Penerapan Algoritma Program Dinamis Optimasi Penyuntingan *String* dalam Penentuan Saran Kata pada Mesin Pencarian

Deborah Aprilia Josephine - 13516152
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl.Ganesha 10 Bandung 40132, Indonesia
13516152@std.stei.itb.ac.id

***Abstrak*—Perkembangan teknologi informasi saat ini sangat pesat. Salah satu bentuk perkembangannya ada mesin pencarian pada internet. Saat ini begitu banyak optimasi kerja yang dilakukan pada mesin pencarian. Mulai dari algoritma untuk menentukan hasil pencarian, menentukan prioritas hasil pencarian bahkan dalam hal *suggestion word*. *Suggestion word* adalah salah satu fitur yang kini sering muncul di mesin pencarian. Selanjutnya pada makalah ini penulis akan menyebutnya dengan istilah saran kata. Untuk menentukan saran kata yang akan disampaikan pada pengguna, penulis menggunakan algoritma program dinamis dalam optimasi ongkos penyuntingan *string*. Algoritma ini tidak menjadi satu-satunya cara mesin pencarian menentukan saran kata. Namun dapat menjadi salah satu algoritma untuk menentukan saran kata, atau bahkan dapat dikombinasikan dengan algoritma lainnya. Penulis juga menuliskan implementasi algoritma dalam bahasa pemrograman Java untuk mempermudah pembaca memahami algoritma program dinamis yang digunakan.**

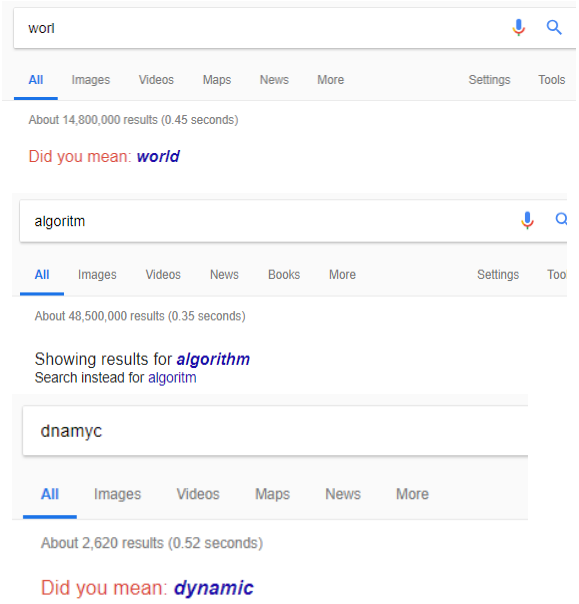
***Kata kunci*—*string*; optimal; ongkos; minimum; penyuntingan.**

I. PENDAHULUAN

Dewasa ini kita mengenal perkembangan teknologi informasi dengan begitu pesat. Salah satu perkembangan yang kita bisa nikmati adalah internet. Internet menjadi salah satu tempat seseorang masa kini mencari tahu berbagai hal yang belum pernah diketahuinya. Perkembangan internet bermula dari keinginan untuk menyatukan jaringan komputer yang ada di suatu negara berkembang pesat hingga dapat digunakan untuk seluruh dunia. Salah satu perusahaan yang bergerak di bidang produk internet ini adalah Google. Beberapa produk internet yang mungkin paling sering kita gunakan adalah mesin pencarian Google. Bahkan di Indonesia sendiri, istilah "Google" lebih akrab dipakai jika seseorang ingin melakukan pencarian di internet. Demikian pesatnya dan berpengaruhnya perkembangan teknologi informasi dalam hidup setiap kita.

Tentu perkembangan ini akan terus berjalan untuk semakin memudahkan manusia dalam bekerja atau belajar. Mungkin semula manusia hanya bisa melakukan pencarian secara tidak optimal, namun lambat laun dengan berkembangnya ilmu komputasi akan meningkatkan performansi kerja dari mesin pencarian.

