PENERAPAN ALGORITMA RUNUT BALIK DALAM PERMAINAN TEKA-TEKI SILANG

Imaduddin Amin

Departemen Teknik Informatika Institut Teknologi Bandung
Jalan Ganesha No 10 Bandung Indonesia
e-mail: if15067@students.if.itb.ac.id

ABSTRAK

Permainan teka-teki silang (TTS) merupakan permainan asah otak yang sangat diminati. Dalam permainan ini, pemain akan diminta untuk mengisi kotak-kotak di papan permainan. Permainan akan dinyatakan selesai jika pemain mampu mengisi semua kotak-kotak pada papan permainan. Papan permainan sendiri tediri atas kotak-kotak 2 warna biasanya berwarna hitam dan putih. Kotak-kotak ini akan membentuk deretan kotak tempat suatu kata yang merupakan jawaban dari soal akan diisi.

Setiap deretan kotak dapat terhubung dengan deretan kotak lain baik secara mendatar maupun tegak. Setiap deretan kotak yang terhubung dengan deretan kotak lain akan memiliki kesamaan isi karakter pada tempat terhubungnya kotak-kotak tersebut. Hal inilah yang menimbulkan kesulitan dalam permainan ini, sekaligus kesulitan dalam membua soal-soal yang pas dan bersesuaian dengan kotak-kotak permain yang diberikan.

Oleh karena itu, dalam makalah ini, akan disajikan salah satu metode yaitu algoritma runut balik yang bisa digunakan sebagai salah satu cara membuat soal (mencari kata-kata yang pas utuk digeneratekan kedalam kotak-kotak yang diberikan) dalam permainan TTS ini.

Algoritma runut balik dalam program akan mengecek kotak jawaban yang telah disediakan oleh pembuat dan mengecek *data base* kata yang tersedia. Program akan mengecek apakah terdapat kata yang layak dalam *data base* kata untuk dimasukkan kedalam kotak-kotak permainan.

Kata kunci: Algoritma runut-balik, Teka-teki Silang,

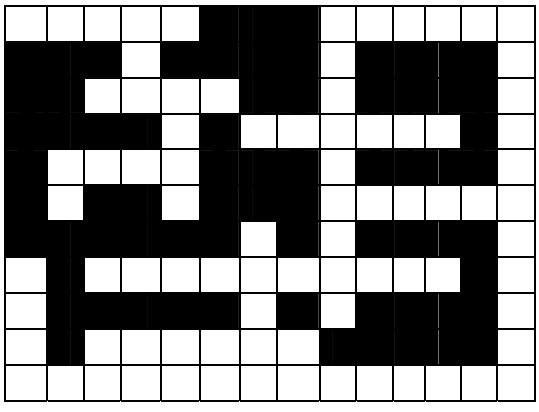
1. PENDAHULUAN

Permainan teka-teki silang merupakan salah satu permainan asah otak yang diminati banyak orang. Teka Teki Silang atau disingkat TTS adalah suatu permainan di mana kita harus mengisi ruang-ruang kosong (berbentuk kotak putih) dengan huruf-huruf yang membentuk sebuah kata berdasarkan petunjuk yang diberikan. Petunjuknya biasa dibagi ke dalam kategori 'Mendatar' dan 'Menurun' tergantung posisi kata-kata yang harus diisi. (Wikipedia)

Dalam permainan teka-teki silang terdapat papan permainan utama. Papan permainan sendiri terdiri atas kotak-kotak berwarna hitam dan putih.Sebagai mana telah dijelaskan bahwa kotak-kotak putih yang membentuk deretan blok baik mendatar maupun menurun merupakan tempat pemain mengisi jawaban. Setiap deretan kotak akan mempunyai nomor dan soal yang diberikan. Permainan akan dinyatakan selesai jika, pemain mampu mengisi semua deretan kotak-kotak putih mendatar dan menurun tersebut.

