Teknik Pencocokan Pola dalam Bidang *Bioinformatics*

Yeksadiningrat Al Valentino (13514055)
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13514055@std.stei.itb.ac.id

***Abstract*—Teknik pencocokan pola sudah cukup banyak menyelesaikan masalah-masalah di bidang komputer sains seperti kompresi dan enkripsi data. Sekarang pencocokan pula juga digunakan dalam bidang informatik terutama pada pencarian sekuens DNA. Tujuan dari tulisan ini adalah menunjukkan bagaimana teknik pencocokan pola menyelesaikan beberapa masalah yang ada di bidang Bioinformatics.**

Keywords—Biology, informatics, bioinformatics, kmp, boyer-moore, string matching, pattern matching, DNA

I. PENDAHULUAN

Pencocokan pola adalah salah satu teknk yang cukup sering digunakan di bidang komputer sains. Banyak sekali hal yang dapat dilakukan oleh teknik ini seperti *voice recognition, handwriting recognition, object recognition* dan lain sebagainya.

Tidak hanya di komputer sains, teknik pencocokan pola juga dapat digunakan di bidang-bidang lain dan pada tulisan kali ini akan ditunjukkan kegunaan dari teknik pencocokan pola di bidang biologi.

Diketahui pada makhluk hidup terdapat sekuens dari DNA yang dapat direpresentasikan sebagai kumpulan dari karakter. Jika ada suatu keadaan dimana suatu pola terjadi pengulangan secara berkali-kali diatas batas normal maka terdapat keanehan pada makhluk tersebut, hal ini dapat lebih cepat ditemukan dengan bantuan dari teknik informatika lebih spesifik lagi dengan pencocokan pola.

Selain untuk mencari pola yang berulang pada DNA, pencocokan pola juga dapat digunakan dalam *Polymerase Chain Reaction* (PCR) yang digunakan oleh para peneliti untuk menggandakan DNA secara spesifik.

II. DASAR TEORI

A. *Bioinformatics*

Bioinformatika adalah bagaimana teknik dari bidang informatika membantu mengerti dan mengorganisir informasi pada bidang biologi terutama pada makromolekul.

DNA

Deoxyribonucleic acid (DNA) adalah molekul yang mengandung intruksi biologis yang membuat masing masing spesies unik, dimana intruksi tersebut selalu diturunkan dari orang tua ke anak-anaknya.

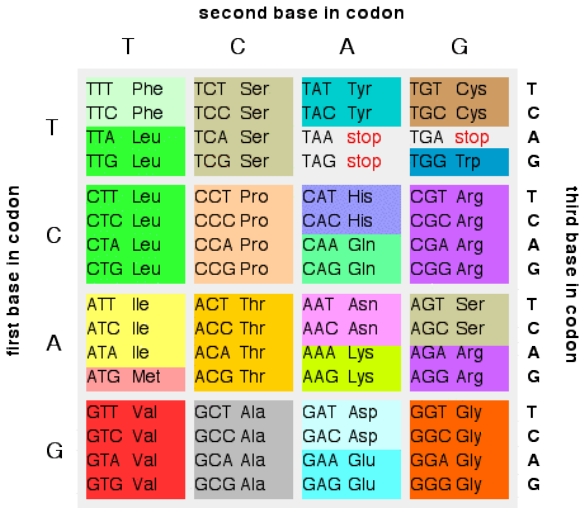
DNA dapat ditemukan di tempat khusus di dalam sel yang bernama nucleus dan karena sel sangat kecil dan organisme memiliki banyak molekul DNA per sel, setiap molekul DNA di kemas dalam kromosom.

Nukleotida adalah molekul yang membentuk DNA. Molekul ini terbagi menjadi 3 bagian yaitu bagian fosfat, bagian gula dan salah satu dari 4 tipe basa nitrogen. Keempat tipe basa nitrogen yang dapat ditemukan di nukleotida adalah adenine (A), sitosin (C), guanine (G), timin (T).

Sekuens DNA adalah proses untuk mendapatkan susunan lengkap nukleotida rantai DNA. Intruksi yang dikandung oleh DNA diperlukan oleh organisme untuk tumbuh, bertahan hidup dan berkembang biak. Untuk menjalankan fungsi tersebut sekuens DNA harus dikonversi menjadi pesan yang dapat digunakan untuk menghasilkan protein, dan proteinlah yang melakukan hampir setiap pekerjaan di tubuh kita.