Salah satu teknologi yang dapat kita rasakan adalah penentuan saran kata (*suggestion word*) pada mesin pencarian. Kemungkinan besar teknologi ini muncul setelah Google berkembang begitu pesat. Ongkos untuk melakukan pencarian semakin berkurang (dalam hal ini kita dapat asumsikan waktu), sehingga muncul suatu tren baru yaitu kesalahan menulis kata kunci. Penentuan saran kata menjadi salah satu teknologi informasi yang digunakan untuk menyelesaikan permasalahan ini. Berikut beberapa contoh penentuan saran kata yang familiar kita temukan dalam mesin pencarian Google.



Gambar 1 Beberapa contoh penentuan saran kata pada mesin pencarian Google (Sumber : Koleksi penulis)

Tren ini muncul kemungkinan setelah orang memiliki akses yang mudah, dan tidak mengambil begitu banyak ongkos dalam melakukan sebuah pencarian di Google. Namun tentu tren ini jika dilakukan oleh hampir semua orang di dunia, dengan waktu penggunaan yang intensitasnya tinggi, akan memberatkan *server*. Belum lagi jika orang-orang merasa kesalahan kata ini cukup menghambat waktu mereka dalam melakukan pencarian. Kedua hal ini akan membuat mesin

pencarian yang paling baik dan memanjakan penggunanya yang berkuasa dalam pasar produk internet.

Jika kita melakukan pencarian seperti gambar di atas, maka kita akan melihat hasil pencarian dengan menggunakan kata yang disarankan (*suggestion word*). Salah satu skenario yang dapat penulis berikan adalah sebagai berikut. sebelum melakukan penentuan saran kata dipilihlah beberapa kata yang memiliki banyak hasil pencarian, mungkin sering dicari oleh orang lain. Namun penentuan ini juga memerhatikan bagaimana penambahan, pengurangan, penukaran huruf pada kata.

Untuk menentukan kata ini, ada beberapa teori komputasi yang digunakan. Salah teori komputasi yang dilakukan adalah dengan menggunakan algoritma program dinamis. Algoritma program dinamis yang dapat digunakan untuk menentukan jumlah penyuntingan kata optimal adalah dengan menggunakan teori optimasi penyuntingan *string*. Setelah kita menentukan kata mana yang paling minimal dalam ongkos penyuntingannya, kata tersebut dapat dijadikan sebagai saran kata yang mungkin dimaksud oleh pengguna. Namun tentu penentuan saran kata dengan menentukan jumlah penyuntingan kata optimal adalah salah satu dari berbagai algoritma yang digunakan untuk menentukan saran kata ini.

Salah satu cara untuk melakukan optimasi kita dapat menggunakan algoritma program dinamis. Bagian 2 dari makalah ini akan menjelaskan mengenai algoritma program dinamis dalam optimasi penyuntingan *string*. Bagian 3 dari makalah ini akan menjelaskan bagaimana contoh studi kasus dan hasil komputasi yang sesuai dengan teori optimasi penyuntingan *string*. Bagian 4 dari makalah ini, kita akan melihat bagaimana implementasi kode program algoritma pemrograman dinamis untuk penyuntingan *string*, hasil eksekusi, dan analisis hasilnya. Bagian 5 dari makalah ini akan menyimpulkan keseluruhan isi makalah.

II. Teori Program Dinamis Optimasi Penyuntingan *String*

A. *Algoritma Program Dinamis*

Teori program dinamis adalah salah satu algoritma yang dikembangkan oleh seorang ahli matematika bernama Richard Bellman pada tahun 1950-an[1]. Algoritma ini pada dasarnya melakukan proses optimasi penentuan pilihan pada banyak *level* atau tahap. Algoritma ini biasanya digunakan untuk menyelesaikan permasalahan dengan upamasalah yang saling tumpang tindih. Biasanya upamasalah ini diselesaikan secara rekurens dari upamasalah yang terkecil kemudian berkembang hingga menjadi solusi untuk keseluruhan persoalan.

Seperti yang telah penulis sebutkan pada bagian 1, algoritma program dinamis menekankan pada penentuan solusi optimal dari persoalan. Optimal menjadi kata kunci yang penting pada algoritma program dinamis. Bellman menyebutnya sebagai *principle of optimality* (prinsip optimalitas). Prinsip ini berbunyi bahwa solusi optimal dari sebuah persoalan dapat dihasilkan dari gabungan solusi optimal dari upamasalah.

