Algoritma Pencocokan String dalam Permainan Hangman

Andrei Dharma Kusuma/13508009
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
if18009@students.if.itb.ac.id

***Abstrak*—Algoritma pencocokkan string merupakan algoritma yang digunakan dalam pencarian string dalam sebuah string yang lebih besar. Ada dua algoritma pencocokkan string yang terkenal, yaitu algoritma Knutt-Morris-Pratt dan algoritma Boyer-Moore. Hangman merupakan sebuah permainan kata yang bertujuan untuk menebak kata apa yang dimaksud penanya dengan menebak huruf satu per satu. Dalam makalah ini, akan dibahas bagaimana memecahkan permainan Hangman dengan algoritma pencocokkan string.**

***Kata Kunci*—Pencocokkan String, Hangman**

I. PENDAHULUAN

Makalah ini dibuat karena ketertarikan dan rasa penasaran penulis akan permainan Hangman. Hangman merupakan sebuah permainan sederhana dengan kesulitan yang rumit. Inti dari permainan Hangman ini hanya menebak kata apa yang dimaksudkan oleh penanya dengan cara mengisi kata-kata yang kosong dengan pilihan satu dari 26 huruf yang ada.

Penulis juga kagum tentang algoritma pencocokkan string dikarenakan fungsinya yang lebih dari sekedar mencari string dalam sebuah string yang lebih besar. Dalam implementasinya, algoritma string digunakan sebagai pencocokkan sidik jari, juga untuk semua hal yang berhubungan dengan *recognition* atau pencocokkan.

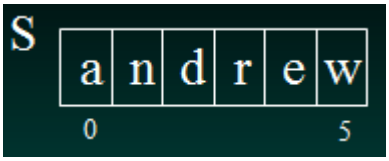
Dalam implementasi sebenarnya, pemecahan masalah hangman tidak hanya menggunakan algoritma pencocokkan string, tapi beragam algoritma juga digunakan, seperti algoritma bruteforce yang nanti akan digunakan untuk menghitung jumlah kata, algoritma greedy dalam pemilihan huruf yang akan diberikan dan masih banyak lagi.

II. ALGORITMA PENCOCOKKAN STRING

Pencocokkan string merupakan sebuah masalah yang sering diketemukan dalam kehidupan sehari-hari. Beberapa Aplikasi yang menggunakan algoritma ini ialah text editor, web search engine, ataupun image analysis.

String merupakan sekumpulan karakter yang tergabung menjadi satu. Substring S[i..j] dari S merupakan bagian dari string antara indeks i sampai j. Sebuah prefiks S merupakan substring dari indeks 0 sampai i. Sedangkan sebuah sufiks S adalah substring antara indeks i sampai indeks terakhir pada string.

Sebagai contoh yaitu sebuah string S = andrew memiliki indeks dari 0 sampai 5.

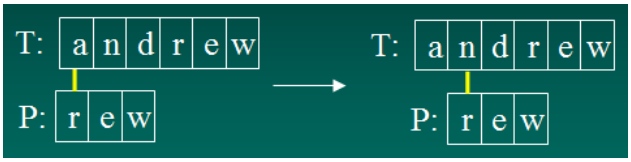


Substring S[1..3] ialah “ndr”. Prefiks yang mungkin dari S ada “andrew”, “andre”, “andr”, “and”, “an” dan “a”. Sedangan sufiks yang mungkin dari S ialah “andrew”, “ndrew”, “drew”, “rew”, “ew”, dan “w”.

Ada beberapa cara populer yang digunakan dalam algoritma pencocokkan string.

Cara pertama yang digunakan ialah algoritma sederhana, yaitu algoritma Brute Force. Cara kedua ialah algoritma Knutt-Morris-Pratt dan ketiga algoritma Boyer-Moore.

Algoritma Brute Force bekerja dengan mencocokkan semua posisi dari teks T untuk melihat apakah string P merupakan substringnya.



Secara iteratif string “rew” digeser dengan membandingkan karakter pertamanya dengan teks T. Jika ternyata berbeda, maka string P digeser dan dibandingkan dengan indeks +1 dari teks T. Pencarian ini dilakukan hingga ditemukan pola P dalam teks T atau hingga pola P tidak dapat digeser lagi dikarenakan sudah mencapai batas akhir teks T.

