Penerapan Algoritma Program Dinamis untuk Penyelesaian Persoalan *Edit Distance*

Obbie Hadrian (13506027)

Program Studi Teknik Informatika Institut Teknologi Bandung
Jalan. Ganesha 10, Bandung
e-mail: if16027@students.if.itb.ac.id

ABSTRAK

Kata 'penyucian' dan 'pencucian' sangat mirip, dan pengubahan pada 1 huruf, yaitu huruf c->y akan mengubah kata pertama menjadi kata kedua. Kata 'balik' dapat diubah menjadi kata 'baik' dengan menghilangkan huruf 'l', begitu pula sebaliknya kata 'baik' dapat diubah menjadi 'balik' dengan menambahkan huruf 'l'. Hal tersebut berkaitan dengan fitur-fitur di berbagai perangkat lunak yang berfungsi memberikan umpan balik pada pengguna tentang perbaikan atau perpanjangan dari teks yang tengah disorot pengguna. Pada tulisan-tulisan yang beredar secara umum, persoalan ini disebut dengan *Edit-Distance.* Salah satu teknik yang digunakan untuk menyelesaikan persoalan ini adalah dengan algoritma Program Dinamis yang menghitung mengecek dan menetapkan *string* yang paling mendekati dengan *string* yang dimasukkan. Pengembangan teknik ini diterapkan dalam banyak hal terutama yang menitikberatkan pada kecocokan *string* seperti pada deteksi plagiat, biologi molekul, dan pengenalan suara. Pada makalah ini akan dibahas langkah-langkah teknis algoritma Program Dinamis dalam memecahkan masalah ini, serta bagaimana kompleksitas ruang dan waktu dari langkah-langkah tersebut.

Kata kunci: Program Dinamis, *Edit Distance,* Needleman–Wunsch *Algorithm, Dynamic Programming Alignment Algorithm,* pencocokan *string*, bioinformatika.

1. PENDAHULUAN

Dewasa ini, perbaikan *string* secara otomatis lumrah ditemui pada berbagai aplikasi, terutama aplikasi yang berkaitan dengan *word processing.* Aplikasi tersebut akan memberikan umpan balik pada pengguna pada saat-saat tertentu yang dipicu oleh suatu kata diketik secara salah, penggunaan tanda baca tidak tepat, struktur kalimat tidak tepat, atau pengetikan kata yang masih pendek. Berikut ini contoh beberapa kasus dari pemakaian fasilitas tersebut:

1. Bila pengguna aplikasi Ms-Word mengaktifkan fitur *auto correction* , maka pada saat pengguna salah mengetik 'datang', akan langsung dikoreksi menjadi '*dating*'. Terlepas dari maksud pengguna tadi ingin menulis bahasa Indonesia maupun Inggris akan langsung dikoreksi kata-katanya secara otomatis ke bahasa Inggris.
2. Bila pengguna aplikasi Ms-Word mengaktifkan fitur *Spelling and Check Grammar,* maka pada saat pengguna salah mengetik kata atau struktur kalimat tidak tepat maka kata/kalimat tersebut akan digarisbawahi.
3. Fitur *search engine'* Yahoo!' untuk memberikan *query* alternatif pada saat pengguna memberikan masukan seperti 'Program Dinamis*', search engine* akan mengembalikan hasil yang relevan dan juga umpan balik berupa pilihan untuk mencoba opsi kata kunci lainnya seperti 'Program Dinamis *algorithm'*.