Untuk menentukan solusi optimal persoalan, kita dapat menurunkan solusinya secara rekursif dari upamasalah yang lebih kecil. Tahap menurunkan relasi rekurens inilah yang paling penting dan utama dari algoritma program dinamis. Untuk menyelesaikan persoalan dengan algoritma program dinamis diperlukan pemahaman yang tepat bagaimana menyelesaikan persoalan secara rekursif.

Algoritma program dinamis dapat menyelesaikan berbagai macam persoalan sebagai contoh *Knapsack Problem*, Persoalan Optimasi Keuntungan, *Binary Search Tree* dengan Optimasi, dan lain sebagainya. Salah satu penyelesaian persoalan yang juga menggunakan algoritma program dinamis adalah *Optimal String Editing*. Bagian B akan menjelaskan secara detail tentang *Optimal String Editing* menggunakan *Minimum Edit Distance.*

B. *Teori Optimasi Penyuntingan* String

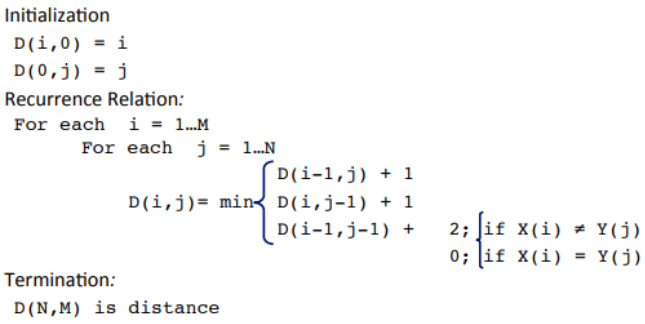
Minimum Edit Distance atau selanjutnya penulis akan menggunakan istilah ongkos penyuntingan minimum adalah salah satu perhitungan yang digunakan untuk menghitung berapa ongkos yang digunakan dalam menyunting sebuah *string*. Menyunting sebuah *string* yang dimaksud adalah dengan *insert* (menambahkan), *delete* (menghapus), atau *swap* (menukar). Penyuntingan ini terjadi ketika kita membandingkan dua buah *string* yang tidak persis sama. Kita akan melakukan penyuntingan agar kedua *string* yang berbeda dapat menjadi sama. Salah satu bentuk penyuntingan yang mungkin pernah kita alami adalah ketika kita mengetik sebuah kata yang kurang tepat pada Microsoft Word namun Microsoft Word langsung memperbaiki kata secara otomatis.

Ongkos yang dihitung adalah ongkos minimum penyuntingan antar dua *string* berbeda hingga akhirnya menjadi *string* yang sama. Biasanya ongkos yang ditentukan untuk menambah karakter adalah 1, untuk menghapus karakter adalah 1 dan untuk menukar adalah 2. Namun nilai ongkos ini dapat diubah sesuai dengan persoalan.

Untuk menentukan ongkos penyunting minimum kita perlu menetapkan *initial state*, *operators*, *goal state* dan *path cost*. *Initial state* adalah *string* yang ingin kita samakan, sedangkan *operator* adalah operasi penambahan, penghapusan dan penukaran. *Goal state* adalah *string* yang kita tetapkan ingin diubah, dan *path cost* adalah apa yang ingin kita optimasi yaitu ongkos penyuntingan minimum.

Program dinamis biasanya menggambarkan dengan tabel bagaimana relasi rekurens terjadi. Untuk menyelesaikan persoalan dengan menggabungkan solusi optimal pada upamasalah. Penggabungan solusi optimal pada upamasalah ini dilakukan dengan cara *backtrace* dari ujung kanan atas tabel sampai ke ujung kiri bawah tabel.

Untuk menentukan nilai pada tabel, kita perlu memahami algoritma berikut.



Gambar 2 Algoritma perhitungan ongkos penyuntingan minimum [2]

Misalkan kita memiliki dua buah *string* dengan panjang M dan N. Untuk menentukan tabel hasil perhitungan ongkos penyuntingan, kita perlu menginisialisasi tabel ukuran [0…M][0…N]. Indeks 0 pada baris dan kolom menandakan nilai *NULL*. Lalu kita inisialisasi setiap baris 0 dan kolom 0 dengan indeks kolom atau baris yang bersesuaian dengan baris 0 dan kolom 0. Berikut contoh langkah inisalisasi tabel ongkos penyuntingan antara *string* "ball" dan "bold".

