Implementasi Algoritma BFS dalam permainan Rangkai Kata

Fajar Dwi Anggara

Departemen Teknik Informatika Institut Teknologi Bandung
Jalan Ganesha No 10 Bandung Indonesia
e-mail: if15039@students.if.itb.ac.id

ABSTRAK

Permainan kata merupakan salah satu jenis permainan yang menarik karena dapat menantang pemainnya untuk mengeluarkan kreativitasnya dalam mengolah kata. Rangkai kata merupakan salah satu contohnya. Rangkai kata merupakan kata acak yang dibentuk dari kata lain yang telah diubah susunan atau jumlah karakter di dalamnya. Pada pengolahan kata ini, pemain akan memasukan kata awal yang ingin dibangkitkan Rangkai kata nya. Kemudian pemain akan menebak kata yang merupakan Rangkai kata dari kata awal. Untuk menemukan semua solusi berupa sekumpulan kata ini, kami mencoba membangkitkan Rangkai katanya dengan algoritma Breadth First Search. Algoritma Breadth First Search yang mengeksplorasi setiap cabang dari setiap node sangat tepat untuk diterapkan pada permasalahan ini. Algoritma ini akan membangkitkan pohon solusi secara dinamis bersamaan dengan dilakukannya proses pencarian solusi.

Kata kunci: Rangkai kata, Breadth First Search, acak.

1. PENDAHULUAN

Teknologi *game* kini berkembang pesat sejalan dengan perkembangan teknologi informasi. Tidak semua *game* hanya mengandalkan ketangkasan pemain dalam menggerakkan tetikus atau menekan tombol-tombol *keyboard*. Ada *game* yang mengandalkan kemampuan logika dan kreativitas pemain untuk menemukan solusinya. Permainan kata Rangkai kata ini menuntut kemampuan dari pemain dalam mengolah kata. Tujuan dari permainan ini adalah mencari kata yang bisa dibangkitkan dari kata awal.

Algoritma Breadth First Search saya implementasi dalam permainan ini sendiri adalah algoritma penelusuran pohon dengan cara mengunjungi semua anak pada kedalaman tertentu. Kedalaman pohon tidak akan bertambah apabila anaknya belum dikunjungi semua.

2. METODE

2.1 Terminologi

1. Status persoalan (*problem state*)
Simpul-simpul di dalam pohon dinamis yang memenuhi kendala (*constraints*).
2. Status solusi (*solusi state*)
Semua simpul telah dikunjungi.
3. Ruang solusi (*solution space*)
Himpunan solusi.

2.2 Peraturan permainan

Pertama pemain akan diminta untuk memasukkan kata yang ingin dibangkitkan Kata Acaknya. Setelah itu pemain akan memasukkan tebakan kata yang diasumsikan sebagai kata yang dibangkitkan dari kata awal. Pemain akan mendapat nilai apabila kata tebakannya merupakan salah satu kata yang dibangkitkan oleh algoritma BFS. Nilai yang didapat bergantung dari jumlah karakter kata tebakan. Semakin banyak karakter tebakan, semakin sulit dan apabila jawaban benar, pemain akan memperoleh nilai yang tinggi pula. Dalam implementasinya, diperlukan struktur data Queue dan pohon.

2.3 Graf Ruang Solusi

Agar lebih jelas, akan digunakan contoh untuk menentukan Graf Ruang Solusinya

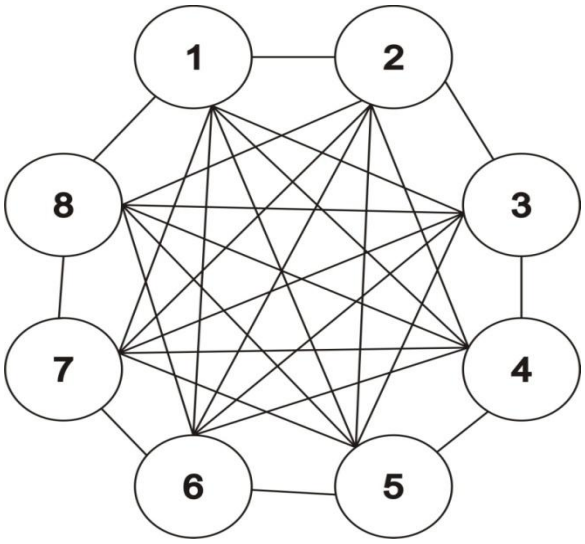
Misal :
Kata awal yang ingin dibangkitkan : katakana
Tebakan pemain : anak

Setiap karakter dalam kata awal diwakili oleh satu symbol status. Dalam hal ini setiap karakter diberi penomoran sebagai berikut :

Tabel 1 Tabel pemetaan status karakter

K	A	T	A	K	A	N	A
1	2	3	4	5	6	7	8

Dari penomoran status di atas dapat dibentuk graf ruang solusi sebagai berikut :



Gambar 1. Graf Ruang Solusi

Setelah melihat graf tersebut, terdapat banyak sekali kemungkinan solusi, maka saya menggunakan algoritma Breadth First Search untuk mencari solusinya.

