sayfası (ncbi.nlm.nih.gov web adresi, 2015)

Seçilen nükleotid dizisi kısa bir dizi olup toplam 5 satırdan oluşmakta ve 232 adet baz çiftinden (232 bp) müteşekkildir

3. DNA DİZİLEME VE DİZİLEME YÖNTEMLERİ

3.1. DNA Dizileme

Bu bölümde DNA dizilemenin önemine ve kullanılan yöntemlere kısaca değinilmiştir. DNA dizilemesi(sekanslama), bir DNA molekülündeki nükleotid bazlarının (A,T,G,C) sırasının belirlenmesi işlemine verilen addır. DNA dizileme;

- Bireysel genotiplerin tanımlanması
- DNA Polimorfizminin tespit edilmesi
- Genetik çeşitlilik çalışmaları
- Gen/genom haritalarının çıkarılması
- Kantitatif özellikteki gen lokuslarının saptanması
- Hastalık ve genetik bozuklukların teşhisi
- Ebeveyn ve akrabaların tespiti
- Filogenetik çalışmalar
- Moleküler arkeoloji
- Adli tıp çalışmaları vb. alanlarda kullanılmaktadır.

DNA dizilerinin öğrenilmesi biyoinformatik, biyoteknoloji, adli bilişim, temel biyoloji, ve tıbbi tanı koyma gibi alanlarda önemli bir hâle gelmiştir. DNA dizilemesi biyolojik keşif ve incelemeleri oldukça hızlandırmıştır. DNA dizilemede kullanılan modern teknolojilerin imkân verdiği hızlı DNA dizilemeyle İnsan Genom Projesi adıyla insan gen haritası çıkarılabilmiştir. Buna benzer çalışmalarla pek çok bitki, hayvan ve mikrobun tam gen dizisi elde edilmiştir.

1970'lerde DNA dizilerinin ilk örnekleri akademisyenler tarafından iki boyutlu kromatografi kullanılarak oldukça zor yöntemlerle elde edilmiştir. Sistematik analiz kullanan boya tabanlı dizileme metotlarının gelişmesiyle (Olsvik, Wahlberg, Petterson, 1993) DNA dizilemesi çok daha kolay hale gelmiş ve birkaç kat daha hızlı yapılabilmiştir (Pettersson, Lundeberg, Ahmadian, 2009).

DNA dizilemesi ve analizi ile genetik denetim mekanizması ve gen yapısı ile ilgili birçok bilgi elde edilebilmiştir. Rastgele bir organizmadan fazla miktarda ham DNA elde edilmesine olanak veren rekombinant DNA tekniğinin gelişmesine ile birlikte DNA dizisi inceleme metotları da gelişmiştir. 1960'lı yıllarından itibaren DNA dizileme ile ilgili incelemeler kabaca Çizelge 3.1'de özetlenmiştir (Zülal, 2001).

Çizelge 3.1. DNA Dizi Analizi Kronolojik Gelişmeler (Zülal, 2001)

Yıl	Geliştirici/ Kuruluş	Açıklama	Kaynakça
1965	Robert HOLLEY	74 nükleotidlik bir tRNA dizi analizi yapıldı	(Holley, 1965)
1977	A. MAXAM – W. GILBERT ve F. SANGER	2 adet DNA dizi analizi metodu uygulandı	(Maxam, Gilbert, 1977; Sanger, 1977)
1982	Akiyoshi WADA	DNA dizi analizinin otomatik olarak yapılması fikri ortaya atıldı	(Kenneth, 1991)
1986	L. HOOD ve L. SMITH (California Teknoloji Enstitüsü)	DNA dizi analizinde kullanılmak üzere otomatik dizileme makinesi yapıldı	(Smith, 1986)
1991	Edward UBERBACHER	GRAIL adında gen bulma programı kullanılmaya başlandı	(Evelyn, 2013)
1992	Daniel COHEN ve Fransız Ekibi	İnsanın 21. kromozomun DNA dizi analizi tamamlandı	(Evelyn, 2013)
1995	Craig VENTER, Claire FRASER ve Hamilton SMITH	*Haemophilus influenzea*' ya ait ilk DNA dizisi yayınlandı	(Fleischmann vd., 1995)
1996	Ulusal bir birlik	*S.cerevisiae* adında ekmek mayasının DNA dizisi yayınlandı	(Galibert, 1996)
1998	Washington Üniversitesi ve Sanger Center	*Caenorhabditis elegans'*ın DNA dizisi tespit edilip açıklandı	(C. Elegans Sequencing Consortium, 1998)
1999	ABD, İngiltere ve Japonyalı bilim adamları	İnsana ait 22. kromozomun DNA dizisi tamamlandı	(Dunham vd., 1999)
2000	Celera ve birlikte çalıştığı üniversiteler	*Drosophila melanogaster'* in DNA dizisi elde edilip açıklandı	(Adams, 2000)
2000	Celera ve işbirliği içinde olduğu üniversiteler	İnsan Genom Projesi çalışanları ve Celera kurumu insan gen haritası taslağını tamamladığını açıklandı	(White House Press Release, 2000)
2000		DNA dizisi açıklanan ilk bitki "*Arabidopsis thaliana"*	(Kaul vd., 2000)
2003	David PAGE ve çalışma arkadaşları (Whitehead Enstitüsü)	Y kromozomu dizi analizi çalışmalarını tamamlandığını duyurdu	(Whitfield, 2003)
2004	Linda J Mullins, John J Mullins	Tarla faresi genomunun 'yüksek kalitede' taslağı yayınlandı. Genom normal fare genomuna göre daha büyük fakat insan genomundan daha küçüktür.	(Mullins, 2004)
2005	Uluslararası HapMap Konsorsiyumu	Nature'da HapMap (İnsan Genetiği Varyasyon Haritası) raporu yayınlandı	(The International HapMap Consortium, 2005)
2007		DNA dizileme hızını 70 kat arttıran yeni bir DNA dizileme teknolojisi (NGS) tanıtıldı	http://www.yourgenome.org/facts/timeline-history-of-genomics
2008		1000 Genom Projesi başlatıldı. NGS ile dizileme maliyeti ciddi oranda düştü.	http://www.yourgenome.org/facts/timeline-history-of-genomics
2009	Michael R. Stratton, P. Andrew Futreal, Peter	İlk kapsamlı kanser genomu analizi yayınlandı.	(Michael, 2009)

	J. Campbell		
2010		10 yılda 4000 sağlıklı, 6000 genetic hastalıklı 10000 insan genomunun karşılaştırılacağı "WellCome Trust UK10K" projesi duyuruldu.	(WellCome Trust, 2010)
2012	Nature Publishing Group	ENCODE projesi insan genomunun aktif bölgelerini açıklayan 30 araştırma yayınladı.	(Nature Publishing Group, 2012)
2013		ABD Yüksek Mahkemesi doğal olan DNA'nın patentlenemeyeceğine hükmetti.	(The Guardian, 2013)

3.2. DNA Dizileme Yöntemleri

DNA dizileme ve analizinde günümüzde dört yöntem kullanılmaktadır. Bu dört yöntem;

a. Maxam ve Gilbert Dizileme Yöntemi (Maxam, 1977)
b. Sanger ve Coulson Dizileme Yöntemi (Sanger, 1977)
c. Shotgun Dizileme Yöntemi
d. Pyrosekanslama (Durmaz, 2010)

2000'li yılların başlarına kadar Sanger–Coulson'un yöntemi Maxam-Gilbert yönteminden daha yaygın bir şekilde kullanılmaktadır (Durmaz, 2001).

3.2.1. Maxam ve Gilbert Dizileme Yöntemi

Uzunlukları farklı DNA parçalarının birleşmesi ile sonlanmış DNA'yı kesmek için dimetil sülfat, formik asit veya hidrazin kimyasallarının kullanıldığı yönteme denir (Maxam, 1977). Tek bir DNA dizisi jelinde 40 klonun incelebilmesini sağlar. Metodun ana prensibi, kimyasallar kullanılarak değiştirilen DNA'da bazlarının ve daha sonra farklılığa uğratılmış piperidin gibi nükleotidlerin bulunduğu noktalar üzerinden zinciri kırması temeline dayanır.

Dimetil sülfat pürinlerin kırılması için kullanılan kimyasaldır. DNA, asidik ortamda adenin bazından, bazik ortamda guanin bazından kırılmaktadır. Hidrazin ile de primidin bazları kırılır. DNA'yı timin ve sitozin bazından hidrazin kırabilir. Bazik ortamda ve fazla miktarda tuz yoğun ortamda DNA sitozin bazı ile kırılır.

Allan MAXAM ve Walter GILBERT tarafından geliştirilen metodun temeli dimetil sülfat formik asit veya hidrazinin DNA'da bulunan bazları değiştirip ve sonrasında bunlara eklenen piperidinin farklılığa uğrayan nükleotidlerin olduğu bölgelerden var olan zinciri kırmasına dayanmaktadır (Maxam, 1977).

Çizelge 3.2. Maxam ve Gilbert Yönteminin Kimyasalları (Durmaz, 2001)

İlgili baz	Baza has kimyasal	Baz ayırmada kullanılan	Zincir kırmada kullanılan
G	Dimetil sülfat	Piperidin	Piperidin
A+G	Asit	Asit	Piperidin
C+T	Hidrazin	Piperidin	Piperidin
C	Hidrazin+baz	Piperidin	Piperidin

Bu yöntemde; 1. adım: Nükleotid dizisi tespit edilecek DNA önce 5' ucundan floresan boya veya 32P ile işaretlenir. DNA'nın çift sarmalı ayrılır veya DNA kullanışlı bir restriksiyon enzimiyle kesilerek DNA'nın sadece bir ucundan işaretlenmesi sağlanır. 2. adım: DNA molekülleri 4 farklı tüpe ayrılarak A, C, G veya T nükleotidlerini değiştirmek ve kırmak için zorunlu reaksiyonlar uygulanır. Reaksiyon için kısıtlı bir süre verilerek her tüpte farklı pozisyonlardaki hedef nükleotidlerden kırılmış DNA parçaları elde edilir.

Sonuç olarak kırılmanın bulunduğu yere göre hepsi 5' yönünde işaretli fakat boy olarak birbirinden farklı DNA dizi parçası elde edilmiş olur. Bu elde edilmiş boyca gittikçe kısalmış DNA dizileri, jel elektroforeziyle birbirlerinden büyüklüğe göre ayrılır. Otoradyografi uygulanması suretiyle bantlar görüntülenmektedir (Klug, Cummings, 2000).

3.2.2. Sanger ve Coulson Dizileme Yöntemi

Enzimatik DNA sentezine dayanan bu yöntem belli zamanlara kadar çok fazla kullanılan dizi analiz tekniklerinden biridir. Bu yöntemde DNA dizisi tespit edilecek olan DNA ipliği, sentezlenecek DNA ipliği için kalıp olarak kullanılmaktadır.

DNA sentezini yapmak için ters transkriptaz, taq DNA polimeraz, klenov veya sequenaz enzimlerinden biri kullanılmaktadır. Bu metodun temeli, DNA polimerazın dNTP ve ddNTP'leri substrat olarak kullanabilmesi temeline dayanmaktadır.

Teknik açıdan dizi analizi üç aşamadan oluşmaktadır.

a. Polimeraz zincir tepkimesi
b. Jel elektroforezi ve bilgisayarda değerlendirme
c. Dizileme tepkimesi

DNA tek zincir haline getirilip, tepkimeye girecek olan karışımda çok miktarda dört türde normal nükleotid (A,T,C,G için d[A,T,C,G]TP) bulunmaktadır. Karışımda aynı anda diziyi rastgele sonlandırmada kullanılan değişik renkte flouresan kimyasallarla işaretlenmiş 4 tür dideoksi nükleotidler (A,T,C,G için dd[A,T,C,G]TP) vardır.

DNA polimeraz I ise sentezde zincirin uzaması aşamasında gerekli olup, hedef DNA PZR ile denatürasyon, bağlanma ve uzama basamakları yapılarak çoğaltılır.

Dizileme için gerekli olanlar;

- Deoksinükleotid (dNTP)
- Dideoksinükleotid (ddNTP)
- DNA kalıbı
- Taq DNA polimeraz
- Primer

Primer, DNA kalıbı, DNA polimeraz ve dNTP'ler ortama konulurak işaretli nükleotidin yapıya katılımı temin edilir. Burada kullanılan ddNTP'lerin yoğunluğu diğer maddelere göre düşük olmalıdır. Primere de işaretleme yapılabilir. İşaretlemeden sonrası zincir sonlanması tepkimelerine geçilir. Elde edilen karışım dört bölüme ayrılarak ayrı ayrı tüplere konulur. Bu 4 tüpe gerekli enzimler faktörleriyle beraber düşük derişimli farklı ddNTP'ler eklenir ve inkübe edilir. dNTP'ler ve ddNTP'ler aynı karışıma konulursa aralarında bir rekabet olur. Substrat olarak dNTP'ler kullanılan süre boyunca uzama sürer. Sentezin herhangi bir aşamasında yapıya dideoksi girer ise tepkime durur. 4 tüpte de aynı anda bağımsız olarak birçok tepkime oluşur. Sonuç olarak primerin sonuyla başlayıp prematüre sonlanmaların bölgelerine doğru türlü uzunlukta DNA kısımları oluşur. Dizi analiziyle elde edilmiş DNA dizileri jel üzerinde radyoaktif, flouresan ve gümüş boyalarla işaretlenip tespit edilebilir.

Dizileme tepkimesi PZR benzeri üç ana aşamada ve 30-40 çevrimde oluşur. Sentezlenen DNA'ya bir ddNTP'nin eklenmesi 3" doğrultusunda OH grubu olmadığından bu durum sentezi durdurur. Tepkime sonunda deoksinükleotidler ile uzamış ve ddNTP'lerle sonlanmış DNA dizileri elde edilir. Bazlar (sonlandırıcı özellikli) flouresan boyalar ile işaretlenebilir.

Oluşan DNA dizilerine jel elektroforezi uygulanabilir ve otomatik dizi analizi aletlerince okumalar yapılabilir. DNA parçaları jel elektoforezinde elektriksel alanda uzunluğa göre sıralanır ve DNA dizisi jelden okunabilir. Poliakrilamid jeller yüksek ayırım gücüne sahip olup uygun voltaj ve uygun sürede tek bir nükleotid farkını dahi ayırabilir.

Sonuç olarak iki yöntemde (Maxam-Gilbert ve Sanger-Coulson) üç temel aşamadan meydana gelmektedir.

a. DNA'nın hazır hale getirilmesi

Her iki inceleme esnasında da tek sarmallı DNA parçaları hazırlanır. DNA dizi analizi metodu esnasındaki ana değişiklik DNA parçalarının üretilme şeklinden kaynaklanır.

b. Tepkimeler

- Hem Maxam-Gilbert hem de Sanger yönteminde genel kural, DNA'yı işaretlenmiş 4 parçaya ayırmaktır.
- Her parçayı oluşturan tepkime baza özgüdür; belli bir bazın DNA dizisi bulunduğu konuma uygun uzunlukta bir parça oluşturur.
- Örnek olarak 5I-pAATCGACT-3I şeklinde bir oligo nükleotid için sadece C ile sonlanan parçalar oluşturan bir tepkime, 4 ve 7 nükleotid uzunluğunda parçalar (pAATC ve pAATCATC) oluşturur.
- Aynı oligo nükleotid için G ile sonlanmış parçalar oluşturan bir tepkime ise sadece 5 nükleotidli bir parça (pAATCG) oluşturur.

c. Jel elektroforezi (Yüksek Voltajlı)

DNA'da mevcut olan dört baza (A,G,C,T) uyan işaretlenmiş parçaların elektroforetik olarak ayrıldıklarında dizinin direkt okunabildiği bir bantlar merdiveni meydana gelir ve böylece nükleotid dizisi tespit edilmiş olur.

3.2.3. Shotgun Dizileme Yöntemi

Fazla miktarda klonlanmış DNA parçalarının birçok parçaya bölünerek alt klonlar şeklinde dizilemenin yapıldığı bir yöntemdir. DNA parçaları dizilendikten sonra gerçek DNA'nın yeniden yapılandırılması sağlanır. Bu yöntemin amacı; hız kazanmak ve doğruluk oranı çok yüksek olan sonuçlara ulaşmaktır. Hemem hemen 10000 bazda 1 baz hata oranı ile çalışma yapıldığı varsayılmaktadır. Özellikle kromozom incelemelerinde ve genom çalışmalarında tercih edilmektedir.

Avantajları;

- Eşleme aşamalarını ortadan kaldıran tüm genom Shotgun dizilemesi, klon-klon dizilemeden çok daha hızlı işlem yapmaktadır.
- Tüm genom Shotgun dizilemesi klon-klon dizilemesi ihtiyacı olduğu DNA'nın bir kısmını kullanır.
- Var olan referans dizisi varsa tüm genom Shotgun dizileme özellikle etkilidir. Var olan referans genomu hizalayarak genom dizisini bir araya getirmek çok daha kolaydır.
- Shotgun dizileme yöntemi genetik harita gerektiren yöntemlerden çok daha hızlı ve daha ucuz bir yöntemdir.

Dezavantajları;

- Shotgun dizilerini bir araya getirmek için devasa bilgisayar gücü ve gelişmişi yazılım gereklidir. Bir memelinin genomunu dizilemek için (milyarlarca baz) yaklaşık 60 milyon DNA dizi okumasına ihtiyaç vardır.
- Bir araya getirilmiş genomdaki hatalar genetik bir harita kullanılamadığından çok fazladır. Ancak genellikle bu hataları diğer metotlara göre çözmek daha kolaydır ve referans genom kullanılabilirse bu hatalar minimize edilebilir.
- Uygun referans bir genom varsa tüm genomun Shotgun dizilemesi gerçek şekilde yapılabilir aksi takdirde var olan bir genom olmaksızın tüm genomu bir araya getirmek çok zordur.
- Tüm genom Shotgun dizilemesi, diğer klon-klon dizileme gibi çok yoğun emek isteyen dizileme türlerinin çözebildiği hatalara neden olabilmektedir.
- Tekrarlı genom ve dizileri bir araya getirmek çok daha zor olabilmektedir.

3.2.4. Pyrosekanslama

Dizi analizinde sık kullanılan metotlardan biri olan Sanger yöntemini zaman alması, birçok aşama içermesi gibi farklı dezavantajları ortadan kaldırmak için, Pal Nyrén tarafından 1986 yılında geliştirilen bir yöntemdir (Nyrén, 2007). Tek nükleotid eklenmesi (Single-nükleotide addition –SNA) yöntemi ile dizi analizi yapan bir yöntemdir. Sentezleme ile dizi analizi yapma ilkesine dayanmaktadır. DNA sentezi sırasında ortaya çıkan pirofosfat'ların tespit edilmesi temeline dayanan gerçek zamanlı kantitatif dizi analizi yöntemidir. Bu yöntemde işlem PZR ürünlerinin tek zincir DNA(ssDNA)'ya dönüşümü ile başlar. Tek iplikli DNA kalıp olarak kullanılıp izole edilir ve her bir primer çifti 5" ucundan biotin ile işaretlenir (Durmaz, 2010).

Kullanılan Alanları

- Heteroplazmik DNA dizilemesi
- Adli tıp incelemeleri
- Fungal tiplendirme ve direnç
- Metilasyon incelemeleri
- Bakteriyal tiplendirme ve direnç
- İnsersiyon, delesyon saptanmaları
- Viral tiplendirme ve direnç

Rastgele bir organizmadan elde edilen DNA'ya uygulanabilir. Örneğin otoimmün hastalıklar, koroner arter hastalığı, diabet, alzheimer hastalığı gibi klinik araştırmaları kolaylaştırmakta ve hızlandırmaktadır.

Mikrobiyoloji Yönünden Yararları

- Gerçek DNA dizilim bilgilerinin klinik olarak müsait bir zaman içinde elde edilmesi
- Mikrobiyal dizilemesi 1 saat içinde yapılabilmektedir
- Tür tanımlaması ve direnç karakterizasyonunun tek bir çatı içinde yapılabilmesi,
- Antibiyotik direnç tanımlaması, çok kopyalı genlerin miktar tayinleri, viral ve fungal yükleri ölçülebilir.

3.3. DNA Dizi Analizinin Otomatik Yapılması

İnsan Genom Projesi çok sayıda DNA dizi analizi yapılmasını gerektiren büyük projelerdir. Bundan dolayı yüksek iş gücü, artan inceleme sayısı, uzun süre gerektirmektedir. Bu gereksinimlerden dolayı otomasyonun kullanılması kaçınılmazdır. Otomatik DNA dizi analizleri süre kazandırmasının yanında, daha normal çalışma şartları ve elde edilen sonuçların daha iyi değerlendirilmesinde de faydalı olmuştur. Sanger'in enzimatik DNA sentezini uygulayan zincir sonlanma yöntemi otomatik dizi analizinde de kullanılmıştır. DNA dizi analizini otomatik yapan aygıtlar, bir bilgisayarda yüklü yazılımlar ile bu yazılımlarınların yönettiği elektroforez sistemini içermektedir. Burada lazer ışık ile monokromatik bir ışık oluşturulur. DNA'nın bulunduğu jelmatriks bu ışık ile taranır. Elektroforez boyunca DNA'ya bağlanan floresan boya taranan bölgeye geldiğinde uyarılır. Uyarılan bu boya karakteristik dalga boyunda bir ışığı geri yansıtır. Yansıyan ışık demeti bir detektör aracılığıyla kaydedilir.

Kaydedilmiş bilgiler bilgisayar yazılımı ile değerlendirilip sonuçlar grafik olarak veya matematiksel şekilde bilgisayar ekranına aktarılır. DNA dizi analizi aygıtlarında 6-1000 baz aralığında güvenli okumalar yapılabilmektedir (Sambrook, Maniatis, 1989).

3.4. Yeni Nesil Dizileme Teknikleri

Yukarıda verilen dizileme yöntemleri halen kullanılan yöntemler olmasına rağmen bu yöntemlerle birlikte çeşitli teknikler geliştirilmiştir. Özellikle bilgisayar bilimlerinin katkısı bu tekniklerin geliştirilmesinde etkili olmuştur. 1998 yılında ortaya çıkan otomatik dizileme araçları ve kılcal dizileme makineleri kullanılarak 2001 yılında insan genom projesinin tamamlanması ilgili yazılım ve Sanger dizileme teknolojisinin başlıca araçları olmuştur. Yeni nesil dizileme (NGS) teknolojilerinin geniş uygulamalarla hızlı gelişimi ile genomik dizi bilgisi, hayat sırlarını çözmek gibi hedeflere ulaşılmasını yardım etmek için, daha iyi bitkiler elde etmek için, patojenlerin tespiti ve yaşam kalitelerini iyileştirmesi için kullanılmaktadır. NGS sistemleri Life Science şirketinden "SOLiD/Ion Torrent PGM", Illumina şirketinden "Genome Analyzer/HiSeq 2000/MiSeq" ve Roche şirketinden "GS FLX Titanium/GS Junior" programları ile temsil edilmektedir. NGS sistemleri Roche 454, AB

SOLiD, Illumina GA/HiSeq ve Compact PGM Sequencers başlıkları altında incelenebilir (Lin Liu, 2012).

Çizelge 3.3.a. Dizileyicinin Avantajı ve Mekanizması (Lin Liu, 2012)

Dizileyici	**454 GS FLX**	**Hiseq 2000**	**SOLiDv4**	**Sanger 3730xl**
Dizileme mekanizması	Pyrosekanslama	Sentez ile sekanslama	Ligasyon ve 2 baz kodlama	Dideoksi zincir sonlandırma
Okuma uzunluğu	700 bp	50SE, 50PE, 101PE	50 + 30 bp veya 50 + 50 bp	400 ~ 900 bp
Doğruluk	%99.9	%98, (100PE)	%99.94 ham veri	%99.999
Okumalar	1 M	3 G	1200 ~ 1400 M	-
Çıkış verisi	0.7 Gb	600 Gb	120 Gb	1.9~84 Kb
Zaman	24 Saat	3~10 Gün	SE için 7 Gün, PE için 14 Gün	20 Dakika ~ 3 Saat
Avantajı	Uzun okuma, hızlı	Yüksek başarım	Doğruluk	Yüksek kalite, büyük okuma uzunluğu
Dezavantajı	6'dan daha fazla polibaz hata oranı, yüksek maliyet, düşük başarım	Kısa okuma montajı (bir araya getirme)	Kısa okuma montajı (bir araya getirme)	Yüksek maliyet, düşük başarım

Çizelge 3.3.b. Bileşenler ve Dizileyici Maliyeti (Lin Liu, 2012)

Dizileyici	**454 GS FLX**	**Hiseq 2000**	**SOLiDv4**	**Sanger 3730xl**
Enstrüman fiyatı	Enstrüman 500.000$, çalışma başına 7.000$	Enstrüman 690.000$ (30 kat) insan genomu başına 6.000$	Enstrüman 490.000$, 100 Gb başına 15.000$	Enstrüman 95.000$, 800 bp reaksiyon başına 4$
İşlemci	2* Intel Xeon X5675	2* Intel Xeon X5560	8* processor 2.0 GHz	Pentium IV 3.0 GHz
Bellek	48 GB	48 GB	16 GB	1 GB
Sabit Disk	1.1 TB	3 TB	10 TB	280 GB
Hazır otomasyon kütüphanesi	Var	Var	Var	Yok
Diğer gerekli araç	REM e sistem	cBot sistem	EZ beads sistem	Yok
Maliyet /milyon baz	10$	0.07$	0.13$	2400$

Çizelge 3.3.c. Dizileyici Uygulaması (Lin Liu, 2012)

Dizileyici	454 GS FLX	Hiseq 2000	SOLiDv4	Sanger 3730xl
Yeniden Dizileme	-	Evet	Evet	-
De novo	Evet	Evet	-	Evet
Kanser	Evet	Evet	Evet	-
Dizi	Evet	Evet	Evet	Evet
Yüksek GC Örneği	Evet	Evet	Evet	-
Bakteriyel	Evet	Evet	Evet	-
Büyük genom	Evet	Evet	-	-
Mutasyon tespiti	Evet	Evet	Evet	Evet

Yeni nesil dizileme teknikleriyle dizileme maliyeti dramatik şekilde düşüş göstermiştir. Çizelge 3.3'de yeni nesil DNA dizileme tekniklerinin (NGS) detaylı bir karşılaştırması yapılmıştır (Lin Liu, 2012).

