Penggunaan Pattern Macthing dalam Penentuan Gen pada BLAST

Rama Febriyan 13511067
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13511067@std.stei.itb.ac.id

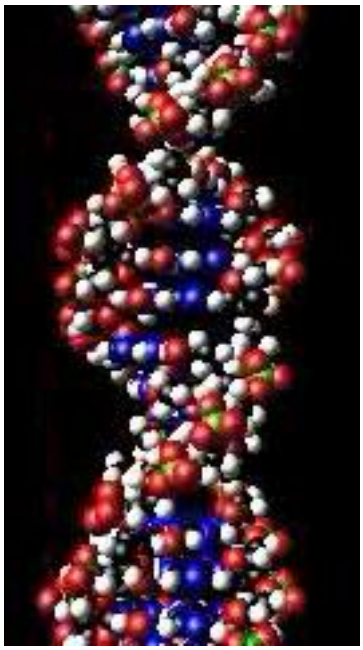
***Abstract*—Makalah ini membahas tentang penerapan strategi *pattern matching* dalam pencocokan struktur DNA yang digunakan pada kakas bernama Basic Local Alignment Search Tool (BLAST).**

***Index Terms*—gen, BLAST, DNA, FASTA, Smith-Waterman.**

I. PENDAHULUAN

Makhluk hidup, termasuk manusia memiliki ciri khas masing-masing. Semua ciri khas tersebut berasal dari sebuah kompunen kecil yang di sebut DNA.

Secara garis besar, DNA di dalam sebuah sel adalah sebagai materi genetik, dengan kata lain, DNA merupakan *blueprint* bagi setiap aktivitas sel. Hal ini berlaku bagi seluruh organisme.



Gambar 1 DNA

Setiap makhluk hidup memiliki ciri khas tertentu pada DNA mereka. Dengan mengetahui perbedaan dan ciri khas tersebut, DNA dapat digunakan dalam berbagai bidang. Misalnya dalam forensik, DNA yang terdapat dalam darah, sperma, kulit, liur, dapat digunakan untuk melakukan identifikasi pelaku kejahatan. Proses ini di sebut fingerprinting genetika. Metode ini merupakan tekni yang paling terpercaya meskipun tidak selalu sempurna, misal jika tempat kejadian merupakan tempat yang terkontaminasi oleh DNA dari banyak orang.

Selain bidang forensik, keunikan DNA juga dapat digunakan di bidang bioinformatika. Info yang terdapat di dalam DNA dapat di gunakan untuk mengidentifikasi sebuah DNA tak dikenal yang ditemukan di sebuah lokasi penelitian. Dengan melakukan pencocokan terhadap DNA yang terdapat pada basis data dengan DNA temuan, dapat diketahui apakah DNA temuan tersebut berasal daru organisme yang sudah dikenal atau sebuah organisme baru.

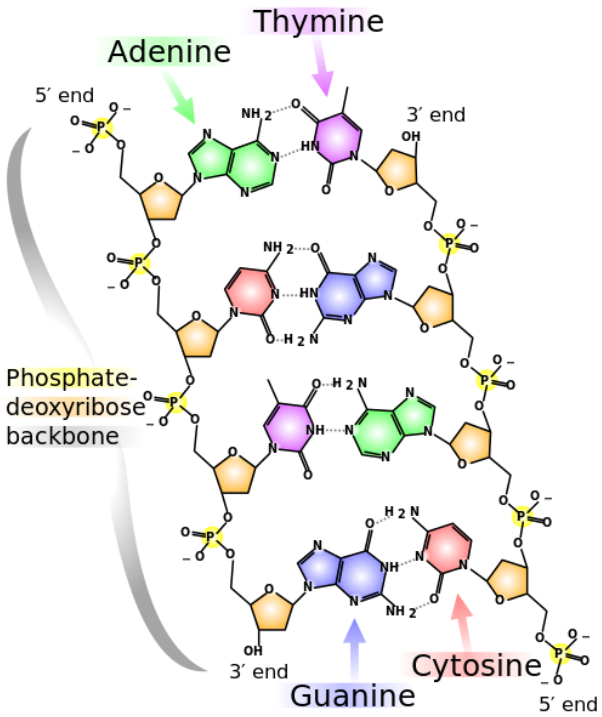
II. DNA

DNA (deoxyribonucleic acid) atau asam deoksiribonucleat adalah sejenis asam nukleat yang tergolong boimolekul utama dari organisme. Pada umumnya, DNA terletak di dalam inti sel

DNA teriri dari tiga komponen utama, yaitu:

- Gugus fosfat
- Gula deoksiribosa
- Basa nitrogen, yang terdiri dari:
  - Adenin (A)
  - Guanin (G)
  - Sitosin (S)
  - Timin (T)

Satu unit monomer DNA dinamakan nukleotida. Basa nitrogen pada rantai DNA saling berkaitan. Adenin barkaitan dengan Timin, sedangkan Guanin berkaitan dengan Sitosin.