Algoritma brute force dalam pencocokkan string ini memiliki kompleksitas algoritma O(mn) dalam kasus terburuk, seperti mencari pola “aaaaaaab” dalam teks

“aaaaaaaaaaaaaaaaaaaaaaaaaaab”. M dan n berturut-turut ialah panjang pola dan panjang teks. Sedangkan pada umumnya, pencarian hanya memiliki kompleksitas algoritma O(m+n) yang dapat disebut sangat cepat. Hal ini terjadi ketika jumlah alfabet yang digunakan dalam pencarian sangat banyak, seperti A..Z, a..z, 1..9, dan lainnya. Dan sangat lambat ketika jumlah alfabet dalam pencarian sedikit, seperti 0, 1 (seperti dalam file biner, file gambar, dan lainnya).

Algoritma Knuth-Morris-Pratt (KMP) dikembangkan oleh D.E Knuth, bersama-sama dengan J.H.Morris dan V.R. Pratt. Pada algoritma brute force, setiap kali ditemukan ketidakcocokan pattern dengan teks, maka pattern digeser satu karakter ke kanan. Sedangkan pada algoritma KMP, kita memelihara informasi yang digunakan untuk melakukan jumlah pergeseran. Algoritma KMP menggunakan informasi tersebut untuk membuat pergeseran yang lebih jauh. Tidak hanya satu karakter seperti pada algoritma brute force.

```
                123456789..
Teks:           bimbingan belajar atau bimbel
Pattern:        bimbel
                  ↑
                j = 5
                123456789..
Teks:           bimbingan belajar atau bimbel
Pattern:          bimbel
                    ↑
                  j = 2
```

Misalkan A adalah alphabet dan x=x1x2…xk adalah string yang panjangnya k yang dibentuk dari karakter-karakter di dalam alphabet A. Maka prefiks dari x adalah u = x1x2…xj-1 dan sufiks dari x ialah v = xj-bxj-b+1…xk. Pinggiran (border) dari x adalah substring r sehingga r = x1x2…xj-1 dan u = xj-bxj-b+1…xj, j merupakan himpunan 1,2,…k.

Misalkan *x* = `abacab`.

Awalan dari *x* adalah

□, `a`, `ab`, `aba`, `abac`, `abaca`

Akhiran dari *x* adalah

□, `b`, `ab`, `cab`, `acab`, `bacab`

Pinggiran dari *x* adalah

□, `ab`

Pinggiran □ mempunyai panjang 0, pinggiran `ab` mempunyai panjang 2.

Kompleksitas waktu algoritma KMP ialah :

- Menghitung fungsi pinggiran : O(m)
- Pencarian string : O(n)

Sehingga kompleksitas total dari algoritma KMP ialah O(m+n).

Keunggulan dari KMP ialah kecepatannya yang tinggi. Algoritmanya tidak pernah bergerak mundur dalam teks T. Hal ini membuat algoritma ini bagus untuk pemrosesan file yang sangat besar yang dibaca dari perangkat eksternal atau melalui transfer jaringan.

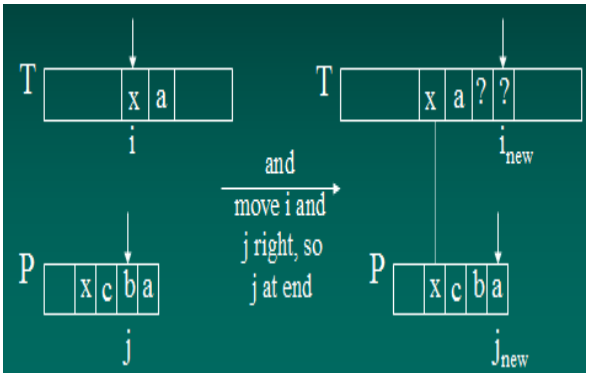
Kejelekkan dari KMP ialah tidak bekerja dengan baik sebagai ukuran dari alfabet yang semakin besar. Semakin sering terjadi ketidakcocokan, dan ketidakcocokan dalam perbandingan sering terjadi pada awal pola, tetapi KMP cepat ketika ketidakcocokkan terjadi pada bagian akhir.