Pseudo Code dan prosedur

```
Procedure WordUnscramble (String tebakan)
        Push nodeAwal ke Queue
        n=0
        While (n<jmlAllNodes) and
(not(isSolusi(CurrentNode)))
                Pop kepala Queue
                Bangkitkan anak
                m = 0
                While(m<jmlAllNodes-1 and
        not(node))
                        If(isBagian(Node,Tebakan))
        then
                                Push ke Queue
                        Else Nextm
                Endwhile
        Endwhile

        if (isSolusi(CurrentNode))
        then Solusi ditemukan
        else
        Solusi tidak ditemukan
```

```
Function isBagian(input : kata, tebak : string)-
>Boolean
        i=0
        cek :boolean
        cek<-true
        While(i<length)and(cek)
                If(not(katai = tebaki)) then
                        cek<-false
                else next i
        EndWhile
        ->cek
```

Penjelasannya adalah sebagai berikut

Fungsi Utama :

Prosedur WordUnscrambler(Tebakan : String, node: int)

Prekondisi :

Dalam prosedur ini kami mengambil asumsi bahwa setiap node telah merepresentasikan karakter yang diwakilinya, Perubahan dari status ke status (dari node ke anaknya) diasumsikan sebagai operasi string untuk membentuk kata. Kemudian kata tebakan harus lebih kecil dari kata awal.

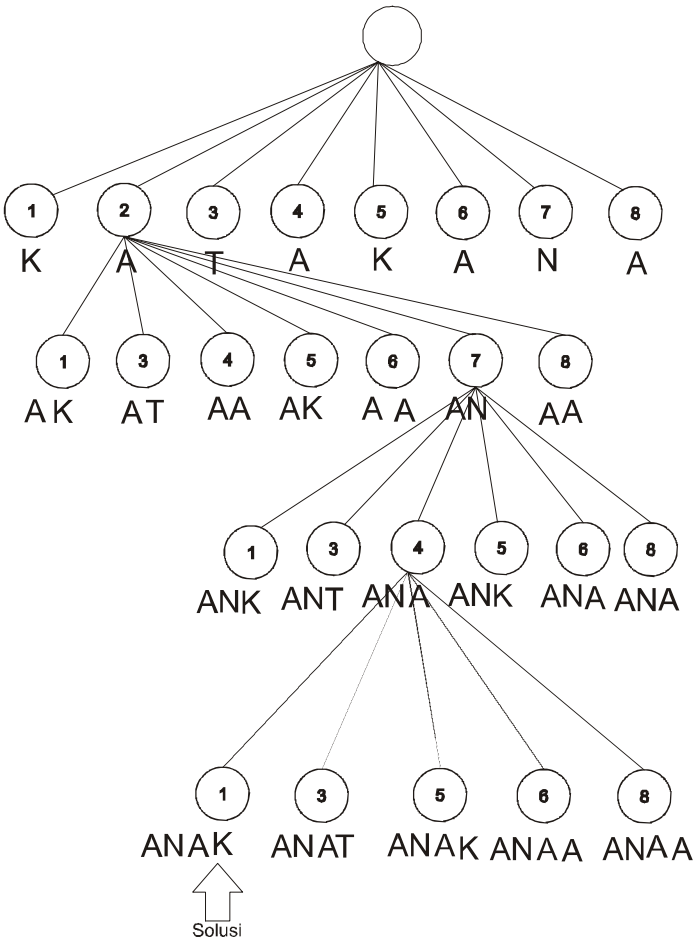
1. Masukkan node awal ke Queue
2. Set Kedalaman dengan 0
3. Ambil kepala Queue
4. Bangkitkan semua anak
5. Jika anak memenuhi syarat push ke Queue
6. Jika memenuhi solusi stop, jika tidak ulangi langkah 3 sampai kedalaman maks
7. Cetak solusi dan perhitungan skor berdasarkan kedalaman

Fungsi Bantuan :

