Penggunaan Algoritma *String Matching* untuk Mendeteksi *Genomic repeats*

Kevin Yauris - 13513036
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
Kevin_Yauris@itb.ac.id

Tujuan algoritma *string matching* adalah untuk mendapatkan kemunculan suatu pattern pada sebuah text. Teknik ini juga telah digunakan pada bidang analisis DNA, salah satunya untuk menentukan apakah ada kelainan pada DNA seseorang, dengan cara menghitung *genomic repeats* yang ada di dalam DNA seseorang. *Genomic repeats* yang terjadi melewati batas mengindikasikan suatu kelainan genetik. Algoritma *string matching* berperan penting dalam analisa DNA ini, sehingga semakin hari semakin banyak orang yang mencoba membuat *string matching* yang efektif untuk melakukan analisa DNA. Dalam makalah ini akan dibahas mengenai bagaiman suatu tahap kelainan genetika dideteksi melalui *genomic repeats* dan peran algoritma *string matching* dalam pendetektisian *genomic repeats* disertai beberapa contoh sederhananya.

***Index Terms—genomic repeats*, kelainan genetik, genetika, DNA, *string matching*.**

I. Pendahuluan

Di era modern sekarang ini, dimana setiap orang dituntut untuk bekerja secara cepat. Kadang kala banyak orang memiliki gaya hidup yang tidak sehat seperti memakan makanan cepat saji, kurang olahraga, tidur tidak teratur, terpapar terlalu banyak radiasi elektromagnetik ataupun terpapar logam saat bekerja. Hal-hal ini memicu terjadinya kelainan genetik yang apabila tidak cepat-cepat ditangani dapat menjadi penyakit yang sulit disembuhkan. Bukan hanya berbahaya bagi pemilik kelainan genetik, tetapi juga berbahaya bagi keturunannya nantinya. Untuk mencegah sebuah kelainan genetik menjadi sebuah penyakit genetik, akhir-akhir ini dikembangkan *gen therapy* atau terapi gen, yaitu suatu upaya menginjeksikan DNA atau RNA ke dalam sel seseorang untuk memperbaiki kelainan genetika yang terjadi dalam DNA seseorang. Tetapi untuk dapat melakukan suatu terapi gen, harus dilakukan analisa terlebih dahulu untuk menemukan kelainan gentika yang tejadi, mengenali jenis kelainan genetik dan penyebab kelainan genetik tersebut terjadi. Salah satu cara mendeteksi adanya suatu kelainan genetika ini adalah dengan cara menganalisa *genomic repeats* yang terjadi dalam DNA seseorang. *Genomic repeats* adalah pengulangan suatu rangkaian basa nitrogen dalam suaut DNA. Untuk dapat melakukan analisa *genomic repeats* diperlukan suatu cara untuk menghitung kemunculan suatu rangkain basa nitrogen. Suatu DNA terdiri dari banyak sekali basa nitrogen, oleh karena itu dibutuhkan cara yang efektif untuk menghitung kemunculan suatu rangkaian basa nitrogen. Banyak cara yang dikembangkan untuk melakukan hal ini salah satunya adalah dengan menggunkan algoritma *string matching*.

String matching adalah teknik pencocokan *string*, dimana suatu suatu *pattern* dicari kemunculannya dalam suatu teks. Dalam pencarian *genomic repeats*, DNA akan dijadikan sebagai suatu teks dan rangkaian asam basa yang akan dicari dijadikan *pattern* yang akan dicari.

Dalam makalah ini akan dijelaskan mengenai bagaimana *String matching* ini digunakan untuk pencarian *genomic repeats* dengan menggunakan algortima *string matching* Boyer-Moore dan algortima *string matching* Knuth-Morris-Pratt.

II. Dasar Teori

2.1 Genom

Genom adalah himpunan materi genetik dalam suatu organisme secara keseluruhan. Jadi genom suatu organisme adalah kumpulan semua gen yang dimiliki oleh organisme[1]. Gen sendiri adalah unit pewarisan sifat bagi organisme hidup. Bentuk fisik dari gen adalah urutan DNA yang menyandi suatu protein, polipeptida, atau seuntai RNA yang memiliki fungsi bagi organisme yang memilikinya.