Setiap sekuens DNA yang mengandung intruksi untuk membuat protein disebut sebagai gen. Ukuran dari setiap gen sangat bervariasi. Gen hanya membentuk sekitar 1 persen dari DNA sekuens. DNA sekuens selain dari 1 persen ini terlibat dalam kapan, bagaimana dan seberapa banyak protein harus dibuat.

Instruksi dari DNA digunakan untuk membuat protein dalam proses 2 tahap. Pertama enzim membaca informasi di dalam molekul DNA dan menuliskannya menjadi sebuah molekul perantara yang bernama *messenger ribonucleic acid* atau biasa disebut mRNA. Selanjutnya informasi yang terkandung dalam mRNA diterjemahkan menjadi ‘bahasa’ dari asam amino yang dimana terdiri dari beberapa blok protein. Bahasa ini yang memberitahu pembuat protein gugus asam amino mana yang harus disambungkan untuk membuat sebuah protein yang spesifik. Ada 20 gugus asam amino yang dapat membuat bermacam-macam jenis protein

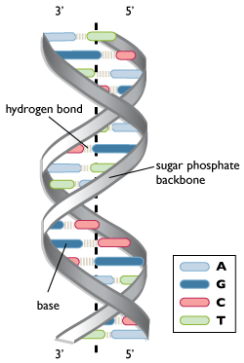


Gambar 1. Tabel Gugus Asam Amino 1

Sumber :

http://www.chemguide.co.uk/organicprops/aminoacids/dnacode.gif (diakses tgl 8 Mei 2016 10:07)

Peneliti menggunakan istilah *double helix* untuk mendeskripsikan lekukan DNA. Untuk mengerti *double helix* dari sudut pandang kimia bayangkan sisi pinggir dari tangga sebagai untaian dari gula dan fosfat dan pada tengah tangganya adalah dua basa nitrogen berpasangan dengan ikatan hidrogen. Struktur unik dari DNA ini memungkinkan dia untuk menggandakan dirinya sendiri saat pembelahan sel. Saat sell bersiap untuk membelah helix dari DNA membelah pada bagian tengah menjadi dua untaian, masing-masing untaian ini yang bertanggung jawab sebagai template untuk kedua DNA baru.



Gambar 2. Struktur DNA

Sumber :

http://cyberbridge.mcb.harvard.edu/images/dna1_7.png (dikases 8 Mei 2016 pukul 12.40)

DNA Polymerase

DNA Polymerase adalah enzim yang membuat molekul DNA dengan cara menyusun nukleotida, blok bangunan dari DNA. Enzim ini penting untuk mereplikasi DNA dan biasanya bekerja berpasangan untuk membuat 2 untaian DNA yang sama persisdari satu molekul DNA originalnya.

Polymerase Chain Reaction

PCR adalah cara yang dikembangkan oleh Kary Mullis pada tahun 1980 untuk mensistensis untaian baru dari DNA komplemen dari untaian DNA template menggunakan kemampuan DNA polymerase. Karena DNA polymerase hanya bisa menambahkan nukleotida hanya kepada grup 3'-OH yang sudah ada saja maka dibutuhkan primer sebagai nukleotida pertama.

Primer adalah sebuah sekuens DNA yang digunakan dalam PCR untuk mengidentifikasi lokasi dari sekuens DNA yang akan di gandakan.

B. Pattern Matching/String Matching

Inti dari teknik pencocokan pola atau pencocokan string adalah sebagai berikut: diberikan sebuah text dan sebuah pola, dan tentukan apakah pola tersebut ada di text. Ada beberapa algoritma yang sudah dikembangkan untuk masalah ini dengan kelebihan dan kekurangannya dan kompleksitasnya sendiri-sendiri, akan dibahas beberapa diantaranya :

Brute Force (*Naïve Algorithm)*

Brute force akan mencocokkan setiap karakter satu persatu dimulai dari kiri. Apabila karakter sama maka geser kanan di text begitu juga di pattern, apabila ada yang salah maka kembali ke huruf yang pertama kali sama dan geser kanan pada text namun pada pattern kembali ke huruf pertama, apabila sudah ada yang cocok dengan pattern masukkan index awal text yang cocok dengan pattern ke dalam array solusi. Apabila sudah sampai ujung pada text dan masih tidak ada yang cocok dengan pattern maka solusi tidak ditemukan.