Çizelge 3.3'de verilen detaylı karşılaştırmada aşağıda belirtilen hususlar gözetilmiştir (Lin Liu, 2012).

1. Tüm veri BGI(Beijing Genomics Institute)'de günlük ortalama performans ile alınmıştır. Dizilemeciler(Sekanslayıcılar) (genelde Hiseq 2000) yaklaşık %80 ortalama ile çalıştığında ortalama günlük sekans veri çıkışı BGI'de yaklaşık 8 TB'dır.
2. 454 GS FLX Titanium reaktif maliyeti 400 bp sekanslama baz alınarak, Hiseq 2000 reaktif maliyeti 200 bp sekanslama baz alınarak ve SOLiDv4 reaktif maliyeti 85 bp sekanslama baz alınarak hesaplanmıştır.
3. HiSeq 2000, 50SE, 50PE veya 101PE gibi sekanslama tiplerinde daha esnektir.
4. Özellikle 30 kattan daha fazla kapsamalarda SOLiD yüksek doğruluğa sahiptir. Bu yüzden yeniden sekanslama, hedeflenmiş yeniden sekanslama ve transkriptom sekanslamada varyasyonların tespitinde yaygın şekilde kullanılmaktadır. Şeritler maliyeti azaltmak için bağımsız olarak çalıştırılabilir.

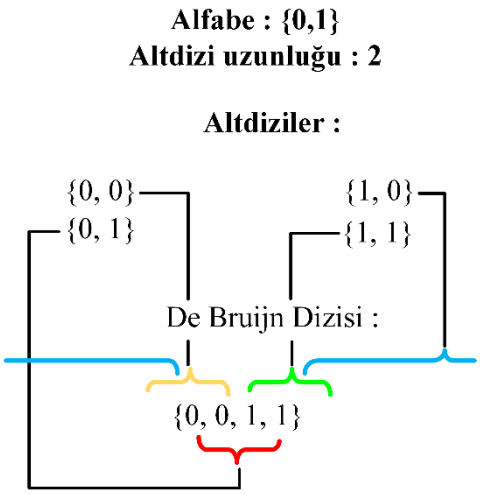
Sanger okumaları için belirgin başarıya rağmen, De Bruijn graf temelli yaklaşım yaygın olarak kullanılmamıştır. Bu durum yeni nesil dizileme (NGS) teknolojilerinin gelmesiyle değişmiştir. NGS araçları milyonlarca küçük parçalı DNA dizilimlerini hızlı

ve ucuz maliyetli olarak kodlayabilmiştir. 4. bölümde yeni nesil dizileme teknolojilerinin çok sık kullanıldığı De Bruijn graf temelli yaklaşımlara daha detaylı değinilmiştir.

4. DE BRUIJN DİZİLERİ VE GRAFLARI

4.1. De Bruijn Dizileri

Hollandalı matematikçi Nicolaas Govert de Bruijn[1]'den ismini alan kombinatoryal matematikte (çözüm kümesi ayrık olan optimizasyon problemleri -NP zor problemler-) n dereceli k-lı bir De Bruijn dizisi *B(k,n)*, verilen k boyutlu bir A alfabesinde tüm olası n uzunluğundaki altdiziler için ard-arda tüm altdizileri içeren bir çevrim dizisidir.



Şekil 4.1. k=2 ve n=2 için De Bruijn Dizisi

Her *B(k,n)* De Bruijn dizisi k^n uzunluğundadır. B(k,n) De Bruijn dizisi için $\frac{(k!)^{k^{n-1}}}{k^n}$ tane farklı dizi vardır. Nicolaas Govert de Bruijn (de Bruijn, N. G., 1975)'e göre, yukarıdaki özellikleriyle birlikte her bir derece için (n) De Bruijn dizilerinin varlığı ilk defa ispat edilmiştir. 2 elemanlı alfabeler sayesinde 1894'te Camille Flye Sainte-Marie (Flye Sainte-Marie, 1894) ve Tanja van Aardenne-Ehrenfest[2]'den dolayı daha geniş alfabelere genelleştirme yapılmıştır.

De Bruijn dizisinin ilk bilinen örneği Sanskrit ölçüsünden gelir. Milattan önce yaşamış Pingala'nın çalışmasına göre, uzun ve kısa ünlülerin olası her üç heceli desenine bir isim verilir. Örneğin kısa-uzun-uzun için 'y', uzun-kısa-uzun için 'r'. Bu isimleri hatırlamak için *yamātārājabhānasalagaṃ* ipucu kullanıldı. İsmin başında her üç heceli desende; '*yamātā*' deseni kısa-uzun-uzun, '*rājabhā*' deseni uzun-kısa-uzun şeklinde gider. Bu ipucu 3-başlıklı ikili bir De Bruijn dizisine denktir.

[1] Hollandalı matematikçi ve Eindhoven Teknoloji üniversitesinde emekli profesör olan **Nicolaas Govert (Dick) de Bruijn** (9 Temmuz 1918 – 17 Şubat 2012) analiz, sayı teorisi, kombinatorik ve lojik alanlarında önemli katkıları olmuştur.

[2] **Tatyana Pavlovna Ehrenfest**, sonraları **van Aardenne-Ehrenfest**, (Vienna, 28 Ekim 1905 – Dordrecht, 29 Kasım 1984), Hollandalı matematikçi.

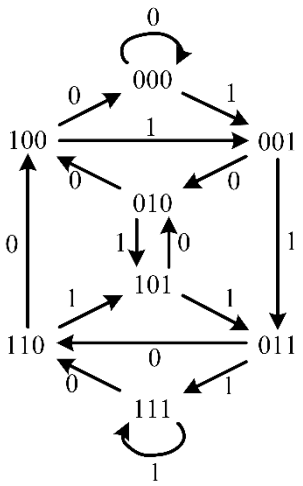
1894'te A. De Rivière Fransız problem dergisi L'Intermédiaire des Mathématiciens'daki bir makalesinde n uzunluğunda tüm 2^n ikili dizileri içeren 2^n uzunluğundaki dairesel dizilişinin varlığı sorusunu gündeme taşıdı. Aynı yıl C. Flye Sainte-Marie tarafından $2^{2^{n-1}-1}$ sayısı ile problem çözüldü. Bu çözüm unutuldu ve 2 boyutlu genel bir alfabe için böyle çevrimlerin varlığını bunları inşa eden bir algoritma ile 1934'te Martin (Martin, 1934) ispatlamıştır. Sonuç olarak 1944'te K. Posthumus $2^{2^{n-1}-1}$ sayısını tahmin etti. 1946'da De Bruijn bu tahmini ispat etmiştir.

4.1.1. De Bruijn Dizi Örnekleri

- A = {0, 1} alfabesi için $B(2, 3)$ De Bruijn dizisinde biri diğerinin tersi veya negatifi olan iki farklı dizi vardır: **00010111** ve **11101000**
- Aynı alfabede B(2,5) için 2048 De Bruijn dizisinin olası 2 dizisi: **00000100011001010011101011011111** ve **00000101001000111110111001101011** 'dir.

4.1.2. De Bruijn Dizisinin İnşası

De Bruijn dizileri k sembollü n dereceli bir Hamilton yolu baz alınarak oluşturulabilir. (n-1 dereceli De Bruijn grafının bir Euler çevrimi) Her 4 sembollü dizi, tam olarak bu dizinin her **köşesi** dolaşılır ve tekrar başlangıç noktasına dönülmesiyle oluşmaktadır. (Euler çevrimi) Amaç, 3-boyutlu De Bruijn grafının Euler çevrimini kullanarak 2^4=16 uzunluğundaki B(2,4) De Bruijn dizisini oluşturmaktır.



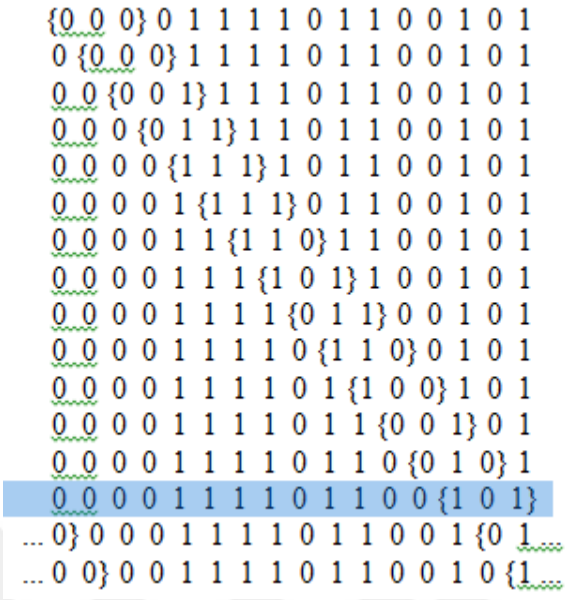
Şekil 4.2. Örnek Bir De Bruijn Grafı

Örneğin; Şekil 4.2'ye göre varsayalım ki aşağıdaki düğümler sırasıyla Euler yolu olsun.

000, 000, 001, 011, 111, 111, 110, 101, 011, 110, 100, 001, 010, 101, 010, 100, 000.

k uzunluğundaki çıkış dizisi; **0 0 0 0, _ 0 0 0 1, _ _ 0 0 1 1** olur. Bu şekilde son düğüme kadar bu sürdürülürse, De Bruijn dizisi Şekil 4.3'teki gibi olur;

0 0 0 0 1 1 1 1 0 1 1 0 0 1 0 1 (Şekil 4.3'de mavi olan)



Şekil 4.3. De Bruijn Dizisinin Bulunması

4.1.3. De Bruijn Dizi İnşası İçin Algoritma

Frank Ruskey'in Kombinatoryal Nesil (Ruskey, 2012) için baz aldığı De Bruijn dizisi elde etmede kullanılan Phyton kodu Çizelge 4.1'de verilmiştir.

Çizelge 4.1. Frank Ruskey'in De Bruijn Dizisi Elde Etme Kullanılan Pyhton Kodu

```
def de_bruijn(k, n):
a = [0] * k * n
sequence = []
def db(t, p):
if t > n:
      if n % p == 0:
            for j in range(1, p + 1): sequence.append(a[j])
else:
      a[t] = a[t - p]
      db(t + 1, p)
      for j in range(a[t - p] + 1, k):
            a[t] = j
            db(t + 1, t)
db(1,1)
return sequence
print(de_bruijn(2, 3))
```

Çigelge 4.1'e göre çıkış: **[0, 0, 0, 1, 0, 1, 1, 1]** şeklinde olacaktır. Bu kaba kodun karmaşıklığı O(n)=n.logn'dir.

4.1.4. De Bruijn Dizilerinin Kullanım Alanları

De Bruijn dizisi “enter” tuşu olmayan bir PIN-like kod kilidi üzerinde bir kaba-kuvvet (brute-force) saldırısını kısaltmada kullanılabilir ve girilen bu son n sayı kabul edilir. Örneğin 4-sayı kodlu sayısal bir kapı kilidi 10.000 uzunluğunda B(10,4) dizisi ile çözülür. Böylece sadece 10.000+3 = 10.003 (çözümlerin çevrimi gibi) tuşlamayla kapı açılır. Denenecek tüm kodlar ayrı olarak 4 x 10.000 =40.000 tuşa basmayı gerektirir.

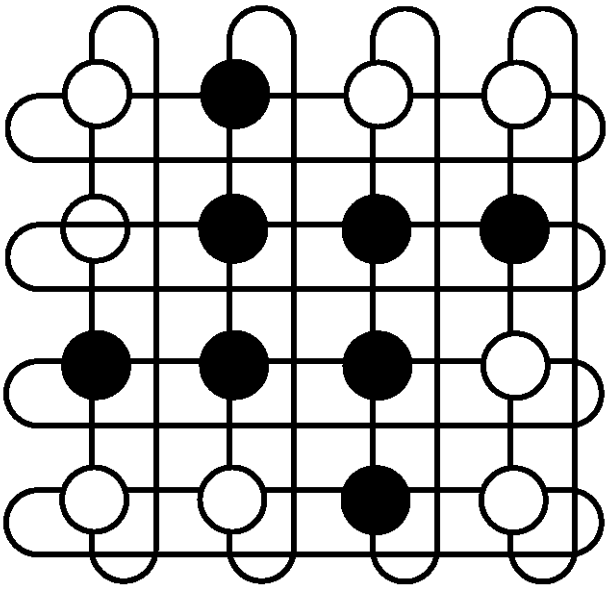
Bir dairesel nesnenin etrafında yazılmış bir De Bruijn dizisinin sembolleri (örneğin bir robotun tekerleği gibi) sabit bir noktaya bakan n ardışık sembolleri incelenerek açısını tanımlamak için kullanılabilir. Gray kodu benzer döner pozisyon kodlama mekanizması olarak kullanılabilir.

De Bruijn çevrimlerinin sinir sistemleri üzerinde uyarıcı düzenin etkisini inceleyen nöroloji ve psikoloji deneylerinde genel kullanımı vardır (Aguirre, 2011) ve bu çevrimler özellikle fonksiyonel manyetik rezonans görüntülemede kullanmak için hazırlanmıştır (cfn.upenn.edu web adresi). De Bruijn dizisi bir sözcükteki ilk veya son biti hızlıca bulmada kullanılabilir (Anderson, 1997-2009).

4.1.5. De Bruijn Torusu

Bir De Bruijn torusu her k dizisi için m.n matrisi tam olarak bir kez oluşan özelliği ile toroidal bir dizidir. Dizinin toroidal olarak ifade edilmesine gerek yoktur. 2 boyutlu bir dizide haritalanabilir. Çünkü dizinin toroidal’ı 4 tarafı da sarmaktadır.

Böyle bir model, döner kodlama için yukarıda tarif edilene benzer bir şekilde iki boyutlu pozisyon kodlama için kullanılabilmektedir. Konum, bitişik sensöre doğrudan m.n matrisi incelenerek belirlenebilir ve De Bruijn torusu üzerindeki pozisyonu hesaplanabilir. Şekil 4.4’de 2x2’li matrisi için de Bruijn torusu gösterilmiştir.



Şekil 4.4. 2x2 İkili Matris için De Bruijn Torusu

4.1.6. De Bruijn Kod Çözme

Bir De Bruijn dizisi veya torusunun belli bir başlık veya matrisinin pozisyonunu hesaplama, De Bruijn kod çözme problemi olarak bilinir. Bu kod çözme algoritmalarının karmaşıklığı O(nlogn) olup, diziler yinelemeli olarak inşa edilir (Tuliani, 2001) ve iki boyutlu duruma genişletilir (Hurlbert, 1993). Büyük diziler ve toruslarda pozisyonel kodlamanın kullanıldığı durumlarda De Bruijn kod çözmenin ilgi alanına girmektedir.

4.2. De Bruijn Grafları

Graf teorisinde, m sembollü, n boyutlu bir De Bruijn grafı sembol dizileri arasında örtüşmeleri gösteren yönlü bir graftır. Bu grafın verilen sembollerin olası tüm n uzunluğundaki dizisinden oluşan m^n adet düğümü vardır. Aynı sembol bir dizide birden çok görülebilir. $S=\{s_1,s_2,\ldots,s_m\}$ şeklinde sembol kümesi olsun, buna göre düğümlerin kümesi;

$V=S^n=\{(s_1,\ldots,s_1, s_1), (s_1,\ldots,s_1, s_2),\ldots, (s_1,\ldots,s_1, s_m), (s_1,\ldots,s_2, s_1),\ldots, (s_m,\ldots,s_m, s_m)\}$ şeklinde olur.

Düğümlerden biri, tek bir yerde tüm sembollerin sola kayması ve bu düğümün sonuna yeni bir sembol ekleyerek başka bir düğüm olarak ifade edilebilir. Bundan sonra elde edilen düğümün önceki düğümden yönlü bir kenarı vardır. Böylece yönlü kenarların kümesi;

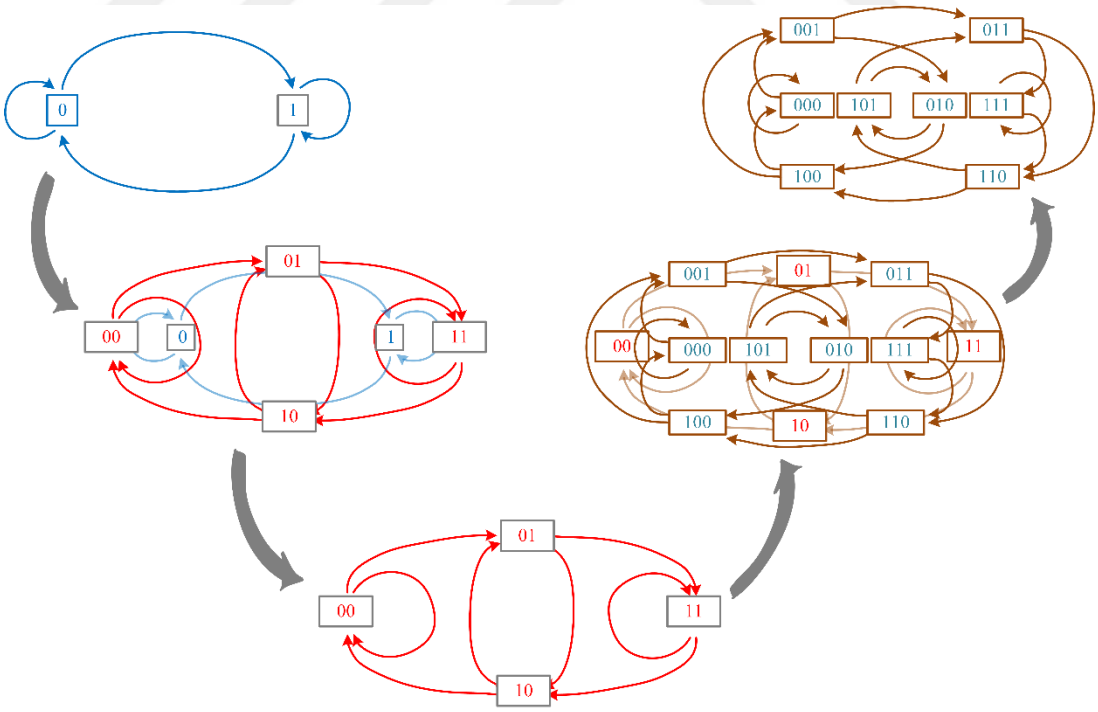
$E=\{((v_1,v_2,\ldots,v_n),(w_1,w_2,\ldots,w_n)):v_2=w_1, v_3=w_2,\ldots, v_n=w_{n-1}\}$ şeklinde olur.

De Bruijn grafları Nicolaas Govert de Bruijn'den ismini almasına rağmen, De Bruijn (de Bruijn, 1946). ve I. J. Good (Good, 1946) her ikisi tarafından bağımsız olarak geliştirilmiştir. Çok daha öncesinde, Flye Sainte-Marie (Flye Sainte-Marie, 1894) tarafından bu grafların özellikleri dolaylı olarak kullanılmıştır.

De Bruijn Graf Özellikleri

- n=1 ise herhangi iki düğüm arasında boş kenar olmaması şartıyla tüm düğümler toplam m^2 kenar ile bağlıdır.
- Her düğümün m adet giriş ve çıkış kenarı vardır.
- Her n boyutlu De Bruijn grafı aynı sembollere sahip n-1 boyutunda yönlü çizgi (line) grafıdır.
- Her De Bruijn grafı Euler ve Hamilton grafıdır. Bu grafın Euler ve Hamilton çevrimleri (line grafın inşası yoluyla bir diğerinin eşdeğeri) De Bruijn dizisidir.

Üç tane n küçük ikili De Bruijn grafının çizgi graf inşası Şekil 4.5'te gösterilmiştir. Şekil 4.5'te görüldüğü gibi, n-1 boyutunda De Bruijn grafının bir kenarı n boyutlu De Bruijn grafının bir düğümüne karşılık gelir. n-1 boyutlu De Bruijn grafında iki kenarlı bir yola, n boyutlu De Bruijn grafında bir kenara karşılık gelir (Zhang,1987).



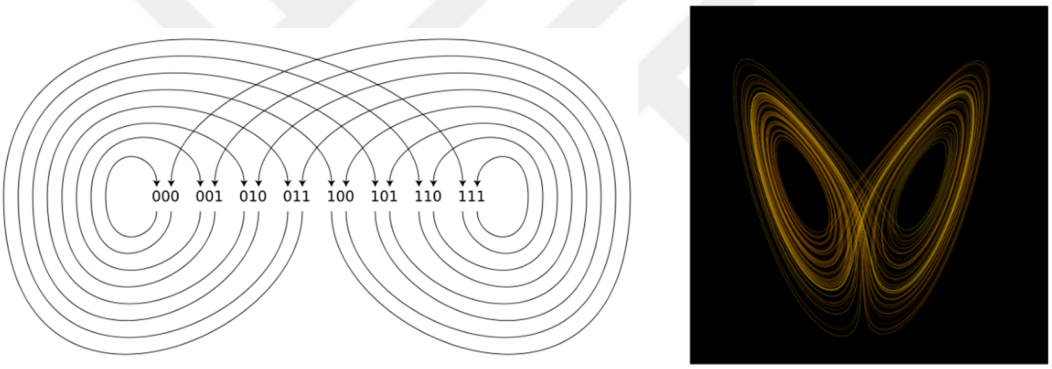
Şekil 4.5. İkili De Bruijn Grafının İnşası

4.2.1. De Bruijn Grafları ile Ölçülebilir Genom Yerleşimi

Tam bir kez grafın tüm düğümlerini ziyaret eden bir döngü bulma (Hamilton döngüsü problemi olarak adlandırılır) zor bir hesaplama sorunudur; ancak, tam olarak bir kez bir grafın tüm kenarlarını ziyaret eden bir döngü bulmak daha kolaydır. Bu algoritmik yaklaşım DNA parçalarını doğru yerleştirmede bilgisayar bilimcilerini motive etmiştir. Burada Hamilton çevriminde kullanılan her düğüm için bir k-harfli atamak yerine her düğümü (k-1)-harfli olarak tanımlayıp her kenara k-harfli atanarak De Bruijn grafının ideal olarak inşası sağlanmıştır. Bu algoritmik yaklaşımda bir Euler yolunun varlığını bulmak çok kolay olmakta ve genom yerleştirmede popüler çözüm halini almaktadır (Phillip, 2011).

4.2.2. Dinamik Sistemler

İkili De Bruijn grafları dinamik sistemler teorisinde benzer nesnelerdeki gibi Şekil 4.6'da gösterilen şekilde çizilebilmektedir.



Şekil 4.6. Dinamik Sistemler ve Lorenz Atraktörü

Bu benzetme kesin yapılabilir: n boyutlu m sembollü De Bruijn grafı Bernoulli haritasının bir modelidir.

$$x \rightarrow mx \bmod 1$$

Bu Bernoulli haritası (m=2 için 2x mod 1 haritası olarak adlandırılır) bir m-adik (Leroux, 2002) sayısının tek seferde anlaşılabildiği ergodik bir dinamik sistemdir.

4.2.3. De Bruijn Grafları Kullanım Alanları

- Bazı grid ağ topolojilerinde kullanılır.
- Dağıtık özüt (hash) tabloları Koorde protokolü De Bruijn graflarını kullanılır.

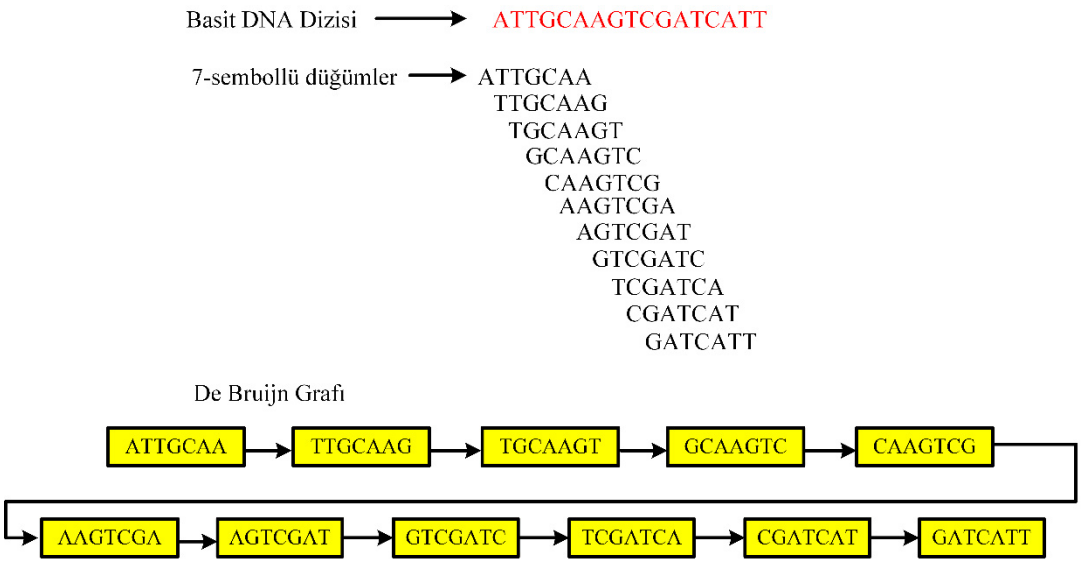
- Biyoinformatikte bir genomun okunan dizilerinin de novo (yeniden) yerleşimi için De Bruijn grafları kullanılır (Pevzner, Tang, Waterman, 2001; Pevzner, Tang, 2001; Zerbino, Birney, 2008).
- Bir genomun sadeleştirilmesinde kullanılır.

4.3. Basit DNA Dizileriyle De Bruijn Graflarının Oluşturulması

Kısa okuma yerleştirici (short read assembly) için yeni algoritmalarda çoğunlukla gen dizisi verisini gösteren ve saklayan De Bruijn grafları kullanılır. De Bruijn graflarının ne olduğu ve neden DNA kısa okuma dizilerinin bu kadar popüler olduğu burada açıklanmaya çalışılmıştır (homolog.us web adresi).