4-L	4				
3-L	3				
2-A	2				
1-B	1				
0-NULL	0	1	2	3	4
	0-NULL	1-B	2-O	3-L	4-D

Tabel 1 Tabel inisialisasi ongkos penyuntingan

Setelah kita melakukan inisialisasi, kemudian kita mengisi sel-sel yang ada pada tabel sesuai dengan algoritma yang terdapat pada Gambar 2. Secara ringkas berikut langkah yang dapat kita perhatikan dalam mengisi sel-sel pada tabel.

a) Jika baris ke-i, dan kolom ke-j sama dengan baris ke-(i-1) dan kolom ke-(j-1) maka kita mengisi sel baris i kolom j sama dengan nilai pada sel baris i-1 dan j-1.

b) Jika tidak termasuk pada kasus di atas, maka kita perlu menentukan nilai minimum antara nilai :

   i. Sel baris i-1 kolom j ditambah 1

   ii. Sel baris i kolom j-1 ditambah 1

   iii. Dan sel baris i-1 kolom j-1 ditambah 2

Berikut salah satu bentuk pengisian tabel optimasi penyuntingan *string* pada tahap berikutnya.

4-L	4				
3-L	3				
2-A	2	minimum antara (2+1),(0+1), (1+2) = 1			
1-B	1	0	minimum antara (0+1),(2+1), (1+2) = 1		
0-NULL	0	1	2	3	4
	0-NULL	1-B	2-O	3-L	4-D

Tabel 2 Tabel ongkos penyuntingan *string*

Setelah memahami algoritma pengisian tabel, lakukan pengisian hingga didapat nilai paling ujung kanan atas atau sel baris ke n kolom ke m. Setelah itu kita lakukan *backtrace* hingga mendapatkan operasi apa saja yang harus dilakukan untuk mendapatkan *string* yang sama dengan ongkos operasi minimum. Berikut hasil tabel pengisian nilai penyuntingan *string* minimum dan *backtrace* yang dilakukan.

4-L	4	3	4	3	**4**
3-L	3	2	3	**2**	3
2-A	2	1	**2**	3	4
1-B	1	**0**	1	2	3
0-NULL	**0**	1	2	3	4
	0-NULL	1-B	2-O	3-L	4-D

Tabel 3 Tabel ongkos penyuntingan *string*

Dari tabel di atas, dapat kita simpulkan bahwa ongkos penyunting *string* minimum dari *string* "ball" dan "bold" adalah 4. Jika kita lakukan *backtrace* maka operasi yang dilakukan adalah penukaran pada huruf A dan penukaran pada huruf D.

Langkah di atas menjelaskan bahwa dengan melakukan kedua operasi di atas, penyuntingan *string* hingga mencapai *string* "bold" merupakan operasi yang paling minimum untuk mengubah *string* "ball" menjadi "bold".

III. Studi Kasus

Pada bagian 2, kita dapat mengatakan bahwa penyuntingan minimum yang dilakukan antara *string* "ball" dan "bold" adalah penukaran pada dua karakter di *string* "bald". Pada topik yang sedang kita bicarakan di makalah ini, yang kita perlukan hanyalah ongkos minimum penyuntingan *string* pada kata-kata tertentu. Maka dari itu langkah pertama yang dilakukan adalah menentukan kata-kata yang akan dibandingkan, kemudian melakukan komputasi penyuntingan *string* minimum dan membandingkan hasil komputasi penyuntingan *string* minimum,

A. Kata yang Digunakan

Untuk menentukan kata yang digunakan kita perlu memerhatikan beberapa kata yang mungkin sering dicari dalam mesin pencarian. Misalnya kita ingin mencari hasil pencarian

dari kata “dynamic” namun karena kesalahan tulis, kata kunci yang kita tuliskan di mesin pencarian adalah “dynamyc”.

Mesin pencarian yang penulis bayangkan memiliki algoritma sendiri ketika menemukan kata kunci yang janggal, maka dari itu akan mencari *alternate keyword* yang sebenarnya diinginkan oleh pengguna. *Alternate keyword* atau yang penulis sebutkan sebagai saran kata pada bagian sebelumnya ditentukan berdasarkan kata serupa yang sering dicari, mungkin sering terdapat kesalahan atau memiliki kata yang serupa.