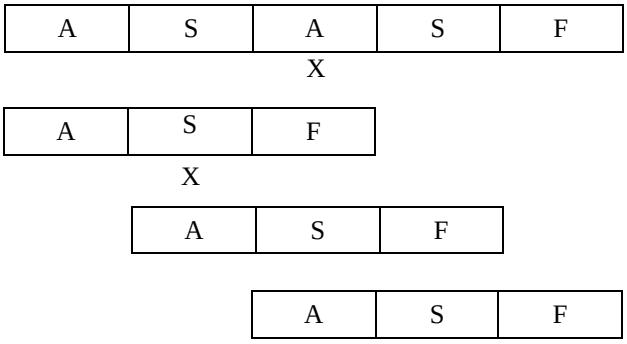
Kompleksitas waktu terbaik bruteforce : O(n)

Kompleksitas waktu terburuk bruteforce : O(mn)

Pseudocode

```
function brute_force(text[],pattern[]{
  //let n be the size of text and m the size
of the pattern

 for (i = 0 ; i < n ; i++){
    for(j = 0 ; j < m && i + j < n ; j++)
       if(text[i + j] != pattern[j]) break
       //if mismatch break loop
       if(j == m) //match found
 }
}
```

Contoh :
Text = ‘asasf’
Pattern = ‘asf’



Jumlah Perbandingan : 7

KMP(Knuth-Morris-Pratt)

Ide utama dari KMP adalah kita bisa menidentifikasi posisi mulai dari masing masing karakter dari pattern agar tidak terjadi perbandingan yang sia sia.

Ide utama dari KMP adalah sebagai berikut, misalkan terdapat sebuah string :

A B A B A C

Sebagai pattern, lalu kita list semua dari prefixnya

0 /empty string/

1 A

2 A B

3 A B A

4 A B A B

5 A B A B A

6 A B A B A C

Lalu sekarang ktia list bagaimana list string tersebut(prefix) adalah suffix terpanjang(dari string tersebut juga) sekaligus prefixnya juga..

0 /empty string/

1 /empty string/

2 /empty string/

3 A

4 A B

5 A B A

6 /empty string/

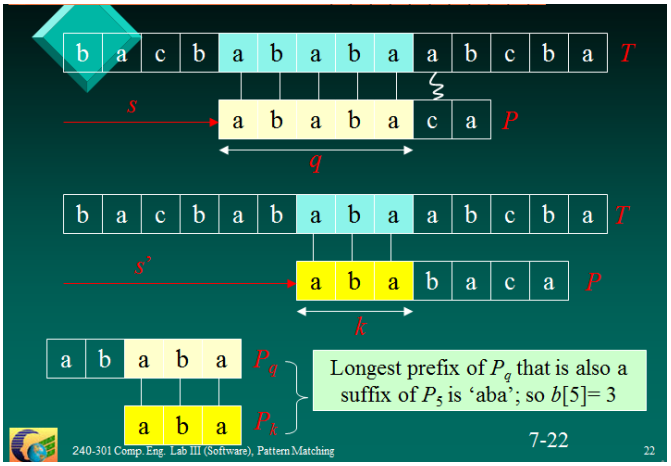
Hal diatas adalah yang biasa disebut KMP *failure function* yang akan digunakan nanti saat pencocokan pola dengan KMP

Pseudocode dari mebuat failure function

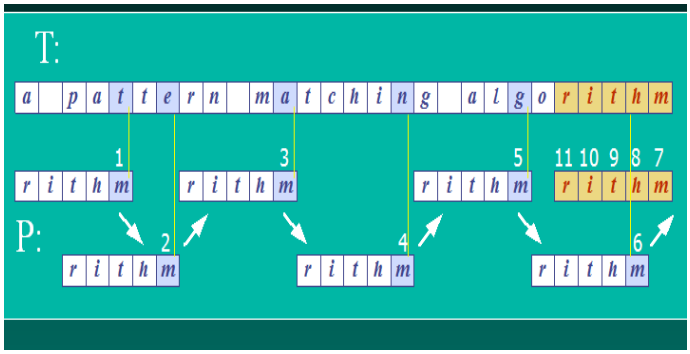
```
function build_failure_function(pattern[])
{
  // let m be the length of the pattern

  F[0] = F[1] = 0; // always true

  for(i = 2; i <= m; i++) {
    // j is the index of the largest next
partial match
    // (the largest suffix/prefix) of the
string under
    // index i - 1
    j = F[i - 1];
    for( ; ; ) {
      // check to see if the last character
of string i -
      // - pattern[i - 1] "expands" the
current "candidate"
      // best partial match - the prefix
under index j
      if(pattern[j] == pattern[i - 1]) {
        F[i] = j + 1; break;
      }
      // if we cannot "expand" even the
empty string
      if(j == 0) { F[i] = 0; break; }
      // else go to the next best
"candidate" partial match
      j = F[j];
    }
  }
}
```