2.1.2 DNA

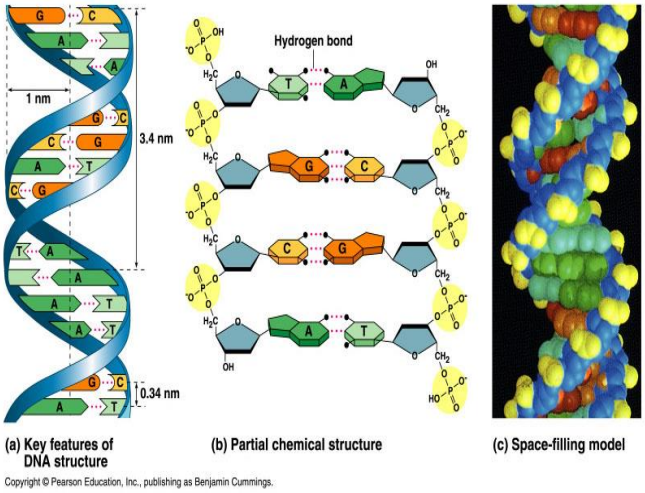
DNA(***d**eoxyribo**n**ucleic **a**cid)* adalah suatu materi genetik yang berada dalam tubuh manusia, tepatnya di dalam inti sel, yang berupa sebuah rantai asam deoksiribonukel. Biomolekul ini menyimpan dan menyandi instruksi-instruksi genetika setiap organisme. Instruksi-instruksi genetika ini berperan penting dalam pertumbuhan, perkembangan, dan fungsi organisme.

DNA merupakan asam nukleat bersamaan dengan protein dan karbohidrat, asam nukleat adalah makromolekul esensial. DNA terdiri dari dua unting biopolimer yang berpilin satu sama lainnya membentuk heliks ganda, yang dikenal sebagai polinukleotida karena keduanya terdiri dari satuan-satuan molekul yang disebut nukleotida. Tiap-tiap nukleotida terdiri atas gila monosakarida, gugus fosfat dan salah satu jenis basa nitrogen yaitu :

- Adenin (A)

- Guanin (G)
- Sitosin (C)
- Timin (T)

Dua unting DNA yang saling berpilin bersifat anti-paralel, yang berarti bahwa keduanya berpasangan secara berlawanan. Masing-masing untaian terdiri atas rangka utama dan basa nitrogen. Kedua untaian pada heliks ganda DNA disatukan oleh ikatan hydrogen Antara basa-basa yang terdapat pada kedua untaian tersebut. Empat basa yang ditemukan pada DNA (Adenin(A), Guanin(G), Sitosin(C), dan Timin(T)) masing-masing akan berikatan dengan pasangannya.



Gambar 1 : Struktur DNA

Hubungan antar basa nitrogen, seperti yang bisa dilihat pada Gambar 1 adalah sebagai berikut :

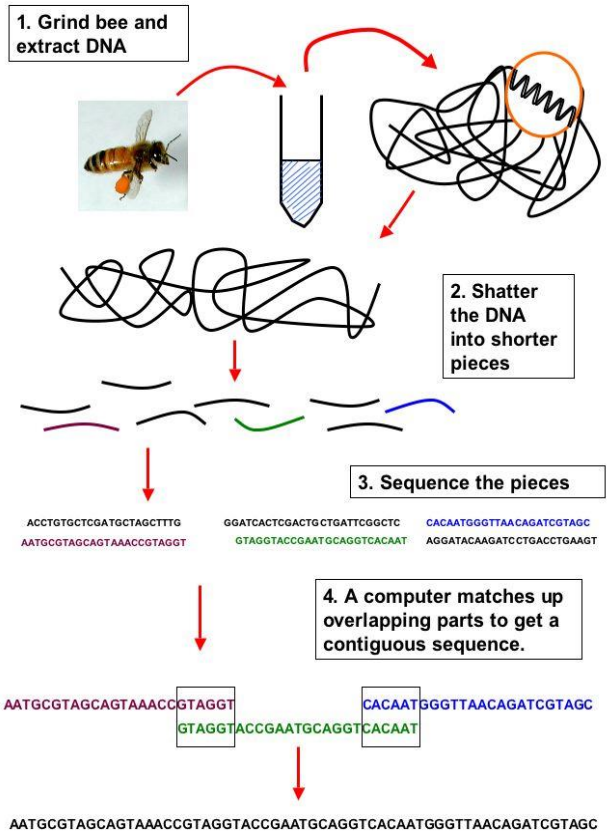
- Adenin(A) >< Timin(T)
- Guanin(G) >< Sitosin (C)

Segmen polipeptida dari DNA disebut gen, yang biasanya merupakan sebuah bentuk dari molekul RNA. Berbeda dengan DNA, struktur gula yang membentuk RNA adalah ribosa.

