Algoritma Pencarian String dengan Menggunakan metode Brute Force yang diperkaya

Mohammadd Akbar[1] , Ipam Fuadina[2], Jaya Pramadesa[3]

Departemen Teknik Informatika, Institut Teknologi Bandung
Jl. Ganesha 10, Bandung

E-mail : if12053@students.if.itb.ac.id[1], if12079@students.if.itb.ac.id[2], if13011@students.if.itb.ac.id[3]

Abstrak

Pencarian string adalah sebuah masalah klasik dalam dunia teknologi informasi. Kebutuhan untuk mencari informasi yang berguna dalam suatu data yang besar adalah salah satu masalah klasik ini. Selain untuk mencari informasi yang tepat, pencarian string juga diharapkan untuk dapat menuntaskan pekerjaannya dengan cepat. Sehingga hasil dari pencarian itu, dapat digunakan tepat, cepat, dan efisien. Contoh dari penggunaan pencarian string, antara lain adalah search engine untuk website, pencarian kata di suatu dokumen, pembuatan program kamus, dan data mining.
Pada kali ini, kami akan mencoba membuat suatu algoritma baru untuk mencari string dengan berbasis algoritma brute-force dan menambahkan kemampuan untuk memproses karakter dalam string.

***Kata kunci**: pencarian string, brute-force, tabel*

1. Pendahuluan

Pencarian string adalah sebuah masalah klasik dalam dunia teknologi informasi. Kebutuhan untuk mencari informasi yang berguna dalam suatu data yang besar adalah salah satu masalah klasik ini. Selain untuk mencari informasi yang tepat, pencarian string juga diharapkan untuk dapat menuntaskan pekerjaannya dengan cepat. Sehingga hasil dari pencarian itu, dapat digunakan tepat, cepat, dan efisien. Contoh dari penggunaan pencarian string, antara lain adalah search engine untuk website, pencarian kata di suatu dokumen, pembuatan program kamus, dan data mining.
Untuk memecahkan masalah ini, dibuatlah suatu algoritma yang memanfaatkan kemampuan komputer untuk menjalankannya. Algoritma ini melakukan pencarian dengan suatu jalan tertentu yang terurut, tidak ambigu, dan dapat dijalankan dengan baik oleh komputer. Contoh algoritma yang sekarang sering digunakan adalah :
1. algoritma brute-force
2. algoritma KMP

Algoritma pertama sering menjadi rujukan, karena ia bersifat *straight* dalam melakukan komputasi, sehingga mudah dimengerti. Selain itu, ia dapat merupakan suatu basis, atau dasar dalam membuat suatu algoritma yang lebih cerdas. Hal ini menyebabkan ia sering menjadi dasar dalam mengukur waktu komputasi untuk berbagai algoritma pencarian string. Sedangkan algoritma KMP, merupakan algoritma yang sering dipakai dalam melakukan pencarian string dalam dekade belakangan, disebabkan keefektifannya dalam pencarian tersebut.
Pada kali ini, kami akan mencoba membuat suatu algoritma baru untuk mencari string dengan berbasis algoritma brute-force dan menambahkan kemampuan untuk memproses karakter dalam string.

2. Dasar Pemikiran

Dasar dari pembuatan algoritma ini, adalah dengan menggunakan algoritma pencarian string dengan metode brute force, kemudian menambahkan beberapa fitur untuk merekam hasil pencarian string pada suatu saat, untuk kemudian dapat dipakai untuk melakukan pemilihan keputusan search yang akan diambil.
Algoritma pencarian string dengan metode brute-force melakukan pencocokan karakter per karakter, dengan cara, pada pertama kali eksekusi program, algoritma akan mencocokkan karakter pertama pattern dengan karakter pertama teks, dan jika cocok, maka algoritma akan mencocokkan karakter kedua pattern dengan karakter kedua teks. Tetapi jika tidak terdapat kecocokan, maka pencocokan akan dimulai lagi dari karakter kedua karakter teks, dan dicocokkan dengan karakter pertama teks. berikut ditampilkan algoritma pencarian string dengan metode brute-force.

```
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 direpresentasika 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  { catatan: jika indeks array dimulai dari 0,
                idx ← s }
  else
     idx←-1
  endif
```

algoritma ini mempunyai kompleksitas pada kasus terbaik O(n) dan pada kasus terburuk O(mn). Dengan m ialah panjang string pattern dan n adalah panjang string teks.

