Boggle Solver dengan Algoritma Depth First Search Dalam Java

Gerry Kastogi – 13513011
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung Jl. Ganesha 10 Bandung 40132, Indonesia
13513011@std.stei.itb.ac.id

Abstrak – Algoritma Depth First Search atau yang lebih sering dikenal dengan DFS adalah algoritma yang digunakan untuk pencarian dalam struktur data Page 1 of 6berupa pohon maupun graph. Pada makalah ini akan diulas mengenai penggunaan algoritma DFS untuk menyelesaikan permainan 'Boggle'. Adapun 'Boggle' merupakan sebuah permainan kata seperti scrabble yang tujuannya adalah mencari semua kata yang mungkin dibentuk. Kata yang dapat dibentuk adalah kata yang terdapat pada kamus.

Selain itu pada makalah ini juga akan diulas mengenai alasan mengapa boggle solver ini menggunakan algoritma Depth First Search, bukan menggunakan algoritma Breadth First Search.

Kata kunci – Algoritma, Depth First Search, DFS, Pohon, Graph, Boggle, Scrabble, Breadth First Search.

1. PENDAHULUAN

1.1 Latar Belakang

Permainan Boggle merupakan permainan kata – kata. Tujuannya adalah menemukan semua kata yang mungkin dibentuk dari sekumpulan huruf yang direpresentasikan dalam bentuk matriks. Pada dasarrya cara bermain Boggle sama dengan cara bermain Scrabble.

Kesulitan pada permainan Boggle akan meningkat seiring jumlah kolom dan baris pada matriks yang digunakan untuk menyimpan huruf – huruf. Oleh karena itu Boggle solver ini diciptakan untuk membantu menyelesaikan permainan ini berapapun banyaknya kolom dan baris matriks.

Untuk membuat Boggle solver ini ada beberapa algoritma yang dapat digunakan seperti algoritma *Brute Force, DFS, BFS,* maupun *Branch and Bound.* Namun kali ini akan dibahas satu dari banyak cara penyelesaian Boggle.

1.2 Tujuan

Tujuan dibuatnya makalah ini adalah untuk membantu memahami bagaimana cara penggunaan algoritma *Depth First Search* pada persoalan nyata yang dihadapi sehari – hari. Dalam makalah kali ini permainan Boggle. Selain itu makalah ini juga akan membahas lebih dalam pada permasalahan seperti apa algoritma DFS sebaiknya digunakan.

2. LANDASAN TEORI

2.1 Algoritma Depth First Search

Algoritma Depth First Search atau DFS merupakan algoritma yang digunakan untuk pencarian dalam struktur data berupa pohon maupun graph. Pencarian dengan metode DFS merupakan pencarian yang dilakukan dengan menelusuri terlebih dahulu suatu pohon atau graph sedalam – dalamnya. Artinya algoritma ini akan melakukan pencarian sampai simpul paling dalam. Setelah tidak bisa dilanjutkan barulah algoritma DFS melakukan backtrack ke simpul – simpul tetangga.

Kompleksitas waktu algoritma DFS adalah $O(b^m)$ dimana b adalah jumlah sumpul dan m adalah kedalaman maksimum dari pohon atau graph. Kompleksitas waktu tersebut berlaku pada kasus terburuk.

Sedangkan kompleksitas ruang dari algoritma DFS pada kasus terburuk adalah O(bm) karena kita hanya perlu menyimpan satu buah lintasan tunggal dari akar sampai daun, ditambah dengan simpul – simpul tetangga yang belom dikembangkan.

Algoritma Depth First Search dapat diimplementasikan dengan dua cara yaitu diimplementasikan secara rekursif dan tidak dengan rekursif.

Berikut adalah pseudocode untuk algoritma Depth First Search secara umum :

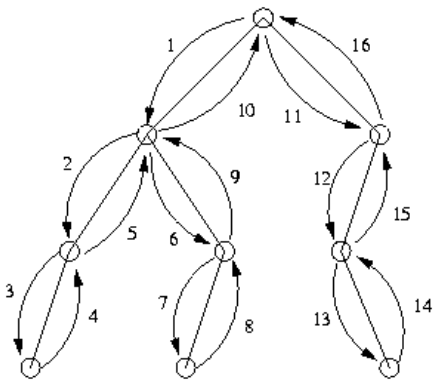
Implementasi algoritma DFS dengan rekursif :

```
procedure DFS(G,v){
  label v as discovered
  for all(edges from v to w in
  G.adjacentEdges(v)) do{
    if (vertex w is not labeled as
    discovered) then
       recursively call DFS(G,w)
  }
}
```

Implementasi algoritma DFS tanpa menggunakan rekursif

```
procedure DFS-iterative(G,v){
  let S be a stack
  S.push(v)
  while S is not empty{
    v = S.pop()
    if (v is not labeled as
    discovered)
      label v as discovered
      for all (edges from v to w in
      G.adjacentEdges(v)) do{
        S.push(w)
      }
}
```

Berikut ilustrasi bagaimana cara penalaran dengan metode DFS

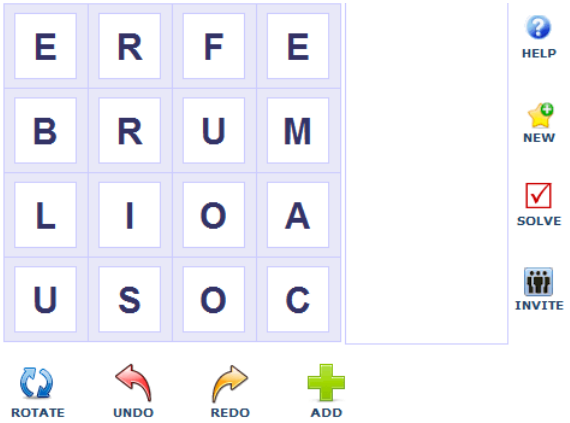


Gamber 2.1 DFS

2.2 Boggle

Boggle merupakan sebuah permainan kata – kata yang didesain oleh Alan Turoff dan diperkenalkan oleh Parker bersaudara. Awalnya permainan ini dimainkan menggunakan dadu plastik yang bertuliskan huruf huruf tertentu. Namun sekarang sudah permainan Boggle ini sudah dimainkan dengan berbagai cara maupun bentuk. Bahkan boggle sudah ada di berbagai platform seperti pc maupun android dengan berbagai macam perubahan.