2.1.2 Sekuens DNA

Informasi biologis seseorang didapatkan dari menganalisis bagian dari makhluk hidup yang membawa informasi genetik, dan informasi ini berada di dalam DNA yang membawa informasi genetik tiap makhluk hidup.Karena DNA merupakan ciri kunci makhluk hidup, pengetahuan mengenai sekuens DNA dapat berguna dalam setiap penelitian.

Sekuens DNA adalah sebuah proses untuk mendapatkan susunan lengkap nukleotida rantai DNA dari suatu makhluk hidup. Sekuens DNA menyandikan informasi yang diperlukan bagi makhluk hidup untuk melangsungkan hidup dan berkembang biak. Pengetahuan akan sekuens DNA berguna untuk mengetahui sekuens asam amino yang disandikan oleh gen. Mengetahui sekuens DNA suatu organisme sangat berguna dalam bidang kedokteran karena dapat digunakan untuk mengidentifikasi, mendiagnosis, dan mengembangkan pengobatan terhadap penyakit genetis. Hampir semua sampel biologis dapat memberikan sebuah salinan DNA yang dapat digunakan dalam proses sekuens DNA.



Gambar 2 : Proses sekuens DNA untuk mendapatkan susunan nukleotida lebah

Ada beberapa metode yang digunakan dalam proses sekuens DNA, di antaranya adalah :

- *Maxam-Gilbert sequencing*
  Ditemukan oleh Allan Maxam dan Walter Gilbert pada tahun 1977. Metoda ini didasarkan kepada modifikasi secara kimia terhadap DNA dan pembelahan pada bagian dasar tertentu. Dikenal dengan nama *chemical sequencing*. Metode ini awalnya cukup popular karena dapat langsung menggunakan DNA hasil pemurnian, sedangan metode Sanger pada waktu itu memerlukan kloning untuk membentuk DNA untai tunggal. Sayangnya, kerumintan pengerjaan teknis yang harus dilakukan saat menggunakan metode Maxam-Gilbert yang ini sangat rumit karena menggunakan unsur radioaktif serta kesulitannya untuk mengidentifikasi DNA yang kompleks, maka metode ini mulai ditinggalkan

- *Chain-termination methods (Sanger sequencing)*
  Ditemukan oleh Frederick Sanger pada tahun 1977 Saat ini, hampir semua pengerjaan sekuensing DNA dilakukan dengan menggunakan metode terminasi

rantai. Hal ini karena metode ini relatif muda dan handal. Teknik ini menggunakan terminasi atau penghentian reaksi sintesis DNA menggunakan substrat nukleotida yang telah dimodifikasi. Metode dasar ini sekarang telah dikembangkan menjadi banyak metode lanjut yang lebih handal seperti SOLiD *sequencing, Heliscope single molecule sequencing* dan *SMRT(Single molecule real time) sequencing.*

2.1.3 *Genomic Repeats*

Genomic repeats atau *repeated DNA sequences* adalah suatu keadaan dimana suatu *pattern* dari asam nukleat ditemukan berulang kali dalam suatu genom. Suatu *pattern* dari asam nukleat dalam suatu organisme memiliki batas jumlah wajar ditemukan dalam suatu untaian DNA yang sama. Jika suatu *pattern* dari asam nukleat ditemukan berulang kali dan meleibihi batas jumlah wajar, maka mengindikasikan suatu kelainan genetik. Kelainan genetika yang terjadi tergantung dengan jenis perulangan *pattern* yang terjadi dan jumlah pengulanan *pattern* yang terjadi. Penyebab terjadinya *genomic repeats* ini diantaranya adalah karena terjadinya *genomic rearrangements* yang dapat menyebabkan *genomic disorders.Genomic disorders* ini apabila cukup parah dapat menyebabkan *genomic disorders disease,* seperti cacat mental, *down syndrome*, cacat fisik, ketidakmampuan untuk mengendalikan emosi dan bahkan tumor. Hal ini tidak selalu menyerang pemilik *genomic disorders,* kadang-kadang dampaknya turun kepada turunan dari pemilik gangguan ini. Tetapi dengan kemajuan teknologi saat ini bisa dilakukan terapi gen dengan berbagai macam metode untuk memulihkan kelainan gen yang terjadi ini. Tetapi untuk dapat melakukan terapi gen yang sesuai harus dapat diidentifikasi terlebih dahulu apakah seseorang mengalami *genomic disorders* yang serius dan perlu ditangani atau tidak dan jenis *genomic disorders* seperti apa yang dialami oleh orang itu. Penanganan yang cepat dan tepat dapat mengurangi kerusakan lebih jauh kepada pemilik *genomoc disorders* ini maupun keturunannya nanti.