Algoritma Boyer-Moore merupakan algoritma pencarian string dengan dua teknik :

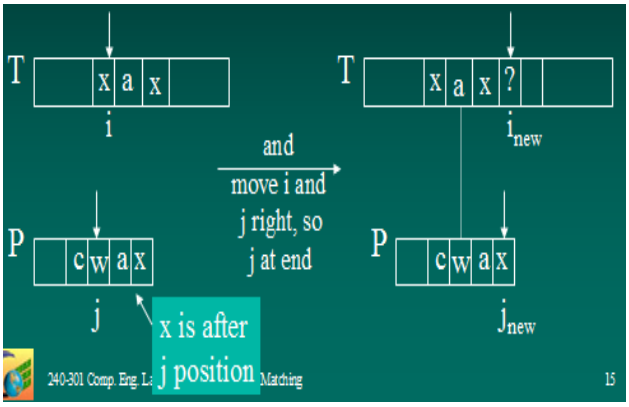
1. Teknik Looking-glass
   - mencari P dalam T dengan bergerak mundur, dimulai pada akhir dari P.
2. Teknik character-jump
   - ketika ketidakcocokkan terjadi pada T[i] == x
   - karakter P[j] tidak sama dengan karakter T[i]

Ada tiga kemungkinan kasus yang dapat terjadi :

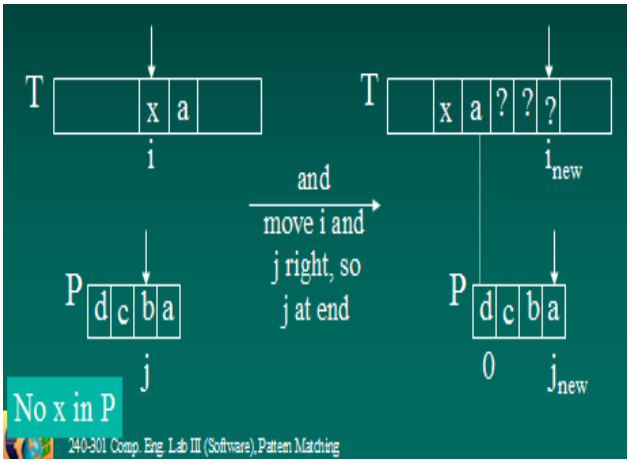
1. Jika P memiliki x di suatu tempat, maka geser P ke kanan hingga kemunculan terakhir x di P dengan T[i]



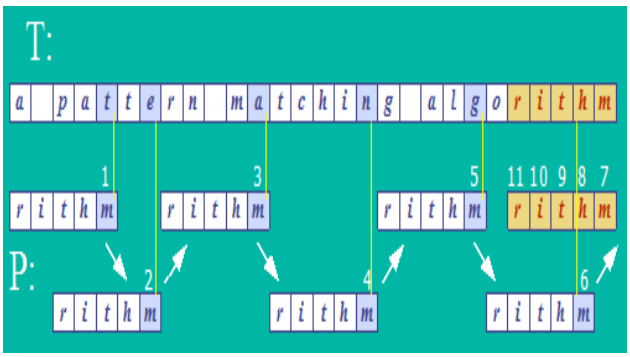
2. Jika P memiliki x di suatu tempat, tetapi pergeseran ke kanan hingga kemunculan terakhir tidak dimungkinkan, maka geser P satu karakter ke T[i+1].



3. Jika kasus pertama dan kasus kedua tidak dapat dilakukan, maka geser P hingga P[0] dengan T[i+1]



Contoh algoritma Boyer-Moore :



Algoritma Boyer-Moore memiliki kompleksitas algoritma untuk kasus terburuk ialah O(nm + A). Tetapi Boyer-Moore cepat ketika alfabet (A) besar, dan pelan ketika alfabet kecil. Boyer-Moore jauh lebih cepat dibandingkan dengan brute force untuk pencarian dengan alfabet yang besar.

III. ALGORITMA BRUTE FORCE

Algoritma brute force adalah sebuah algoritma dengan pendekatan yang lempang untuk memecahkan suatu masalah. Biasanya algoritma ini didasarkan pada pernyataan masalah, dan definisi konsep yang dilibatkan. Algoritma brute force memecahkan masalah dengan sangat sederhana, langsung, dan jelas.

Dalam makalah ini, algoritma brute force yang akan digunakan ialah dengan algoritma pencarian beruntun (Sequential Search)

Sebagai contoh permasalahan ialah : Diberikan sebuah list yang berisi n buah bilangan bulat (a1, a2, ….,an). Carilah nilai x di dalam list tersebut. Jika x ditemukan, maka keluarannya adalah indeks elemen senarai, jika x tidak ditemukan maka keluarannya adalah 0.

Dengan algoritma brute force (sequential search), setiap elemen list dibandingan dengan x. Pencarian selesai jika ditemukan atau elemen list sudah habis diperiksa.