Permainan ini memang cukup mudah untuk dimainkan, namun sayangnya untuk dapat membuat soal yang valid merupakan hal yang sulit. Untuk itu dalam makalah ini akan dijelaskan penyelesaian masalah tersebut dengan bantuan program komputer. Pembuat soal cukup memasukkan *data base* berupa kata-kata jawaban berikut soalnya dan membuat deretan-deretan kotak putih tempat jawaban di papan permainan.

Salah satu cara untuk menyelesaikan permasalahan tersebut adalah dengan menggunakan algoritma runut balik. Algoitma runut balik (*back tracking*) akan mampu memberikan hasil apakah deretan-deretan kotak jawaban yang telah dibuat sudah cocok dengan deretan jawaban kata yang disediakan.



Gambar 1.1 Papan permainan Teka-teki Silang

2. ALGORITMA RUNUT BALIK DAN PENERAPANNYA

2.1 Algoritma Runut Balik

Algoritma runut balik pertama kali diperkenalkan oleh D.H. Lehmer pada tahun 1950. Algoritma ini cukup mangkus untuk digunakan dalam beberapa penyelesaian masalah dan juga untuk memberikan kecerdasaan buatan dalam *game*. Beberapa *game* populer semisal Sudoku, Labirin, Catur juga bisa diimplementasikan dengan menggunakan algoritma runut balik.

Algoritma runut balik (*back tracking*) merupakan algoritma yang digunakan untuk mencari solusi persoalan secara lebih mangkus daripada menggunakan algoritma *brute force*. Algoritma ini akan mencari solusi berdasarkan ruang solusi yang ada secara sistematis namun tidak semua ruang solusi akan diperiksa, hanya pencarian yang mengarah kepada solusi yang akan diproses. . (Rinaldi Munir, Diktat Srategi Algoitmik, Teknik Informatika ITB, 2005)

Algoritma Runut Balik berbasis pada DFS (*Depth First Search*) sehingga aturan pencariannya akan mengikut kepada aturan pencarian DFS yaitu dengan mencari solusi dari akar ke daun (dalam pohon ruang solusi) dengan pencarian kedalam. Simpul-simpul yang sudah dilahirkan (diperiksa) dinamakan sipmup hidup (*live node*). Simpul hidup yang sedang diperluas dinamakan simpul-E atau *Expand Node*.

Dalam diktat Strategi Algoritmik Teknik Informatika ITB oleh Rinaldi Munir, dijelaskan bahwa algoritma runut balik memiliki property umum yaitu :

- Solusi persoalan
  Solusi dinyatakan sebagai vektor dengan *n-tuple*:
  $X = (x_1, x_2, \ldots, x_n)$, $x_i \in S_i$.
  Mungkin saja $S_1 = S_2 = \ldots = S_n$.
  Contoh: $S_i = \{0, 1\}$, $x_i = 0$ atau 1
- Fungsi pembangkit nilai xk
  Dinyatakan sebagai:
  T(k)
  T(k) membangkitkan nilai untuk xk, yang merupakan komponen vektor solusi.
- Fungsi pembatas (pada beberapa persoalan fungsi ini dinamakan fungsi kriteria)
  Dinyatakan sebagai B(x1, x2, …, xk)
  B bernilai true jika (x1, x2, …, xk) mengarah ke solusi. Jika true, maka pembangkitan nilai untuk xk+1 dilanjutkan, tetapi jika false, maka (x1, x2, …, xk) dibuang dan tidak dipertimbangkan lagi dalam pencarian solusi.

Solusi persoalan adalah kemungkinan solusi yang didapatkan dari permasalahan yang diberikan, sedangakan fungsi pembatas merupakan fungsi yang akan menentukan langkah selanjutnya berupa penerusan pencarian solusi ataupun melakukan *backtrack*.