2.2 Pencocokan *String (String Matching)*

Algoritma pencocokan string bertujuan untuk menentukan letak sebuah pattern di dalam text. Ada banyak algoritma yang telah dikembangkan untuk mencari algoritma apa yang terbaik dalam menyelesaikan permasalahan *string matching* ini, di antaranya adalah :

- Algoritma *Brute Force (Naïve Algorithm)*

  Brute Force akan mencocokkan string di setiap karakter untuk menentukan apakah pattern yang dimasksud terdapat di posisi tersebut

  Dengan algoritma Brute Force, pattern akan maju 1 langkah ke kanan dan mulai mencocokan lagi sampai bertemu dengan karakter yang tidak cocok, pattern yang dimaksud sudah ditemukan, atau pencarian sudah mencapai ujung teks.

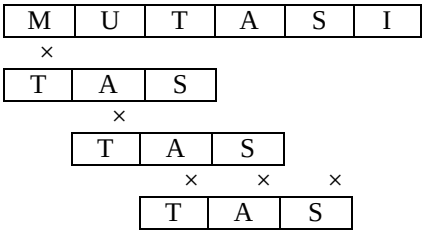
  Kompleksitas waktu terburuk: O(MN)

  Kompleksitas waktu terbaik: O(N)

Pseudocode :

```
do
if (text letter == pattern letter)
compare next letter of pattern to next
letter of text
else
move pattern down text by one letter
  while (entire pattern found or end
of text)
```

Contoh :
Teks = “MUTASI”
Pattern = “TAS”



Jumlah perbandingan : 5

- Algoritma Knuth-Morris-Pratt (KMP)

  Algoritma Knuth-Morris-Pratt (KMP) berbeda dengan algoritma Brute Force karena algoritma ini menyimpan informasi dari hasil perbandingan yang dilakukan sebelumnya untuk menghindari perbandingan yang sia-sia.

  KMP menggunakan menggunakan prefix dan suffix dari pattern untuk mengoptimasi pergeseran pattern dalam pencarian.

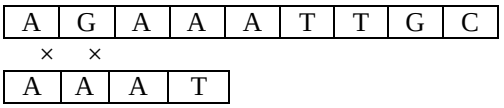
  Kompleksitas waktu total : $\boldsymbol{O}(n + m)$

  Contoh :
  Teks : AGAAATTGC
  Pattern : AAAT

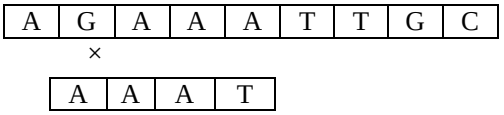
  Tabel Fungsi Pembatas:

p	0	1	2
b(p)	0	1	2

  f(MATCH) = MATCH – b(MATCH-1) untuk MATCH >= 1
  f(MATCH) = 1 untuk MATCH = 0



Pergeseran = f(MATCH) = 1-b(0) = 1



A	G	A	A	A	T	T	G	C

× × × ×

A	A	A	T

Jumlah perbandingan : 7

Pseudocode :

```
KMPMatch(T,P)
Input: Strings T (text) with n
characters and P
(pattern) with m characters.
Output: Starting index of the first
substring of T
matching P, or an indication that P is
not a
substring of T.
f ← KMPFailureFunction(P) {build
failure function}
i ← 0
j ← 0
while i< n do
if P[j] = T[i] then
if j = m - 1 then
return i - m - 1 {a match}
i ← i + 1
j ← j + 1
else if j > 0 then {no match, but we
have advanced}
j ← f(j-1) {j indexes just after
matching prefix in P}
else
i ← i + 1
   return "There is no substring of T
               matching P"
```