Kompleksitas algoritma brute force untuk sequential search ialah O(n).

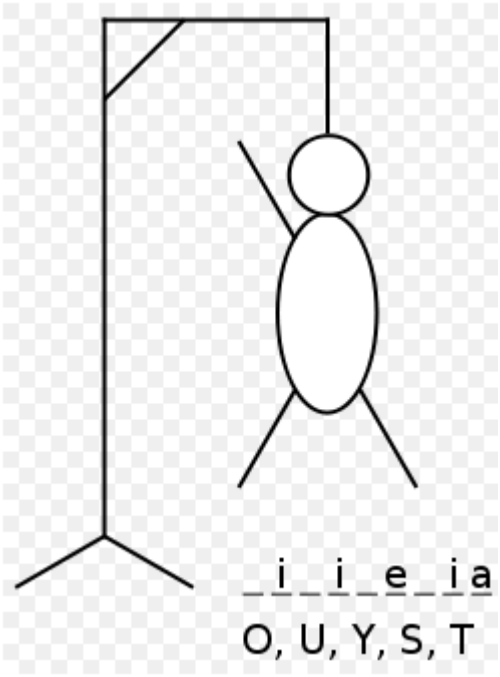
IV. PERMAINAN HANGMAN

A. Deskripsi Permasalahan

Hangman pada dasarnya merupakan sebuah permainan kertas dan pensil sederhana untuk dua atau lebih pemain. Seorang memikirkan huruf untuk ditebak dan yang lain mencoba menebak kata apa yang dipikirkan si pembuat soal. Namun dewasa ini, permainan hangman berkembang menjadi permainan yang menggunakan desktop sebagai media utama permainan. Dengan computer sebagai pembuat soal yang di-generate secara random dan pemain yang berusaha menebak kata yang di-generate.

Kata yang akan di tebak, biasanya direpresentasikan dengan garis sejumlah dengan jumlah huruf yang dimaksudkan. Jika penebak menebak sebuah huruf yang berada dalam kata yang dimaksud, maka huruf tersebut dituliskan menggantikan garis dalam kata. Jika ternyata huruf yang ditebak tidak terdapat pada kata yang dimaksud, maka digambarkan sebuah elemen dari diagram hangman, yaitu kepala, badan, sepasang kaki, dan sepasang tangan, yang memberikan enam kesempatan bagi penebak untuk salah.

Gambar berikut merupakan sebuah contoh permainan hangman yang belum selesai.



Dalam gambar di atas terdapat sembilan digit huruf yang harus ditebak. Dan penebak sudah menebak delapan huruf, dengan 3 huruf i,e, dan a berhasil ditebak, dan huruf o,u,y,s,t gagal ditebak sehingga digambarkan lima elemen dari hangman pada gambar, yaitu kepala, badan, sepasang kaki, dan sebuah tangan. Penebak akan kalah jika ia sekali lagi salah menebak huruf yang

dimaksudkan penanya. Karena lengkap sudah elemen orang pada diagram hangman.

Tidak hanya ketika penebak berhasil menebak kata yang dimaksud, tetapi permainan pun berhenti ketika ia gagal menebak kata yang dimaksud.

B. Strategi

Permainan dapat diselesaikan oleh seorang manusia biasa. Dalam Bahasa Indonesia, setiap huruf pasti mempunyai huruf vokal, a, i, u, e, dan o. Sehingga menebak salah satu dari huruf tersebut memperbesar kemungkinan kita untuk menebak jawaban dari kata yang dimaksud. Ketika kira-kira sudah tertebak satu atau dua huruf vokal, kita dapat melanjutkan dengan menebak huruf kapital yang sering terdapat pada kata-kata dalam Bahasa Indonesia seperti huruf n,m,s,t,r, dan p. Kemudian dengan pengetahuan tentang huruf yang ada, kita dapat mengikuti pola dari kata untuk menebak apa kata yang dimaksud.

C. Contoh Permainan

Berikut ini ialah contoh permainan hangman dimulai dari awal hingga selesai.