3. Algoritma pencarian string metode brute-force yang diperkaya

Algoritma pencarian string dengan metode brute-force yang diperkaya ini, adalah algoritma brute-force yang dilengkapi kemampuan untuk memanfaatkan hasil dari pencarian sebelumnya, dan berbasis elemen pertama pattern.
Berikut akan dijelaskan bagaimana algoritma ini bekerja, tetapi sebelumnya akan didefinisikan struktur data yang akan digunakan.
Struktur data yang akan digunakan yaitu, 2 buah tabel kontigu untuk menyimpan string Pattern dan string Teks, sedangkan untuk menyimpan hasil pemrosesan terhadap pencarian string digunakan 2 buah tabel untuk menyimpan nilai boolean dari pemrosesan string Pattern, karena itu mempunyai panjang sebesar string pattern, tabel pertama bernama tabValidPattern, tabel kedua bernama tabRuntimePattern, dan 1 buah tabel untuk menyimpan nilai boolean dari pemrosesan dari string teks, dengan panjang sesuai dengan panjang string teks, yang selanjutnya kita sebut sebagai tabValidTeks.
Algoritma pada awal eksekusi akan menginisialisasi nilai dari semua tabel boolean dengan nilai true dan false sesuai dengan spesifikasi. Tabel tabValidPattern berisi kecocokan nilai karakter ke-i dengan karakter ke-1, tabValidPattern ini bernilai true jika cocok dan false jika tidak cocok. sedangkan untuk tabValidTeks, semuanya nilainya diisi dengan true, yang berarti bahwa apabila pencarian dimulai pada karakter ke-i pada teks, maka mungkin akan terdapat kecocokan, sedangkan jika false, tidak mungkin akan cocok. dan tabel tabRuntimePattern, berisi kecocokan string pada karakter ke-i pada pattern dan karakter ke-(posisi+i) pada teks. Posisi merupakan nilai dari karakter awal pencarian pada teks.
Setelah inisialisasi, algoritma kemudian memulai proses pencocokan string. Pada pass dengan nilai posisi pada karakter ke-1, algoritma mencocokkan semua karakter pada pattern dengan karakter pada teks dengan urutan yang bersesuaian. Pencocokan ini mempunyai persamaan tes pattern[i]=teks[posisi+i]. setelah itu, akan didapatkan nilai dari tabRuntimePattern, yang berisi true, jika Pattern[i]=teks[posisi+i] dan false, jika tidak memenuhi persamaan. Kemudian, dari tabel itu, akan dilakukan pencocokan dengan menggunakan operator XOR. Yang dideskripsikan sebagai berikut.

Indeks pattern	1	2	3
Nilai	T	T	F

tabValidPattern

Indeks pattern	1	2	3
Nilai	T	F	F

tabRuntimePattern

hasilnya dari operasi XOR pada kedua tabel itu akan dimasukkan pada tabel tabValidTeks

Indeks teks	1	2	3
Nilai	T	F	T

tabValidTeks

algoritma ini akan mempunyai kompleksitas pada kasus terburuk O(pjgPattern . pjgTeks). Sedangkan pada kasus terbaik, kompleksitasnya bernilai O(pjgPattern). Dengan panjang pattern adalah pjgPattern dan pjgTeks adalah pjgTeks. Kompleksitas algoritma ini menghitung proses pencocokan karakter per karakter pada bagian pengisian nilai tabRuntimePattern.