De Bruijn grafı k-harfli bileşenlerle bir diziyi göstermenin etkili bir yoludur. De Bruijn grafları geniş ölçekte problemler için kullanılmasına rağmen şimdilik nükleotid dizileriyle sınırlı tutulmuştur. Çoğunlukla makaleler De Bruijn graflarından gen dizisini elde etme ve kısa okumalardan (short reads) De Bruijn graflarını oluşturma üzerine yazılmıştır. Burada ilk önce bir gen dizisinin De Bruijn grafı ile başlayıp sonrasında bu graftan kısa okumaların nasıl yapıldığı açıklanmıştır.

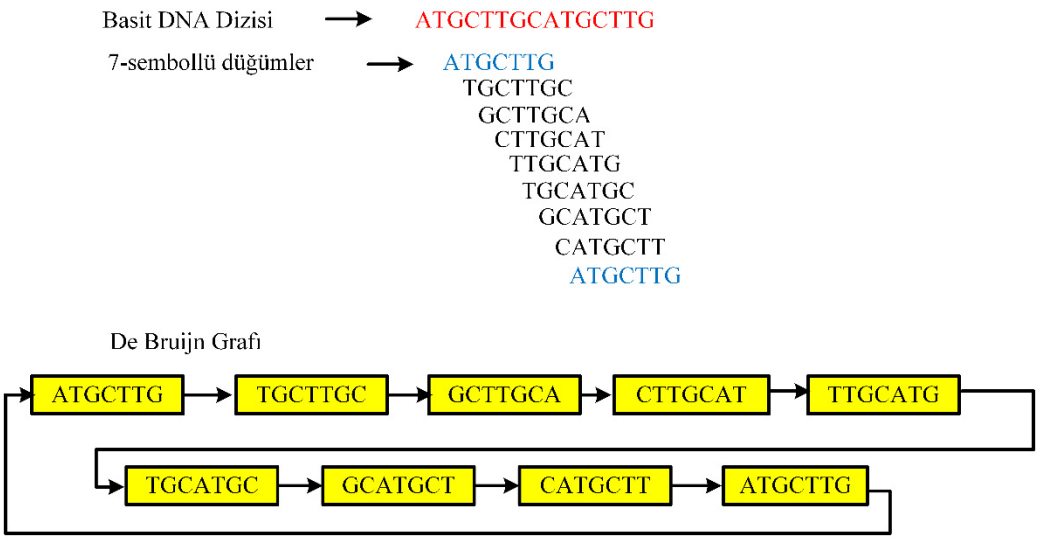
Bir De Bruijn grafı, uzun veya kısa herhangi bir gen dizisi için oluşturulabilir. Şekil Şekil 4.7'deki örnekte "ATTGCAAGTCGATCATT" gen dizisi, içinde örtüşme olan (örnekte k=7) 11 adet k-harfliye bölünmüş yönlü bir grafın düğümleri her biri 7 harfle oluşturulmuştur. Kenarlar orijinal dizi üzerinde 7-harfli komşular (düğümler) arasında çizilmiştir. Bu metot bağlı düğümlerdeki 6 (=k-1) nükleotidin örtüşmesini sağlamaktadır. Şekil 4.7'de basit bir örnek verilmiştir.



Şekil 4.7. k=7 için De Bruijn Grafı

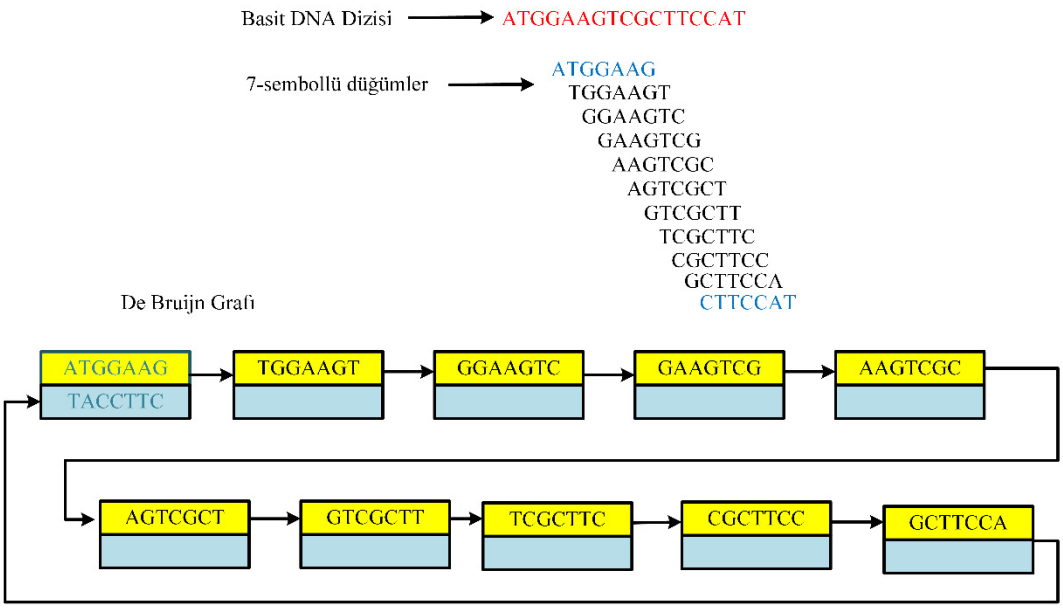
Orijinal gen dizisinde tekrar eden 7-harfli düğüm olmadığından Şekil 4.7'deki örnek basit

kalmıştır. Şekil 4.8'deki örnekte bazı tekrar eden düğümler verilmiştir. Bu örnekte en yüksek 5' açılı 7-harfli düğüm, en yüksek 3' açılı 7-harfli düğüm'de görülmüştür (her ikisi de mavi olarak gösterilmiştir).



Şekil 4.8. k=7 için De Bruijn Grafında Döngü

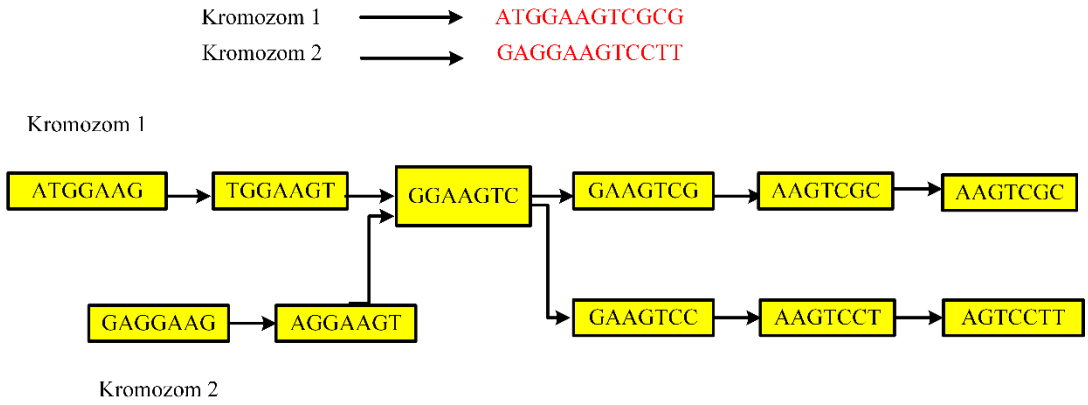
Bu durumda De Bruijn grafında bir döngü oluşur. Şekil 4.8'deki örnekte gösterilen düğümler gen dizisinin her iki sarmalı için gösterilmemesine rağmen, gerçekte her düğüm Şekil 4.9'da gösterildiği gibi çift sarmallıdır.



Şekil 4.9. k=7 için Çift Sarmallı De Bruijn Grafı

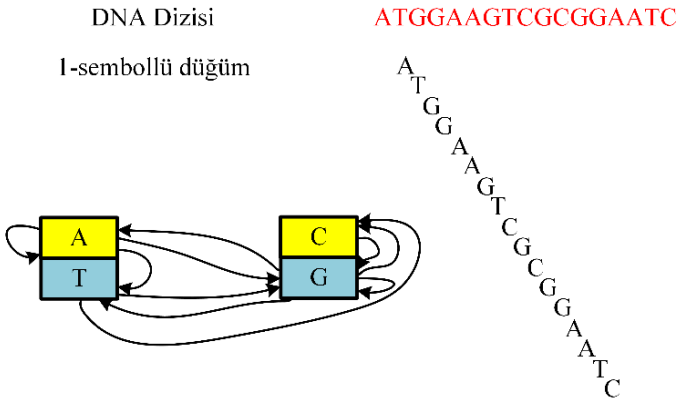
Şekil 4.9'daki örnekte en büyük 3' açılı 7-harfli düğüm en büyük 5' açılı 7-harfli düğümün tamamlayıcısının tersidir. Çift sarmallı De Bruijn graflarının Şekil 4.10'da görülen adımlar herhangi boyutta büyük bir gen dizisinin De Bruijn graflarıyla oluşturulması tekrar edilebilir.

Bir gen dizisi 2 ayrı kromozoma sahip olsa da De Bruijn grafları bu kromozomların Şekil 4.10'da görüldüğü gibi k-harfli düğümler örtüşürse ayrı kalmayabilir ve bağlantılar buna göre oluşturulur.



Şekil 4.10. k=7 için 2 Ayrı Kromozomlu De Bruijn Grafı

Gösterilen örneklerin hepsinde k=7 alınmıştır, fakat k çok küçük veya çok büyük bir tamsayı olabilir. k oldukça küçük veya 1 olabilir. Ancak Şekil 4.11'den görülebileceği gibi k=1 için De Bruijn grafı çok kullanışlı değildir.



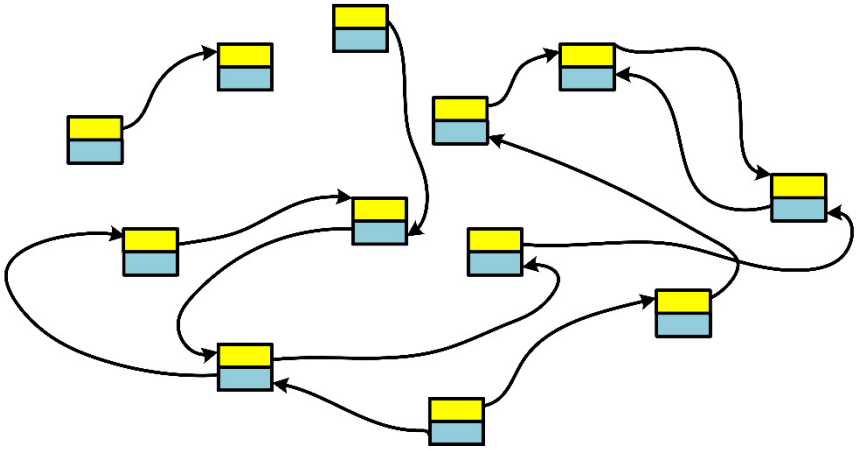
Şekil 4.11. k=1 için De Bruijn Grafı

Burada De Bruijn grafları basit örneklerle açıklanmıştır. Buradan hareketle bu graflar ile ilgili şu sonuçlara varılabilir.

1. Verilen herhangi bir gen dizisi ve k-harfli ile basit yapıda bir De Bruijn grafı oluşturulabilir.
2. Daha büyük k-harfliler için tek bir gen dizisini De Bruijn graflarına dönüştürmek daha kolaydır.
3. Çok yüksek boyutlu k-harfliler için genellikle, grafı saklamak ve işlem yapmak için daha fazla bilgisayar belleğine gereksinim duyulmaktadır. Ayrıca k değeri için üst sınırı ayarlanabilen ne kadar belleğe sahip bir bilgisayara ihtiyaç duyulması bir tartışma konusu olup donanım kısıtlamaları göz önünde bulundurulmalıdır.

4.4. De Bruijn Grafları İleri Konular

4.3'teki örneklerde gösterildiği gibi herhangi bir gen dizisi De Bruijn graflarına kolaylıkla dönüştürülebilir.



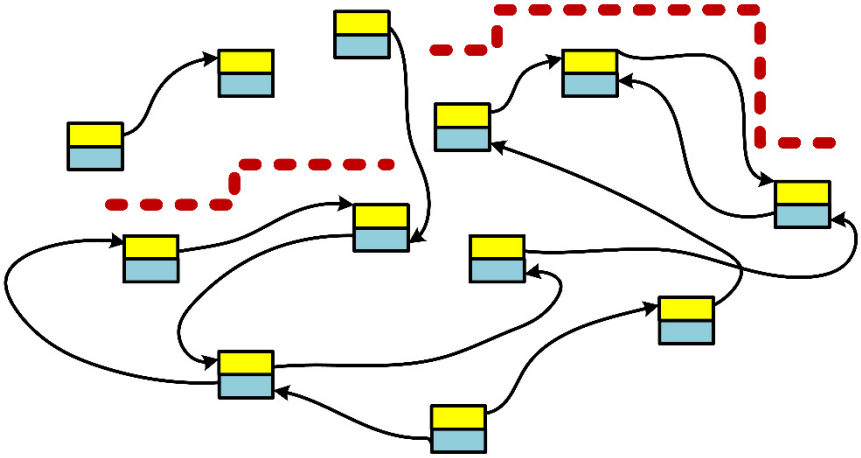
Şekil 4.12. De Bruijn Grafının Kaba Görünümü

Bu metot "niçin genom veya transkriptom yerleşiminde kullanılan kısa-okumalarda bu kadar popülerdir" ve ayrıca Velvet veya SOAPdenovo gibi geleneksel kısa-okuma genom yerleştiricilerin neden doğrudan transkriptom genlerine uygulanamadığı açıklanabilir.

4.3.'de gösterildiği gibi herhangi bir genom, De Bruijn grafına dönüştürülebilir. Bu graf genomun ne kadar büyük olduğuna bağlı olarak büyük veya küçük olabilir fakat temel özellikleri tüm genom için benzerdir.

Şekil 4.12'deki De Bruijn grafı için eşit kromozom parçaları alınabilir ve bu parçaları popüler kısa okuma teknolojilerinden biri kullanılarak küçük sekanslar elde edilebilir. Her bir küçük sekans, De Bruijn grafına dönüştürülürse; genomun bu De Bruijn grafı parçası ile eşleşecektir. Durumu basitleştirmek için, okumalarda dizileme(sekanslama) hatası olmadığı varsayılır.

Şekil 4.13'de, iki kısa okuma kırmızı ve mor ile gösterilmiştir. Ayrıca bu okumalardan De Bruijn grafının eşleşen bölgeleri görülmektedir.



Şekil 4.13. De Bruijn Grafında Kısa Okumaların Kırmızı ve Mor ile Gösterimi

Kavramsal olarak bunun hakkında düşünülürse, tüm genomu temsil eden eşit kısa-okumalardan oluşan milyarlarca dizi (sekansımız) varsa devasa bir De Bruijn grafı elde edilecektir. Bu De Bruijn grafı tüm genomun De Bruijn grafının aynısı gibi gözükecektir. Bundan dolayı konu De Bruijn grafından genom dizisini (sekansını) elde etmek olacaktır.

Bu De Bruijn grafını oluştururken her düğüm ile eşleşen kaç tane kısa-okumanın izi sürülebilir. Bu genomda hiçbir tekrar yoksa dizileme(sekanslama), eşit 50 okumada mükemmel olacaktır ve bu grafın her düğümü tam olarak 50 kısa-okuma tarafından ziyaret edilebilecektir. Bu durumda bu graftan genom sekansının yeniden inşası önemsiz bir hal alacaktır.

Gerçek dünyada tüm genomlar tekrarlı bölgelere sahiptir ve tüm kısa okuma kütüphanelerinde hatalar vardır. Bunları çözmek, çeşitli kısa-okuma yerleştirme(montaj) algoritmalarının ele aldığı en temel problemlerden biridir.

Tekrarlar ile ilgili, ilginç bir gözlem ile karşı karşıya kalınmaktadır. De Bruijn grafının neredeyse tüm düğümlerini 50 kısa-okuma tarafından ziyaret edilirse; bu 200 kısa-okuma tarafından ziyaret edilen düğümlerin küçük bir alt kümesi olup, genomun tekrarlı bölgelerinden oluşan düğümlerin alt kümesine yakın bir diziyi(sekansı) teşkil ettiği iddia edilebilir ve bu genomda 4 kez gösterilmiştir denilebilir. Bu yüzden tekrarlardan kaçınmak yerine geleneksel gen yerleştiricilerde yapıldığı gibi De Bruijn grafları gen yerleşimi sırasında tekrarlı bölgelerin tekrar frekanslarını tahmin etmemizi sağlamaktadır.

Bahsedilen gözlemler genom yerleşimine karşı transkriptom yerleşimi konusunu gündeme getirmektedir. Bir genomun kısa-okuma kütüphanesinde, genomun tüm bölgeleri eşit şekilde temsil edilmektedir. Buna göre, k-harfli sayısı yüksek olursa, genomun tekrarlı bölgeleri olasıdır. Öte taraftan k-harfli sayısı düşük olursa, dizileme hataları olasıdır. Kolaylık ve

basitlik için, çoğu kısa-okuma genom yerleştirme programı ortalama k-harfli sayısını ayarlayabilir (dizileme derinliğine eşit, örnekte 50 alınmış olan) ve çok büyük veya çok küçük k-harfli sayısı reddedilmektedir.

Bir transkriptomun De Bruijn grafı hakkında düşünülürse, çeşitli genlerin ekspresyon seviyelerinden dolayı doğal olarak graf düzgün çıkmayacaktır. Çok büyük k-harfli sayısı ile yüksek derecede ifade edilen genler gelebilir ve küçük k-harfli sayısı ile düşük sayıda genler gelebilir. Bu aşırılıklar transkriptom verisini daha ilginç kılan öğelerdir. Bir genom yerleştirme programı iki ucunda aşırı bol k-harfli sayısını kaldırırsa, gen ifadesindeki değişimi düzgün şekilde gösteremeyecektir. Bundan dolayı genom yerleştiricilerin transkriptom yerleşimi için modifiye edilmesi gerekmektedir.

Tüm bu durumlar göz önünde alınırsa, De Bruijn graflarının tüm biyoinformatikçilerin cephaneliğini içeren faydalı araçlar olduğu izlenimi oluşmaktadır. Ancak açıkça belirtilmeyen, "niçin bu graflar tüm popüler kısa-okuma yerleştiricileri tarafından uygulamada en temel araç oldu" açıklanmaya çalışılırsa bunun için Şekil 4.14'teki örnek ele alınmıştır. Şekil 4.14 bir genomun De Bruijn grafını göstermektedir. Bu genom ve De Bruijn grafı çok az kısa-okuma ile hizalanmıştır. Şekil 4.14'te hiç dizileme hatası olmadığı varsayımı ile başlanılsın. Eğer böyle ise sonuç önceden bu genomu bilmeyle veya kısa-okuma verisinden genomun de novo (yeniden) yerleşimini yapmaya çalışmakla değişmeyecektir.

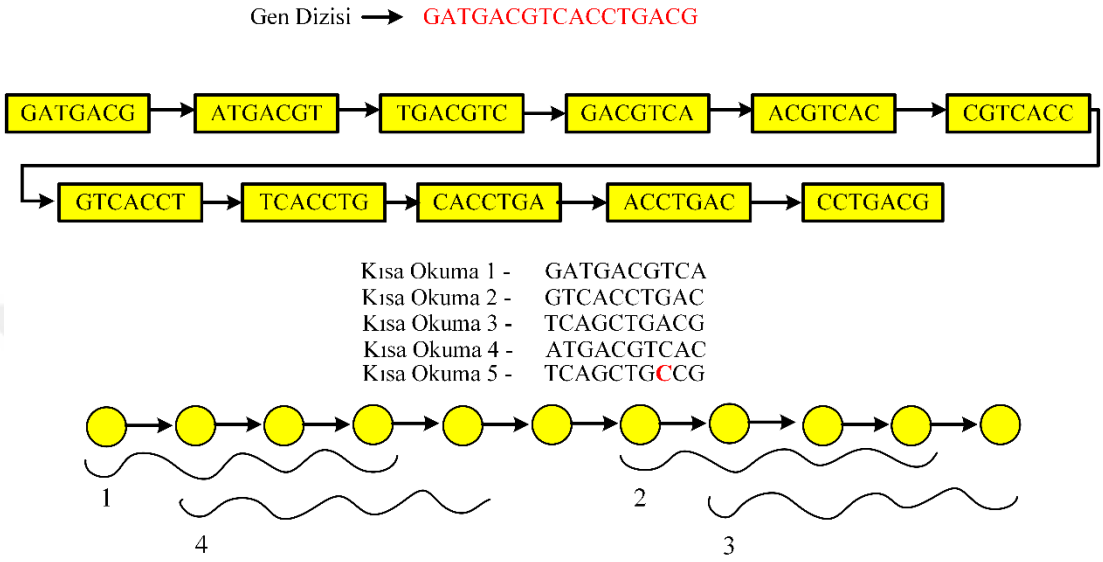
Tüm okumalar mükemmel ise, gerçek genom ile De Bruijn grafı tam eşleşecek ve Şekil 4.14'teki grafa hiç düğüm ve kenar ekleme ihtiyacı olmayacaktır. Böylece bu genom dizisinin derinliğinin 10X veya 1000X olup olmadığı önemsenmeden De Bruijn grafının boyutu esas genomun veri hacmi ile değil, boyutu ile sınırlanmış olacaktır.

Bu gözlem sonuçları hakkında düşünüldüğünde araştırmacılar sıklıkla Hiseq verisinden bir genomun yerleşimi için ne kadar bilgisayar belleğine ihtiyaç duyulacağı sorusunu sormaktadır. Cevap De Bruijn grafına dayanan bir algoritma ile yerleştirilmiş (monte edilmiş) genomun boyutuna bağlı fakat dizi(sekans) veri hacmine bağlı değildir olacaktır. 3 şeritli bir Hiseq verisinden maya genomunun yerleşimi, bir omurgalı genomunun yerleşiminden çok daha az bilgisayar belleğine ihtiyaç duymaktadır. Ancak, mükemmel bir dünyada yaşanılmamakta ve tüm dizileme (sekanslama) kütüphanelerinin bazı açılardan hatalara sahip olduğu bilinmektedir. 4.4.1.'de dizileme hatalarıyla De Bruijn grafının yapısının nasıl değişeceğini ve artan bilgisayar belleği gereksinimini ele alınmıştır. Yanlış okumalarda bile, De Bruijn graflarını kullanan yeni nesil yerleştirme (montaj) programları pek çok okumalar üzerinde çalışarak daha az bellek gereksinimine ihtiyaç duymaktadır. Burada verilen bilgisayar belleği anahtar rol oynamaktadır.

4.4.1. Dizileme Hatalarının De Bruijn Grafları Üzerindeki Etkisi

4.3. ve 4.4. kısımlarında kavramsal olarak resmi basit tutmak için dizileme hataları göz ardı edilmiştir. Ancak, dizi analizi gerçekte dizileme hatalarıyla doludur. Bu nedenle, bu hataların De Bruijn graflarını nasıl değiştirebileceği bu kısımda gösterilmeye çalışılmıştır.

Şekil 4.14'te orijinal genom dizisinden elde edilen 7-harfli De Bruijn grafıyla başlanılsın.

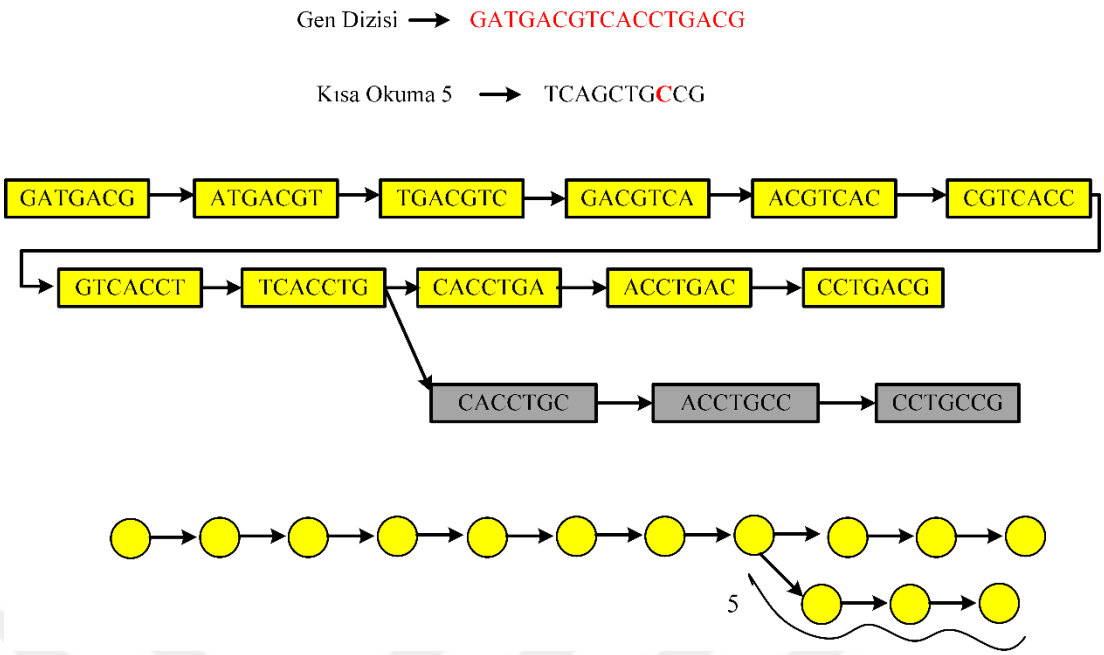


Şekil 4.14. De Bruijn Grafı ve 5 tane Kısa-Okuma

Şekil 4.14'teki 11 düğümlü kenarları bağlı De Bruijn grafı verilmiş olsun. Varsayım olarak bu çok kısa genom 10 harfli kısa okumalarla sıralanmış olsun. Şekil 4.14'te görülmektedir ki 5 adet kısa okuma vardır ve bunlardan ilk 4'ü mükemmel fakat 5. Kısa-okuma tek bir nükleotid hatasına sahiptir. (kırmızı ile işaretlenmiş)

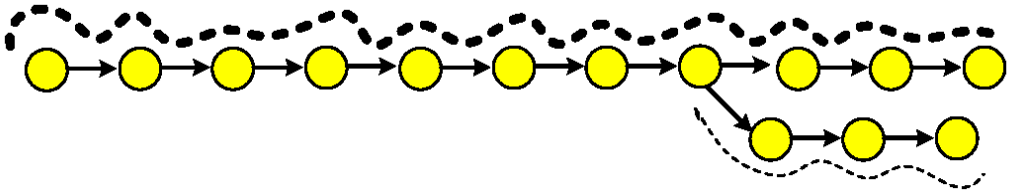
Kolaylık sağlamak için Şekil 4.14'te altta daireler ve oklar kullanılarak De Bruijn grafı yeniden çizilmiştir. Bu grafın 11 düğümü ve 11 kenarı sayılabilmektedir. Düğümler içindeki 7-harf gözükmemektedir. 4 kısa-okumayla genom ve De Bruijn grafı Şekil 4.14'te görüldüğü gibi tam eşleşmektedir.

