Penerapan Algoritma Branch and Bound pada String Error Correction

Ardhana Aryomukti Wibowo[1], Herianto[2], Meirza Auriq[3]

Laboratorium Ilmu dan Rekayasa Komputasi
Program Studi Teknik Informatika, Institut Teknologi Bandung
Jl. Ganesha 10, Bandung
E-mail : if14004@students.if.itb.ac.id, if14077@students.if.itb.ac.id, if14103@students.if.itb.ac.id

Abstrak

String adalah suatu komponen yang sangat sering menimbulkan masalah karena keragamannya. Salah satu masalah yang sering ditemukan adalah pada kesalahan pengetikan string. Sering kali kita menjumpai susunan karakter-karakter pembentuk string yang salah sehingga menimbulkan string error baik dari standar pengejaan ataupun makna yang ditimbulkan. Melalui makalah ini, penulis mencoba memberikan alternatif solusi string error correction dengan menerapkan algoritma branch and bound. Standar pengejaan yang menjadi referensi pada string correction adalah Kamus Besar Bahasa Indonesia.

***Kata kunci**: string, branch and bound, brute force,*

1. Pendahuluan

Di era komputasi dan digital seperti sekarang ini, mengetik adalah suatu pekerjaan yang tidak asing lagi bagi semua orang. Dewasa ini banyak sekali perangkat digital yang meminta input data kepada user melalui keyboard. Hal ini tentulah mengharuskan user tersebut mengetik. Baik komputer, PDA, maupun telepon genggam mengharuskan usernya untuk dapat mengetik dengan cepat, tepat, dan efisien. Sayangnya kemampuan untuk mengetik cepat dan tepat secara umum masih belum banyak dimiliki masyarakat luas. Terkadang hal ini menjadi kendala ketika seseorang berada dalam keadaan terjepit/terdesak untuk melakukan pengetikan yang cepat dan tepat dalam penyelesaian suatu tugas atau pekerjaan. Dalam kondisi ini maka seorang user rawan sekali melakukan kesalahan dalam mengetik sebuah string. Kesalahan yang terjadi dalam pengetikan sebuah string dapat menyebabkan kesalahpahaman/salah pengertian dari pihak-pihak yang membacanya. Selain itu upaya user untuk mengoreksi string-string yang telah diinputkan juga akan memakan waktu yang lama dan berakhir kepada ketidakefisienan. Setelah melihat fenomena string error yang sering terjadi dalam kegiatan mengetik maka penulis mengajukan sebuah alternatif solusi untuk memecahkan masalah tersebut, sehingga string error dapat dihindari dan kegiatan mengerik dapat berlangsung secara tepat, cepat, dan efisien. Secara umum makalah ini akan membahas dua metode yang dipakai yaitu :

- String matching secara Brute Force
- Pencarian string solusi dengan Algoritma Branch and Bound

2. String Matching secara Brute Force

Pencocokan String secara Brute Force adalah metode dengan asumsi bahwa teks berada di dalam array T[1..n] dan pattern berada pada array P[1..m]. Pada String Error Correction metode ini akan diapliksikan ke dalam Algoritma Branch and Bound sebagai pen-generate root yang akan dianalisis.
Contoh :

Teks : nobody noticed
Pattern : **not**

```
            nobody noticed
s = 0       not
s = 1         not
s = 2           not
s = 3             not
s = 4                 not
s = 5                   not
s = 6                     not
s = 7                       not
```

Kesimpulan : pattern *not* ditemukan pada indeks 8 dari awal teks.

Pseudo-code

```
Procedure BruteForceSearch(input m,n : integer, input P : array[1..m] of char, input T : array[1..n] of char , output idx : integer)
{
mencari kecocokan pattern P di dalam teks T. Jika ditemukan P di dalam T, lokasi awal kecocokan disimpan di dalam peubah idx.
Masukan : pattern P yang panjangnya m dan teks T yang panjangnya n. Teks T
```

```
            direpresentasikan sebagai string (array of
            character).
Keluaran : posisi awal kecocokan (idx). Jika P tidak
            ditemukan, idx = -1.
}

Deklarasi
    s, j      : integer
    ketemu : boolean

Algoritma :
    S <=  0
    ketemu <=  false
    while (s ≤ n-m) and (not ketemu) do
          J <= 1
          while (j ≤ m) and  ( P[j] = T[s+j] ) do
             J <= j +1
          endwhile {j > m or  P[j] ≠ T[s+j]}

          if j = m then {kecocokan string ditemukan}
             ketemu <= true
          else
             s <= s+1 {geser pattern satu karakter ke
             kanan teks}
          endif
endfor
{s>n-m or ketemu}

if ketemu then
    idx <= s+1  {cat. Jika indeks array dimulai dari 0,
            maka idx <= s}
else
    idx <= -1
endif

{<= berarti diassign nilainya ←
}
```