- Algoritma *Boyer-Moore*
  Algoritma pencocokan string Boyer Moore didasarkan pada 2 teknik, yaitu :
  1. *Looking-glass technique*
     Teknik ini merupakan cara untuk menemukan sebuah pattern dalam teks dengan memulai pencocokan dari akhir string pattern.
  2. *Character-jump technique*
     Saat terjadi ketidakcocokan, pencarian akan dilanjutkan setelah menggeser pattern sebesar nilai tertentu untuk menghindari pencocokan yang sia-sia

  Contoh :
  Teks : TAAGTAAGC
  Pattern : AAG

T	A	G	G	T	A	A	G	C

× × ×

A	A	G

Pergeseran = 1, karena pattern "AAG" tidak mengandung huruf "T" sehingga pattern dimajukan sebanyak 1 indeks dengan letak indeks pertama pattern terletak di indeks teks+1

T	A	G	G	T	A	A	G	C

× ×

A	A	G

Pergeseran = 1, karena pattern mengandung huruf "G" tapi sudah terlewatkan

T	A	G	G	T	A	A	G	C

×

A	A	G

Pergeseran = 3, karena pattern "AAG" tidak mengandung huruf "T" sehingga pattern dimajukan sebanyak 1 indeks dengan letak indeks pertama pattern terletak di indeks teks+1

T	A	G	G	T	A	A	G	C

× × ×

A	A	G

Jumlah perbandingan : 9

Pseudocode: :

```
BOYER_MOORE_MATCHER (T, P)
Input:    Text with n characters
and Pattern with m characters
Output: Index of the first
substring of T matching P
   Compute function last
   i ← m-1
   j ← m-1
   Repeat
       If P[j] = T[i] then
           if j=0 then
               return i
   // we have a match
           else
               i ← i -1
               j ← j -1
       else
           i ← i + m - Min(j, 1 +
   last[T[i]])
           j ← m -1
   until i > n -1
   Return "no match"
```

III. Perhitungan perulangan suatu genom dengan menggunakan string matching

Untuk dapat menemukan berapa jumlah pengulanan suatu *pattern* asam nukleat perlu dilakukan sedikit perubahan pada algortima KMP ataupun BM, agar tidak langsung berhenti saat *pattern* ditemukan tetapi akan terus mencari *pattern* sampai DNA selesai diperiksa secara keseluruhan. Selain itu keluaran dari algoritma juga harus diganti, bukan hanya satu 1 indeks, tetapi kumpulan indeks dimana *pattern* ditemukan. Setelah mendapat kumpulan indeks dapat diketahui berapa jumlah pengulanan *pattern* yang terjadi pada DNA dan juga jenis *genomic repeats* apa yang terjadi. Hasil perubahan yang dilakukan adalah sebagai berikut (program ditulis dalam bahasa *java*)

```
public class PatternMatching {
   //KMP Algorithm//
  public static ArrayList<Integer> kmpMatch (String text, String pattern){
     String tempText = text.toLowerCase();
      String tempPattern = pattern.toLowerCase();
       int n = tempText.length();
       int m = tempPattern.length();
       int fail[] = computeFail(tempPattern);
       int i = 0;
       int j = 0 ;
      ArrayList<Integer> found = new ArrayList<>();

       while(i < n ){
          if (tempPattern.charAt(j) == tempText.charAt(i)){
             if (j == m -1){
                  found.add(i-m+2); // match
             i++;
             j=0;

                        } else {
                                 i++;
                                 j++;
                        }
          }else if (j > 0 )
                j = fail[j-1];
          else i++;
       }
                                 return found; // nomatch
   }// end of kmpMatch

   //fungsi pembantu KMP Algorithm
   public static int[] computeFail(String pattern){
      int fail[] = new int[pattern.length()];
      fail[0]=0;
      int m = pattern.length();
      int j = 0;
      int i = 1;

      while ( i < m ){
         if (pattern.charAt(j) == pattern.charAt(i)){ // j+1 chars match
            fail[i]=j+1;
            i++;
            j++;
         }else if ( j > 0 ) //j follows matching prefix
            j = fail[j-1];
         else { //no match
            fail[i]=0;
            i++;
         }
      }
      return fail;
   }// end of computeFail()

   public static ArrayList<Integer> bmMatch (String text, String pattern){
                        String tempText = text.toLowerCase();
                        String tempPattern = pattern.toLowerCase();
      int last[] = buildLast(tempPattern);
      int n = tempText.length();
      int m = tempPattern.length();
      int i = m-1;
     ArrayList<Integer> found = new ArrayList<>();

      if(i>n-1) return found;
      int j = m-1;
      do {
         if ( tempPattern.charAt(j)==tempText.charAt(i)){
            if ( j == 0) {

              found.add (i+1);
                                                                  i++;
                                                          }
            else { //looking glass technique
               i--;
               j--;
            }
         }else{ //char jump technique
            while (tempText.charAt(i) > 128){
              i++;
              if (i == tempText.length()){
                return found;
              }
            }
            int lo = last[tempText.charAt(i)];
            i = i +m-Math.min(j,1+lo);
            j=m-1;
         }
      }while(i <= n-1);
      return found;
   }//end of bmMatch()

   public static int[] buildLast(String pattern){
      int last[] = new int[128];
      for (int i=0;i<128;i++) last[i]=-1; //init array
      for (int i=0;i<pattern.length();i++) {
        //if (pattern.charAt(i) < 128){
          last[pattern.charAt(i)]=i;
        //}
      } //init array
      return last;
```