5. kısa-okuma bir düğüm hariç De Bruijn grafı ile eşleşmemektedir. Bu nedenle hatalı kısa-okumayı tamamen birleştirmek için ek düğümlerin oluşturulmasına ihtiyaç duyulur. Bu ek düğümler Şekil 4.15'de gri ile gösterilmiştir.



Şekil 4.15. De Bruijn Grafında Hatalı 5. Kısa Okuma

Şekil 4.16'da 5. okumanın hizalanışını gösteren basitleştirilmiş De Bruijn grafı gözükmektedir.



Şekil 4.16. 5. Kısa Okumanın Basitleştirilmiş De Bruijn Grafı

Şekil 4.15'deki örnekte genom dizisi ve De Bruijn grafı önceden bilinmektedir. Bu yüzden hatalı okumayı temsil eden dallanmayı söylemek kolaydır. Gen yerleştirme probleminde, De Bruijn grafı kısa-okumalardan ilk kez inşa edilir ve sonrasında De Bruijn grafından genom dizisi elde edilir. Burada "yanlış dallanmayı bulmak ne kadar mümkündür" şeklinde bir soru akla gelmektedir.

Bazı hatalarla genomdan dizilenmiş binlerce kısa-okumadan De Bruijn grafının inşası kavramsal olarak düşünülürse, okumaların büyük çoğunluğu mükemmel şekilde eşleşecektir. Bunlar genomun De Bruijn grafının eşleşmiş yolları ile pek çok örtüşme oluşturacaktır. Çok az hatalı okuma, yoğun olmayan çapraz geçişli fazladan dallanmalar oluşturacaktır. Bunun sonucu olarak, pek çok okumadan oluşturulmuş De Bruijn grafı Şekil 4.16'daki gibi gözükecektir.

De Bruijn grafındaki yoğun bölgeler örtüşen çoğu okumayı temsil etmekte, olası hatalardan oluşmuş çok az okuma da yoğun olmayan bölgelerde gözükmektedir. Bundan dolayı

yerleştirme algoritmaları çok hafif şekilde dallanmış bölgeleri (yolları) budar ve sonrasında yoğun şekilde dallanmış bölgelerden genomu yeniden oluşturur.

Şekil 4.16'daki örnekte görülen hatadan oluşmuş dallanma orijinal De Bruijn grafından çıkarılır. Başka bir ihtimal de vardır. Bazen bu okumalar De Bruijn grafının k-harfli boyutuna göre yeterli uzunluktadır ve hata ortada olabilir. Böyle durumlarda, hatalı okuma genomun De Bruijn grafının her iki ucu ile hizalanabilir fakat ortadaki hata farklıdır ve bir döngü oluşturur. İyi kısa-okuma yerleştirme algoritmaları hafif şekilde olan dallanmaları ve döngüleri budamaktadır.

Okuma hatasının olmadığı De Bruijn grafının boyutu ele alınan genomun boyutuyla sınırlı olup ve dizileme derinliği değişmezdir. Ama hatalar olduğunda bu durum geçerli değildir. Hatalar De Bruijn grafına dallanmalar ekleyerek grafın boyutunu büyütmektedir. Bundan dolayı, grafı yüklemek için daha fazla bilgisayar beleğine ihtiyaç duyulur. Bu ek bilgisayar belleği gereksinimi, okumaların sayısı ile neredeyse doğrusal olarak artacaktır. Çünkü okuma sayısı arttıkça doğrusal olarak hataların sayısı da artacaktır.

4.5. Yeni Nesil Dizileme Teknikleriyle Yeniden Gen Yerleşimi

Yeni nesil dizileme teknikleri;

- Geleneksel Metotlar
- Hash Aramaları
- De Bruijn Graf Temelli Yaklaşımlar

adında 3 başlık altında incelenebilir.

4.5.1. Geleneksel Metotlar

Gen yerleşiminde geleneksel bir yaklaşımda örtüşen grafın kullanımı formülize edilebilir (Myers, 1995). Bu yapıda her bir okuma, iki yönlü bir kenar tarafından bağlanmış açık bir örtüşmeyi temsil eden 2 okumanın olduğu ayrı bir düğüm olarak temsil edilmektedir. Örtüşme-uzlaşma-düzen yaklaşımı özellikle uzun okumalarda hem sezgiseldir hem de sağlamdır. Örtüşmenin hakiki olup olmadığını belirlemek çeşitli deneylerle olasıdır. Çünkü her DNA baz çifti hizalaması uzun diziler üzerinden yapılmıştır (phrap.org web adresi).

Ancak bu yaklaşım özellikle yeni nesil mikro-okuma dizilemede çeşitli dezavantajları da beraberinde getirmektedir. Gerçekte örtüşen DNA baz çiftlerinin hesaplanması deneyler (Pearson, Lipman, 1988) veya filtreler (Rasmussen, 2005) tarafından optimize edilebilmesine rağmen doğal olarak 2. dereceden bir karmaşıklığa sahiptir. Sadece EDENA (Hernandez, 2008) isimli mikro-okuma yerleştirici bu yaklaşım kullanılarak geliştirilmiştir.

Bir diğer okuma üzerinden tam uzunlukta hizalanan okumayı da içeren okumalar graftan kaldırılmak zorundadır (Myers, 2005). Bu, örtüşen bir graf ile doğrudan gerçekleştirilemeyen karışık uzunlukta dizileme gerektirir. Çeşitli programlar örtüşen graflar üzerinden karışık uzunlukta gen dizisi yerleşimlerini gerçekleştirmeye çalışmaktadır (Chevreux, 2005; Reinhardt, 2009) fakat bu programlar bu okumaların kullanımında bir asimetri ile başlamak zorundadır. Ne kısa okumalar uzun okuma örtüşmeleri üzerinden basitçe haritalanır ne de bu programlarla ayrı olarak gen dizisi yerleşimi yapılabilir.

4.5.2. Hash Aramaları

Örtüşen grafların oluşturulması için gereken 2. dereceden hesaplama miktarını azaltmak için, bazı geliştiriciler bilgisayar veri yapılarının iyi çalışan bir sınıfı prefix ağaçlarını kullanmayı tercih etmiştir.

Ana hedef greedy araması yoluyla bir örtüşme kümesini (kontig) yinelemeli olarak genişletmektir. Program başlangıçta tek bir biçimde rasgele bir okuma seçer. Her bir adımda, mevcut örtüşme kümesine doğru olarak örtüşen okumaları bunun veritabanı aracılığıyla arar. Buradan okumalar kümesi azaltılır ve sonrasında program uygun genişlemeyi belirlemek için bir tahmin (höristik) kullanır ve mevcut örtüşme kümesine ekler. Bu süreç tüm olası okumalar bitene kadar birkaç kez çalıştıktan sonra tüm örtüşmeler PHRAP gibi geleneksel gen yerleştirme yazılımı kullanılarak birleştirilebilir. Çizelge 4.1'de yayınlanmış metotların bir özeti verilmiştir.

Çizelge 4.2. Arama Tabanlı Gen Yerleştirme Programları

Program	**Kaynak**	**Tahmin Kriteri**
SSAKE	R.L. Warren (Warren, 2007)	Kapsama, örtüşme uzunluğu
VCAKE	W.R. Jeck (Jeck, 2007)	Kapsama, örtüşme uzunluğu, çoğunluk sayısı
SHARCGS	J.C. Dohm (Dohm, 2007)	Örtüşme uzunluğu, uzlaşma
QSRA	D.W. Bryant Jr. (Bryant Jr., 2009)	Kapsama, örtüşme uzunluğu, kalite puanları, çoğunluk sayısı

4.5.3. De Bruijn Graf Temelli Yaklaşımlar

1995'te Idury ve Waterman (Idury, Waterman, 1995) gen yerleşimini göstermek için bir dizi grafının kullanımı tanıttı. Onlar, melezleşme tarafından dizilemesi verilen bir genomda bir nükleotid dizisinde tüm k'lı nükleotid sözcüklerinin (k-harfli olarak da bilinen) tespit edildiği alternatif dizileme teknikleri için bir gen yerleştirme algoritması sunmuştur. Bunların çözüm metodu tespit edilen her sözcük için bir düğüm oluşturmayı ve sonra örtüşen k-

harflilere karşılık bu düğümleri bağlamayı içermekteydi. Bunlar sonra dallanan bağlantıların yok olmasından dolayı k-harfli kesin üretilen örtüşme kümelerinin örtüşen zincirlerini raporlayabilmiştir.

Pevzner 2001'de bu fikri genişletti. İlk olarak De Bruijn grafı olarak adlandırılan çok az farklı formüle edilmiş dizi grafını önerdiler. Aynı grupta olan algoritmaları geliştirmek için arka arkaya yayınlar yapıldı ve De Bruijn grafındaki hatalar düzeltildi (Pevzner, Tang, 2001). Burada iki yönlü (paired-end) okumalar (Pevzner, Tang, 2001) ve kısa-okumalar (Chaisson, Pevzner, 2008) kullanıldı. Klasik Sanger metodunda ve Hamilton çevrimi metotlarında sonuç genomu elde etmek oldukça zordur ve NP-tam bir problemdir. De Bruijn graf temelli NGS teknolojileri Euler yolu ile sonuç genomu elde etmeyi daha kolay hale getirmiştir.

4.6. Velvet Assembler

Velvet genomun yeniden yerleştirilmesi ve kısa-okuma dizileme hizalanmalarını çözmek için tasarlanmış bir algoritma paketidir. Bu algoritma hataların kaldırılması ve tekrarlı bölgelerin basitleştirilmesi yoluyla genom dizisini yerleştirmek için De Bruijn graflarının manipülasyonuyla gerçekleştirildi (Zerbino, 2008). 2008 yılında Daniel Zerbino[3] ve Ewan Birney[4] tarafından geliştirilmiştir.

Yeni nesil dizileyicilerin (NGS) gelişimi çok kısa okunan DNA dizileri üzerinde maliyetlerin düşmesine yol açtı. Hizalama için bir metot olarak De Bruijn graflarının manipülasyonu daha gerçekçi bir hal aldı fakat sonraki gelişmeler hatalar ve tekrarlar ile ilgili konulara odaklanmayı gerektirmiştir (Miller, 2010). Bu da İngiltere'deki Avrupa Biyoinformatik Enstitüsünden Daniel Zerbino ve Ewan Birney tarafından Velvet'in geliştirilmesine öncülük etmiştir (Zerbino, 2008).

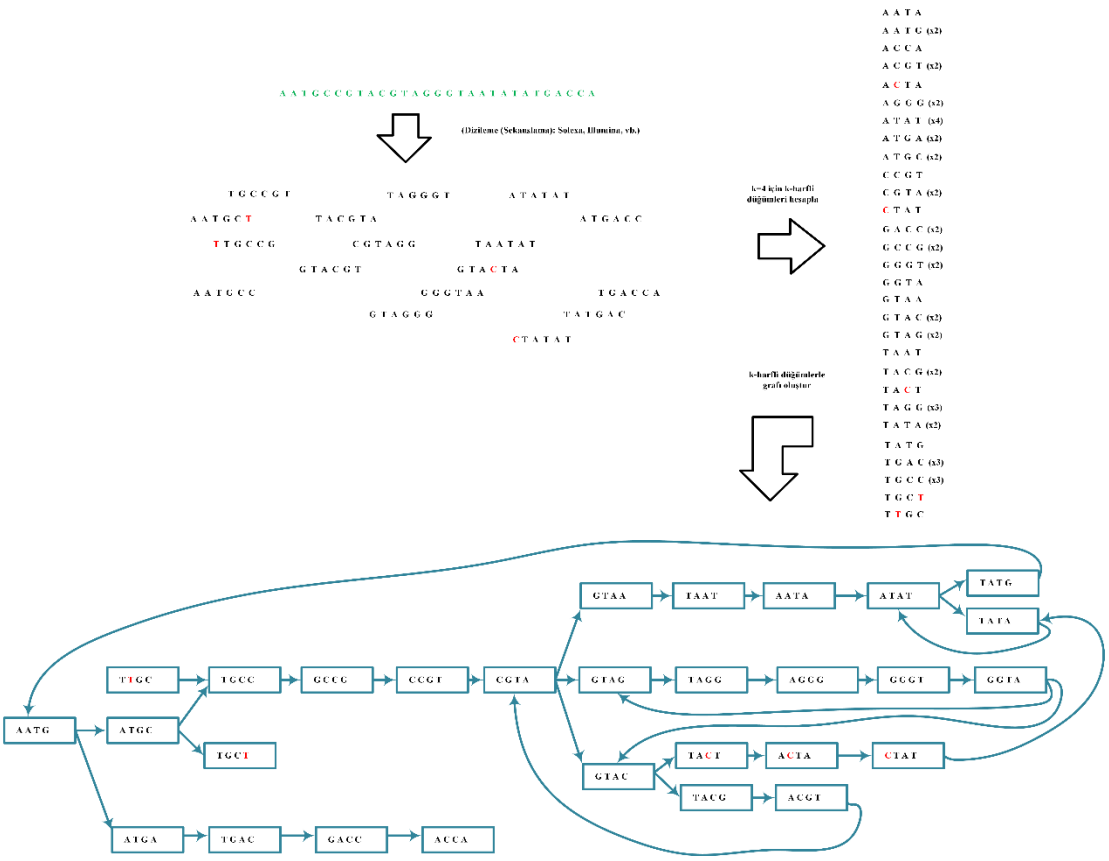
Velvet basitleştirme ve sıkıştırma yoluyla De Bruijn graflarının verimli şekilde manipülasyonuyla grafta bilgi kaybı olmaksızın yolları kesişmeyen tek düğümleri yakınsayarak çalışmaktadır. Algoritma birlikte birleşmiş dizileri bir hata düzeltme algoritması kullanarak ortadan kaldırıp, tekrarları çözebilir. Tekrarlar, yerel örtüşen ayrı yollarda tekrarları çözücüyle sonradan diziden kaldırılır. Kısa okumaların kombinasyonu ve okuma çiftleri Velvet'in küçük tekrarları çözmesine ve makul uzunlukta kontiglerin üretilmesine izin vermektedir. Velvet'in bu uygulaması N50 uzunluğunda 50 kb 'lık prokaryotik DNA baz çifti verisi ile kontigleri üretebilir.

[3] Matematik ve Bilgisayar Mühendisi, EMBL-EBI'da Genom Analizi Takım Lideri, Velvet Assembler Geliştiricisi
[4] 1972 doğumlu, Biyokimyacı, EMBL-EBI'da yönetici, Velvet Assembler Geliştiricisi

4.6.1. Velvet Algoritması

Velvet zaten bahsedildiği gibi kısa-okumaları yerleştimek için De Bruijn graflarını kullanır. Daha ayrıntılı olarak Velvet graftaki tek bir düğümde okumalardan elde edilen her farklı k-harfliyi temsil eder. Eğer bu k-harfliden k-1'i örtüşürse iki düğüm birbirine bağlıdır. Başka bir ifadeyle k-harfliden son k-1 karakteri varsa A düğümünden B düğümüne bir bağlantı vardır ve B ile temsil edilen düğümün k-harfli ilk k-1 karakteri A ile temsil edilen düğümde temsil edilir. Şekil 4.17'de Velvet ile oluşturulmuş bir De Bruijn grafı örneği gösterilmiştir.

Aynı işlem karşı ipliklerin okumaları arasında örtüşmeler dikkate alınarak tüm k-harflilerin tamamlayıcısı ile eş zamanlı yapıldı. Basitleştirme ve hata silmeyi içeren birkaç iyileştirme bu graf üzerinde yapılabilir.



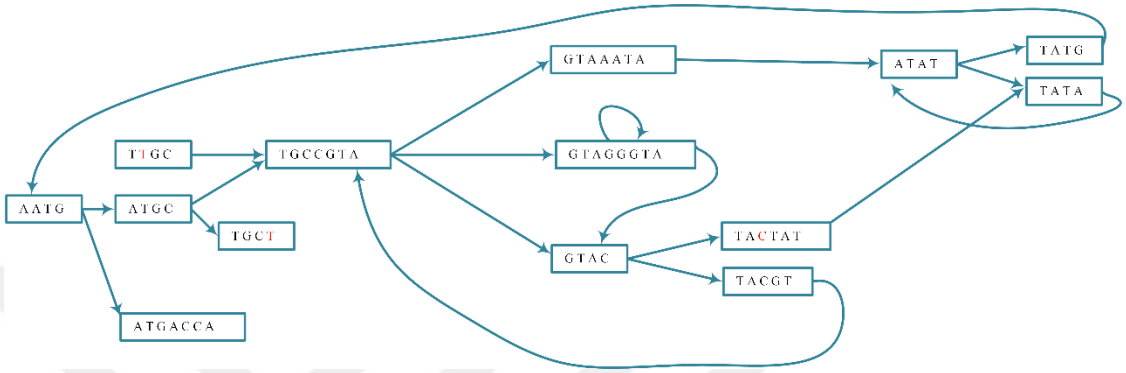
Şekil 4.17. Velvet ile De Bruijn Grafının İnşası

Bu işlemler şunlardır;

- Basitleştirme (Simplication)
- Hata Kaldırma (Error Remove)
  - Kuyruklar (Tips)
  - Kabarcıklar (Bubbles)
  - Hatalı Bağlantılar (Erroneous Connections)

4.6.2. Basitleştirme

Bellek maliyetini düşürmenin kolay yolu grafımızda, oluşturulmuş yolu etkilemeyen düğümleri birleştirmektir. Örneğin A düğümünden B düğümüne sadece çıkış yönünde kenarları olan düğümler varsa bu düğümler birleştirilir ve bu düğüme sadece bir giriş yapılır. (A düğümünden birleştirilmiş düğüme). Bu düğüm, bu aradaki düğümlerin tüm bilgisini temsil eder. Şekil 4.18'de başlangıç De Bruijn grafının basitleştirilmesi işlemi gösterilmiştir.



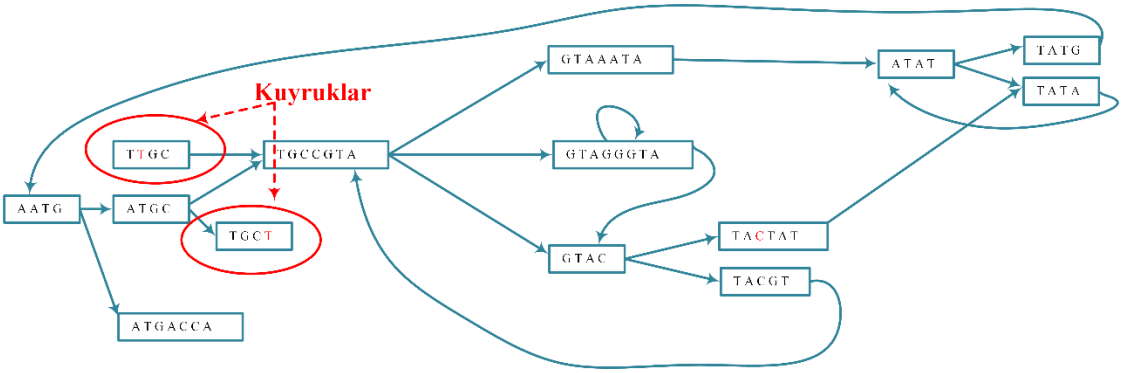
Şekil 4.18. Basitleştirme Sonrası De Bruijn Grafı

4.6.3. Hata Kaldırma

Grafımızda, dizileme işlemi yoluyla veya bazı hataları içeren biyolojik örneklerin (örneğin polimorfizm) basitleştirilmesinden kaynaklanan hatalar bulunmaktadır. Velvet algoritması bununla ilgili 3 çeşit hatayı tanımaktadır. Bunlar; kuyruklar, kabarcıklar ve hatalı bağlantılardır.

4.6.3.1. Kuyruklar

Uçlarından birinin bağlantısının olmadığı kuyruk olarak değerlendirilen bir düğüm graftan silinebilir. Bu tür düğümlerde saklanan bilgi 2k'dan daha kısa olmalıdır. Bu tür düğümlere gelen kenarlar düşük yoğunluktadır (bu kenarların sayısı grafın inşası esnasında bulundu) ve sonuç olarak diğer alternatif yollarla mukayese edilemez. Bu hatalar kaldırıldıktan sonra graf tekrar basitleştirme işlemine tabi tutulur.



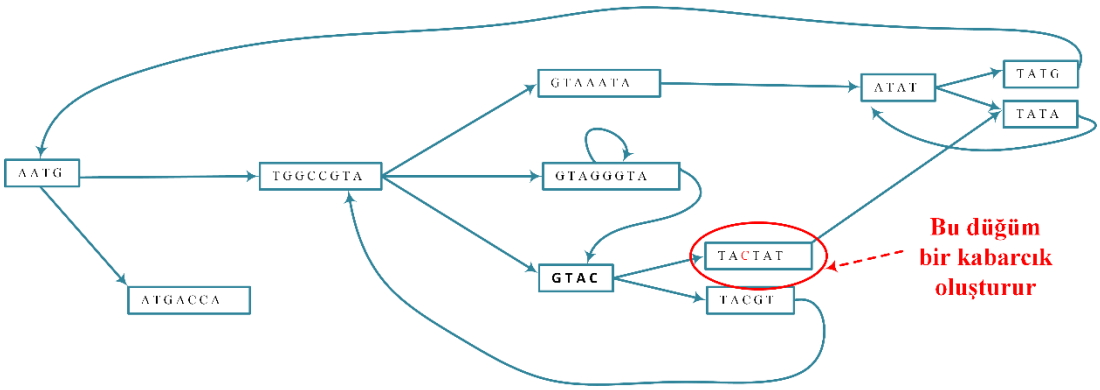
Şekil 4.19. Kuyruk Türleri

4.6.3.2. Kabarcıklar

İki farklı yolla başlayıp aynı düğümde sonlandığında kabarcıklar oluşur. Normalde kabarcıklara hatalar veya biyolojik varyasyonlar sebep olur. Bu hatalar Tour Bus algoritması kullanılarak kaldırılır. Tour Bus algoritması Dijkstra algoritmasına benzer, hangi düğümün silineceğini belirlemede ve en iyi yolu tespit etmede BFS (geniş öncelikli arama) metodunu kullanır. Şekil 4.20'de bununla ilgili basit bir örnek gösterilmiştir.

Şekil 4.20. Kabarcığın Kaldırılması

Kabarcıklar Şekil 4.17'de ve Şekil 4.18'de gösterilen örnekler üzerinden Şekil 4.21'de de gösterilmiştir.



Şekil 4.21. Kabarcık Tespiti

4.6.3.3. Hatalı Bağlantılar

Hatalı bağlantılar grafta, doğru yollar üretemeyen bağlantılar veya hiçbir şekilde tanınmayan yapılardır. Velvet algoritması, Tour Bus algoritması işlemini tamamladıktan sonra bu hataları, kullanıcı tarafından tanımlanması gereken basit bir kapsama, kesme uygulayarak silmektedir.

5. DNA DİZİLERİYLE DE BRUIJN GRAF UYGULAMALARI

5.1. Giriş

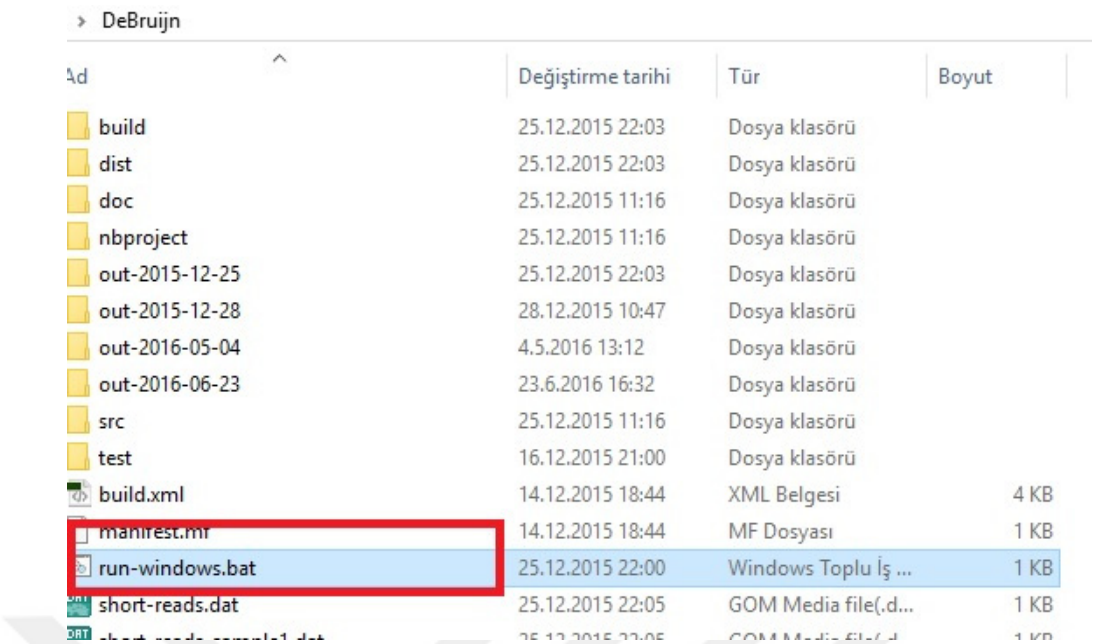
Reel dünyada DNA dizileri tam doğruluğa sahip değildir. Elde edilen DNA, RNA verileri biyoinfornatik çevrelerininde kabul ettiği gibi %100 doğruluğa sahip olması beklenemez. DNA verisinin doğruluk oranını artırmak için bir organizmadan alınan örnek Referans Genom'un farklı bölgelerinden(kontig) bir veya fazla sayıda kısa-okumalar(short-reads) yapılır. Referans genomdan en yüksek doğruluğa sahip gerçek genom elde edilebilmesi için yeteri sayıda kısa-okumalar yapılmalıdır.