2.2 Penggunaan Algoritma Runut Balik dalam permainan Teka-Teki Silang

Dalam penyelesaian kasus pembuatan soal teka-teki silang ini dapat diterapkan algoritma *brute force*. Algoritma *Brute force* adalah sebuah pendekatan yang lempang (straightforward) untuk memecahkan suatu masalah, biasanya didasarkan pada pernyataan masalah (problem statement) dan definisi konsep yang dilibatkan. (Rinaldi Munir, Diktat Srategi Algoitmik, Teknik Informatika ITB, 2005)

Namun sayangnya hal ini tidak efisien karena semua kemungkinan akan dicoba. Sebagai contoh : Jika terdapat **n** buah kata dalam *data base* kata dan terdapat **m** jumlah deretan koak yang harus diisi maka jumlah proses untuk setiap deretan kata, harus dicoba sebanyak **n!** dari keseluruhan katadidalam *data base*, sehingga jumlah proses yang diperlukan adalah **m x n !** dan kompleksitas algoritmanya menjadi **0(mn!)**. Untuk itulah diperlukan algoritma lain yang lebih efisien, dalam hal ini salah satunya adaalah algoritma runut balik.

Algoritma runut balik dalam permainan ini akan digunakan untuk mengisi kotak-kotak permainan yang sebelumnya telah dibuat. Kotak-kotak ini bisa direpresentasikan dengan struktur data matriks sehingga setiap kotak akan memiliki indeks. Indeks ini akan digunakan untuk melakukan pencarian kata yang cocok

Pada pengisian kata kedalam kotak-kotak, pertama-tama program akan menentukan deretan kotak awal yang ingin diisi. Program akan menghitung jumlah kotak pada deretan kotak tersebut kemudian akan mencari kata (didalam *data base* yang terdiri atas kumpulan kata (jawaban)) yang memiliki jumlah karakter sama dengan jumlah kotak tersebut.

Dalam pencarian data kata-kata mungkin akan terdapat beberapa kata yang cocok untuk dimasukkan kedalam satu deretan kotak, untuk itu program akan memilih kata yang berada lebih awal dalam *data base* kata. Langkah selanjutnya, program akan mengidentifikasi indeks pada deretan kotak yang terhubung dengan deretan kotak lainnya. Program akan mencatat dimana letak hubungan antar deretan kotak tersebut kemudian mencatat indeks dan mengambil karakter yang terdapat di dalamnya untuk dibandingkan kembali dengan deretan kata yang ada di dalam *data base* kata.

Jika kata yang dimasukkan berikutnya cocok maka pencarian akan dilanjutkan, namun jika tidak terdapat kata yang cocok maka program akan mematikan kemungkinan jawaban berdasarkan pencarian tersebut dan program akan melakukan *backtrack*.

Backtrack dilakukan dengan cara program akan menghapus kata yang terakhir dimasukkan kedalam deretan kotak, kemudian program akan mengganti kata tersebut dengan kata lain yang juga bisa diisikan kedalam deretan kotak tersebut dan kemudian program akan melakukan pencarian ulang.

Langkah-langkah diatas akan terus dilakukan secara rekursif, sampai program menemukan solusi dari permasalahan (seluruh kotak terisi) atau program tidak menemukan solusi (tidak ada kemungkinan jawaban yang valid).