Gambar 2 Struktur kimia DNA

III. STRING MATCHING

String matchng merupakan sebuah teknik yang digunakan untuk melakukan untuk mencari sebuah pola tertentu dalam teks

Untuk melakukan proses ini dibutuhkan 2 jenis data yaitu:

1. Teks, yaitu string sepanjang n karakter sebagai sumber pencarian
2. Pattern, yaitu string sepanjang m dengan m < n yang akan dicari di dalam teks

Contoh:

```
Pattern:rumah
Teks: kami pulang ke rumah sore hari
```

Bagian bercetak tebal di atas adalah string yang ditemukan sesuai pola.

String matching dapat di temukan dalam kehidupan sehari-hari. Google Search yang sering kita gunakan, memanfaatkan string matching dalam melakukan pencarian. Pertama pengguna memasikkan kata kunci yang akan di cari. Kata kunci ini digunakan sebagai *pattern*. Kemudian dengan menggunakan algoritma tertentu, kata kunci di cocokkan dengan data yang dimiliki oleh Google. Semua dokumen yang memiliki kecocokan dengan *pattern* yang telah diberikan, ditampilkan sebagai hasil pencarian dengan pola pengurutan tertentu.

String matching juga digunakan dalam kriminalitas. Pada tempat kejadian perkara, terkadanag tersisa sidik jari dari pelaku kejahatan. Sidik jari tersebut diambil dan disimpan dalam format tertentu. Kemudian hasil simpanan diubah dalam bentuk string dan dibandingkan dengan string yang terdapat dala m basis data sidik jari yang dimiliki pihak kepoilisan. Dengan ini tersangka kejahatan dapat dibatasi ruang pencariannya.

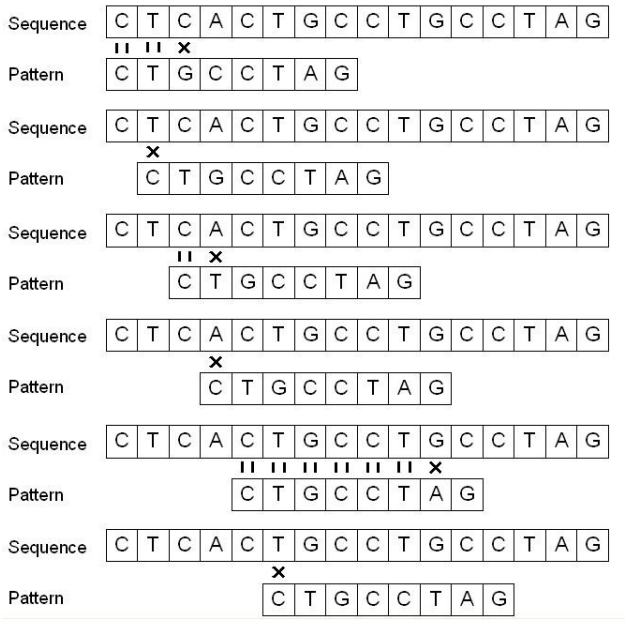
IV. ALGORITMA STRING MATCHING

Terdapat banyak algoritma yang digunakan dalam string matching. Seperti *Brute-force, Knuth-Morris-Pratt, Boyer-Moore.* Algoritma-algoritma ini memiliki karakteristik tersendiri dalam melakukan pencarian.

A. Brute-force

Algoritma Brute-force merupakan algoritma paling sederhana dalam melakukan string matching. Algoritma ini berkerja denagn cara berikut:

1. Mula-mula,pattern di cocokkan pada awal teks
2. Dengan bergerak dari kiri ke kanan, setiap katrakter pada pattern dibandingkan dengan karakter yang bersesuaian dengan teks, sampai semua karakter sama, atau ditemukan ketidakcocokan.
3. Apabila ditemukan ketidakcocokan dan teks belum habis, maka pattern digeser satu karakter ke kanan dan langkah 2 diulang.