Bir organizmanın DNA verisinin doğruluğunu tespit etmenin çeşitli yöntemleri vardır. Bunun için DNA hizalama (kısa-okumaların örtüştürülmesi) ve string graflar kullanılmaktadır. Gerçek DNA verileri çok büyük miktarda olduğundan bu yöntemler kullanılırken çok büyük miktarda bellek kullanılmakta ve bu çok fazla çalışma zamanı almaktadır. Bundan dolayı bellek kullanımını ve çalışma zamanını azaltmak için yeni yöntemlere gereksinim duyulmuştur. Bunlardan biri de De Bruijn grafları kullanılmasıdır. Bu şekilde kısa-okumaların De Bruijn grafı elde edilerek De Bruijn Grafı üzerinde kuyruk(tips), kabarcık(bubbles) gibi yapıların yardımı ile bu kısa okumalardaki hatalı bazları (A,T,G,C) tespit edip daha güvenilir bir genom (consensus genome) elde edilebilmektedir. Yazılan yazılım ile graf teorisindeki Euler yolu tekniği baz alınarak De Bruijn grafı inşa edilmiştir. Ayrıca burada kuyrukları tespit edip silmek için Round Table algoritması kullanılmış ve düğümleri azaltmak için basitleştirme işlemi yapılmıştır. Yapılan örneklerde bu yöntemin diğer yöntemlere nazaran daha verimli olduğu anlaşılmıştır. Yazılım aracı ile bu durumu göstermek için 4 uygulama örneği sunulmuştur.

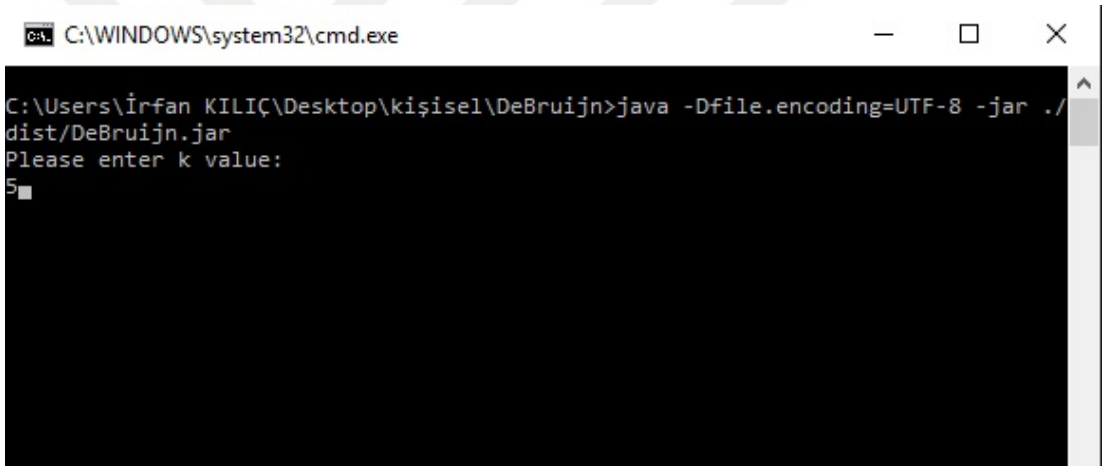
5.2. Yazılım Aracının Geliştirilmesi

Bu yazılım kodlanırken NetBeans IDE 8.1 platformunda Java programlama dili kullanılmıştır. Kodda oluşturulan ".py" uzantılı Python dosyalarında Graphviz kütüphanesi fonksiyonları kullanılarak grafların görsel çizimleri yapılmıştır.

Yazılım çalıştırılmadan önce bilgisayarınızda Java'nın yüklü olması gerekmektedir. Yüklü değilse https://java.com web adresinden yüklenebilir. Yazılımımızı "DeBruijn" adında klasörden açalım. Şekil 5.1'de işaret edilen "run-windows.bat" dosyası çalıştırılırsa Şekil 5.2'deki konsol ekranı çıkacaktır.

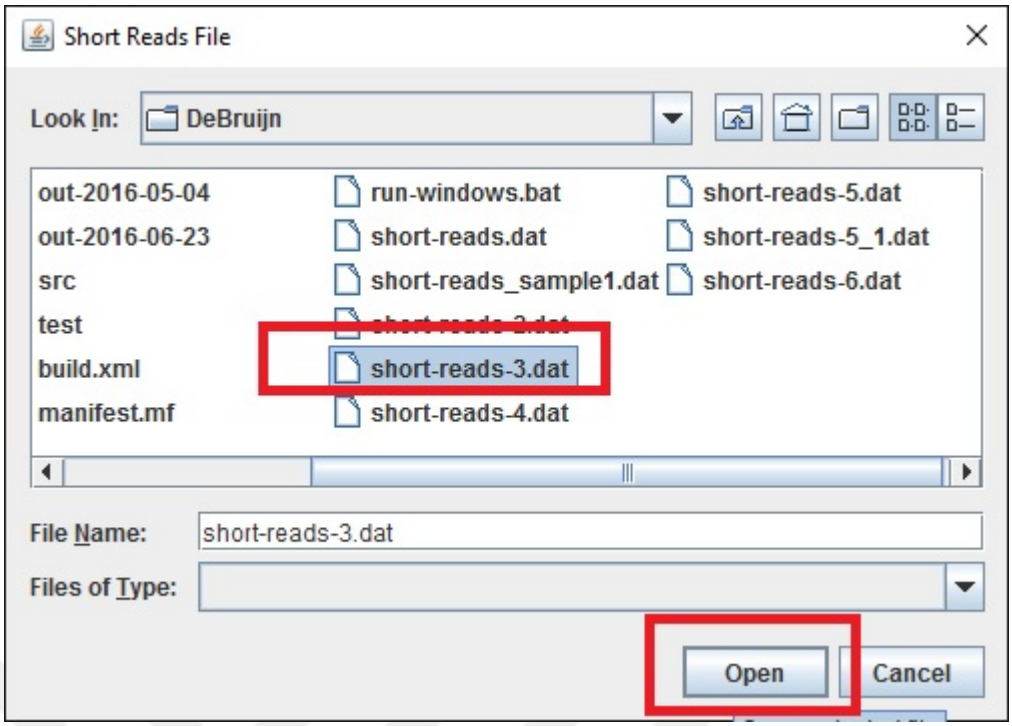


Şekil 5.1. DeBruijn Klasörü

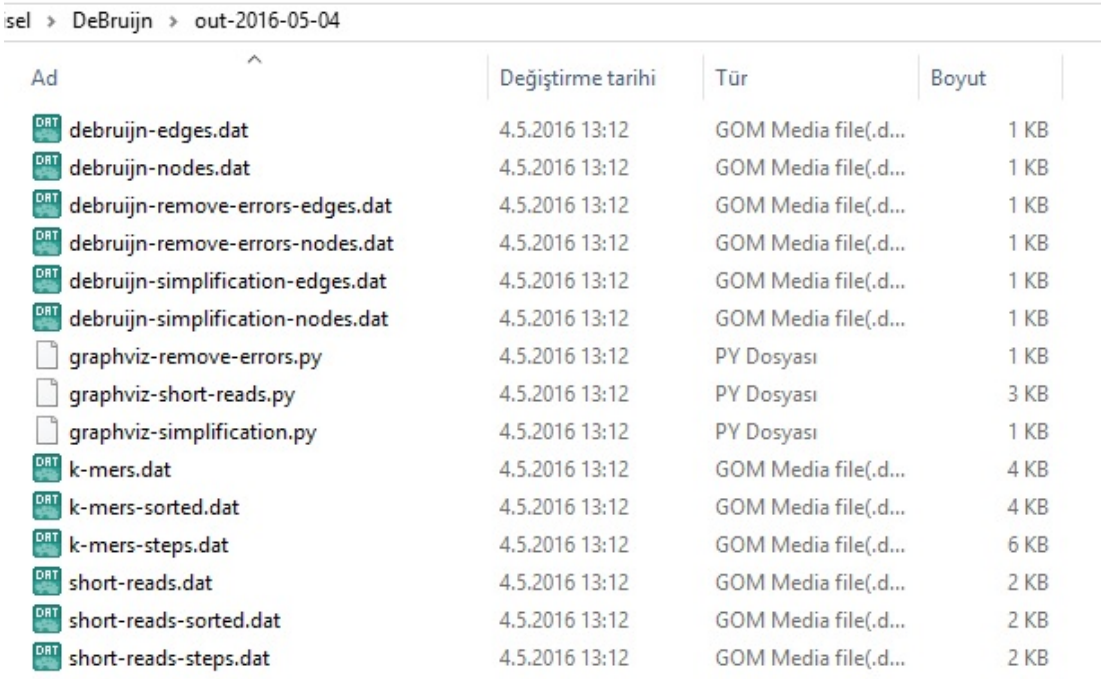


Şekil 5.2. Yazılım Konsol Ekranı

Şekil 5.2'deki konsol ekranında k-harfli sayısını girmemiz istenecektir. Belirtilen k değeri k-harfli kenar sayısını ve k-1-harfli düğüm sayısını ifade etmektedir. k=5 değeri girilip enter tuşuna basılır. Şekil 5.3'de De Bruijn grafı çizilecek DNA kısa-okuma dosyası seçilir. DNA kısa-okuma dosyası seçildikten sonra yazılım çalışacak ve konsolda Çizelge 5.1'deki loğu oluşturacaktır. Yazılım "DeBruijn" klasöründe "out-2016--05-23" formatında yazılımın çalıştırıldığı tarihi belirten bir klasör oluşturacak ve bu klasörde ".dat" uzantılı De Bruijn grafı dosyalarını ve ".py" uzantılı Python dosyalarını oluşturacaktır. Şekil 5.4'de bu klasörün içeriği gösterilmiştir.



Şekil 5.3. DNA Kısa-okuma Dosyası Seçimi



Şekil 5.4. Yazılım Çalıştırıldığında Oluşan Dosyalar

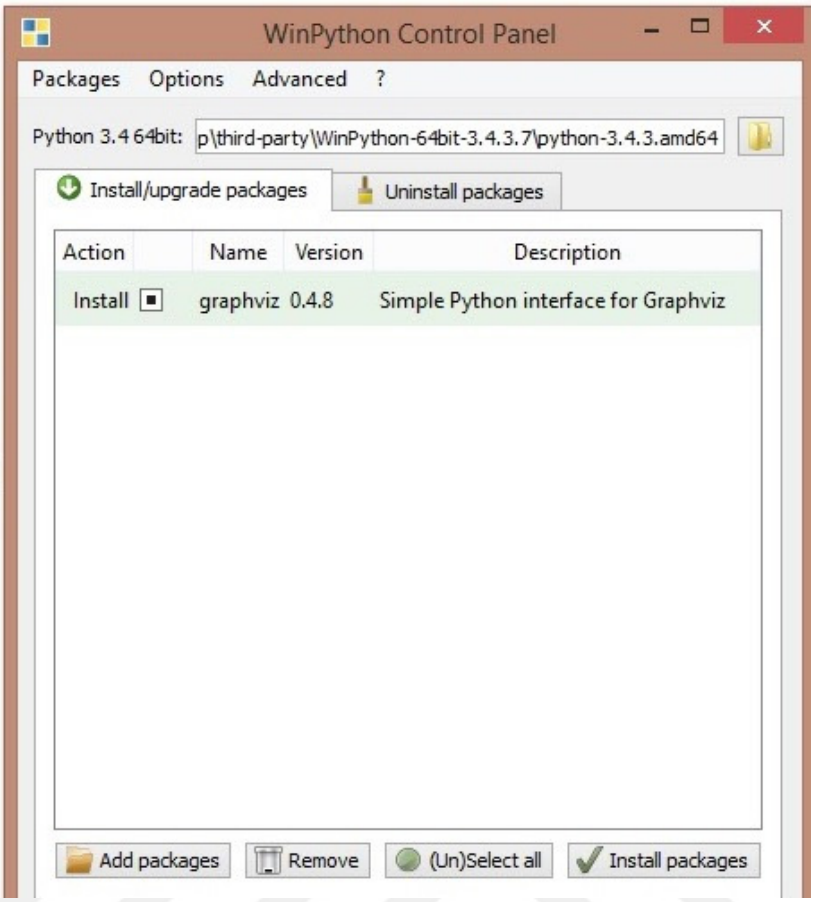
Şekil 5.4'de kısa-okuma dosyaları (short-reads….dat), k-harfli dosyaları (k-mers-….dat), DeBruijn grafı düğüm ve kenar dosyaları (debruijn-nodes.dat, debruijn-edges.dat) ve De Bruijn grafına uygulanan algoritma dosyaları (debruijn-remove….dat, debruijn-simplication…..dat) görülmektedir. Ayrıca De Bruijn grafını görsel olarak göstermek için yazılım tarafından ".py" uzantılı Python kodu dosyaları oluşturulmuştur (graphviz-short-

reads.py, graphviz-remove-errors.py, graphviz-simplication.py). De Bruijn grafının ilk halinnin görseli oluşturulmak istenirse “graphviz-short-reads.py” dosyası açılır. Şekil 5.5’de bu dosyanın içeriği görülmektedir. Şekil 5.5’deki Python kodu Graphviz kütüphanesini kullanmaktadır. Bu kodu Windows işletim sisteminde çalıştırmak için WinPython-64bit-3.4.3.7 sisteme yüklenmeli ve “WinPython Control Panel.exe” çalıştırılarak “graphviz-2.38” paketi Şekil 5.6’da gösterildiği gibi kurulmalı ve “graphviz-2.38.msi” dosyası Şekil 5.7’de gösterildiği gibi konsolda yönetici modunda kurulmalıdır. Ayrıca sistemin ortam değişkenleri PATH’ine “C:\Program Files (x86)\Graphviz2.38\bin” değişkeni eklenmelidir.

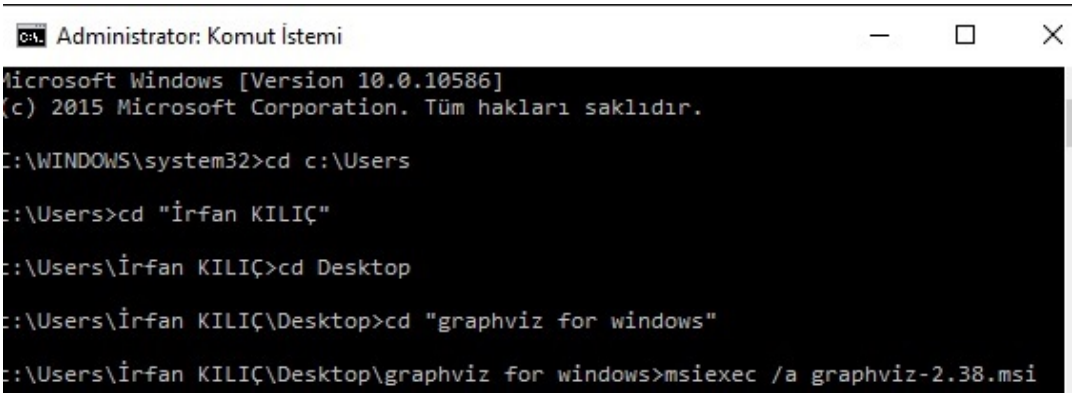
graphviz-short-reads.py

```
1  from graphviz import Digraph
2  dot = Digraph(comment='The Round Table')
3  dot  #doctest: +ELLIPSIS
4  dot.node('ACGC', 'ACGC')
5  dot.node('CGCA', 'CGCA')
6  dot.node('ACGT', 'ACGT')
7  dot.node('CGTA', 'CGTA')
8  dot.node('GTAT', 'GTAT')
9  dot.node('TATT', 'TATT')
10 dot.node('GCAT', 'GCAT')
11 dot.node('CATT', 'CATT')
12 dot.node('ATTA', 'ATTA')
13 dot.node('TTAC', 'TTAC')
14 dot.node('TTAG', 'TTAG')
15 dot.node('TAGC', 'TAGC')
16 dot.edge('ACGC', 'CGCA')
17 dot.edge('CGCA', 'GCAT')
18 dot.edge('ACGT', 'CGTA')
19 dot.edge('CGTA', 'GTAT')
20 dot.edge('GTAT', 'TATT')
21 dot.edge('TATT', 'ATTA')
22 dot.edge('GCAT', 'CATT')
23 dot.edge('CATT', 'ATTA')
24 dot.edge('ATTA', 'TTAC')
25 dot.edge('ATTA', 'TTAG')
26 dot.edge('TTAG', 'TAGC')
27 dot.render('DeBruijn/graphviz-short-reads.py.gv', view=True)
```

Şekil 5.5. “graphviz-short-reads.py” Dosyası



Şekil 5.6. WinPython'a Graphviz Paketinin Eklenmesi



Şekil 5.7. Graphviz-2.38.msi'in Kurulması

Şekil 5.5'de verilen Python kodu WinPython-64bit-3.4.3.7 klasöründe bulunan "IPython Qt Console.exe" dosyası çalıştırılarak Şekil 5.8'de açılan konsola kopyalanır ve enter tuşuna 2 defa basılarak çalıştırılır ve ".pdf" uzantılı De Bruijn grafı görseli oluşturulup Şekil 5.9'da gösterildiği gibi açılır.

Jupyter QtConsole

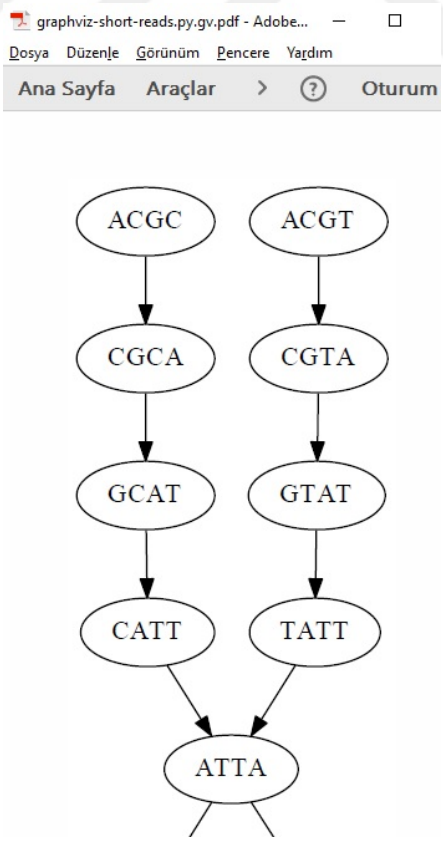
File Edit View Kernel Window Help

```
Jupyter QtConsole 4.1.1
Python 3.4.3 (v3.4.3:9b73f1c3e601, Feb 24 2015, 22:44:40) [MSC v.1600 64 bit
(AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 4.0.1 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.
%guiref   -> A brief reference about the graphical user interface.

In [1]: from graphviz import Digraph
   ...: dot = Digraph(comment='The Round Table')
   ...: dot  #doctest: +ELLIPSIS
   ...: dot.node('ACGC', 'ACGC')
   ...: dot.node('CGCA', 'CGCA')
   ...: dot.node('ACGT', 'ACGT')
   ...: dot.node('CGTA', 'CGTA')
   ...: dot.node('GTAT', 'GTAT')
   ...: dot.node('TATT', 'TATT')
   ...: dot.node('GCAT', 'GCAT')
   ...: dot.node('CATT', 'CATT')
   ...: dot.node('ATTA', 'ATTA')
   ...: dot.node('TTAC', 'TTAC')
   ...: dot.node('TTAG', 'TTAG')
   ...: dot.node('TAGC', 'TAGC')
```

Şekil 5.8. QtConsole'da Pyhton Kodunun Çalıştırılması



Şekil 5.9. De Bruijn Grafı Görseli

Yazılan Java koduna göre 4 farklı örnek verilmiştir. Çizelge 5.1'de yazılım çalışırken yazılım tarafından oluşturulan log bilgileri verilmiştir.

Çizelge 5.1. Uygulama Kodu Logları

1	run: Please enter k value: 5 k value: 5
2	Selected Short Reads File: C:\Users\İrfan KILIÇ\Desktop\DeBruijn\short-reads_sample1.dat
3	Reading Short Reads Reead File: C:\Users\İrfan KILIÇ\Desktop\DeBruijn\short-reads_sample1.dat Save File: C:\Users\İrfan KILIÇ\Desktop\DeBruijn\out-2015-12-25\short-reads.dat Sorting Short Reads Save File: C:\Users\İrfan KILIÇ\Desktop\DeBruijn\out-2015-12-25\short-reads-sorted.dat
4	Printing Short Reads Steps Save File: C:\Users\İrfan KILIÇ\Desktop\DeBruijn\out-2015-12-25\short-reads-steps.dat Creating k-mers Save File: C:\Users\İrfan KILIÇ\Desktop\DeBruijn\out-2015-12-25\k-mers.dat Sorting k-mers Save File: C:\Users\İrfan KILIÇ\Desktop\DeBruijn\out-2015-12-25\k-mers-sorted.dat Printing k-mers Steps Save File: C:\Users\İrfan KILIÇ\Desktop\DeBruijn\out-2015-12-25\k-mers-steps.dat
5	Creating Short Reads Graphs Save File: C:\User s\İrfan KILIÇ\Desktop\DeBruijn\out-2015-12-25\graphviz-short-reads.py Save File: C:\Users\İrfan KILIÇ\Desktop\DeBruijn\out-2015-12-25\debruijn-nodes.dat Save File: C:\Users\İrfan KILIÇ\Desktop\DeBruijn\out-2015-12-25\debruijn-edges.dat
6	Starting Simplification Save File: C:\Users\İrfan KILIÇ\Desktop\DeBruijn\out-2015-12-25\graphviz-simplification.py Save File: C:\Users\İrfan KILIÇ\Desktop\DeBruijn\out-2015-12-25\debruijn-simplification-nodes.dat Save File: C:\Users\İrfan KILIÇ\Desktop\DeBruijn\out-2015-12-25\debruijn-simplification-edges.dat

7	Starting Remove Error(s) Save File: C:\Users\İrfan KILIÇ\Desktop\DeBruijn\out-2015-12-25\graphviz-remove-errors.py Save File: C:\Users\İrfan KILIÇ\Desktop\DeBruijn\out-2015-12-25\debruijn-remove-errors-nodes.dat Save File: C:\Users\İrfan KILIÇ\Desktop\DeBruijn\out-2015-12-25\debruijn-remove-errors-edges.dat

Çizelge 5.1 kısaca açıklanırsa 1. satırda k değerini girilmesi istenmektedir. k=5 değeri verildi. 2. satırda DNA dizisini içeren dosya seçilir. 3. satırda seçilen dosyadaki kısa-okumalar sıralanmaktadır. 4. satırda sıralanan kısa-okumalar ve k-harfliler dosyalara yazılır. 5. satırda grafın dosyaları(düğüm ve kenarlar) ve graf çizimi için Python dosyası oluşmaktadır. 6. satırda oluşan graf üzerinde basitleştirme işlemi yapılmaktadır. 7. satırda graftaki kuyrukları(tips) kaldırma işlemi yapılmakta ve grafın son hali oluşturulmaktadır.

5.3. DNA Kısa-Okuma Verileriyle De Bruijn Graf Uygulamaları

Örnek DNA kısa-okuma veri dosyaları kullanılarak farklı durumları göstermek için yazılım aracı ile 4 uygulama yazılım üzerinde çalıştırılmıştır.

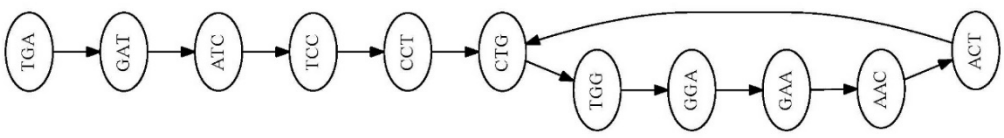
5.3.1. Uygulama 1

DNA dizisi “TGATCCTGGAACTG” şeklinde verilen veri dosyasının içeriği Çizelge 5.2’deki gibi verilsin.

Çizelge 5.2. Uygulama 1 DNA Kısa-Okuma Dosyası

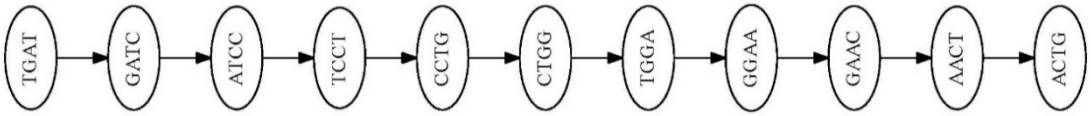
```
##################################
# TGATCCTGGAACTG -> DNA Sequence
##################################
# 12345... -> Position
# TGATCCTGGAACTG -> DNA Sequence
TGATCCTGGAACTG,1,14
```

Bu dosya algoritmaya verilsin. k-harfli(k-mers)=4 değeri verilsin. Burada k-harfli=4 verildiğinde her düğüm 3 bazdan oluşacak ve 4. bazdan geldiğinde sonraki düğüme bir kenar oluşacaktır. Şekil 5.10’da sırasıyla: TGA, GAT, ATC, TCC, CCT, CTG, TGG, GGA, GAA, AAC, ACT, CTG düğümlerinden oluşan grafın çıktısı verilmiştir.