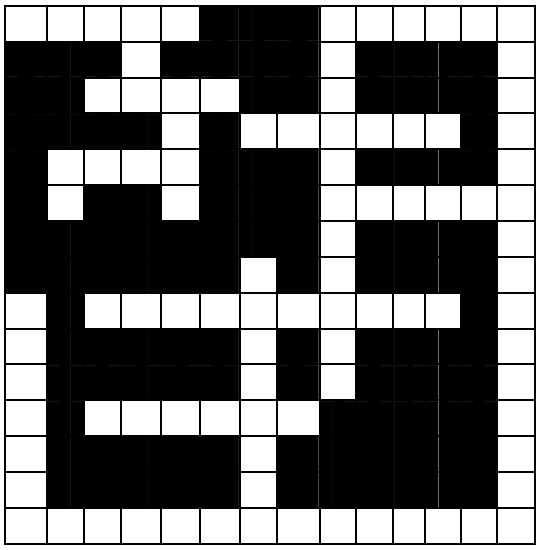
```
//Pseudocode

//Kamus
Datakata array of String;
Integer kolom, baris;
Papan array of integer of integer;
//algoritma
Procedure tampilkanpapan();
//tidak diidefinisikan
Prosedure membuatkotak();
//tidak didefinisikan
Prosedure membuatdatabase();
//membuat array dari masukan kata
Prosedure caridata(integer);
//mencari kata berukuran masukan di
data base, dengan iteratif
Prosedure kirimdata();
//mengirim data ke papan
Prosedure dapatkanukuran(integer
indeks)
//mendapatkan ukuran dari deretan kotak
Procedure inputmasukan()
Begin
String kata;
Integer I = 0;
While(kata.equal("selesaiiiiii")
Begin
      Input(kata);
      Datakata[i] = kata;
      I++;
End;

Prosedure backtrack(integer indeks)
Kamus
     Boolean solusi;
Begin
      While (indeks>0) && (!gerak)
      Begin
          Dapatkanukuran(papan,indeks);
          Caridata(ukuran)
          If (caridata==null)
          Begin
            Backtrack(indeks--);
          Geraktrue();
          End;
      Else
          If (Ceklayak())
          Begin
              Kirimdata();
              Indeks++;
          End;
          Else
              Randomkata();
              Ceklayak();
          End;
      End;
End;
```

Contoh pencarian solusi:

- Terdapat *data base* kata Z dan papan permainan

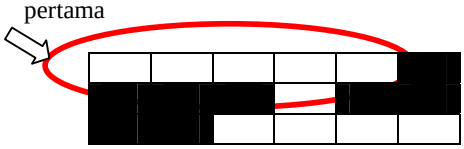
No	Kata	Jumlah Karakter
1	Kuliah	6
2	Di	2
3	Informatika	12
4	masih	5
5	Menyenangkan	12
6	Namun	5
7	Mahasiswa	9
8	tidak	5
9	Menyadarinya	5
10	Termasuk	8
11	ane	3
12	Itu	3
13	dulu	4

Tabel 2.2.1 Data base kata permainan



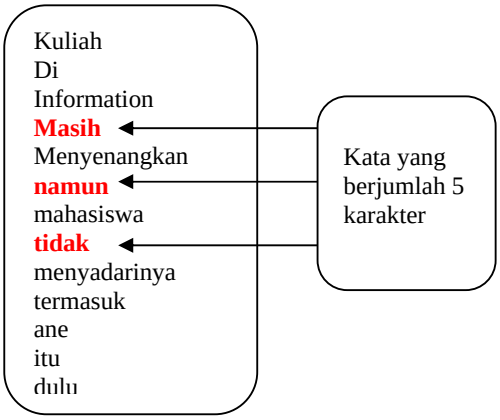
Gambar 2.2.1 Kondisi awal kotak permainan

- Program akan menghitung jumlah kotak pada deretan pertama

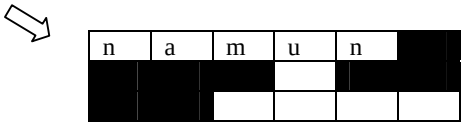


Gambar 2.2.2 Deretan kotak yang dicek panjangnya

- Program akan mencari di *data base*, kata yang berjumlah 5 karakter.



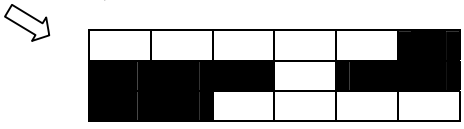
- Program akan memilih kata secara acak dan memasukkannya ke dalam deretan kotak.