Berikut ditampilkan algoritma pencarian string dengan metode brute-force yang diperkaya dalam bentuk pseudo code.

```
procedure SearchString(input pjgPattern, pjgTeks : integer,
input Pattern : array[1..pjgPattern] of char,
input Teks : array[1..pjgTeks] of char,
input/output tabValidPattern : array[1..pjgPattern] of
                               char,
input/output tabValidTeks : array[1..pjgTeks] of char)
input/output tabRuntimePattern : array[1..pjgPattern] of
                               char,
output idx : integer)
{
 Mencari kecocokan pattern Pattern di dalam teks Teks.Jika
ditemukan Pattern di dalam Teks, lokasi awal kecocokan
disimpan di dalam peubah idx.

Masukan: panjang pattern Pattern adalah pjgPattern dan panjang
teks Teks adalah pjgTeks.Teks Teks direpresentasikan sebagai
string (array of character)
Keluaran: posisi awal(idx) kecocokan pattern Patter di teks
Teks. Jika Pattern tidak ditemukan, idx = -1.
}
Deklarasi
  posisi, i : integer
  ketemu, salahPertama : boolean

Algoritma:
```

```
{ melakukan pemeriksaan string }
posisi←0
ketemu←false
while (posisi ≤ pjgTeks-pjgPattern) and (not ketemu) do
{ cek karakter per karakter, untuk karakter ke-posisi di teks
Teks }
  Ketemu←True
  i←1
  while (i ≤ pjgPattern) do
    if Pattern[i]=Teks[posisi+i] then
      tabRuntimePattern[i]←True
    else
      tabRuntimePattern[i]←False
      ketemu←False
    endif
  endwhile
  { i > pjgPattern }

  { cek nilai kebenaran teks untuk mencari pattern }
  if (not ketemu) then
    i←1
    salahPertama←true
    while (i ≤ pjgPattern) do
      if tabValidPattern[i] ≠ tabRuntimePattern[i] then
        tabValidTeks[i]←True
      else
        tabValidTeks[i]←False
        ketemu←false
        if salahPertama then
          posisi←i-1
          salahPertama←false
        endif
      endif
    endwhile
  endif
endfor
{ s > (pjgTeks-pjgPattern) or ketemu }

 { procedure menghasilkan nilai }
if ketemu then
  idx←posisi+1
else
  idx←-1 { pattern Pattern tidak ditemukan di teks T }
endif


procedure TandaiKemunculan(input pjgPattern : integer,
input Pattern : array[1..pjgPattern] of char,
output tabValidPattern : array[1..pjgPattern] of integer)
{ Menghitung nilai b[1..m] untuk pattern P[1..m] }

Deklarasi
  i : integer

Algoritma:
{ menandai kesamaan karakter pertama dgn semua karakter pada
pjgPattern }
tabValidPattern[1]←true
for i←2 to pjgPattern do
  if Pattern[1]=Pattern[i] then
    tabValidPattern[i]←true
  else
    tabValidPattern[i]←false
  endif
endfor


procedure inisialisasi(input pjgPattern, pjgTeks : integer,
input/output tabValidPattern : array[1..pjgPattern] of char,
input/output tabValidTeks : array[1..pjgTeks] of char)
{ Menghitung nilai b[1..m] untuk pattern P[1..m] }

Deklarasi
  i : integer

Algoritma:
for i←1 to pjgTeks do
  tabValidTeks[i]←false
endfor

for i←1 to pjgPattern do
  tabValidPattern[i]←false
endfor

tandaiKemunculan(pjgPattern, Pattern, tabValidPattern)
```

4. Kesimpulan

Algoritma pencarian string dengan metode brute-force yang diperkaya ini adalah sebuah algoritma pencarian string metode brute-force dengan menambahkan kemampuan untuk merekam hasil suatu tahap pencarian dan memanfaatkannya untuk tahap selanjutnya. Tetapi walaupun begitu, algoritma ini masih mempunyai kompleksitas yang relatif sama dengan algoritma brute-force biasa, khususnya untuk pattern dan teks yang pendek. Tetapi akan mempunyai kelebihan, apabila digunakan untuk pattern dan teks dengan panjang yang lumayan besar. Selain itu, algoritma ini juga lebih banyak makan memori daripada algoritma brute-force biasa, karena menggunakan lebih banyak tabel.

5. Terima Kasih

Terima kasih penulis ucapkan kepada dosen mata kuliah strategi algoritmik, kepada diko aldillah, semua teman-teman 2003 dan 2002, dan kepada yang sudah direpotkan agar makalah ini bisa selesai.

6. Daftar Pustaka

1. Rinaldi Munir, algoritma pencarian string, bahan kuliah ke-15, diktat kuliah IF2251, 2003.