Şekil 5.10. k=4 için De Bruijn Grafı

Eğer algoritmada k=5 değeri verilirse Şekil 5.11'deki gibi bir graf elde edilecektir. Burada dikkat edilirse hiç dallanma yoktur ve graf bir Euler yoluna sahiptir. Bu DNA dizisi için oluşturulan De Bruijn grafı k=5 değerinde ideal bir sonuç verecektir.



Şekil 5.11. k=5 için De Bruijn Grafı

Şekil 5.11'deki grafta basitleştirme(simplication) işlemi yapıldığında Şekil 5.12'deki graf elde edilir. Bu graf tek düğümlü olup dikkat edilirse başlangıçta verilen 14 bazlık DNA dizisi ile birebir aynıdır.

Şekil 5.12. k=5 için Basitleştirme Sonrası De Bruijn Grafı

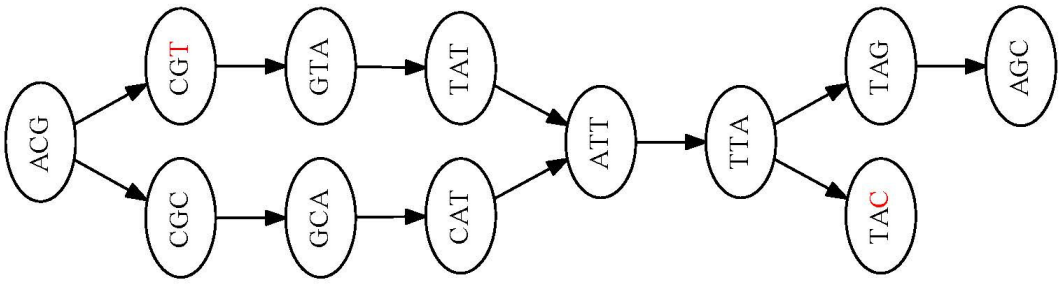
5.3.2. Uygulama 2

Belli bir DNA'nın farklı yerlerinden alınmış kısa-okumaların veri dosyası Çizelge 5.3'te verilmiştir.

Çizelge 5.3. Uygulama 2 DNA Kısa-Okuma Dosyası

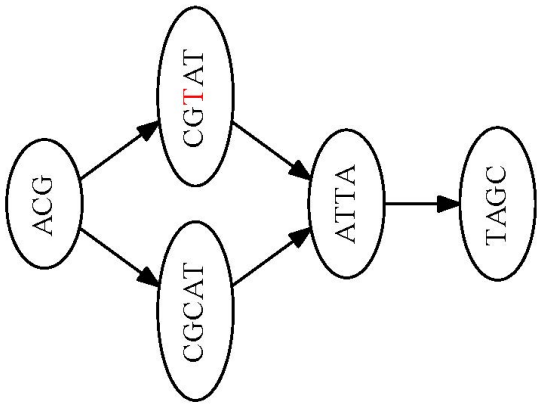
```
##################################
# 23.12.2015
##################################
# ACGCATTAGC -> DNA Sequence
# ACGCA      -> Short Read-0/0
#  CGCATT    -> Short Read-1/0
#     ATTAGC -> Short Read-2/0
# ACGTATT    -> Short Read-3/0
#   GCATT    -> Short Read-4/0
#    CATTAC  -> Short Read-5/0
#      TTAGC -> Short Read-6/0
##################################
# 123456789... -> Position
#ACGCATTAGC    -> DNA Sequence
ACGCA,1,5
CGCATT,2,7
ATTAGC,5,10
ACGTATT,1,7
GCATT,3,7
CATTAC,4,9
TTAGC,6,10
```

Bu örnekte dikkat edilmesi gereken her kısa okumanın pozisyon bilgisinin verilmiş olmasıdır. Örneğin “ACGCA, 1, 5” kısa-okumasının başlangıç pozisyonu 1’dir. Bu örnekte kırmızı ile işaretlenmiş olan bazlar hatalı okunan bazlardır. Örneği k=4 için çalıştırılsın. İlk oluşan graf Şekil 5.13’deki gibi olacaktır.



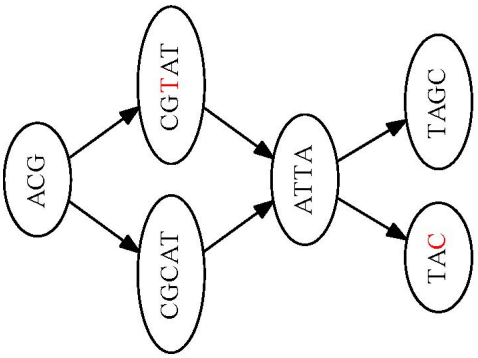
Şekil 5.13. k=4 için De Bruijn Grafı

İlk bakışta bakıldığında hatalardan dolayı bir kabarcığın ve bir kuyruğun oluştuğu görülmektedir. Şekil 5.13’deki graf üzerinde basitleştirme işlemi yapılırsa Şekil 5.14’deki graf oluşacaktır.



Şekil 5.14. Basitleştirme Sonrası De Bruijn Grafı

Burada dikkat edilirse ilk oluşan graf 12 düğümlü iken basitleştirme sonrası 6 düğüme inmiştir. Özellikle bellek kullanımı açısından bu büyük bir kazançtır. Bu aşamadan sonra hatalı kuyruğu silme işlemi yapılmıştır. Algoritma çalıştığında oluşan log incelenirse bir kuyruk bulunduğu görülecektir. Hatalı kuyruğun kaldırılması sonucu Şekil 5.15’deki graf elde edilir.



Şekil 5.15. Hatalı Kuyruğun Silinmesi

Tespit Edilen Hatalı Kuyruk:

Starting Remove Error(s)

Removing Node: TAC 1 0 9

Removing Edge: ATTA TAC 6 9

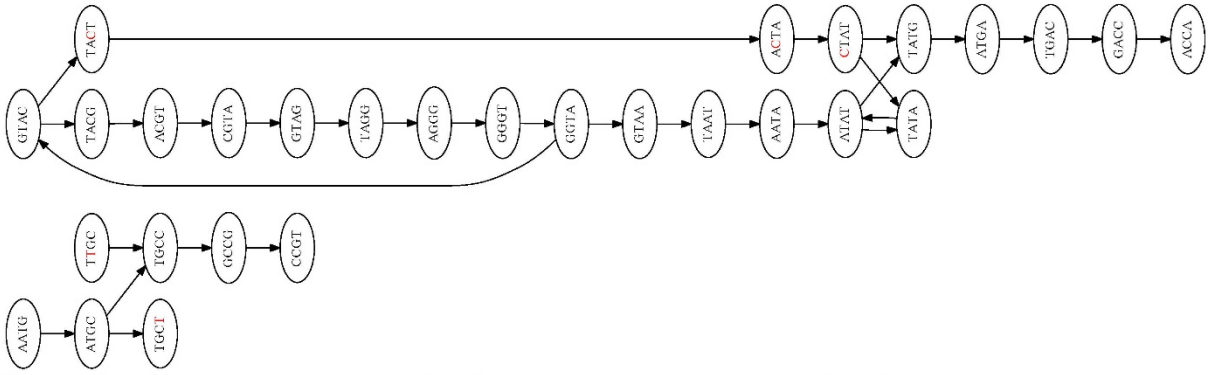
Bu aşamadan sonra Şekil 5.15'de görüldüğü gibi okuma hatasından dolayı bir kabarcık(bubble) bulunmaktadır. Bu işlemde yapılırsa "ACGCATTAGC" DNA dizisi elde edilecektir.

5.3.3. Uygulama 3

Çizelge 5.4. Uygulama 3 DNA Kısa-Okuma Dosyası

```
##################################
# 14.12.2015
##################################
# AATGCCGTACGTAGGGTAATATATGACCA -> DNA Sequence
#   TGCCGT   TAGGGT ATATAT      -> Short Read-0/0, Short Read-0/1,
Short Read-0/2
# AATGCT TACGTA         ATGACC  -> Short Read-1/0, Short Read-1/1,
Short Read-1/2
#  TTGCCG  CGTAGG TAATAT        -> Short Read-2/0, Short Read-2/1,
Short Read-2/2
#       GTACGT   GTACTA         -> Short Read-3/0, Short Read-3/1
# AATGCC       GGGTAA    TGACCA -> Short Read-4/0, Short Read-4/1,
Short Read-4/2
#           GTAGGG     TATGAC   -> Short Read-5/0, Short Read-5/1
#                   CTATAT      -> Short Read-6/0
##################################
# 123456789...                  -> Position
# AATGCCGTACGTAGGGTAATATATGACCA -> DNA Sequence
TGCCGT,3,8;TAGGGT,12,17;ATATAT,19,24
AATGCT,1,6;TACGTA,8,13;ATGACC,23,28
TTGCCG,2,7;CGTAGG,10,15;TAATAT,17,22
GTACGT,7,12;GTACTA,16,21
AATGCC,1,6;GGGTAA,14,19;TGACCA,24,29
GTAGGG,11,16;TATGAC,22,27
CTATAT,19,24
```

Çizelge 5.4'deki kısa-okumalardan oluşan DNA kısa-okuma dosyası incelenirse; çizelge 5.4'te dikkat edilirse aynı sırada birden fazla kısa-okuma yapılarak dosya oluşturulmuştur. Kırmızı renkle işaretli olanlar hatalı okumalardır. k=5 için programı çalıştıralım. İlk aşamada Şekil 5.16'da görülen graf elde edilir.



Şekil 5.16. k=5 için De Bruijjn Grafı

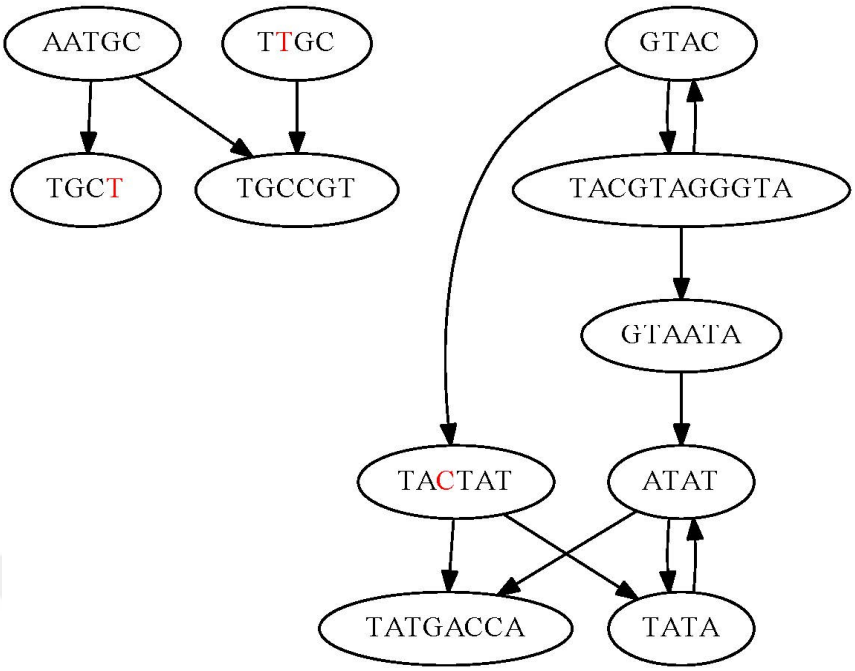
Şekil 5.16'daki grafta bir kopukluk gözükmektedir. "short-reads-steps.dat" dosyasında tüm kısa-okumalar pozisyonlarına göre gösterilmiştir.

```
Short Reads Steps
AATGCT
AATGCC
 TTGCCG
  TGCCGT
    GTACGT
      TACGTA
        CGTAGG
         GTAGGG
          TAGGGT
            GGGTAA
              GTACTA
               TAATAT
                 ATATAT
                 CTATAT
                    TATGAC
                     ATGACC
                      TGACCA
```

Şekil 5.17. Kısa-Okumaların Pozisyonları

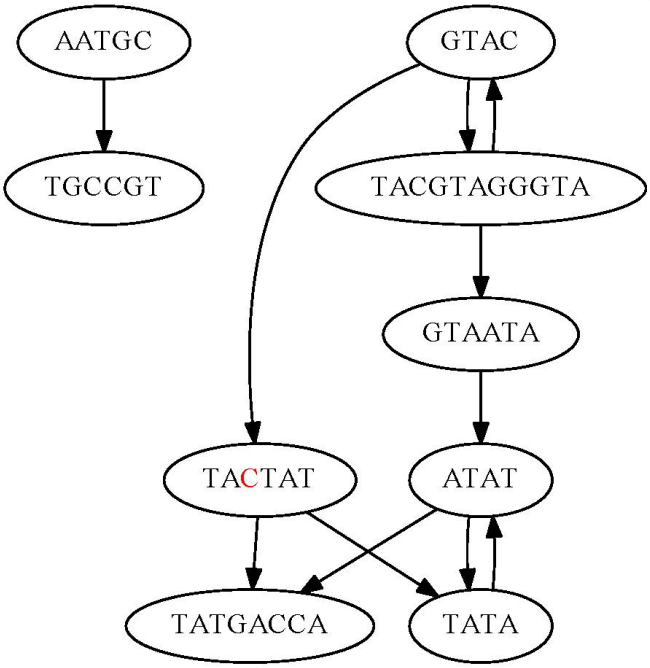
Şekil 5.17'de de görüldüğü gibi "TGCCGT" ile "GTACGT" kısa okumalarındaki "CCGT" düğümün son 3 bazı ile "GTAC" ilk 3 bazı örtüşmemektedir(pozisyon olarak aynı hizada değildir). Bundan dolayı grafta kopukluk olmuştur. Bu kopukluğu önlemek için k değeri üzerinde oynama yapılabilir.

Bu graf üzerinde basitleştirme işlemi sonucu Şekil 5.18'deki graf elde edilecektir. Şekil 5.16'daki grafta 29 düğüm vardır. Basitleştirme işleminden sonra düğüm sayısının 11'e düştüğü görülmektedir. Bu işlemle düğüm sayısı büyük değişiklik göstermiştir.



Şekil 5.18. Basitleştirme Sonrası De Bruijn Grafı

Log dosyası ve graflar incelenirse hatalı okuma kaynaklı 2 adet kuyruk, hatalı kuyruk kaldırma işlemiyle silinir. Bu işlemden sonra Şekil 5.19'daki graf elde edilir.



Şekil 5.19. Hatalı Kuyrukların Silinmesi

Bu aşamadan sonra buradaki hatalı kabarcık tespit edilip kaldırıldıktan sonra verilen kısa-okumalardan “AATGCCGTACGTAGGGTAATATATGACCA” DNA dizisi elde edilir.

5.3.4. Uygulama 4

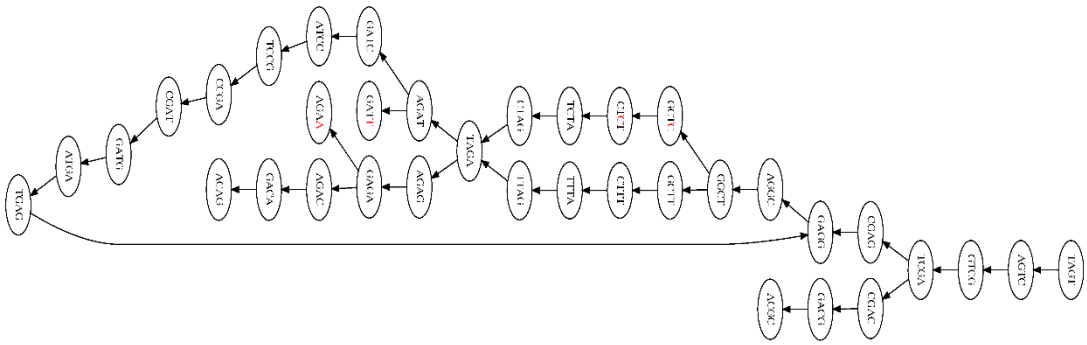
Çizelge 5.5’te her biri 7-bazdan oluşan kısa-okuma dosyası incelenirse;

Çizelge 5.5. Uygulama 4 DNA Kısa-Okuma Dosyası

```
##################################
# 25.12.2015
##################################
# TAGTCGAGGCTTTAGATCCGATGAGGCTTTAGAGACAG -> DNA Sequence
#  AGTCGAG CTTTAGA  CGATGAG CTTTAGA       -> Short Read-0/1, Short
Read-0/2, Short Read-0/3, Short Read-0/4
#   GTCGAGG  TTAGATC  ATGAGGC    GAGACAG -> Short Read-1/1, Short
Read-1/2, Short Read-1/3, Short Read-1/4
#       GAGGCTC   ATCCGAT AGGCTTT GAGACAG -> Short Read-2/1, Short
Read-2/2, Short Read-2/3, Short Read-2/4
#  AGTCGAG    TAGATCC ATGAGGC  TAGAGAA    -> Short Read-3/1, Short
Read-3/2, Short Read-3/3, Short Read-3/4
# TAGTCGA  CTTTAGA CCGATGA    TTAGAGA     -> Short Read-4/1, Short
Read-4/2, Short Read-4/3, Short Read-4/4
#      CGAGGCT  AGATCCG TGAGGCT  AGAGACA  -> Short Read-5/1, Short
Read-5/2, Short Read-5/3, Short Read-5/4
# TAGTCGA GCTTTAG TCCGATG  GCTCTAG       -> Short Read-6/1, Short
Read-6/2, Short Read-6/3, Short Read-6/4
#     TCGACGC    GATCCGA GAGGCTT AGAGACA  -> Short Read-7/1, Short
Read-7/2, Short Read-7/3, Short Read-7/4
# TAGTCGA     TTAGATC GATGAGG TTTAGAG      -> Short Read-8/1, Short
Read-8/2, Short Read-8/3, Short Read-8/4
#   GTCGAGG TCTAGAT   ATGAGGC  TAGAGAC    -> Short Read-9/1, Short
Read-9/2, Short Read-9/3, Short Read-9/4
#       AGGCTTT  ATCCGAT AGGCTTT GAGACAG -> Short Read-10/1, Short
Read-10/2, Short Read-10/3, Short Read-10/4
#  AGTCGAG   TTAGATT  ATGAGGC   AGAGACA  -> Short Read-11/1, Short
Read-11/2, Short Read-11/3, Short Read-11/4
#         GGCTTTA  TCCGATG    TTTAGAG     -> Short Read-12/1, Short
Read-12/2, Short Read-12/3
#     CGAGGCT TAGATCC  TGAGGCT   GAGACAG -> Short Read-13/1, Short
Read-13/2, Short Read-13/3, Short Read-13/4
#  AGTCGAG  TTTAGATC  ATGAGGC TTAGAGA    -> Short Read-14/1, Short
Read-14/2, Short Read-14/3, Short Read-14/4
#      GAGGCTT  GATCCGA GAGGCTT  GAGACAG -> Short Read-15/1, Short
Read-15/2, Short Read-15/3, Short Read-15/4
##################################
# 123456789...                          -> Position
# TAGTCGAGGCTTTAGATCCGATGAGGCTTTAGAGACAG -> DNA Sequence
AGTCGAG,2,8;CTTTAGA,10,16;CGATGAG,19,25;CTTTAGA,27,33
GTCGAGG,3,9;TTAGATC,12,18;ATGAGGC,21,27;GAGACAG,32,38
GAGGCTC,6,12;ATCCGAT,16,22;AGGCTTT,24,30;GAGACAG,32,38
AGTCGAG,2,8;TAGATCC,13,19;ATGAGGC,21,27;TAGAGAA,30,36
TAGTCGA,1,7;CTTTAGA,10,16;CCGATGA,18,24;TTAGAGA,29,35
CGAGGCT,5,11;AGATCCG,14,20;TGAGGCT,22,28;AGAGACA,31,37
TAGTCGA,1,7;GCTTTAG,9,15;TCCGATG,17,23;GCTCTAG,26,32
TCGACGC,4,10;GATCCGA,15,21;GAGGCTT,23,29;AGAGACA,31,37
TAGTCGA,1,7;TTAGATC,12,18;GATGAGG,20,26;TTTAGAG,28,34
GTCGAGG,3,9;TCTAGAT,11,17;ATGAGGC,21,27;TAGAGAC,30,36
AGGCTTT,7,13;ATCCGAT,16,22;AGGCTTT,24,30;GAGACAG,32,38
AGTCGAG,2,8;TTAGATT,12,18;ATGAGGC,21,27;AGAGACA,31,37
```

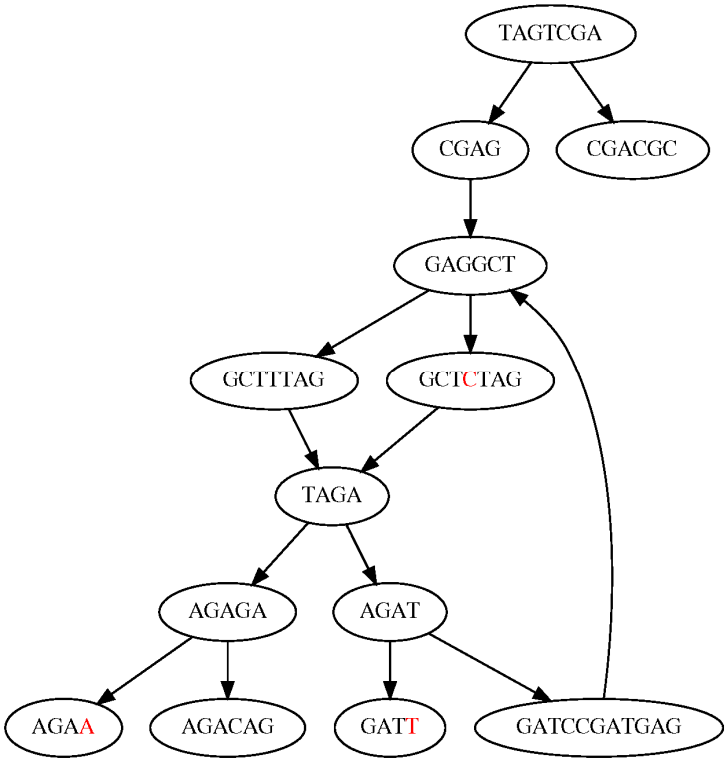
```
GGCTTTA,8,14;TCCGATG,17,23;TTTAGAG,28,34
CGAGGCT,5,11;TAGATCC,13,19;TGAGGCT,22,28;GAGACAG,32,38
AGTCGAG,2,8;TTTAGATC,11,18;ATGAGGC,21,27;TTAGAGA,29,35
GAGGCTT,6,12;GATCCGA,15,21;GAGGCTT,23,29;GAGACAG,32,38
```

Çizelge 5.5'te kırmızı ile işaretlenen bazlar hatalı okunan bazlardır. Yazılım çalıştırılıp, k=5 (yani her düğüm k-1-harfli=4, her kenar k-harfli=5 olacak) değeri verilsin. İlk aşamada Şekil 5.20'deki graf elde edilir.



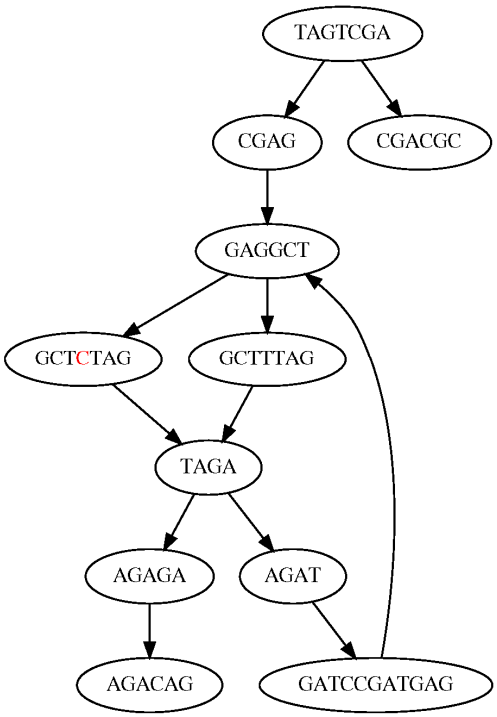
Şekil 5.20. k=5 için De Bruijn Grafı

2. aşamada 1. basitleştirme işlemi yapılmaktadır. Bu işlem sonrasında Şekil 5.21'deki graf elde edilmektedir.



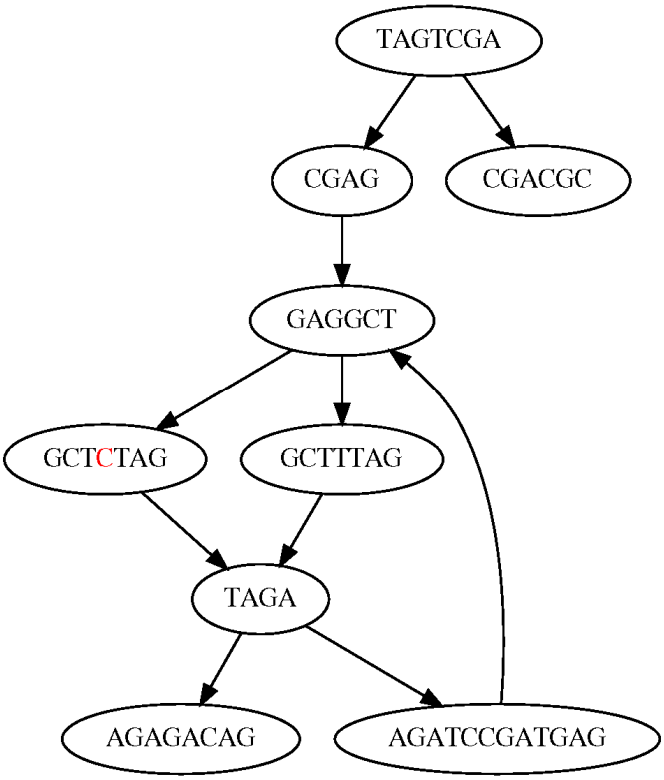
Şekil 5.21. 1. Basitleştirme Sonrası De Bruijn Grafı

Şekil 5.20'de dikkat edilirse başlangıçta 36 düğümden oluşan graf 1. basitleştirme işleminden sonra 13 düğüme düşmüştür. Bundan sonra yazılım ile hatalı kuyrukların silinmesi işlemini gerçekleştirilmektedir. Bu işlemden sonra Şekil 5.22'deki graf elde edilecektir.