3. Penerapan Algoritma Branch and Bound

Algoritma Branch and Bound adalah algoritma pencarian dimana ruang solusinya diorganisasikan ke dalam pohon ruang status yang dibangun dengan skema Breadth First Search(BFS). Setiap simpul diberi *cost* dan simpul berikutnya yang akan diekspansi tidak berdasarkan urutan pembangkitannya, tetapi simpul yang memiliki *cost* terkecil.

```
Procedure BranchAndBound(input akar : array [1..n]
of  string, output hasil : array [1..m] of string)

{mencari solusi correct string dengan
membandingkan string yang ada dalam library lau
mengeluarkannya sebagai alert dimana syaratnya
adalah string tersebut berada dalam satu level}

Deklarasi
          daun      : array [1..x] of string
          depth, i  : integer
          ketemu  : boolean

          Procedure Generate(input akar : array [1..n]
          of string)
          {menghasilkan array daun dan jumlahnya}
          Procedure Library(input daun : string)
          {menghasilkan true jika daun[i] ada dalam
          kamus}
          Procedure  CariMirip(input  daun  :  array
          [1..n] of string )
          {menentukan daun-daun yang kemiripann-
          nya paling besar dengan akar}

Algoritma :
          depth <= 0
          ketemu <= false

          while (not ketemu) and (depth < 10) do
                Generate (akar)
                i traversal [1..NbAnak]
                if (Library(daun[i])) then
                      ketemu <= true
                      hasil <= daun[i]
                endtraversal
                if (not ketemu) then
                      CariMirip(daun)
                      depth <= depth + 1
                      akar <= daun

          endwhile

{<= berarti diassign nilainya ←
}
```

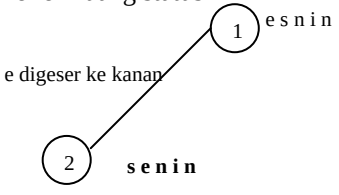
Correct string yang dikeluarkan oleh program ini adalah string-string yang sesuai dengan String library(Kamus Besar Bahasa Indonesia) dengan kedalaman yang paling rendah.
Berikut disertakan beberapa sampel pengkoreksian string error, sehingga didapatkan string yang benar sesuai dengan library.

Tabel Ruang Sampel

String Sampel	Correct String	String Library
Esnin	Senin	Senin
Keapal	Kepala	Kepala, Kelapa
alma	Lama, alam	Lama, alam

Penyelesaian

1. e s n i n
   depth = 1
   - e digeser ke kanan (s = 2)
     s e n i n = ketemu
   Pohon ruang status



2. k e a p a l

depth = 1

- k digeser ke kanan (s = 2)

  e k a p a l

- e digeser ke kiri (tidak perlu karena sudah muncul)
- e digeser ke kanan (s = 3)

  k a e p a l

- a digeser ke kanan (s = 4)

  k e p a a l

- p digeser ke kanan (s = 5)

  k e a a p l

- a digeser ke kanan (s = 6)

  k e a p l a

depth = 2

digenerate dari kepaal

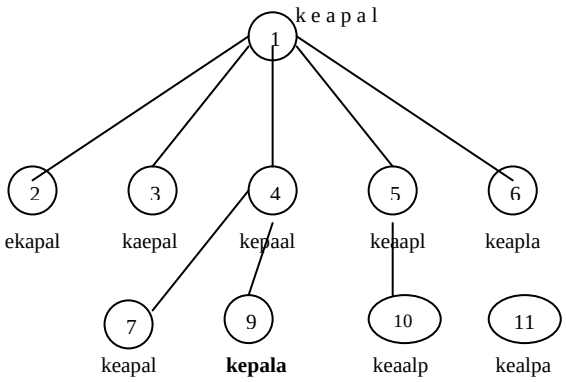
- p digeser ke kanan (s = 7)

  k e a p a l

- a digeser digeser ke kanan (s = 8)

  k e p a l a = ketemu

Pohon ruang status



3. alma

depth = 1

- l digeser ke kiri (s = 2)

  l a m a = ketemu

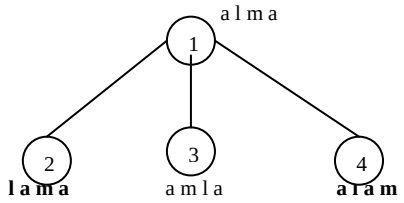
- l digeser ke kanan (s = 3)

  a m l a

- m digeser ke kanan (s = 4)

  a l a m = ketemu

Pohon ruang status



5 Kesimpulan

Dalam melakukan koreksi terhadap string error, penulis menggunakan StringMatching secara brute force dan untuk mengkoreksi string yang dimasukkan oleh user dengan string yang terdapat didlam library, penulis menerapkan algoritma Branch and Bound. Dengan metode ini string error yang terjadi dapat diatasi/dikoreksi dengan syarat sudah terdapat string yang benar dalam library.

6. Referensi

1. Munir,Rinaldi,*Dikat Kuliah IF2251 Strategi Algoritmik*, Program Studi Teknik Informatika Sekolah Tinggi Elekto Dan Informatika Institut Teknologi Bandung, 2006.