1. Bangkitkan anak
   Prosedur ini membangkitkan anak dari Node dengan elemen status yang tersisa

2. Fungsi isBagian(input : String,String)
   Fungsi ini memeriksa apakah Node tersebut memenuhi syarat atau merupakan bagian kata dari tebakan

Sebagai contoh, Saya akan menggunak contoh persoalan yang telah ada di atas. Secara umum pohon yang dibentuk adalah sebagai berikut :

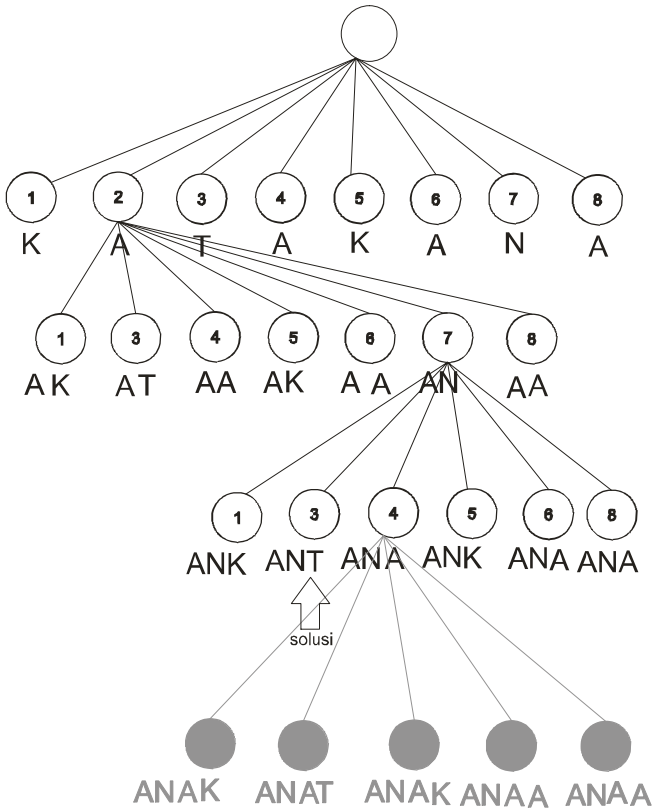


Gambar 2 Pohon dinamis pencarian solusi ANAK

Pada kedalaman 1, anak yang memenuhi kriteria akan dimasukan ke dalam queue. Lalu dari anak yang masuk queue tadi dibangkitkan anaknya. Dalam gambar di atas status 2 akan dimasukkan ke queue karena memenuhi criteria. Namun sebenarnya yang dibangkitkan anaknya bukan cuma status 2, namun juga status 4,6,8. Namun karena status-status itu merepresentasikan hal yang sama, maka tidak saya gambarkan, karena saya anggap hasilnya akan sama dengan anak-anak yang dibangkitkan oleh status ke 2.Setiap kedalaman akan dilakukan operasi yang sama, sampai kedalaman maksimal atau saat CurrentNode adalah solusi.

Dengan melihat gambar di atas terlihat jelas keuntungan BFS dalam pencarian solusi ini, karena BFS mengunjungi anak secara melebar. Sehingga untuk kasus dimana tebakan yang lebih sedikit dimasukkan pohon statsu akan berhenti pada node tersebut tanpa harus melangkah lebih dalam.

Misalkan kata tebakan yang dimasukkan adalah ANT



Gambar 3 Pohon dinamis pencarian solusi ANT

IV. KESIMPULAN

Algoritma Breadth First Search saya nilai masih belum mangkus untuk menyelesaikan persoalan ini. Karena untuk mencari satu solusi saja, algoritma ini harus mengunjungi semua anak yang dibangkitkan. Algoritma yang lebih cocok untuk diterapkan dalam persoalan ini adalah Depth First Search yang menelusuri anak tertentu saja yang telah memenuhi kriteria. Dari anak tersebut akan dicari sampai pada kedalaman yang sesuai dengan jumlah karakter tebakan. Karena solusi yang dicari hanyalah satu, sebenarnya yang harus dikunjungi tidak harus semua anak yang dibangkitkan, maka algoritma BFS saya nilai tidak mangkus dalam hal ini. Namun untuk melihat bagaimana kata itu dibentuk, algoritma BFS dengan sangat jelas bisa menunjukkannya, karena akan menampilkan semua solusi

REFERENSI

[1] Munir, Rinaldi. "*Strategi Algoritmik*", Lab Ilmu Rekayasa Komputasi, Departemen Teknik Informatika ITB.2005