Şekil 5.22. Hatalı Kuyrukların Silinmesi

2. basitleştirme işleminden sonra Şekil 5.23'deki graf elde edilecektir. De Bruijn grafı 9 düğüme düşmüştür. Bu örnekte başlangıçta 36 düğümlü bir graf bu yazılım ile 9 düğüme düşürülmüştür.



Şekil 5.23. 2. Basitleştirme Sonrası De Bruijn Grafı

5.4. Uygulama Yazılımı Kod Yapısı

Çizelge 5.6 ve Çizelge 5.7'te yazılımın genel kod yapısı verilmiştir.

Çizelge 5.6. De Bruijn Graf Uygulama Yazılımı Genel Kod Yapısı

DeBruijn.java

```
public class DeBruijn {
    private final int k_value;//# k
value
    private List<KMer> k_mers;
    private     List<Edge>     edges;//#
multimap from nodes to neighbors
    private List<Node> nodes;//# maps k-
1-mers to Node objects

    //Iterate over nodes; tally #
balanced, semi-balanced, neither
    private int balanced;
    private int semiBalanced;
    private int neither;
    private Node head, tail;

    private final int NO_EDGE = 0;
    private final int SINGLE_EDGE = 1;
    private final int MULTI_EDGE = 2;

    public      DeBruijn(List<ShortRead>
shortReads, int k) {
        this.balanced = 0;
        this.semiBalanced = 0;
        this.neither = 0;
        this.head = null;
        this.tail = null;
        this.k_value = k;
        chop(shortReads, this.k_value);
        createEdges();
        createNodes();
        calcTally();
    }
```

Node.java

```
public class Node {

    private String label;
    private int input;
    private int output;
    private List<Integer>
refInEdges = new
ArrayList<>();//for quick
search
    private List<Integer>
refOutEdges = new
ArrayList<>();//for quick
search

    public Node(String
node) {
        this.label = node;
        this.input = 0;
        this.output = 0;
    }

    public Node(String
label, int input, int
output) {
        this.label = label;
        this.input = input;
        this.output =
output;
    }
```

Çizelge 5.7. De Bruijn Graf Uygulama Yazılımı Kenar Kod Yapısı

Edge.java

```
public class Edge {

    private String source;
    private String destination;
    private int refSource;//for quick
search
    private int refDestination;//for
quick search

    public Edge() {
    }

    public Edge(String source, String
destination) {
        this.source = source;
        this.destination = destination;
    }
```

ShortRead.java

```
public class ShortRead {

    private String data;
    private int startPos;
    private int finishPos;

    public ShortRead(String
data, int startPos, int
finishPos) {
        this.data = data;
        this.startPos =
startPos;
        this.finishPos =
finishPos;
    }
```

5.5. Uygulama Sonuçlarının Değerlendirilmesi

Örneklerden anlaşıldığı gibi De Bruijn graflarının oluşturulmasının diğer yöntemlere göre daha pratik olması uygulamada kolaylıklar sağlamaktadır. Basitleştirme ve hatalı kuyruk kaldırma işlemleriyle graftaki düğüm sayısının önemli ölçüde düşmesi bellek ve işlem açısından avantaj sağlamaktadır. Ayrıca burada k değerinin uygun şekilde verilmesi önem arz etmektedir.

Çizelge 5.8'de bu bölümde anlatılan örneklerin bir özeti verilmiştir. Çizelge 5.8'de de görüldüğü üzere başlangıçta çok yüksek düğüm ve kenara sahip De Bruijn grafları graf teknikleri kullanılarak minimize edilmiştir. Bu sayede konsensüs genomun daha yüksek doğrulukla, daha az zamanda ve daha az donanım kaynağı kullanılarak De Bruijn grafları ile elde edildiği gösterilmiştir.

Çizelge 5.8. De Bruijn Grafı Uygulama Örnekleri Özeti

Örnekler	Başlangıç Düğüm ve Kenar Sayısı	k-harfli sayısı	İşlem Sonrası Düğüm ve Kenar Sayısı	Yapılan İşlem	Düğüm/Kenar Azalma Oranı (%)
Örnek 1	11 / 10	5	1 / 0	Basitleştirme	%91 / %100
Örnek 2	12 / 12	4	6 / 6	Basitleştirme	%50 / %50
Örnek 2	12 / 12	4	5 / 5	Kuyruk Kaldırma	%58 / %58
Örnek 3	29 / 31	5	11 / 13	Basitleştirme	%62 / %58
Örnek 3	29 / 31	5	9 / 11	Kuyruk Kaldırma	%68 / %65
Örnek 4	36 / 37	5	13 / 14	Basitleştirme	%64 / %62
Örnek 4	36 / 37	5	11 / 12	Kuyruk Kaldırma	%70 / %68
Örnek 4	36 / 37	5	9 / 10	2. Basitleştirme	%75 / %73

Çizelge 5.8'den görüldüğü gibi yazılımın uygulandığı örneklerde en az %50 bir iyileştirme gerçekleşmiştir.

6. SONUÇ

Bu tezde günümüzde çok büyük önem kazanan genetik çalışmalarda bilgisayar bilimlerinin katkısının ne derece önemli olduğu gösterilmeye çalışılmıştır. Geleceğin konuları olan yapay doku üretimi, ileri kanser tedavi yöntemleri göz önüne alınırsa bilgisayar bilimlerinde kullanılan teknikler bu çalışmalar için doğruluk ve zaman açısından kritiktir. Özellikle bilgisayar bilimlerinde çok sık kullanılan graflar bu çalışmanın temelini oluşturmuştur.

Bu çalışmada DNA dizilemede yeni nesil dizileme yöntemlerinde çok sık kullanılan grafların özellikle De Bruijn graflarının önceden kullanılan klasik yöntemlere göre avantajları gösterilmeye çalışılmıştır. Bu tez DNA dizilemede kullanılan De Bruijn graflarına açıklık getirmektedir. Bu çalışmada De Bruijn graflarını tüm detayları ile anlatılmaya çalışılmıştır. Bu tez düzenlenirken 6 bölüm halinde düzenlenmiştir. Birinci bölümde tezin konusu üzerine detaylara değinilmiş ve literatürdeki çalışmalardan bahsedilmiştir. Bu bölümde DNA dizileme de De Bruijn graflarını kullanan yazılım paketlerinden bahsedilmiştir. Ayrıca tezin genel bir çerçevesi sunulmuştur. İkinci bölümde temel mikrobiyolojik konulardan bahsedilmiş, DNA dizileme ve DNA dizilemenin önemi, biyoinformatik konuları anlatılmış ve çalışma amaçlı gen bankalarından gen dizilerine nasıl ulaşıldığı anlatılmıştır. Üçüncü bölümde DNA dizileme ve DNA dizileme yöntemleri detaylarıyla anlatılmıştır. Bu bölümde DNA ve DNA dizileme tarihi ile ilgili literatür araştırması verilmiştir. Ayrıca bu bölümde De Bruijn graflarını kullanan yeni nesil dizileme tekniklerinden bahsedilmiştir. Dördüncü bölümde De Bruijn dizileri, De Bruijn grafları ve De Bruijn grafları üzerine geliştirilen yöntemler detaylarıyla anlatılmıştır. DNA dizilemede kullanılan yöntemlerde graflar ve De Bruijn graflarının literatür araştırması yapılmış ve yeni nesil dizileme ile ilgili diğer konular anlatılmıştır. Ayrıca De Bruijn graflarını kullanan Velvet Assembler hakkında De Bruijn graf teknikleri anlatılmıştır. Beşinci bölümde Java ile kodlanan yazılım anlatılmış ve örnek DNA kısa-okuma verileri kullanılarak yazılım üzerinde örneklerle De Bruijn graf tekniklerinin nasıl çalıştığı gösterilmiştir. Ayrıca bu çalışmada Ek A, Ek B ve Ek C'de AbySS, Velvet ve IDBA yazılımlarının kurulum ve kullanımı ile ilgili kılavuz verilmiştir.

Bu çalışma ile De Bruijn graflarının genetik çalışmalarda önemi anlaşılmıştır. Özellikle yeni nesil dizileme teknolojilerinde çok sık başvurulduğu gösterilmiştir. Bu çalışmanın bu konuda çalışmak isteyenlere bir başlangıç olabileceği gibi mevcut teknikleri geliştirmelerinde yardımcı olması beklenmektedir.

Bu çalışma ile gerçek DNA verilerinin birkaç GB olduğu hesap edildiğinde bu DNA verilerinin kısa-okuma dosyaları kullanılarak De Bruijn grafları ile incelenmesinin çok büyük katkısının olduğu görülmüştür.

KAYNAKLAR

Adams, M.D., Celniker, S.E., Holt, R.A., … Myers, E.W., Rubin, G.M., Venter, J.C. "The genome sequence of Drosophila melanogaster" Science. 2000 Mar 24;287(5461):2185-95.

Aguirre, G.K., Mattar, M.G., Magis-Weinbergn, L. (2011) "de Bruijn cycles for neural decoding".. NeuroImage 56: 1293–1300.

Anderson, S.E. (1997-2009). "Bit Twiddling Hacks". Stanford University. https://github.com/hazirguo/BitHacks, 12 Şubat 2009'da erişildi.

Bayat, A. Science, medicine and the future: Bioinformatics. BMJ, 2002; 324 p: 1018-1022.

Bcgsc.ca web adresi, http://www.bcgsc.ca/platform/bioinfo/software/abyss web adresi, 20 Mayıs 2015 tarihinde erişildi.

Boost.org web adresi, http://www.boost.org/users/download web adresi, 20 Mayıs 2015 tarihinde erişildi.

Brooksbank C, Camon E, Midori AH, Magrane M, Martin MJ, Mulder N, O'Donovan C, Parkinson H, Tuli MA, Apweiler R, Birney E, Brazma A, Henrick K, Lopez R, Stoesser G, Stoehr P. and Cameron G, Nucleic Acids Research, 2003; Vol. 31, No. 1, p:43-50.

Bryant Jr., D.W.,Wong,W.-K., and Mockler, T. C. (2009). QSRA - a quality value guided de novo short read assembler. BMC Bioinformatics In press.

Buttle, A.J. Challanges in bioinformatics, Trends Biotechnol. 2001;19, p:159-160.

C. Elegans Sequencing Consortium," Genome sequence of the nematode C. elegans: a platform for investigating biology" Science. 1998 Dec 11;282(5396):2012-8.

Cfn.upenn.edu web adresi, https://cfn.upenn.edu/aguirre/wiki/public:de_bruijn_software, 13 Ocak 2013'te erişildi.

Chaisson, M.J., Pevzner, P.A., (2008). Short read fragment assembly of bacterial genomes. Genome Research 18:324–330.

Chevreux, B. (2005). MIRA: An Automated Genome and EST Assembler. Ph.D. thesis, The Medical Faculty of Heidelberg of the Ruprecht-Karls-University.

Chikhi, R., Rizk, G. "Space-efficient and exact de Bruijn graph representation based on a Bloom filter" Algorithms for Molecular Biology 2013 8:22. Doi: 10.1186/1748-7188-8-22.

Code.google.com web adresi, https://code.google.com/archive/p/hku-idba/wikis/InstallationGuideAndUserGuide.wiki web adresi, 21 Mayıs 2015 tarihinde erişildi.

Cornish-Bowden, A. "Nomenclature for incompletely specified bases in nucleic acid sequences: recommendations 1984" Nucleic Acids Res. 1985 May 10; 13(9): 3021–3030.

De Bruijn, N. G. (1975), Acknowledgement of Priority to C. Flye Sainte-Marie on the counting of circular arrangements of 2n zeros and ones that show each n-letter word exactly once, T.H.-Report 75-WSK-06, Technological University Eindhoven.

De Bruijn, N.G. (1946). "A Combinatorial Problem". Koninklijke Nederlandse Akademie v. Wetenschappen 49: 758–764.

Demir, M., Karcı, A., Özdemir M. (2011), Fidan Gelişim Algoritması Yardımı ile DNA Motiferinin Keşfi, Çankaya University Journal of Science and Engineering, 8(1), 51-62.

Dnabaser.com web adresi, http://www.dnabaser.com/help/samples/what%20is%20a%20chromatogram.html, 27 Haziran 2015 tarihinde erişildi.

Doerry, E.., Douglas, S.A., Kirkpatrick, A.E., Westerfield M. Participatory Design for Widely-Distributed Scientific Communities. Proceedings of the 3rd Conference on Human Factors and the Web 1997.

Dohm, J. C., Lottaz, C., Borodina, T., and Himmelbauer, H. (2007). SHARCGS, a fast and highly accurate short-read assembly algorithm for de novo genomic sequencing. Genome Research 17:1697–1706.

Dunham, I., Hunt, A. R., Collins J. E., …Tapia, C. E. Bruder & K. P. O'Brien "The DNA sequence of human chromosome 22" Nature 1999 Dec 402, 489-495 doi:10.1038/990031

Durmaz, R., Klinik Mikrobiyolojide Pyrosekans Kursu 2010 Kurs Özet Kitabı 57.

Durmaz, R., Uygulamalı Moleküler Mikrobiyoloji, Akademisyen Kitapevi, İstanbul, 2001.

Ebi.ac.uk web adresi (a), https://www.ebi.ac.uk/~zerbino/velvet/velvet_1.2.10.tgz web adresi, 22 Mayıs 2015 tarihinde erişildi.

Ebi.ac.uk web adresi (b), https://www.ebi.ac.uk/~zerbino/velvet/Manual.pdf web adresi, 22 Mayıs 2015 tarihinde erişildi.

Evelyn, B. Kelly, Encyclopedia of Human Genetics and Disease Volume I, 1. Cilt, ABC-CLIO, LLC, California, 2013, 27.

Feagan, L., Rohrer, J., Garret, A., Amthauer, H., Komp, E., Johnson, D., Hock, A. Bioinformatics process management: informationflow via a computional journal. Source Code for Biology and Medicine. 2007;2:9 p: 2-3.

Fleischmann, Robert D., Adams, Mark, D.; White, O., Clayton, R., . . . Venter, J. Craig (July 28, 1995). "Whole-Genome Random Sequencing and Assembly of Haemophilus influenzae Rd". Science (Washington, DC: American Association for the Advancement of Science) 269 (5223): 496–512.

Flye, Sainte-Marie, C. (1894), "Solution to question nr. 48", L'intermédiaire des Mathématiciens 1: 107–110.

Galibert, F., Alexandraki, D., Baur A., Boles E., Chalwatzis, N., Chuat, J.C., Coster, F., Cziepluch, C., De Haan, M., Domdey, H., Durand, P., Entian, K.D., Gatius, M., Goffeau, A., Grivell, L.A., Hennemann, A., Herbert, C.J., Heumann, K., Hilger, F., Hollenberg, C.P., Huang, M.E., Jacq, C., Jauniaux, J.C., Katsoulou, C., Karpfinger-Hartl, L. 1996 May 1, "Complete nucleotide sequence of Saccharomyces cerevisiae chromosome X" EMBO J. 15(9): 2031–2049.

Gatto, M.L., The changing face of Bioinformatics, Drug Discov. Today 2003; 8, p: 375-376.

Github.com web adresi, https://github.com/bcgsc/abyss#abyss web adresi, 20 Mayıs 2015 tarihinde erişildi.

Good, I.J. (1946). "Normal recurring decimals". Journal of the London Mathematical Society 21 (3): 167–169. doi:10.1112/jlms/s1-21.3.167.

Hernandez, D., François, P., Farinelli, L., Østerås, M., Schrenzel, J. "De novo bacterial genome sequencing: Millions of very short reads assembled on a desktop computer" Genome Res. 2008 May; 18(5): 802–809. doi: 10.1101/gr.072033.107.

Hku-idba.googlecode.com web adresi, http://hku-idba.googlecode.com/files/idba-0.17.tar.gz web adresi, 21 Mayıs 2015 tarihinde erişildi.

Hogue C.W.V. Bioinformatics for Beginners. Trends Biochem. Sci. 2002; 27, 542-543.

Holley, R.W., Everett, G.A., Madison, J.T., Zamir, A. (May 1965). "Nucleotide Sequences In The Yeast Alanine Transfer Ribonucleic Acid" *J Biol Chem* **240** (5): 2122–8.

Homolog.us web adresi, http://www.homolog.us/Tutorials/index.php?p=1.1&s=1, 1 Mayıs 2016 yılında erişildi.

Hurlbert, G., Isaak, G. (1993), "On the de Bruijn torus problem", Journal of Combinatorial Theory, Series A 64 (1): 50–62, doi:10.1016/0097-3165(93)90087-O.

Idury, R. M. and Waterman, M. S. (1995). A new algorithm for DNA sequence assembly. Journal of Computational Biology 2:291–306.

Illumina Sequencer Enables $1,000 Genome. News: Genomics & Proteomics. Gen. Eng. Biotechnol. News (paper) 34 (4). 15 February 2014. p. 18.

Jain, E. A practical Introduction to Bioinformatics, Trends Biotechnol. 2002; 20, p:226.

Jeck, W. R., Reinhardt, J. A., Baltrus, D. A., Hickenbotham, M. T., Magrini, V., Mardis, E. R., Dangl, J. L., and Jones, C. D. (2007). Extending assembly of short DNA sequences to handle error. Bioinformatics 23:2942–2944.

Kaul, S., Koo, Hean L., Jenkins, J., Rizzo, M., Rooney, T., Tallon, Luke J., Feldblyum, T., Nierman, W., Benito M.I., Lin, X.Y. "Analysis of the genome sequence of the flowering plant Arabidopsis thaliana" Nature. 2000 408(6814). p.796-815.

Kenneth, W.A., Advanced Techniques in Chromosome Research, Marcel Dekker, New York, 1991, 110-111.

Klug, S.W., Cummings, W.R., 2000 Concept of Genetics, Prentice Hall, New Jersey 745 p..

Kumar, S. Bioinformatics, http//www.geocities.com/bioinformaticsweb/, 2005 yılında erişilmiştir.

Leroux, P. (2002), Coassociative grammar, periodic orbits and quantum random walk over Z, arXiv:quant-ph/0209100, Bibcode 2002quant.ph..9100P.

Lin Liu, Yinhu Li, Siliang Li, Ni Hu, Yimin He, Ray Pong, Danni Lin, Lihua Lu, Maggie Law, "Comparison of Next-Generation Sequencing Systems" 2012 Volume 2012 (2012), Article ID 251364, 11 pages.

Lodish, H., Berk, A., Zipursky, S.L., Matsudaira, P., Baltimore, D., Darnell, J., Molecular Cell Biology, 4th edition New York: W. H. Freeman; 2000 ISBN-10: 0-7167-3136-3

Luscombe N.M., Greenbaum D., Gerstein M. What is bioinformatics? An introduction and overview. Yearbook of Medical Informatics 2001; s:83-85.

Martin, M. H. (1934), "A problem in arrangements", Bulletin of the American Mathematical Society 40 (12): 859–864, doi:10.1090/S0002-9904-1934-05988-3, MR 1562989.

Maxam, A., Gilbert, W. 1977, A new method of sequencing DNA. Proceedingsof the National Academy of Sciences, 74, 560-4.

Michael, R. Stratton, Peter, J. Campbell, P. Andrew Futreal "The cancer genome" Nature 458, 719-724 (9 April 2009) doi:10.1038/nature07943.

Miller, J.R., Koren, S., Sutton, G. (2010). "Assembly algorithms for next-generation sequencing data". Genomics 95 (6): 315–27. doi:10.1016/j.ygeno.2010.03.001.

Mullins, L.J., Mullins, J.J. "Insights from the rat genome sequence" Genome Biol. 2004; 5(5): 221.

Myers, E. W. (2005). The fragment assembly string graph. Bioinformatics 21:ii79–ii85.

Myers, E.W. (1995). Towards simplifying and accurately formulating fragment assembly. Journal of Computational Biology 2.

Nature.com web adresi, http://www.nature.com/encode/about, 9 Nisan 2016 yılında erişildi.

Ncbi.nlm.nih.gov web adresi, http://www.ncbi.nlm.nih.gov/, 27 Haziran 2015 tarihinde erişildi.

Nyrén P., "The history of pyrosequencing " Methods Mol Biol. 2007;373:1-14.

Pearson, W., Lipman, D.J. (1988). Improved tools for biological sequence comparison. Proc. Nati. Acad. Sci. USA 4:2444–2448.

Pettersson, E., Lundeberg, J., Ahmadian, A. (February 2009). "Generations of sequencing technologies". *Genomics* 93(2): 105–11. doi:10.1016/j.ygeno.2008.10.003.

Pevzner, P.A., Tang, H. (2001). "Fragment Assembly with Double-Barreled Data". Bioinformatics/ISMB 1: 1–9.

Pevzner, P.A., Tang, H. (2001). Fragment assembly with double barreled data. Bioinformatics 17:S225–S233.

Pevzner, P.A., Tang, H., Waterman, M.S. (2001). "An Eulerian path approach to DNA fragment assembly". PNAS 98 (17): 9748–9753. Bibcode 2001PNAS...98.9748P. doi:10.1073/pnas.171285098.

Phillip, E.C.C., Pevzner, P.A., Tesler, G. "How to apply de Bruijn graphs to genome assembly" Nature Biotechnology 29, 987–991 (2011) doi:10.1038/nbt.2023

Phrap.org web adresi, http://www.phrap.org, 11 Nisan 2015'de erişildi.

Polat M., Karahan A.G., Multidisipliner yeni bir bilim dalı: biyoinformatik ve tıpta uygulamaları, S.D.Ü. Tıp Fakültesi Dergisi, 2009:16(3)/ 41-50.

Rasmussen, K., Stoye, J., Myers, E. (2005). Efficient q-gram filters for finding all-matches over a given length. In 9th Conf. on Computational Molecular Biology, pages 189–203.

Reinhardt, J. A., Baltrus, D. A., Nishimura, M. T., Jeck, W. R., Jones, C. D., and Dangl, J. L. (2009). De novo assembly using low-coverage short read sequence data from the rice pathogen Pseudomonas syringae pv.oryzae. Genome Research 19:294–305.

Ruskey, F., Sawada, J., Williams, A., De Bruijn Sequences for Fixed-Weight Binary Strings, SIAM Journal on Discrete Mathematics, 26 (2012) 605-617.

Sambrook, J., Fritsch, E.F., Maniatis, T. 1989 Molecular Cloning, a laboratory manual. Cold spring harbor laboratory Press New York.

Sanger, F., Nicklen, S., Coulson, A.R., DNA sequencing with chain- terminating inhibitors. 1977, *Proceedings of the National Academy of Sciences*, 74, 5463-7.

Searls, D.B. (2010) The Roots of Bioinformatics. PLoS Comput Biol 6(6): e1000809. doi:10.1371/journal.pcbi.1000809

Simpson, J.T., Wong, K., Jackman, S.D., Schein, J.E., Jones, S. J.M., Birol, İ. "ABySS: A parallel assembler for short read sequence data" Genome Res. 2009 Jun; 19(6): 1117–1123. doi: 10.1101/gr.089532.108

Smith, L.M., Sanders, J.Z., Kaiser, R.J., Hughes, P., Dodd, C., Connell, C.R., Heiner, C., Kent, S.B.H., Hood, L.E. (1986) Fluorescence Detection in Automated DNA Sequence Analysis. Nature 321:674-679.

Taylor, D. Leland "Phast (Phage Assembly Suıte And Tutorıal): A Web Based Genome Assembly Teachıng Tool" Computational Biology Center for Interdisciplinary Studies Davidson College May 6, 2012

The International HapMap Consortium "A haplotype map of the human genome" Nature 437, 2005 Oct 1299-1320 doi:10.1038/nature04226.

Theguardian.com web adresi, http://www.theguardian.com/law/2013/jun/13/supreme-court-genes-patent-dna, 9 Nisan 2016 yılında erişildi.

Tuliani, J. (2001), "de Bruijn sequences with efficient decoding algorithms", Discrete Mathematics 226 (1-3): 313–336, doi:10.1016/S0012-365X(00)00117-5.

Warren, R. L., Sutton, G. G., Jones, S. J. M., and Holt, R. A. (2007). Assembling millions of short DNA sequences using SSAKE. Bioinformatics 23:500–501.

Watson, J.D., Crick, F.H.C. (1953). "A Structure for Deoxyribose Nucleic Acid" (PDF). Nature 171 (4356): 737–738. Doi:10.1038/171737a0.

Wellcome.ac.uk web adresi, http://www.wellcome.ac.uk/News/Media-office/Press-releases/2010/WTX060061.htm, 9 Nisan 2016 yılında erişildi.

White House Press Release web adresi, https://www.whitehouse.gov/omb/press_releases, 9 Nisan 2016 yılında erişildi.

Whitfield, J., "Y chromosome sequence completed" Nature. 2003 June doi:10.1038/news030616-16.

Wikipedia.org web adresi, https://tr.wikipedia.org/wiki/Biyoenformatik, 15 Haziran 2015 tarihinde erişildi.

Yourgenome.org web adresi, http://www.yourgenome.org/facts/timeline-history-of-genomics, 9 Nisan 2016 tarihnde erişildi.

Yu Peng, Henry, C. M., Leung, S. M. Yiu, Francis, Y.L. Chin “IDBA – A Practical Iterative de Bruijn Graph De Novo Assembler“ Volume 6044 of the series Lecture Notes in Computer Science 2010 pp 426-440.

Zerbino, D.R.., Birney, E. (2008). "Velvet: algorithms for de novo short read assembly using de Bruijn graphs". Genome Research 18 (5): 821–829. doi:10.1101/gr.074492.107.

Zhang, Fu Ji; Lin, Guo Ning (1987). "On the de Bruijn-Good graphs". Acta Math. Sinica 30 (2): 195–205.

Zhanglab.ccmb.med.umich.edu web adresi, http://zhanglab.ccmb.med.umich.edu/FASTA/, 21 Nisan 2016 tarihinde erişildi.

Zülal, A., 2001, İnsan Genomu, kalıtım şifresinin peşinde 136 yıl, Tübitak Yayınları, Mart; 5-11.