Sedangkan pseudocode dari KMP sendirinya adalah

```
function Knuth_Morris_Pratt(text[],
pattern[])
{
  // let n be the size of the text, m the
  // size of the pattern, and F[] - the
  // "failure function"
  build_failure_function(pattern[]);

  i = 0; // the initial state of the
automaton is
  j = 0; // the first character of the text

  for( ; ; ) {
    if(j == n) break; // we reached the end
of the text

    if(text[j] == pattern[i]) {
      i++; // change the state of the
automaton
      j++; // get the next character from
the text
      if(i == m)
    }
    else if(i > 0) i = F[i];
    else j++;
  }
}
```



Gambar 3. Simulasi Algoritma KMP
Sumber : Slide perkuliahan Pattern Matching dibuat oleh Dr. Andrew Davison dan diperbaharui oleh Dr. Rinaldi Munir



Gambar 4. Simulasi Algoritma Boyer-Moore
Sumber : Slide perkuliahan Pattern Matching dibuat oleh Dr. Andrew Davison dan diperbaharui oleh Dr. Rinaldi Munir

Boyer-Moore

Algoritma Boyer-Moore memulai pengecekan dari kanan ke kiri. Ada 3 kasus dalam algoritma ini yaitu kasus pertama jika mismatch ditemukan dan karakter pada text lalu pada pattern terdapat karakter tersebut maka majukan pattern sehingga posisi karakter tersebut sudah sama dan mulai lagi dari pemeriksaan dari kanan. Kasus kedua yaitu mismatch dan karakter pada text terdapat di bagian kanan pattern maka majukan pattern sebanyak 1 ke kanan. Kasus terakhir dimana karakter yang mismatch pada text tidak ditemukan di dalam pattern maka geser sehingga karakter pertama pattern berada di kanan karakter yang tadinya mismatch.

Pada BM juga terdapat *function last*© yang berisi karakter c dari alphabet dan berada pada urutan keberapa pada pattern. Jika tidak ada pada pattern isi dengan -1

```
function BM(text[], pattern[])
{
  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"  }
}
```