0

Word: _ _ _ _ _ _ _

Guess: E

Misses:

1

Word: _ _ _ _ _ _ _

Guess: T

Misses: e

2

Word: _ _ _ _ _ _ _

Guess: A

Misses: e,t

3

Word: _ **A** _ _ _ **A** _

Guess: O

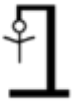
Misses: e,t

4

Word: _ **A** _ _ _ **A** _

Guess: I

Misses: e,o,t

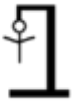


5

Word: _ **A** _ _ _ **A** _

Guess: N

Misses: e,i,o,t



6

Word: _ **A N** _ _ **A N**

Guess: S

Misses: e,i,o,t

7

Word: _ **A N** _ _ **A N**

Guess: H

Misses: e,i,o,s,t

8

Word: **H A N** _ _ **A N**

Guess: R

Misses: e,i,o,s,t

9

Word: **H A N** _ _ **A N**

Guess:

Misses: e,i,o,r,s,t

Guesser loses - the answer was **HANGMAN**.

IV. IMPLEMENTASI ALGORITMA

Sebelumnya penulis telah membahas mengenai algoritma yang dipakai dalam pencocokkan string dan juga mengenai permainan hangman. Dalam bagian ini penulis mencoba membahas bagaimana cara pengaplikasian algoritma pencocokkan string dalam permainan hangman.

Pada permulaan, kita membutuhkan sebuah kamus yang berisi dengan kata-kata. Kamus ini disusun berdasarkan jumlah huruf dan alfabet untuk memudahkan pencarian jawaban.

Kita juga membutuhkan sebuah array untuk menampung kata-kata setelah difilter dengan menggunakan algoritma pencarian string.

Sebelum kita masuk pada penjelasan algoritma yang akan dipakai dalam pemecahan masalah hangman, kita membutuhkan sebuah prosedur untuk mencari nilai rata-rata kemunculan karakter pada setiap string. Prosedur ini dapat di-generate dengan menggunakan algoritma brute force sequential search pada penjelasan sebelumnya. Prosedur ini akan mengembalikan karakter dengan kemunculan rata-rata tertinggi untuk setiap stringnya.

Pemecahan masalah dimulai dengan membandingkan jumlah panjang string yang akan ditebak dengan jumlah string dalam kamus. Hal ini dapat dilakukan dengan mudah dikarenakan kamus sudah terurut berdasarkan jumlah panjang string. Kumpulan string yang didapat setelah perbandingan ini, kita letakkan pada array untuk mempermudah pencarian, mengurangi biaya untuk mengakses storage setiap kali proses perbandingan akan dilakukan.

Langkah pertama sudah kita lewati, kemudian secara rekursif kita akan melakukan hal berikut. Pertama kita menggunakan prosedur pencarian kemunculan rata-rata tertinggi karakter dari string pada array, dan menebak kata pertama dengan huruf yang didapatkan. Hal ini bertujuan memperbesar kemungkinan penebakkan kata pertama yang mendekati solusi akhir. Ada dua kondisi yang dapat terjadi :

1. Kata yang dimaksud tidak terdapat pada solusi akhir,
2. Kata yang dimaksud ada pada solusi akhir.

Jika kita mendapatkan hasil seperti pada nomor 1, maka kita lanjutkan dengan pemilihan kata dengan urutan kemunculan rata-rata tertinggi nomor dua, dan bila gagal lagi dilakukan dengan kata ketiga, keempat, dan seterusnya.

Jika kita mendapatkan hasil seperti pada nomor 2, fungsi rekursif kembali dilanjutkan dengan melakukan pencocokkan string dari pola solusi akhir yang telah didapat dengan sisa string pada array. Hasil yang didapatkan dimasukkan kedalam array, dan menghapus semua string yang tidak cocok dengan pola yang dimaksud.

Fungsi rekursif ini dilakukan hingga jawaban didapat, atau kesempatan menebak sudah habis.

Algoritma pencocokkan string yang digunakan bisa berupa brute force, KMP, maupun Boyer-Moore, namun penggunaan algoritma brute force dirasa cukup dalam pemecahan masalah ini.

V. KESIMPULAN

Kesimpulan dari makalah ini ialah dengan algoritma pencocokkan string, permainan hangman dapat dengan mudah diselesaikan, walaupun kemungkinan gagal menebak juga ada dikarenakan terlalu banyak kemungkinan yang harus dieliminasi. Algoritma Brute Force sederhana dirasa cukup dalam pengimplementasian program hangman solver ini.

REFERENCES

[1] 240-301 Comp. Eng. Lab III (Software), Pattern Matching
[2] http://www.informatika.org/~rinaldi
[3] http://www.wikipedia.org/

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, 29 April 2010

Andrei Dharma K/13508009