EKLER

EK A. ABySS

Abyss de novo assembler(birleştirici)'ın 10 Temmuz 2014 tarihi itibariyle son çıkan versiyonu http://www.bcgsc.ca/platform/bioinfo/software/abyss (bcgsc.ca web adresi) adresinden mevcut Ubuntu 14.04.1 LTS üzerine aşağıdaki komutla kurulmaktadır (github.com web adresi).

#apt-get install abyss

Küçük bir sentetik veri kümesini assemble etmek için;

wget http://www.bcgsc.ca/platform/bioinfo/software/abyss/releases/1.3.4/test-data.tar.gz

tar xzvf test-data.tar.gz

*#abyss-pe k=25 name=test *

in='test-data/reads1.fastq test-data/reads2.fastq'

şeklinde yazılım paketi kullanılabilir.

Assemble olmuş veri kümesindeki contig istatistiklerini hesaplamak için;

#abyss-fac test-unitigs.fa

komutu kullanılır.

Abyss yazılım paketi Boost, sparsehash ve Open MPI kütüphanelerine ihtiyaç duyar. Abyss GCC gibi OpenMP'yi destekleyen C++ Derleyicisine gereksinim duyar. Boost 1.51.0 veya 1.52.0 derlendiğinde hata içerdiği için Abyss yazılım paketinde hata alınacaktır. Sonraki Boost versiyonları hatasız derlenebilmektedir.

Abyss'i GitHub üzerinden derlemek için Autoconf ve Automake araçlarına gerek vardır. Konfigürasyon scriptini üretmek ve dosyaları derlemek için *"./autogen.sh"* komutu girilir. Sonraki adımlarda kaynaktan derleme yapılmalıdır. /usr/local dizininde Abyss'i derlemek ve yüklemek için sırasıyla;

#./configure

#make

#make install

komutları girilir. Başka bir dizinde yükleme yapmak için;

#./configure --prefix=/opt/abyss

#make

#make install

komutları girilir.

Abyss'in paralel olarak çalışması için GCC 4.2 veya daha üstü versiyonlu modern bir derleyici gerektiren OpenMP paketini kullanır. Daha eski versiyonlu derleyici en güncel versiyona çekilmelidir. Bilgisayarda farklı GCC derleyicileri varsa farklı bir derleyici belirtmek için; *"./configure CC=gcc-4.6 CXX=g++-4.6"* komutu girilmelidir.

Abyss Boost C++ kütüphanesine ihtiyaç duyar. Çoğu bilgisayarda Boost yüklü gelir. Eğer yüklü değilse http://www.boost.org/users/download (boost.org web adresi) adresinden indirilebilir. Boost'un derlenmesine gerek yoktur. Boost başlangıç dosya dizini /usr/include/boost dizininde bulunur. Farklı bir dizin olması için *"./configure --with-boost=/usr/local/include"* komutu ile değiştirilebilir.

MPI destekli paralel bir assembler(birleştirici) istiyorsanız. MPI'i bulunduğu /usr/include ve /usr/lib dizininde veya farklı bir yerde derlemek için *"./configure --with-mpi=/usr/lib/openmpi"* komutu kullanılmalıdır.

Abyss'i bellek kullanımını azaltan sparhehash kütüphanesi kullanılarak derlenmesi tavsiye edilir. Abyss'i sparsehash ile derlemek için *"./configure CPPFLAGS=-I/usr/local/include"* komutu girilmelidir.

Varsayılan maksimum k-harfli sayısı 64 olup, derleme zamanında bellek kullanımını azaltmak için bu sayı *"./configure --enable-maxk=96"* komutu arttırılabilir veya azaltılabilir. Arttırma veya azaltma 32'nin katları şeklinde olmalıdır.

Derleme uyarıları ile karşılaşmak istemiyorsanız *"make AM_CXXFLAGS=-Wall"* komutu kullanabilirsiniz.

Abyss'i çalıştırmak için Abyss'in dizini *"/opt/abyss"* ise /opt/abyss/bin dizinini eklemek için *"PATH=/opt/abyss/bin:$PATH"* komutu girilir.

DNA Sarmalını Birleştirmek

reads1.fa ve reads2.fa isimli DNA baz çifti dosyaları ve bunların contigi olan ecoli-contigs.fa dosyasını assemble etmek için;

#abyss-pe name=ecoli k=64 in='reads1.fa reads2.fa' komutu girilir. *"in"* parametresine giriş olarak verilen dosyalar FASTA, FASTQ, qseq, export, SRA, SAM veya BAM biçimi veya

bunların *"gz, bz2 veya xz"* ile sıkıştırılmış tar dosyası olabilir. Assemble olmuş contigler *"${name}-contigs.fa"* adıyla saklanacaktır.

Okunan DNA çiftlerinin 1. ve 2. sarmalını tanımak için bunların adlarının sonuna 1 ve 2 eklenmelidir. Okunan çift sarmal DNA dizisi ayrı dosyalarda olmalı veya tek dosyada arası belirgin olmalıdır.

K Parametresini İyileştirmek

K'nın en iyi değerini bulmak için, Abyss'i çoklu çalıştırmalı ve Abyss'in komşuluk istatistiklerini incelemek gereklidir. Aşağıdaki Shell(kabuk) betiği k'nın 20'dan 40 kadar her değeri için Abyss'i çalıştırır.

```
export k
for k in {20..40}; do
   mkdir k$k
   abyss-pe -C k$k name=ecoli in=../reads.fa
done
abyss-fac k*/ecoli-contigs.fa
```

k için varsayılan maksimum değer 64'tür. Bu sınır derleme sırasında *"enable-maxk"* parametresi ile değiştirilebilir. Hafıza kullanımını azaltmak için bu değer 32, arttırmak için bu değer 96 yapılabilir.

Paralel Çalıştırma

Paralel MPI işi için *"abyss-pe"* komutuna *"np"* parametresi ile işlemci sayısı belirtilebilir. Herhangi bir MPI ayarlaması olmaksızın, tek bir makinede çoklu işlemci çekirdeği kullanılabilir. Abyss'i birden çok makinede paralel kullanmak için "mpirun" kullanım sayfasında açıklandığı gibi bir "hostfile" dosyası oluşturulmalıdır.

Abyss'i 8 işlemci çekirdeğinde çalıştırmak için *"mpirun –np 8 abyss-pe"* komutunu çalıştırmayın bunun yerine *"abyss-pe np=8"* komutunu kullanın. *"abyss-pe"* komutu MPI işlemlerini *"mpirun –np 8 ABYSS-P"* komutu gibi başlatacaktır.

Çift sarmallı dizi birleştirme çok işleçli olup fakat tek bir makinede çalıştırılmalıdır. İşleç sayısı j parametresi ile belirtilebilir ve *"np"* parametresi *"j"* parametresinin varsayılan değeridir.

Abyss'i bir sunucu kümesinde (cluster) çalıştırmak

Abyss aşağıdaki sunucu kümesi iş zamanlayıcıları ile entegre çalışabilmektedir.

- SGE (Sun Grid Engine)
- Portable Batch System (PBS)
- Load Sharing Facility (LSF)
- IBM LoadLeveler

Örneğin, k'nın 51 ve 63 arasındaki tek değerlerini 64 işlemcide her bir iş için iş dizisini çalıştırmak için aşağıdaki betik kullanılabilir.

```
mkdir k{51..63}

qsub -N ecoli -pe openmpi 64 -t 51-63:2 \

  <<<'abyss-pe -C k$SGE_TASK_ID in=/data/reads.fa'
```

Abyss Parametreleri

"abyss-pe" sürücü betiği parametreleri;

- a: bir kabarcığın maksimum dallanma sayısı [2]
- b: bir kabarcığın maksimum uzunluğu (bp) [10000]
- c: bir unitig'in kapsadığı minimum k-harfli ortalaması [sqrt(median)]
- d: bir mesafe tahmini izin verilen hata (bp) [6]
- e: minimum aşınmış k-harfli kapsamı [sqrt(median)]
- E: DNA ipliği başına minimum aşınmış k-harfli kapsamı [1]
- j: işleç (thread) sayısı [2]
- k: k-harfli boyutu (bp)
- l: bir okumanın minimum hizalanmış uzunluğu (bp) [k]
- m: iki unitigin minimum örtüşme sayısı (bp) [30]
- n: kontigleri inşa etmek için gerekli minimum baz çifti sayısı [10]
- N: iskeleyi(scaffold) inşa etmek için gerekli minimum baz çifti sayısı [n]
- p: bir kabarcığı tanımlamak için minimum dizi(sekans) [0.9]
- q: minimum baz cinsi [3]
- s: kontigleri inşa etmek için gerekli minimum unitig boyutu (bp) [200]

- S: iskeleleri inşa etmek için gerekli minimum unitig boyutu (bp) [s]
- t: minimum tip boyutu (bp) [2k]
- v: tüm loglar için v=-v, detaylı tüm loğlar için v=-vv kullanılır [pasif]

Köşeli parantez içinde yazan değerler abyss-pe için varsayılan değerlerdir.

EK B. IDBA

IDBA de novo assembler(birleştirici)'ın Ağustos 2010 tarihi itibariyle son çıkan IDBA 0.17 versiyonu http://hku-idba.googlecode.com/files/idba-0.17.tar.gz (hku-idba.google.com web adresi) adresinden mevcut Ubuntu 14.04.1 LTS üzerine aşağıdaki komutlarla kurulur (code.google.com web adresi).

#wget http://hku-idba.googlecode.com/files/idba-0.17.tar.gz

#tar xzvf idba-0.17.tar.gz

#ln -s idba-0.17.tar.gz idba

#cd idba

#./configure

#make

IDBA 0.17 aracının kullanımı aşağıdaki gibidir.

idba --read okunacak-dosya [--output out] [secenekler]

Kullanılabilen seçenekler;

-h [--help], yardım mesajı verir

-r [--read], okunacak baz dosyasını belirtmek için parametre

-o [--output], çıktının öneki için parametre

--scaffold, kontigleri birleştirmek için baz çifti bilgisini kullanmada kullanılan parametre

--mink, minimum k-harfli değeri parametresi (=25)

--maxk, maksimum k-harfli değeri parametresi (=50)

--minCount, her k-harfli için kullanılan filtreleme eşiği parametresi (=2)

--cover, kontigler için kapsanan kesme sayısı parametresi (=0)

--trim, her okumanın sonunda gözardı edilen "A" bazı sayısı (=0)

--minPairs, 2 kontigi birleştirmek için baz çifti bağlantılarının minimum sayısı parametresi (=5)

--minLength, çıkış için minimum kontig uzunluğu (=100)

Parantez içinde verilen değerler varsayılan değerlerdir.

Çıkış kontigleri *"out-contig.fa"* ve *"out-contig-mate.fa"* dosyalarında bulunacaktır. İlk önce tek bazlı çıktısı sonrada çift bazlı çıktısı oluşacaktır. Eğer çift-bazlı versiyonu çalıştırmak istiyorsanız *"—scaffold"* parametresini kullanmalısınız.

Basit bir veri için örnek aşağıdaki gibidir.

 # bin/idba --read Lacto.reads-mate-30-0.01-75 --output lacto

IDBA çift sarmallı baz okumalarının işlenebilmesi için çift sarmallı bazın iki sarmalı aynı dosyada olmalı ve bunlar ardışık iki satırda verilmelidir. Böyle değilse iki ayrı dosyayı tek dosyada birleştirmek için "mergeReads" komutu aşağıdaki gibi kullanılabilir.

 # bin/mergeReads read-file1 read-file2 merge-read-file

Veri fastq formatında ise ilk önce "fq2fa" aracıyla aşağıdaki gibi dönüştürülmelidir.

 # bin/fq2fa fq-file fa-file

Aynı uzunlukta okumalar tercih edilmekte olup, okumaları normalize etmek için *"normReads"* aracı aşağıdaki şekilde kullanılabilir.

 # bin/normReads read-file output-read-file --length l –mate

EK C. Velvet

Velvet de novo assembler(birleştirici)'ın Nisan 2011 tarihi itibariyle son çıkan Velvet1.2.10 versiyonunu https://www.ebi.ac.uk/~zerbino/velvet/velvet_1.2.10.tgz (ebi.ac.uk web adresi, (a)) adresinden indirebiliriz. Velvet assembler(birleştirici) herhangi bir standart 64bit GCC derleyicili Linux işletim sistemine yüklenebilir. Sunucunun en az 12 GB belleğe sahip olması tavsiye edilmektedir. Teoride 32 bit sistemlere yüklenebilmesine rağmen işlem yapmak için kullanılan fazla bellek gereksinimi ve 32 bit sistemin bellek kısıtlamalarından dolayı tavsiye edilmez. 64 bit Ubuntu 14.04.1 LTS üzerine aşağıdaki komutlarla yüklenebilir (ebi.ac.uk web adresi, (b)).

Velvet'in Kurulumu

#wget https://www.ebi.ac.uk/~zerbino/velvet/velvet_1.2.10.tgz

#tar xzvf velvet_1.2.10.tgz

#ln -s velvet_1.2.10 velvet

#cd velvet

#make

Velvet'in colorspace versiyonu için *"make color"* komutu kullanılmalıdır. Bu komut velveth_de uygulaması hariç diğer tüm uygulama komutları için geçerlidir. Colorspace ve sekans uzayı versiyonları uyumsuz olduğundan uygulama kümeleri ayrı ayrıdır. Başka bir ifadeyle anlamsız sonuçlar almak istenmiyorsa colorspace velvet ile sekans (dizi) dosyaları karılmamalıdır.

Sabit uzunlukta diziler kullanıldığından, değişkenlerin sayısı derleme zamanında ayarlanmak zorundadır. Bağımsız olarak ele alınabilen okuma kategorileri veya kanal sayıları birdir. Bu örneğin farklı eklenmiş kütüphaneler veya beraber olan farklı örneklerin okumalarını ayrıştırmak için kullanışlıdır.

Varsayılan olarak sadece 2 adet kısa okuma kategorisi vardır. Fakat bu değişken farklı ihtiyaca göre genişletilebilir. Örneğin 57 farklı kanal elde etmek için Velvet *"make 'CATEGORIES=57'"* parametresi eklenerek derlenmelidir. Açıkça daha büyük sayıda kategori daha uzun dizilere karşılık gelir ve Velvet'i çalıştırmak için daha fazla belleğe ihtiyaç duyulur. Bellek gereksinimi ve ihtiyaca göre CATEGORIES parametresi ayarlanabilir.

Bir diğer kullanışlı parametre MAXKMERLENGTH parametresidir. Optimal gen yerleşimini yapmak için bu parametre hayatidir. Bu parametre veri kümesine bağlı olarak istediğiniz uzunlukta olabilir. Varsayılan olarak bu parametre 31 bp ile sınırlandırılmıştır. Fakat derleme

anında bu parametre *"make 'MAXKMERLENGTH=57'"* komutu ile ayarlanarak limit artırılabilir. Daha uzun harfli düğümler için Velvet daha fazla belleğe ihtiyaç duymaktadır.

Okuma ID'leri 32-bit işaretli integer olarak saklanmaktadır. Eğer 2.2 milyar okumadan daha fazla bir gen yerleşimi yapılacaksa bunu yapmak için derleme anında *"make 'BIGASSEMBLY=1'"* komutu kullanılmalıdır. Bu aşırı miktarda belleğe gereksinim duyacaktır.

Okuma uzunları 16-bit işaretli integer olarak saklanmaktadır. Eğer 32kb'den daha uzun kontigler assemble edilimek istenirse bunu yapmak için derleme anında *"make 'LONGSEQUENCES=1'"* komutu kullanılmalıdır. Bu işlem de aşırı bellek kullanacaktır.

Velvet'i çok çekirdekli CPU ile paralel çalışmaktadır fakat bu şekil kullanım bellek kullanımını etkilemeyecektir. Derleme anında *"make 'OPENMP=1'"* komutu çalıştırılırsa Velvet programı aynı anda tüm CPU çekirdeklerini kullanacaktır. Bu şekil çalıştırmada çalışma zamanı olarak lineer bir iyileşme beklenmemelidir.

Varsayılan olarak zlib yüklü bir sistemde Velvet bunu kullanır. Eğer zlib sistemde yüklü değilse "make 'BUNDLEDZLIB=1'" komutu ile Velvet derlenirse zlib destekli olacaktır.

Velvet'i Çalıştırma

Velveth komutu

Velveth komutu velvetg ve sistemin işaret ettiğin her bir sekans dosyası için veri kümesinin inşa edilmesine yardımcı olan bir komuttur. Eğer komutun kullanımı unutulursa "> ./velveth" komutu ile yardımcı mesajı alınabilir. Velveth sekans dosyalarının sayısın alır, hash tablosunu üretir sonrasında çıkış dizinine velvetg'nin ihtiyaç duyacağı 2 adet çıkış dosyası (Sequences ve Roadmaps dosyası) oluşturur. Bu komutun sözdizimi aşağıdaki gibidir.

> ./velveth output_directory hash_length [[-file_format][-read_type] filename]

Hash uzunluğu (hash length) k-harfli uzunluğu olarak bilinen baz çiftlerinin uzunluğuna karşılık gelir.

Desteklenen dosya formatları şunlardır; **fasta (varsayılan), fastq, fasta.gz, fastq.gz, sam, bam, eland, gerald**

Okuma kategorileri şunlardır; **short** (varsayılan)**, shortPaired, short2** (short ile aynı fakat ayrı bir ekleme-boyutu kütüphanesi için), **shortPaired2, long** (Sanger, 454 veya referans sekanslar ile), **longPaired**

Özetlenirse, seçenekler stabildir. Başka bir ifadeyle diğer operatörlerle kısıtlanmadığı süre seçenekler gerçektir. Bu birden çok dosya adını yeniden açıklayıcı kullanmaksızın yazılmasına izin verir. Aşağıdaki örnek incelenirse;

> ./velveth output_directory/ 21 -fasta -short solexa1.fa solexa2.fa solexa3.fa -long

capillary.fa

Bu örnekte tüm dosyalar FASTA formatıdır sadece okuma kategorisi değişiktir. Ancak "fasta" ve "short" varsayılan olduğundan yukarıdaki komut aşağıdaki gibi komut gibi yazılabilir.

> ./velveth output_directory/ 21 solexa.fa -long capillary.fa*

Velvet'te Piping

Yeni bir sekans dosyasını oluşturma ve okuma maliyeti olmaksızın başka bir programın çıkışı üzerinden doğrudan velveth komutu çalıştırılmak istenirse bu pipe ile yapılabilir. Bu durumda komut satırında '-' parametresi kullanılır. Aşağıdaki komut yerine

> myprogram > selected_reads.fa

> velveth directory 21 -fastq other_reads.fastq -fasta selected_reads

> velvetg directory (... parameters...)

Aşağıdaki komut kullanılabilir.

> myprogram | velveth directory 21 -fastq other_reads.fastq -fasta -

> velvetg directory (... parameters...)

Eğer DNA iplikçiği özel transkriptom sekanslama protokolü kullanılmak istenirse bu parametre daha iyi sonuçlar verebilir. Aşağıdaki komut ile bu yapılabilir.

> ./velveth output_directory/ 21 (...data files...) -strand_specific

Velvet çeşitli k-harfli uzunlukları için verimli kullanılmak istenirse gereksiz hesaplamalar olmaksızın tüm k-harfli uzunlukları için aşağıdaki komut kullanılabilir.

> ./velveth output_directory/ m,M,s (..data files..)

Burada s'nin bir adımı k, $m \leq k < M$ aralığındadır.

Sıklıkla hash çalıştırılmadan önce sekans dosyasının ön-işlemi yapılabilir. Bu aşamada Velvet basitçe giriş dosyalarını okur ve bunların tümünü içeren bir dosya hazırlar. Bu işlem çok basit bir bilgisayar ile yapılabilir ve bu yüzden güçlü bir bilgisayarın kaynakları israf edilmez. Bu işlemi yapmak için "-noHash" parametresi kullanılır. Örnek komut aşağıdaki gibidir.

> ./velveth output_directory/ 21 (..data files..) –noHash

Büyük veri kümelerinde Velvet kullanılmak istenirse, bilgisayar sekansları diskten veya ağ aracılığıyla okurken çok fazla zamanı okuma erişimine harcayacaktır. Binary (ikili) sekans dosyaları ile çalışılarak Velvet önemli derecede hızlandırılabilir. Bu parametre hashing safhasında velvetg'ye göre ayarlanarak çağrılabilir. Örnek komut aşağıdadır.

> ./velveth output_directory/ 21 -create_binary (..data files..)

Velvetg komutu

Velvetg komutu De Bruijn graflarının inşa edilip, manipüle edildiği Velvet'in ana komutudur. Gereksiz yeniden hesaplamalardan kaçınmak için işlem boyunca velvetg bazı dosyaları kaydetmesine rağmen parametreleri kaydetmez. Komut aşağıdaki gibi kullanılabileceği gibi;

> ./velvetg output_directory/ -cov_cutoff 4

> ./velvetg output_directory/ -min_contig_lgth 100

Farklı olarak aşağıdaki gibi de kullanılabilir.

> ./velvetg output_directory/ -cov_cutoff 4 -min_contig_lgth 100

Bu da hesaplamaları yeniden yapmaksızın rahatça parametrelerle oynamaya imkân verir. Aşağıdaki komutlar incelenebilir.

> ./velvetg output_directory/ -cov_cutoff 4

> ./velvetg output_directory/ -cov_cutoff 3.8

> ./velvetg output_directory/ -cov_cutoff 7

> ./velvetg output_directory/ -cov_cutoff 10

> ./velvetg output_directory/ -cov_cutoff 2

Öte yandan tek bir velvet komutunda parametrelerin sırasının önemi yoktur. Sonuç olarak komuttan emin olunmazsa yardım mesajı görüntülemek için aşağıdaki komutu girilebilir.

> ./velvetg

Tekli okumalar

Başlangıç olarak aşağıdaki çalıştırılırsa;

> ./velvetg output_directory/

Bu komut kontigin bir fasta dosyasını ve birkaç istatistik dosyası oluşturacaktır. Çoğu kısa-okumalarda tecrübeler gösteriyor ki başlangıç düzeltmesinden düşük-kapsamlı düğümler kalmaktadır. İstenilen bir kapsama için kesme değeri (cutoff) verilebilir.

> ./velvetg output_directory/ -cov_cutoff 5.2

Öte taraftan gen yerleşiminde yüksek kapsamlı veri (plasmid, mitokondriyal, kloroplast sekansları) hariç tutulmak istenirse maksimum kapsama kesme değeri kullanılabilir.

> ./velvetg output_directory/ -max_coverage 300 (... other parameters ...)

Bu kapsama kesme değeri otomatik olarak uzunluğun yarısına (ağırlıklı orta kontig kapsama derinliği) ayarlanabilir. Bu parametre daha ileri iterasyonlarla iyileştirilebilir. Bu seçenekle, komutu ilk çalıştırmada hızlı bir şekilde uygun bir assembly (gen yerleşimi) elde edilir. Komut aşağıdaki gibidir.

> ./velvetg output_directory/ -cov_cutoff auto

Uzun okumaları eklemek

Velveth'daki gibi uzun okumalar işaretlenmelidir. Eğer yeterli bir kapsamaya sahip kısa-okumalar varsa, herhangi miktarda uzun okumalar(açıkçası daha derin kapsama ve daha uzun okumalar, daha iyidir) etkili bir biçimde tekrarları çözmek için kullanılabilir. Bu işlemi yapmak için Velvet, ortak sekansın kısa-okumasındaki beklenen kapsamın makul bir tahminine ihtiyaç duymaktadır. Bu değeri elde etmek için en kolay yol basitçe kontiglerin kapsamalarının dağılımlarını gözlemek ve düğümlerin hangi kapsama değerlerini temin ettiğini görmektir. Beklenen kapsama değerini 19x varsayılırsa, bu aşağıdaki komutla kullanılabilir.

> ./velvetg output_directory/ -exp_cov 19 (... other parameters ...)

Eğer uygun biçimde bir örnek üzerinden geçerli nedenler varsa, Velvet size tahmini bir değer verecektir. Aşağıdaki komutla bu istenebilir.

> ./velvetg output_directory/ -exp_cov auto (... other parameters ...)

ÖZGEÇMİŞ

Ad Soyad:	İrfan KILIÇ
Doğum Yeri ve Tarihi:	Elazığ / 01.12.1981
Adres:	Fırat Üniversitesi Rektörlük Kampüsü Bilgi İşlem Daire Başkanlığı Merkez / Elazığ
E-Posta:	ikilic23@gmail.com
Lisans:	Fırat Üniversitesi Mühendislik Fakültesi Bilgisayar Mühendisliği

Mesleki Deneyim ve Özellikler:

- Bilgisayar Mühendisi, Kafkas Üniversitesi (2004-2013)
- Bilgisayar Mühendisi, Fırat Üniversitesi (2013-Devam ediyor)

Yayın Listesi:

A. Uluslararası Toplantıda Sunularak Tam Metin Olarak Yayınlanan Bildiri

1. Ertam F., Kılıç İ., Adli Bilişim Vakası Öncesinde Ağa Yapılan Saldırı ve Anormallik Tespiti: Bir Olay İncelemesi, 1st International Symposium on Digital Forensics and Security (ISDFS'13), 20-21 May 2013, Elazığ, Turkey

B. Ulusal Toplantıda Sunularak Tam Metin Olarak Yayımlanan Bildiri

1. Oğul R., Tamer İ., Kılıç İ., Kağızman'da İnternet Altyapı Durumu ve Bilgisayar Kullanım Değerlerinin Araştırılması, Geçmişten Geleceğe Her Yönüyle Kağızman Sempozyumu, Kafkas Üniversitesi, Kağızman / Kars, 2012
2. Kılıç İ., Karcı A., DNA Dizilerinin De Bruijn Grafları ile İncelenmesi Eleco 2012, Bursa, 2012

C. Ulusal Toplantıda Sunularak Özet Metin Olarak Yayımlanan Bildiri

1. Kılıç İ., PHP CodeIgniter Framework, Akademik Bilişim 2011, İnönü Üniversitesi, Malatya