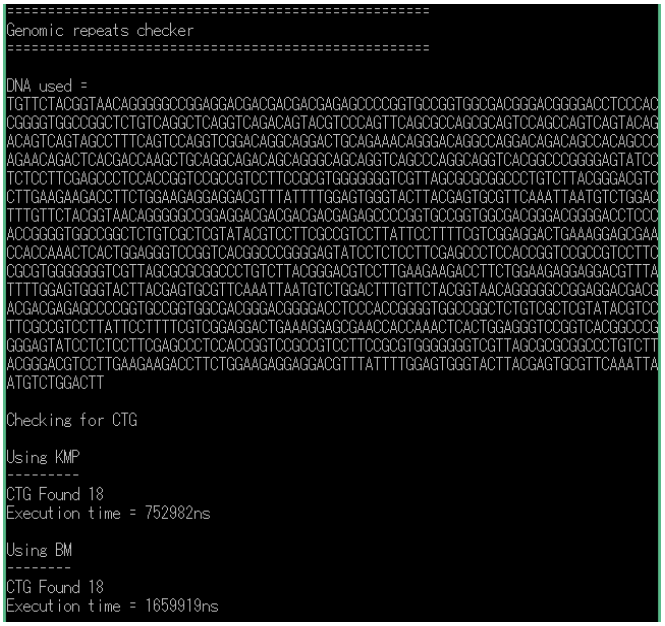
III. PEMBAHASAN

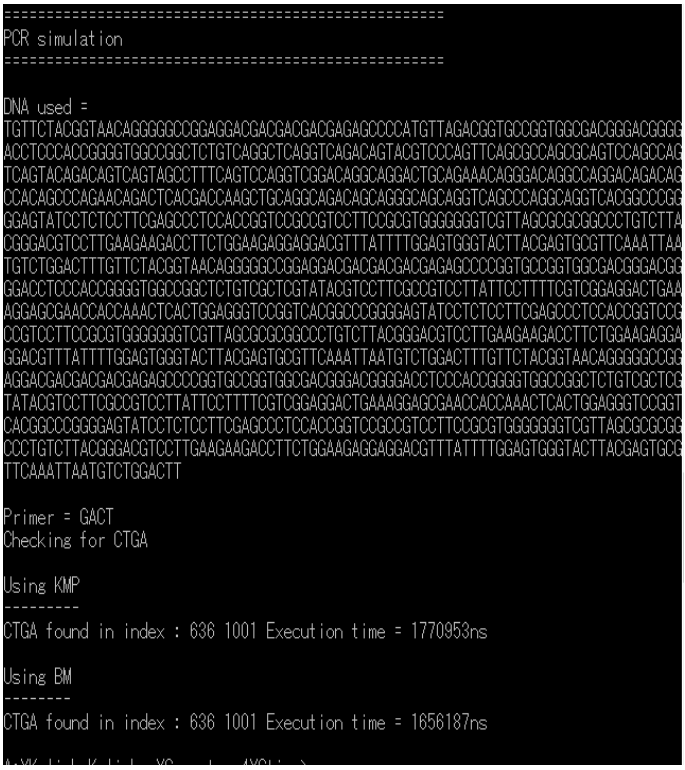
Mencari Sekuens DNA berulang(Pengecekan *Huntington's Disease)*

- Siapkan sebuah string yang merepresentasikan DNA sebuah makhluk hidup
- Hitung berapa kali kodon CAG, CTG berada pada DNA tsb dengan bantuan teknik pencocokan pola
- Analisis berdasarkan hasil yang ditemukan

Hasil :

```
==================================================
Genomic repeats checker
==================================================

DNA used =
TGTTCTACGGTAACAGGGGGCCGGAGGACGACGACGACGAGAGCCCCGGTGCCGGTGGCGACGGGACGGGGACCTCCCAC
CGGGGTGGCCGGCTCTGTCAGGCTCAGGTCAGACAGTACGTCCCAGTTCAGCGCCAGCGCAGTCCAGCCAGTCAGTACAG
ACAGTCAGTAGCCTTTCAGTCCAGGTCGGACAGGCAGGACTGCAGAAACAGGGACAGGCCAGGACAGACAGCCACAGCCC
AGAACAGACTCACGACCAAGCTGCAGGCAGACAGCAGGGCAGCAGGTCAGCCCAGGCAGGTCACGGCCCGGGGAGTATCC
TCTCCTTCGAGCCCTCCACCGGTCCGCCGTCCTTCCGCGTGGGGGGGTCGTTAGCGCCGCGGCCCTGTCTTACGGGACGTC
CTTGAAGAAGACCTTCTGGAAGAGGAGGAGCGTTTATTTTGGAGTGGGTACTTACGAGTGCGTTCAAATTAATGTCTGGAC
TTTTGTTCTACGGTAACAGGGGGCCGGAGGACGACGACGACGAGAGCCCCGGTGCCGGTGGCGACGGGACGGGGACCTCCC
ACCGGGGTGGCCGGCTCTGTCGCTCGTATACGTCCTTCGCCGTCCTTATTCCTTTTCGTCGGAGGACTGAAAGGAGCGAA
CCACCAAACTCACTGGAGGGTCCGGTCACGGCCCGGGGAGTATCCTCTCCTTCGAGCCCTCCACCGGTCCGCCGTCCTTC
CGCGTGGGGGGGTCGTTAGCGCCGCGGCCCTGTCTTACGGGACGTCCTTGAAGAAGACCTTCTGGAAGAGGAGGAGCGTTTA
TTTTGGAGTGGGTACTTACGAGTGCGTTCAAATTAATGTCTGGACTTTGTTCTACGGTAACAGGGGGCCGGAGGACGACG
ACGACGAGAGCCCCGGTGCCGGTGGCGACGGGACGGGGACCTCCCACCGGGGTGGCCGGCTCTGTCGCTCGTATACGTCC
TTCGCCGTCCTTATTCCTTTTCGTCGGAGGACTGAAAGGAGCGAACCACCAAACTCACTGGAGGGTCCGGTCACGGCCCG
GGGAGTATCCTCTCCTTCGAGCCCTCCACCGGTCCGCCGTCCTTCCGCGTGGGGGGGTCGTTAGCGCCGCGGCCCTGTCTT
ACGGGACGTCCTTGAAGAAGACCTTCTGGAAGAGGAGGAGCGTTTATTTTGGAGTGGGTACTTACGAGTGCGTTCAAATTA
ATGTCTGGACTT

Checking for CAG

Using KMP
---------
CAG Found 39
Execution time = 1758358ns

Using BM
--------
CAG Found 39
Execution time = 5523267ns
```