Gambar 3 Contoh Bruteforce String Matching

```
BruteForceStringMatch(T[0...n-1],
P[0...m-1])
      for i ← 0 to n-m do
            j ← 0
                  while j < m and P[j]
= T[i+j] do
                        j++
                  if j = m then return
i
      return -1
```

Algoritm *Brute-force* dalam string matching memiliki kelemahan. Pattern digeser satu karakter ke kanan setiap ditemukan ketidakcocokan. Ini akan memakan waktu yang lama hingga ditemukan ketidakcocokan atau teks habis.

B. Knuth-Morris-Pratt

Algoritma KMP memelihara inforasi yang digunakan saat melakukan pergeseran. Informasi ini digunakan untukmelakukan pergeseran yang lebih jauh, tidak seperti bruteforce yangmelakukan pergeseran perkarakter.

Pergeseran dilakukan berdasarkan suffix kesamaan suffix dan prefix dalam pettern dan yang ditemukan di dalam teks.

```
m: 012345678901234567890 12
S: ABC ABCDAB ABCDABCDABDE
W: ABCDABD
i: 0123456

m: 012345678901234567890 12
S: ABC ABCDAB ABCDABCDABDE
W:     ABCDABD
i:     0123456

m: 012345678901234567890 12
S: ABC ABCDAB ABCDABCDABDE
W:         ABCDABD
i:         0123456

m: 012345678901234567890 12
S: ABC ABCDAB ABCDABCDABDE
W:             ABCDABD
i:             0123456

m: 012345678901234567890 12
S: ABC ABCDAB ABCDABCDABDE
W:                 ABCDABD
i:                 0123456
```

Gambar 4 Pergeseran pada KMP

```
Fungsipinggiran(P)

i ← 1
j ← 0
f(0) ← 0
while i < m do
    if P[j] = P[i]
        f(i) ← j +1
         i ← i +1
          j← j + 1
else if
         j ← f(j - 1)
else

        f(i) ← 0
        i ← i +1
```

```
KMP(P,T)

f ← compute failure function of
Pattern P
i ← 0
j ← 0
while i < length[T] do
    if j ← m-1 then
        return i- m+1    // we have a
match
i ← i +1
j ← j +1
else if j > 0
        j ← f(j -1)
    else
        i ← i +1
```

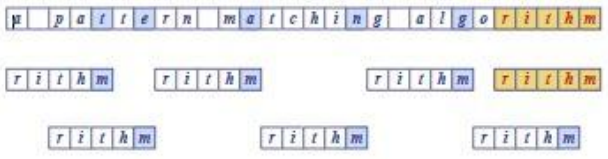
Seperti terlihat pada *pseudocode* diatas, algoritma KMP menggunakan fungsi pinggirn yang mengindikasikan pergeseran terbesar yang mungkin dilakukan dalam melakukan pencarian. Fungsi pingiran ini embuat pencarian lebih efektif dengan menghindari pergesesan tidak berguna seperti halnya algoritma *Brute-force*.

C. Booyer-Moore

Seperti Algoritma KMP, algoritma BM juga memanfaatkan informasi yang dimiliki untuk melakukan pergeseran. Hanya saja, Algoritma BM melakukan pencocokan string dari belakan pattern, tidak seperti algoritma KMP yang melakukannya dari depan.

Secara sistematis, algoritma BM bekerja sesuai langkah berikut:

1. Pattern dicocokkan dari awal teks
2. Dari kanan ke kiri, karakter per karakter dicocokkan
3. Algoritma menggeser pattern dengan memaksimalkan pergeseran *good-suffix* dan pergeseran *bad-character*, kemudian mengulangi langkah 2 sampai ujung teks atau kecocokan di temukan.