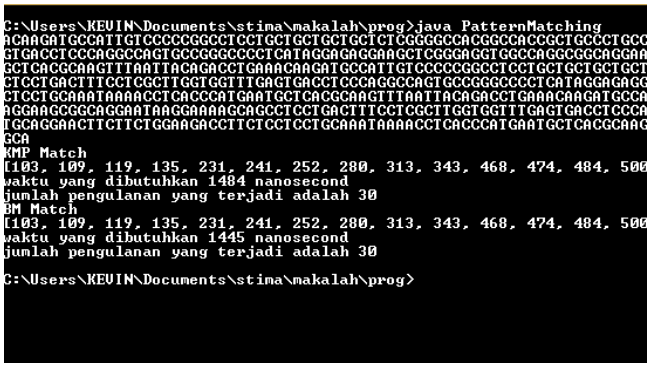
}

Pengujian dilakukan dengan cara memasukkan sebuah rangkaian DNA dalam bentuk *string* yang berlakuk sebagai teks dimana *pattern* akan dicari dan kemudian memasukkan *pattern* berupa kombinasi beberapa basa nitrogen. Program kemudian akan melalukan *string matching* dan menuliskan ke layar berapa kali kemunculan *pattern* tersebut terjadi dan pada indeks mana saja *pattern* tersebut ditemukan.

String matching dilakukan dengan menggunakan algoritma KMP dan BM. Pada setiap kali pencarian selesai dilakukan maka akan ditampilkan juga waktu eksekusi dari tiap algoritma sebagai perbandingan untuk analisa. Pada pengujian yang dilakukan, penulis menggunakan rangkaian DNA sepanjang 1098 karakter yaitu sebagai berikut :