Pencarian DNA yang cocok dengan primer pada Polymerase Chain Reaction

- Siapkan sebuah string yang merepresentasikan DNA sebuah makhluk hidup
- Siapkan sebuah primer, dan sekuens DNA yang akan dicari (komplemen dari primer)
- Cari sekuens DNA tsb pada DNA yang sudah disiapkan pada poin pertama

IV. ANALISIS

Mencari Sekuens DNA berulang(Pengecekan *Huntington's Disease)*

Pada pengujian ditemukan pengulangan sekuens DNA CAG sebanyak 39 kali. Pengulangan sekuens DNA CAG dapat dianalisis untuk menentukan apakah makhluk tersebut terkena penyakit Huntington. Batas normal dari pengulangan CAG adalah 6-35 sehingga pengulangan 39 kali dapat disimpulkan makhluk tersebut terkena penyakit Huntington

Pada pengujian kedua untuk pengecekan CTG adalah untuk menentukan apakah makhluk tersebut terkena *Myotonic Dystrophy*, pengulangan sebanyak 18 kali masih berada di batas normal untuk pengulangan CTG sehingga makhluk aman dari penyakit Myotonic Dystrophy

Pencarian DNA yang cocok dengan primer pada Polymerase Chain Reaction

Pada pencarian DNA untuk membantu Polymerase Chain Reaction hanya dibutuhkan waktu 1719635 ns (tercepat menggunakan KMP) atau 1.7 ms. Pencarian sekuens DNA yang cocok merupakan hal terpenting dalam proses PCR dan hal tersebut dapat dibantu dengan menggunakan pencocokan pola yang dijelaskan pada bidang komputer sains. Perbandingan waktu setiap pencarian juga beragam ada dimana dengan algoritma KMP memiliki waktu lebih cepat dibandung BM begitu pula sebaliknya

ACKNOWLEDGMENT

Penulis ingin berterimakasih kepada Tuhan Yang Maha Kuasa karena berkat dan rahmatnya penulis dapat menyelesaikan makalah ini dengan baik dan tepat waktu Penulis juga menyampaikan banyak terima kasih kepada Bapak Rinaldi Munir dan Ibu Nur Ulva Maulidevi yang telah mengajarkan dasar-dasar teori yang diperlukan penulis untuk menyelesaikan makalah ini.

REFERENCES

[1] https://www.genome.gov/25520880/deoxyribonucleic-acid-dna-fact-sheet/, diakses pada tanggal 8 Mei 2016 pukul 12.00

[2] https://www.topcoder.com/community/data-science/data-science-tutorials/introduction-to-string-searching-algorithms/, diakses pada tangal 8 Mei 2016 pukul 13.12

[3] Rouchka, Eric C, "Pattern Matching Techniques and Their Applications to Computational Molecular Biology -- A Review", IEEE, submitted for publication.

[4] http://bix.ucsd.edu/bioalgorithms/presentations/Ch09_CombinatorialPatternMatching.pdf., diakses pada tanggal 7 Mei 2016 pukul 19.32

[5] https://www.dnalc.org/resources/animations/pcr.html, diakses pada tanggal 8 Mei 2016 pukul 17.21

[6] http://www.ncbi.nlm.nih.gov/probe/docs/techpcr/ diakses pada tanggal 8 Mei 2016 pukul 17.22

[7] http://neuromuscular.wustl.edu/mother/dnarep.htm, diakses pada tanggal 8 Mei 2016 pukul 20.00

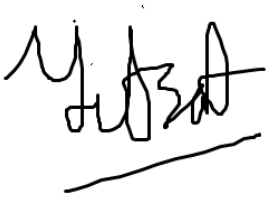
[8] http://www.personal.kent.edu/~rmuhamma/Algorithms/MyAlgorithms/StringMatch/boyerMoore.htm diakses pada tanggal 8 Mei 2016 pukul 09.00

[9] http://users.csc.calpoly.edu/~dekhtyar/448-Spring2013/lectures/lec03.448.pdf diakses pada tanggal 8 Mei 2016 pukul 09.21

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 8 Mei 2016



Yeksadiningrat A.V
13514055