Dalam kasus yang ingin kita gunakan adalah kata “dynamic”, maka beberapa kata yang mungkin sering dicari, memiliki hasil pencarian yang cukup banyak, atau sering terdapat kesalahan yang dapat dibandingkan adalah kata “dynamic”, “pyramid”, dan “dynamo”.

Setelah menentukan kata yang menjadi kandidat saran kata, yang selanjutnya dilakukan adalah melakukan komputasi penyunting *string* dan menentukan hasil perhitungannya.

B. Komputasi Penyuntingan String *Minimum*

Sesuai dengan algoritma penentuan ongkos minimum pada bagian 2, berikut hasil komputasi penyuntingan *string* minimum antara *string* “dynamyc” dan “pyramid”.

C	7	8	7	8	7	6	7	**8**
Y	6	7	6	7	6	5	6	7
M	5	6	5	6	5	4	5	6
A	4	5	4	5	4	5	6	7
N	3	4	3	4	5	6	7	8
Y	2	3	2	3	4	5	6	7
D	1	2	3	4	5	5	6	7
NULL	0	1	2	3	4	5	6	7
	NULL	P	Y	R	A	M	I	D

Tabel 4 Tabel ongkos penyuntingan *string “dynamyc” dan “pyramid”*

Berikut hasil komputasi penyuntingan *string* minimum antara *string* “dynamyc” dan “dynamo”.

C	7	6	5	4	3	2	**3**
Y	6	5	4	3	2	1	2
M	5	4	3	2	1	0	1
A	4	3	2	1	0	1	2
N	3	2	1	0	1	2	3
Y	2	1	0	1	2	3	4
D	1	0	1	2	3	4	5
NULL	0	1	2	3	4	5	6
	NULL	D	Y	N	A	M	O

Tabel 5 Tabel ongkos penyuntingan *string “dynamyc” dan “dynamo”*

Berikut hasil komputasi penyuntingan *string* minimum antara *string* “dynamyc” dan “dynamic”.

C	7	6	5	4	3	2	3	**2**
Y	6	5	4	3	2	1	2	3
M	5	4	3	2	1	0	1	2
A	4	3	2	1	0	1	2	3
N	3	2	1	0	1	2	3	4
Y	2	1	0	1	2	3	4	5
D	1	0	1	2	3	4	5	6
NULL	0	1	2	3	4	5	6	7
	NULL	D	Y	N	A	M	I	C

Tabel 6 Tabel ongkos penyuntingan *string “dynamyc” dan “dynamic”*

C. Analisis Hasil Komputasi

Berdasarkan hasil komputasi di atas dapat dilihat bahwa ongkos penyuntingan minimum pada kata “pyramid” adalah 8. Ongkos ini termasuk besar karena terdiri dari banyak operasi yaitu penukaran pada huruf D, penukaran pada huruf N, penukaran pada huruf Y dan penukaran pada huruf C pada kata “dynamyc”. Dapat kita analisis bahwa untuk mencapai kata “pyramid” terdapat 4 operasi penukaran. Dari jumlah operasi penukaran yang besar ini, dapat kita analisis bahwa kurang masuk akal jika seseorang melakukan kesalahan ketik hingga mencapai 4 karakter. Namun tidak ada salahnya jika kita tetap mencatat ongkos penyuntingan minimum sebagai acuan pertama.

Selanjutnya pada komputasi tabel kedua, kita dapat melihat bahwa ongkos penyuntingan minimum untuk mencapai kata “dynamo” adalah 3. Ongkos ini termasuk sedikit jika dibandingkan dengan ongkos penyuntingan pada tabel 4. Jika kita lakukan *backtrace* maka kita dapat melihat dengan ongkos penyuntingan minimum senilai 3, terdapat 2 operasi di dalamnya yaitu penukaran pada huruf Y dan penghapusan karakter C pada kata “dynamyc”. Sesuai dengan jumlah operasinya, kita dapat melihat bahwa ongkos penyuntingannya sesuai dengan jumlah operasinya. Kata “dynamo” dapat menjadi salah satu kandidat kuat untuk menentukan saran kata karena ongkos penyuntingan yang lebih sedikit dibanding dengan kata “pyramid”.