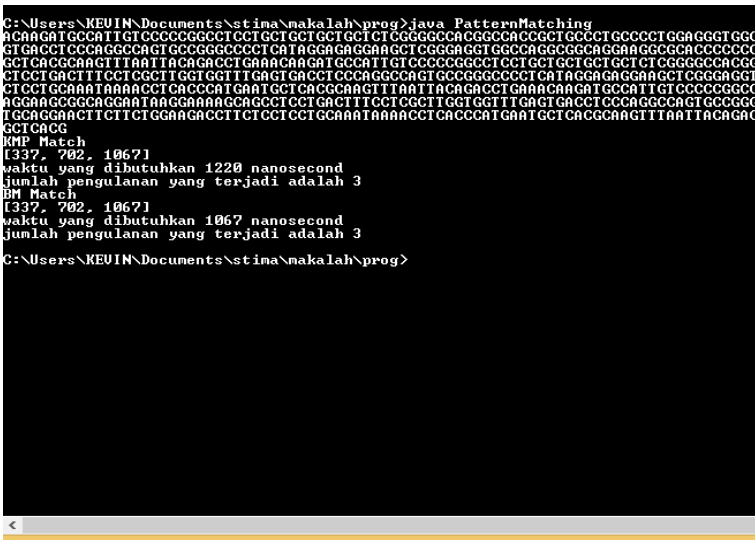
```
ACAAGATGCCATTGTCCCCCGGCCTCCTGCTGCTGCTGCTCTC
GGGGCCACGGCCACCGCTGCCCTGCCCCTGGAGGGTGGCCCCA
CCGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAATAAG
GAAAAGCAGCCTCCTGACTTTCCTCGCTTGGTGGTTTGAGTGA
CCTCCCAGGCCAGTGCCGGGCCCCTCATAGGAGAGGAAGCTCG
GGAGGTGGCCAGGCGGCAGGAAGGCGCACCCCCCCAGCAATCG
CGCGCCGGGACAGAATGCCCTGCAGGAACTTCTTCTGGAAGAC
CTTCTCCTCCTGCAAATAAAACCTCACCCATGAATGCTCACGC
AAGTTTAATTACAGACCTGAAACAAGATGCCATTGTCCCCCGG
CCTCCTGCTGCTGCTGCTCTCGGGGCCACGGCCACCGCTGCCC
TGCCCCTGGAGGGTGGCCCCACCGGCCGAGACAGCGAGCATAT
GCAGGAAGCGGCAGGAATAAGGAAAAGCAGCCTCCTGACTTTC
CTCGCTTGGTGGTTTGAGTGACCTCCCAGGCCAGTGCCGGGCC
CCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAA
GGCGCACCCCCCCAGCAATCGCGCGCCGGGACAGAATGCCCTG
CAGGAACTTCTTCTGGAAGACCTTCTCCTCCTGCAAATAAAAC
CTCACCCATGAATGCTCACGCAAGTTTAATTACAGACCTGAAA
CAAGATGCCATTGTCCCCCGGCCTCCTGCTGCTGCTGCTCTCG
GGGCCACGGCCACCGCTGCCCTGCCCCTGGAGGGTGGCCCCAC
CGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAATAAGG
AAAAGCAGCCTCCTGACTTTCCTCGCTTGGTGGTTTGAGTGAC
CTCCCAGGCCAGTGCCGGGCCCCTCATAGGAGAGGAAGCTCGG
GAGGTGGCCAGGCGGCAGGAAGGCGCACCCCCCCAGCAATCGC
GCGCCGGGACAGAATGCCCTGCAGGAACTTCTTCTGGAAGACC
TTCTCCTCCTGCAAATAAAACCTCACCCATGAATGCTCACGCA
AGTTTAATTACAGACCTGAA
```

Hasil pengujian yang didapatkan adalah sebagai berikut :



Gambar 3 : output program untuk pencarian pattern GCA

Gambar 4 : output program untuk pencarian pattern GCT



Gambar 4 : output program untuk pencarian pattern GCTCACG

IV. Analisis Pencarian *Genomic repeats*

4.1 Analisis *Genomic repeats*

Dari hasil pengujian pertama, ditemukan bahwa terjadi pengulangan rangkaian basa nitrogen GCA sebanyak 30 kali per 1098 rangkaian basa nitrogen. Angka ini tergolong tinggi dan dapat mengindikasikan kelainan genetik seperti *schizophrenia*. Sedangkan untuk pengujian kedua ditemukkan bahwa GCT yang ditemukan 24 kali per 1098 rangkaian basa nitrogen, hasil ini tidaklah mengkhawatirkan karena GCT adalah rangkaian basa nitrogen yang wajar ditemukan berulang kali dan memiliki angka *genomic repeats bound* yang tinggi. Sedangkan untuk rangkaian basa nitrogen GCTCACG ditemukan 3 kali per 1098 rangkaian basa nitrogen, yang merupakan angka yang sangat kecil dan tidak mengindikasikan kelainan genetik apapun.