Gambar 5 Ilustrasi Algoritma Boter-Moore

```
void preBmBc(char *x, int m, int
bmBc[]) {
   int i;

   for (i = 0; i < ASIZE; ++i)
      bmBc[i] = m;
   for (i = 0; i < m - 1; ++i)
      bmBc[x[i]] = m - i - 1;
}
```

```
void suffixes(char *x, int m, int
*suff) {
   int f, g, i;

   suff[m - 1] = m;
   g = m - 1;
   for (i = m - 2; i >= 0; --i) {
      if (i > g && suff[i + m - 1 - f]
< i - g)
         suff[i] = suff[i + m - 1 -
f];
      else {
         if (i < g)
            g = i;
         f = i;
         while (g >= 0 && x[g] == x[g
+ m - 1 - f])
            --g;
         suff[i] = f - g;
      }
   }
}

void preBmGs(char *x, int m, int
bmGs[]) {
   int i, j, suff[XSIZE];

   suffixes(x, m, suff);

   for (i = 0; i < m; ++i)
      bmGs[i] = m;
   j = 0;
   for (i = m - 1; i >= 0; --i)
      if (suff[i] == i + 1)
         for (; j < m - 1 - i; ++j)
            if (bmGs[j] == m)
               bmGs[j] = m - 1 - i;
   for (i = 0; i <= m - 2; ++i)
      bmGs[m - 1 - suff[i]] = m - 1 -
i;
}


void BM(char *x, int m, char *y, int
n) {
   int i, j, bmGs[XSIZE], bmBc[ASIZE];

   /* Preprocessing */
   preBmGs(x, m, bmGs);
   preBmBc(x, m, bmBc);

   /* Searching */
   j = 0;
   while (j <= n - m) {
      for (i = m - 1; i >= 0 && x[i]
== y[i + j]; --i);
      if (i < 0) {
         OUTPUT(j);
         j += bmGs[0];
      }
      else
         j += MAX(bmGs[i], bmBc[y[i +
j]] - m + 1 + i);
   }
}
```

Pada *Boyer-Moore* dikenal istilah *bad-character*. Bad-character adalah karakter pada teks yang menjadikan perbandingan gagal. Pada pattern, dilihat karakter yang sama yang terdekat dengan karakter tempat terjadi kegagalan. Kemudian pattern digeser sehingga karakter pada pattern berada pada posisi yang sama dengan karakter pada teks.

V. ALGORITMA SMITH-WATERMAN

Algoritma *Smith-Waterman* adalah algoritma pencarian yang dikembangkan T.F.Smith dan M.S.Waterman. algoritma ini merupakan algoritma pencarian yang mengimplementasikan program dinamis yang mangambil *alignment* dari panjang dari lokasi, dalam sebuah *sequence* dan menentukan apakah susunan yang optimal dapat ditemukan.

Pada algoritma ini, program dinamis bertugas mencari solusi terkecil dari masalah dan menggabungkannya menjadi solusi final dari keseluruhan masalah.

Algoritma SW membandingkan segmen dengan panjang berbeda, melihat apakah segmen memaksimalkan hasil pengukuran. Bentuk Rekursif dari algoritma ini adalah:

$$H_{i,j} = \max\{H_{i-1,j-1} + s(a_i, b_j); H_{i-k,j} - W_k; H_{i,j-1} - W_1; 0\}$$

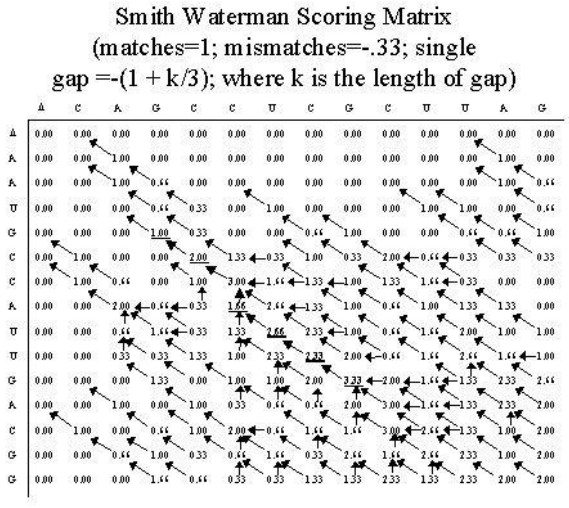
Gambar 6 Bentuk Rekursif dari SW

Misal kita memiliki *sequence* GAGTCGC dan CTATGCA. Pertama, bariskan *sequence* tersebut dan membandingkan jarak. Hitung r dan d (dalam kasus ini bernilai 5) sementara nilai 4 adalah nilai yang sama (dalam kasus ini 4)

```
G - A G T C G C -
r i m d m d m m i
C T A - T - G C A
```

Gambar 7 contoh penggunaan

Pada gambar diatas, r=replacement; m=match; i,d=insertion/deletion.



Gambar 8 Scoring dalam Matriks

VI. BASIC LOCAL ALIGNMENT SEARCH TOOL

Basic Local Alignment Search Tool atau BLAST, adalah sebuah kakas yang di gunakan dalam melakukan perbandingan *sequence* pada info biologis, seperti asam amino, protein atau nukleotida.

BLAST merupakan algoritma yang lebih cepat dibandingkan dengan algoritma Smith-Waterman, meskipun tidak menjamin alignment yang seoptimal Smith-Waterman. BLAST menggunakan format FASTA sebagai input. FASTA merupakan sebuah format berbasis teks sebagai representasi dari nukleotida atau peptida.