Kemudian, pada tabel ketiga kita dapat melihat bahwa ongkos penyuntingan minimum untuk mencapai kata “dynamic” bernilai 2. Dari ketiga kata yang dibandingkan, kata “dynamic”-lah yang memiliki ongkos penyuntingan paling minimum. Jika kita lakukan *backtrace* maka jumlah operasi yang dilakukan hanya 1 yaitu operasi penukaran pada huruf Y yang muncul kedua pada kata “dynamyc”. Jumlah operasi yang sedikit ini sesuai dengan ongkos penyuntingan *string* yang dihitung.

Dari ketiga hasil komputasi di atas kita dapat simpulkan bahwa untuk memperoleh saran kata yang tepat, kita dapat menggunakan nilai paling minimum dari ongkos penyuntingan minimum setiap kata yang ada.

IV. IMPLEMENTASI

Langkah penyelesaian di atas dengan menggunakan algoritma optimasi penyuntingan *string* dapat diimplementasikan dalam kode program. Implementasi ini tentu bersifat rekursif sesuai dengan ciri khas program dinamis yang bersifat rekursif. Untuk menjelaskan implementasi program dari algoritma yang telah penulis jelaskan di bagian 3, penulis akan membaginya ke dalam 3 bagian yaitu penjelasan implementasi, hasil eksekusi dan analisis hasil eksekusinya.

A. *Penjelasan Implementasi Kode Program*

Dalam bagian ini, penulis menggunakan bahasa pemrograman Java untuk melakukan implementasi kode program yang sesuai dengan algoritma program dinamis. Di dalam kode program ini dibagi menjadi satu fungsi utama, satu fungsi pendukung dan satu program utama.

Fungsi pendukung yang diimplementasikan pada kode program ini adalah fungsi minimum antara tiga buah bilangan *integer*. Untuk membandingkan ketiga bilangan *integer* ini, penulis menggunakan perbandingan biasa. Fungsi ini mengembalikan nilai *integer* yang merupakan nilai minimum dari tiga buah *integer*. Berikut potongan program fungsi minimum pada tiga buah *integer*.

```
static int min(int a, int b, int c){
    if (a <= b && a <= c){return a;}
    else {
           if (b <= a && b <= c){return b;}
           else {return c;}
    }
}
```

Fungsi utama yang diimplementasikan adalah fungsi rekursif untuk menentukan nilai pada tabel. Oleh karena itu, pertama-tama pada implementasi program dilakukan inisialisasi matriks ukuran panjang *string* pertama x panjang *string* kedua. Kemudian program akan melakukan pengisian matriks secara iteratif per baris dan per kolom. Pengisian matriks ini ditentukan sesuai dengan algoritma yang telah dijelaskan di bagian 2. Fungsi ini mengembalikan nilai ongkos penyuntingan minimum dari dua buah *string*. Berikut potongan program fungsi editDistanceString yang digunakan.

```
static int editDistanceString(String S1, String S2){
   int m = S1.length();
   int n = S2.length();
   int distEdit[][] = new int [m+1][n+1];
   for (int i = 0; i <= m ; i++){
     for (int j = 0; j <= n ; j++){
       if(i==0){distEdit[i][j] = j ;}
         else {
           if(j==0){distEdit[i][j] = i;}
           else {
             if(S1.charAt(i-1)==S2.charAt(j-1))
   {distEdit[i][j] = distEdit[i-1][j-1];}
             else       {distEdit[i][j]       =
min((distEdit[i-1][j]+1), (distEdit[i][j-1]+1),
(distEdit[i-1][j-1]+2));}
             }
           }
       }
     }
     return distEdit[m][n];
   }
```

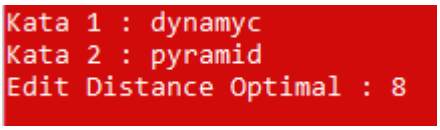
Program utama dari implementasi ini cukup memanggil fungsi editDistanceString yang mengembalikan nilai ongkos penyuntingan minimum. Kemudian kita dapat membandingkan hasil nilai ongkosnya. Berikut potongan program utama dari implementasi kode program.

```
public static void main (String[] args){
   String s1 = "dynamyc"; // Kata 1
   String s2 = "dynamo";  // Kata 2
   System.out.println("Kata 1 : " + s1 +
"\nKata 2 : " + s2 + "\nEdit Distance Optimal :
" + editDistanceString(s1,s2));
   // Menampilkan ongkos penyuntingan minimum
}
```

Program di atas dibuat oleh penulis dengan memerhatikan beberapa referensi [3].