2. *EDIT DISTANCE*

Fungsi *edit distance* dari dua *string*, s1 dan s2, didefinisikan sebagai jumlah minimum *point mutations* yang dibutuhkan untuk mengubah s1 menjadi s2, dimana *point mutation* adalah salah satu dari:

1. Mengubah huruf,
2. Menambahkan huruf, atau
3. Menghapus huruf.

Hubungan rekurens berikut mendefinisikan *edit distance*, d(s1,s2), dari dua *string* s1 dan s2:

```
d('', '') = 0    -- '' = string kosong
d(s, '')  = d('', s) = |s|  -- panjang
s
d(s1+ch1, s2+ch2)
             = min( d(s1, s2) + if
ch1=ch2 then 0 else 1 fi,
           d(s1+ch1, s2) + 1,
           d(s1, s2+ch2) + 1 )
```

Dua aturan pertama jelas benar, sehingga hanya perlu mempertimbangkan yang terakhir. Tidak ada *string* kosong, tiap-tiap memiliki karakter akhir, ch1 dan ch2. Ch1 dan Ch2 harus dijelaskan dalam pengubahan pada

s1+ch1 menjadi s2+ch2. Jika ch1 setara ch2, keduanya dapat dicocokkan tanpa penalti, dan *edit distance* keseluruhan adalah d(s1,s2). Jika ch1 berbeda dengan ch2, maka ch1 dapat diubah menjadi ch2, sehingga ongkos keseluruhannya menjadi d(s1,s2)+1. Kemungkinan lainnya adalah dengan menghapus ch1 dan mengubah s1 menjadi s2+ch2, d(s1,s2+ch2)+1. Kemungkinan terakhir adalah dengan mengubah s1+ch1 menjadi s2 dan kemudian memasukkan ch2, d(s1+ch1,s2)+1.

3. ALGORITMA PROGRAM DINAMIS

Program Dinamis merupakan sebuah algoritma pemecahan masalah dengan cara menguraikan solusi menjadi sekumpulan langkah atau tahapan sedemikian sehingga solusi dai persoalan dapat dipandang dari serangkaian keputusan yang saling berkaitan. Pada penyelesaian metode in kita menggunakan persyaratan optimasi dan kendala untuk membatasi sejumlah pilihan yang harus dipertimbangkan pada suatu tahap.

Algoritma Program Dinamis memiliki karakteristik sebagai berikut:

1. Persoalan dapat dibagi mejadi beberapa tahap, yang pada setiap tahap hanya diambil satu keputusan yang optimal.

2. Masing-masing tahap terdiri dari sejumlah status yang berhubungan dengan tahap tersebut.

3. Hasil keputusan yang diambil pada tahap ditransformasikan dari status yang bersangkutan ke status berikutnya pada tahap berikutnya.

4. Ongkos pada suatu tahap bergantung pada ongkos tahap-tahap sebelumnya dan meningkat secara teratur dengan bertambahnya jumlah tahapan

5. Keputusan terbaik pada suatu tahap bersifat independen terhadap keputusan yang dilakukan tahap sebelumnya.

6. Adanya hubungan rekursif yang mengidentifikasikan keputusan terbaik untuk setiap status pada tahap k memberikan keputusan terbaik untuk tahap sebelumnya.

7. Prinsip optimalitas berlaku pada persoalan tersebut.

Ciri utama dari Program Dinamis adalah prinsip optimalitas yang berbunyi : jika solusi total optimal, maka bagian solusi sampai tahap ke-k juga optimal.

Dari karakteristik poin ke-4 di atas, kita dapat menyimpulkan bahwa algoritma Program Dinamis dapat diaplikasikan apabila peningkatkan ongkos secara linear dan diskrit sehingga optimasi parsial dapat dilakukan.

Dalam menyelesaikan persoalan dengan Program Dinamis, kita dapat menggunakan 2 pendekatan berbeda yaitu :

a. Maju (*forward* atau *up-down*) : bergerak mulai dari tahap 1, terus maju ke tahap 2,3,..,n. Urutan variabel keputusan adalah x1,x2,...,xn

b. Mundur(*backward* atau *bottom-up*) : bergerak mulai dari tahap n, terus mundur ke tahap n-1, n-2,..,2,1. Urutan variabel keputusan adalah xn xn-1,x2,x1.

Adapun kompleksitas waktu pada algoritma ini adalah O(|s1|*|s2|), jika panjang kedua *string* adalah 'n'. Kompleksitas ruangnya juga sama jika seluruh matriks disimpan untuk merunut balik untuk mencari *optimal alignment*. Jika nilai *edit distance* dibutuhkan, hanya dua baris dari matriks yang dialokasi; matriks tersebut dapat mengalami 'daur ulang', dan kompleksitas ruangnya jadi O(n).