4.2 Analisis *String matching*

Dari hasil pengujian pertama algoritma KMP memerlukan waktu 1485 nanosecond dan sedangkan algoritma BM memerlukan waktu 1445 nanosecond. Pada

pengujian kedua algoritma KMP memerlukan waktu 1484 nanosecond dan algoritma BM memerlukan waktu 1372 nanosecond. Dan untuk pengujian terakhir algoritma KMP memerlukan waktu 1220 nanosecond dan algoritma BM memerlukan waktu 1067 nanosecond. Dapat dilihat dari ketiga hasil pengujian bahwa algoritma BM memberikan performa yang lebih baik, hal ini karena kemampuan algoritma untuk melompati beberapa karakter sekaligus apabila tidak karakter pada *pattern* kemunculan terakhirnya jauh dari tempat perbandingan terakhir dilakukan. Karena memiliki kemampuan seperti itu, ditambah panjang *String* yang panjang serta karakter yang muncul berulang kali dalam sebuah *String*, membuat algortima BM lebih efektif apabila diterapkan pada rangkaian basa nitrogen pada DNA untuk mendapatkan *genomic repeats*.

V. Kesimpulan

DNA yang membawa informasi mengenai kondisi tubuh makhluk hidup dapat digunakan untuk mendeteksi kelainan genetik yang dapat menyebabkan berbagai macam penyakit ataupun kelainan. Hal ini dapat dilakukan dengan menggunakan peralatan yang mengimplementasikan algoritma *string matching* untuk menghitung perulangan rangkaian basa nitrogen pada DNA yang melebih batas normal. Algoritma string matching akan mengidentifikasi apakah DNA sampel yang diteliti memiliki perulangan rangkaian basa nitrogen yang melebihi batas atau tidak dan pada bagian mana rangkaian basa nitrogen yang berulang itu ditemukan. Dengan mengetahui kedua hal diatas, dapat dilakukan penanganan ataupun pencegahan terhadap kelainan genetika sebelum memberikan dampak yang besar.

Dari dua algoritma *string matching* yang telah dicoba, dapat dilihat bahwa algoritma boyer-moore lebih efektif dalam mendeteksi *genomic repeats* yang terjadi dalam DNA jika dibandingkan dengan algoritma KMP. Hal ini dapat dilihat dari jumlah waktu yang dibutuhkan untuk mengecek perulangan rangkaian basa nitrogen yang terjadi dalam suatu DNA.

VI. Ucapan Terima Kasih

Penulis ingin mengucapkan kepada Tuhan Yang Maha Kuasa karena telah memampukan penulis untuk menyelesaikan makalah ini dengan baik tepat waktu. Penulis juga menyampaikan banyak terima kasih kepada Bapak Rinaldi Munir dan Ibu Nur Ulva Maulidevi yang telah mengajarkan dasar-dasar teori yang diperlikan penulis untuk menyelesaikan makalah ini. Penulis juga berterima kasih kepada kedua dosen, karena telah mengajari dan membimbing penulis selama satu semester ini serta memberikan banyak ilmu kepada penulis.

References

[1] https://18bios1unsoed.wordpress.com/pokok-bahasan/genom-organisme/pengertian-genom/ -- diakses pada tanggal 01 Mei 2015

[2] "The DNA sequence and biological annotation of human chromosome 1". *Nature* **441** (7091): 315–21

[3] Maxam AM, Gilbert W (February 1977). "A new method for sequencing DNA". Proc. Natl. Acad. Sci. U.S.A. 74 (2): 560–4.Bibcode:1977PNAS...74..560M.doi:10.1073/pnas.74.2.560. PMC 392330.PMID 265521.

[4] Gilbert, W. DNA sequencing and gene structure. Nobel lecture, 8 December 1980.

[5] http://neuromuscular.wustl.edu/mother/dnarep.htm –diakses tanggal 2 Mei 2015

[6] http://www.discoverymedicine.com/Anthony-J-Hannan/2010/10/08/trping-up-the-genome-tandem-repeat-polymorphisms-as-dynamic-sources-of-genetic-variability-in-health-and-disease/ -- diakses tanggal 2 Mei 2015

[7] Munir, Rinaldi.*Strategi Algoritma*.Informatika Bandung, 2009

[8] http://www.bangngangan.com/wallpaper/36073/dna-double-helix-html -- diakses pada tanggal 2 Mei 2015

[9] http://jjco.oxfordjournals.org/content/32/suppl_1/S17.full -- diakses pada tanggal 2 Mei 2015

[10] www.bioalgorithms.info – diakses pada tanggal 1 Mei 2015

Pernyataan

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 17 Mei 2014

Kevin Yauris
13513036