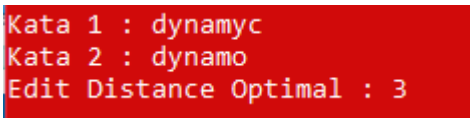
B. *Hasil Eksekusi*

Dengan kata yang sama seperti bagian 3, penulis melakukan perhitungan ongkos penyuntingan minimum pada kata yang mungkin merupakan kandidat saran kata. Berikut hasil eksekusi kode program antara kata “dynamyc” dengan “pyramid”.



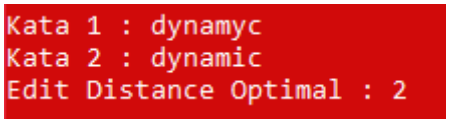
Gambar 3 Hasil Ekseskusi antara kata “dynamyc” dengan “pyramid”

Berikut hasil eksekusi kode program antara kata “dynamyc” dengan “dynamo”.



Gambar 4 Hasil Ekseskusi antara kata “dynamyc” dengan “dynamo”

Berikut hasil eksekusi kode program antara kata “dynamyc” dengan “dynamic”.



Gambar 5 Hasil Ekseskusi antara kata “dynamyc” dengan “dynamic”

C. Analisis Hasil Eksekusi

Dari ketiga hasil eksekusi tersebut, kita dapat melihat hasil yang serupa dengan komputasi menggunakan tabel pada bagian 3. Oleh karena itu tidak perlu dijelaskan secara *detail* hasil analisis terhadap hasil eksekusi implementasi program.

Secara keseluruhan, sama dengan hasil eksekusi pada bagian 3, kata “dynamic” merupakan saran kata yang paling tepat untuk ditampilkan kepada pengguna karena kata ini memiliki ongkos penyuntingan minimum paling kecil jika dibandingkan dengan kata “dynamo” dan “pyramid”.

V. KESIMPULAN

Dari bagian-bagian yang telah dijelaskan di atas, dapat disimpulkan bahwa algoritma program dinamis dapat digunakan untuk menyelesaikan persoalan penentuan saran kata pada mesin pencarian. Tentu untuk meningkatkan “intelegensi” mesin pencarian kita dapat memadukan algoritma program dinamis dengan algoritma lainnya.

UCAPAN TERIMA KASIH

Ucapan terima kasih penulis nyatakan kepada Tuhan Yang Maha Esa, sebab karena kasih karunia-Nya penulis diberi kesempatan untuk belajar di perguruan tinggi Institut Teknologi Bandung dan kembali memberikan kontribusi dalam menuliskan ide pada makalah ini. Kiranya karya ini dapat menjadi kemuliaan bagi nama-Nya.

Penulis juga mengucapkan terima kasih kepada para dosen program studi Teknik Informatika, secara khusus Dr. Nur Ulfa Maulidevi ST,M.Sc. atas dedikasinya dalam mengajar dan memberi motivasi untuk terus mengembangkan diri.

Penulis juga mengucapkan terima kasih kepada para ahli komputasi, matematikawan, begitu juga para akademisi yang telah memberikan kontribusinya untuk mengembangkan ilmu pengetahuan sampai saat ini.

REFERENSI

[1] Anany Levitin, Introduction to The Design and Analysis of Algorithm, 3rd ed., London: Pearson, 2011, pp.283-297.

[2] Stanford CS124 Lecturer, “Minimum Edit Distance”, diakses dari https://www.stanford.edu/ pada tanggal 13 Mei 2018.

[3] GeeksforGeeks, “Dynamic Programming | Set 5 (Edit Distance)”, diakses dari https://www.geeksforgeeks.org/dynamic-programming-set-5-edit-distance/ pada tanggal 13 Mei 2018.

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, 13 Mei 2018

Deborah Aprilia Josephine - 13516152