5. METODE

Nilai *edit distance* D adalah ongkos yang harus dicari solusi optimum parsial maupun globalnya. Misalnya, diketahui 2 buah yaitu *string* W dan *string* V, di mana W=W1…Wi merupakan *string* yang akan diperbaiki, dan V=V1…Vj merupakan *string* yang akan dihitung nilai *edit distance*-nya terhadap *string* W, maka persamaan umum rekursif dari edit distance dengan menggunakan algoritma Program Dinamis prinsip pendekatan maju (*forward* atau *up-down*) dapat kita bentuk sebagai berikut.

$$D(i,j) = \begin{cases} basis: D(0,0) = 0 \\ rekurens: \min \begin{cases} D(i-1,j-1) + P(i,j) \ldots\ldots(1) \\ D(i-1,j) + \delta(i,j) + P(i,j) \ldots\ldots(2) \\ D(i,j-1) + \delta(i,j) + P(i,j) \ldots\ldots(3) \end{cases} \end{cases}$$

Untuk persoalan ini, kita tentukan nilai *mismatch penalty* P(i,j)=0, jika Wi=Vj dan P(i,j)=1, jika Vj ≠ Wi, serta nilai *gap penalty* δ(i,j)=1, ,jika ada karakter '-', dan δ(i,j)=0 ,jika tidak ada.

Pada bagian rekurens, persamaan (1) menunjukkan bahwa perlu dilakukan penggantian (substitution) jika Vj ≠ Wi agar karakter Vj = Wi. Sedangkan, persamaan (2) menunjukkan bahwa perlu dilakukan pemasukkan karakter (insertion) serta persamaan (3)

menunjukkan bahwa perlu dilakukan penghapusan karakter.

Untuk menyimpan nilai-nilai *edit distance* dari setiap langkahnya, maka kita bentuk suatu matriks M berdimensi m x n dimana m=[panjang(V) +1] dan n=[panjang(W) + 1] dimana M[0,0] merupakan tempat untuk menyimpan nilai basis yaitu 0.

Apabila teknik ini diimplementasikan pada aplikasi, maka program akan memberikan umpan balik berupa *string* W dengan nilai D('ASAK', W) yang paling kecil, dimana W merupakan kumpulan *string* yang ada pada database. Jadi, jika kita merujuk ke contoh di atas, program akan mengoreksi *string* 'ASAK' dengan 'ASAL', bukan dengan 'PASAK'.

IV. KESIMPULAN

Algoritma program dinamis bekerja dengan baik dalam menyelesaikan persoalan ini. Akan tetapi algoritma ini bukan tanpa kekurangan, akibat penggunaan fungsi yang bersifat rekursif sehingga dibutuhkan memori yang besar.

Penerapan algoritma ini juga banyak digunakan di bidang bioinformatika. Penerapannya terutama untuk mengindikasi kedekatan kedua *string*. Pengukuran serupa jugadigunakan untuk menghitung *distance* antara sekuens DNA atau protein untuk berbagai tujuan.

REFERENSI

[1] V. I. Levenshtein, "*Binary codes capable of correcting deletions, insertions and reversals*", Doklady Akademii Nauk SSSR **163**(4) p845-848, 1965

[2] S. B. Needleman, C. D. Wunsch,"*A general method applicable to the search for similarities in the amino acid sequence of two proteins*" Jrnl Molec. Biol. 48 p443-453, 1970.

[3] Rinaldi Munir, "Diktat Kuliah IF2251 Strategi Algoritmik", Informatika, 2007.

[4] http://en.wikipedia.org/wiki/Dynamic_programming diakses pada 20 Mei 2008

[5] http://en.wikipedia.org/wiki/Needleman-Wunsch_algorithm diakses pada 20 Mei 2008

[6] http://www.sbc.su.se/%7Emaccallr/thesis/node27.html diakses pada 20 Mei 2008