Keluaran dari BLAST dapat berupa HTML, plain-text, atau XML. Jka dijalankan langsung pada halaman web NCBI, keluaran default adalah HTML. Hasil juga ditampilkan berupa grafis jika dijalankan melalui NCBI, juga tabel berisi pengidentifikasi *sequence* terhadap data yang berkaitan sehingga hasil dari pencarian lebih mudah dibaca.

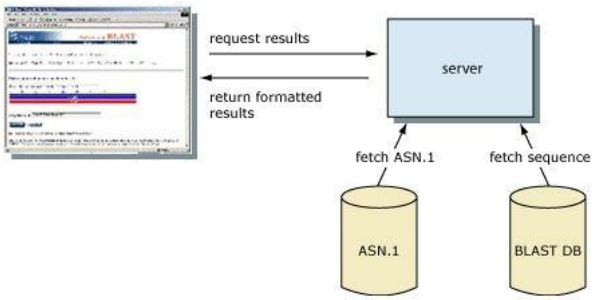
A. Cara Kerja

Kebanyakan protein modular di alam, dengan domain fungsional seringkali berulang dalam protein yang sama begitu juga diantara protein yang berbeda dari spesies yang berbeda. Agoritma BLAST menggunakan informasi ini untuk mempercepat pencarian kesamaan *sequence*. mRNA yang ditemukan dapat di *align* dengan kepingan DNA, karena dibuthkan dalam analisis sebuah genom.

Ketika sebuah query di submit, *sequence* dan beberapa info lainnya dimasukkan kedalam algoritma BLAST di dalam server. BLAST pertama kali dengan membuat sebuah tabel *lookup* dari semua "kata"(sebuah sub-*sequence* dimana protein biasanya terdiri dari tiga huruf) dan "kata terdekat". Ketika kecocokan ditemukan, kecocokan ini digunakan untik menginisiasi *gapped extensions* dari "kata" tesebut.

BLAST tidak mencari di dalam GenBank seara langsug. Setiap entri dibagi dua, salah satu mengandung header dan yang lainnya berisi informasi *sequence*. Inilah data yang digunakan dalam algoritma.

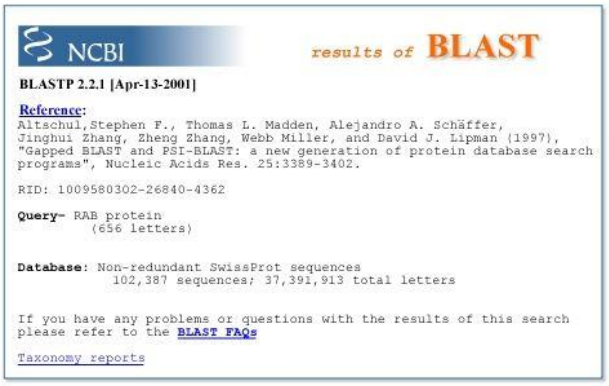
Setelah algoritma selesai melakukan pencarian terhadap semua "kata" yang mungkin dan memperluas secara maksimal, setiap *query-sequence* disusun kembali dan menuliskan informasinya ke sebuah SeqAlign.



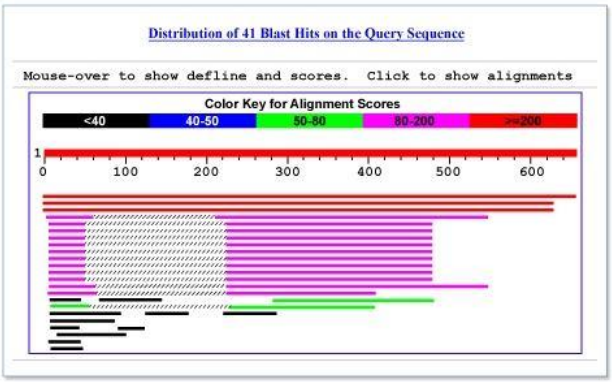
Gambar 9 Bagaimana hasil pada web dibentuk