Gambar 2.2.3 kotak permainan setelah dimasukkan kata "namun"

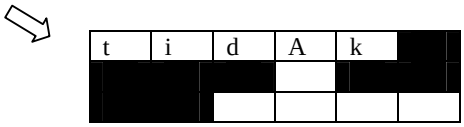
- Program akan melakkan pencarian berikutnya terhadap deretan kotak kedua

- Program tidak dapat menemukan kata yang memungkinkan (berjumlah 3 karaker dan dimulai dengan huruf 'u') sehingga program akan melakukan *backtrack*. (menghapus kata yang sebelumnya diisikan)



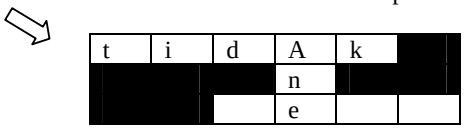
Gambar 2.2.4 kotak permainan setelah dilakukan *backtrack*

- Langkah selanjutnya program akan memilih kata yang lain untuk deretan pertama yang memenuhi sarat, dalam hal ini adalah kata "tidak".



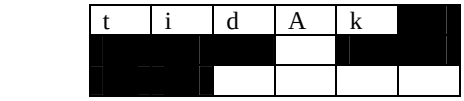
Gambar 2.2.5 kotak permainan setelah dimasukkan kata "tidak"

- Program melakukan pencarian berikutnya di dalam *data base*. Program menemukan kata "ane" dan kembali dimasukkan kedalam kotak permainan.



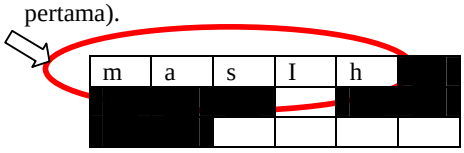
Gambar 2.2.6 kotak permainan setelah dimasukkan kata "ane"

- Pencarian berikutnya dilakukan namun program tidak menemukan kata yang cocok sehingga program akan melakukan *backtrack.*



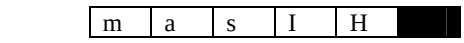
Gambar 2.2.7 kotak permainan setelah dilakukan *backtrack*

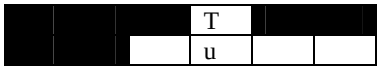
- Karena tidak ada kata lain yang dapat diisikan kembali maka program akan melakukan *backtrack* lagi dan mengganti isi deretan kota pertama dengan kata "masih" (kata yang cocok untuk deretan kotak pertama).



Gambar 2.2.8 kotak permainan setelah dilakukan *backtrack* dan kemudian diisi kembali dengan kata "masih"

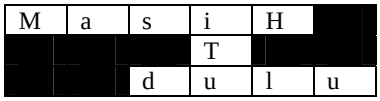
- Program akan mencari kembali kata yang cocok. Dalam hal ini program akan mendapatkan kata "itu".





Gambar 2.2.9 kotak permainan setelah dimasukkan kata "itu"

- Langkah selanjutnya program akan menemukan kata "dulu" di dalam data base kata dan kembali memasukkannya kedalam deretan kotak yang tersedia.



Gambar 2.2.10 kotak permainan setelah dimasukkan kata "dulu"

- Langkah-langkah diatas diulang sampai program selesai menemukan solusi atau sampai solusi tidak mungkin ditemukan. Jika solusi tidak ditemukan maka, program akan meminta user untuk memasukkan kata-kata baru ke dalam *data base* kata kemudian program akan melakukan pencarian solusi ulang.

3. KESIMPULAN

Dalam permainan teka-teki silang ini, algoritma runut balik sudah bisa memberikan jawaban yang pasti sehingga algoritma runut balik ini bisa diimplementasikan. Selain itu algoritma runut balik juga merupakan algoritma yang sederhana namun cukup mangkus. Hal ini disebabkan karena pada prinsipnya, kita tidak perlu memeriksa semua kemungkinan solusi yang ada. Pencarian hanya mengarah pada solusi yang dipertimbangkan saja.

REFERENSI

[1] Munir, Rinaldi. 2005. Strategi Algoritmik. Teknik Informatika ITB : Bandung.
[2] http://id.wikipedia.org/w/index.php?title=Teka-teki_Silang&redirect=no, tanggal akses : 21 mei 2007 pukul 15.00