B. Keluaran

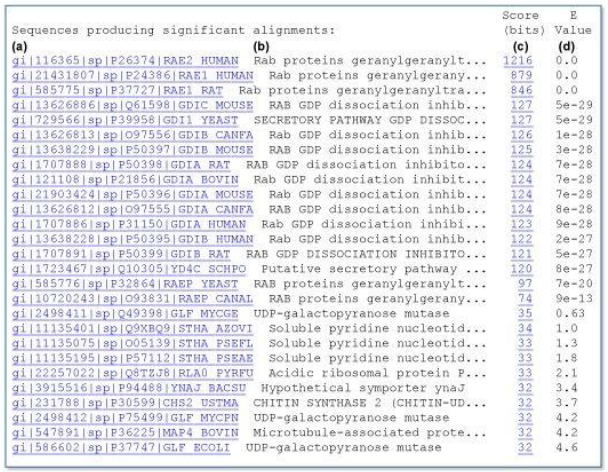
Keluaran dari BLAST dapat berupa informasi "tradisional". Keluaran ini terdiri dari (1) header, yang berisi informasi dari *sequence*, dan grafis menyeluruh; (2) deskripsi dari *sequence* pada database yang sesuai dengan query; (3) alignment dari setiap databse yang sesuai



Gambar 10 Header BLAST



Gambar 11 Hasil dalam grafik

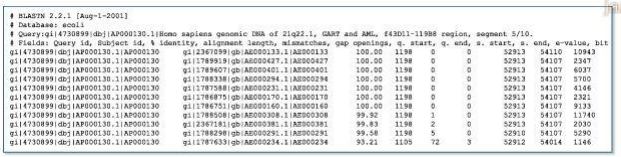


Gambar 12 on-line description

Gambar 13 Sequence yang saling berpasangan

Hasil tradisional ini digunakan untuk mengidentifikasi hasil satu persatu dari karakteristik suatu gen atau protein.

Selain hasil tradisional, juga diberikan keluaran berupa tabel dengan informasi tertentu.



Gambar 14 BLAST hit-table

C. Contoh

```
/* Get default options. */
options =
BLASTOptionNew(blast_program, TRUE);
if (options == NULL)
    return 5;

options->expect_value = (Nlm_FloatHi)
myargs [3].floatvalue;

/* Perform the actual search. */
seqalign =
BioseqBlastEngine(query_bsp,
blast_program, blast_database,
options,

    NULL, NULL, NULL);

/* Do something with the SeqAlign...
*/
MySeqAlignPrint(seqalign, outfp);

/* clean up. */
seqalign = SeqAlignSetFree(seqalign);
options = BLASTOptionDelete(options);

sep = SeqEntryFree(sep);
FileClose(infp);
FileClose(outfp);
```

Keterangan program:

1. BLASTOptionNew mengalokasikan sebuah BLASTOptionBlk dengan default yang telah disediakan program
2. Member expect_value dari BLASTOptionBlk diubah menjadi nilai tertentu dalam *command-line.*
3. BioseqBlastEgine melakukan pencarian dari BioseqPtr(query_bsp) yang bisa di dapat dari database, Entrez, atau FASTA.

VII. KESIMPULAN

Algoritma string-matching dapat diterapkan di berbagai bidang. Mulai dari bidang informatika hingga biologis. Algoritma string matching dapat berupa pengembangan dari algoritma lain, seperti algoritma Smith-Waterman yang berasal dari program dinamis.

Algoritma yang digunakan pada BLAST merupakan pengembangan dari algoritma Smith-Waterman. BLAST memberikan hasil yang lebih cepat daripada Smith-Waterman, meskipun tidak seoptimal Smith-Waterman.

Dengan menggunakan pengetahuan mengenai string matching, manusia dapat melakukan penelitian mengenai genom tak dikenal dengan memanfaatkan BLAST dan potongan gen yang ditemukan serta database yang dimiliki..

REFERENCES

[1] Munir, Rinaldi.2008.*Diktat Kuliah IF2211 Strategi Algoritma.* Program Studi Teknik Informatika, Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung

[2] http://blast.ncbi.nlm.nih.gov/Blast.cgi

[3] http://bioinformatica.upf.edu

[4] http://www.csl.mtu.edu/cs4321/www/Lectures/Lecture%205%20-%20Brute%20Force%20Sorting%20and%20String%20Matching.htm

[5] http://www.personal.kent.edu/~rmuhamma/Algorithms/MyAlgorithms/StringMatch/kuthMP.htm

[6] http://www-igm.univmlv.fr/~lecroq/string/node14.html#SECTION00140

[7] http://www.ncbi.nlm.nih.gov/books/NBK21097/

[8] http://www-cs-faculty.stanford.edu/~eroberts/courses/soco/projects/computers-and-the-hgp/smith_waterman.html

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, 20 Desember 2013

Rama Febriyan, 13511067