Untuk memainkan Boggle pertama – tama kita harus menempatkan huruf – huruf yang telah diacak di tempat yang disediakan. Misalkan pada matriks 4x4 atau 5x5. Setelah itu cari semua kata yang mungkin dibentuk dari matriks huruf tersebut.



Gambar 2.2 Permainan Boggle

2.3 Pohon

Dalam informatika sebuah pohon merupakan struktur data yang digunakan secara luas menyerupai struktur pohon dengan sejumlah simpul yang terhubung.

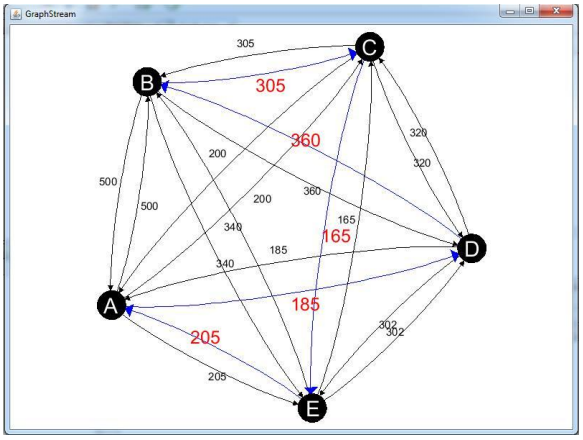
Sebuah simpul mengandung suatu nilai yang menggambarkan struktur data yang terpisah atau sebuah bagian dari pohon itu sendiri. Semua simpul yang berada pada tingkat terendah dari pohon dinamakan daun.

Adapun istilah lain yang sering digunakan ketika menggunakan struktur data pohon adalah istilah upa pohon. Upa pohon merupakan suatu bagian dari pohon struktur data yang dapat dilihat sebagai sebuah pohon lain yang berdiri sendiri.

2.4 Graph

Dalam informatika sebuah graph merupakan sebuah struktur data yang digunakan untuk menggambarkan graph maupun graph langsung yang digunakan dalam matematika.

Banyak algoritma yang diciptakan dengan memanfaatkan struktur data graph. Di antaranya mencari lintasan di antara dua simpul seperti pada algoritma Depth First Search yang akan dibahas kali ini maupun algoritma Breadth First Search. Selain itu algoritma terkenal lain yang menggunakan struktur data graph adalah mencari lintasan terpendek dari suatu simpul ke simpul lainnya seperti algoritma Dijkstra.



Gambar 2.3 Graph

3. ALGORITMA DFS DAN IMPLEMENTASI

3.1 Langkah Penyelesaian

Dalam menyelesaikan permainan Boggle ini terdapat beberapa langkah yang dapat digunakan. Berikut adalah langkah – langkah menyelesaikan permainan Boggle :

1. Mulai dari matriks ke-[1][1]
2. Cari semua kemungkinan kata yang dapat dibentuk yang diawali dengan huruf pada matriks ke-[1][1]
3. Setelah semua kata yang mungkin dibentuk dari matriks[1][1] diperoleh bergerak ke matriks[1][2] lalu lakukan langkah satu sampai dua secara berulang
4. Setelah itu ulangi langkah tiga sampai semua kolom dan baris pada matriks dikunjungi

3.2 Permasalahan yang Dihadapi

Langkah yang harus dilakukan untuk menyelesaikan permainan ini sebenarnya cenderung mudah dan sangat sederhana. Namun permasalahannya adalah tingkat ketelitian pemain yang berbeda – beda. Sehingga bisa saja saat mencari semua kata yang dapat dibentuk pada matriks ada kolom dan baris yang tidak dikunjungi, sehingga ada kata dalam kamus yang tidak ditemukan.

Ditambah lagi tingkat permainan yang semakin meningkat seiring jumlah kolom dan baris yang meningkat pula. Semakin banyak jumlah kolom dan baris pada permainan Boggle maka semakin besar pula kesalahan yang dilakukan pemain karena banyaknya kemungkinan kata yang dapat dibentuk.

3.3 Penyelesaian Masalah dengan Menggunakan Algoritma Depth First Search

Untuk itu dibuat algoritma yang dapat mengerjakan langkah – langkah di atas secara konsisten dan tepat. Dalam makalah ini akan digunakan pendekatan berorientasi objek dalam bahasa java. Algoritma yang akan digunakan digunakan di sini adalah algortitma DFS. Ada beberapa alasan mengapa algoritma DFS yang dipilih dan bukan algoritma BFS, namun penjelasan lebih lanjut akan dibahas pada bab berikutnya.

Pertama – tama untuk membuat simulasi permainan Boggle kita definisikan permainan Boggle sebagai class Boggle

```
public class Boggle {
  private int N;
            //jumlah kolom dan baris
  private char[][] board;
            // matriks of char
  private boolean[][] visited;
            //Cek kolom dan baris yang
              sudah dikunjungi
  private PrefixST st;  //list of kata
```

Setelah itu masukkan huruf ke dalam matriks NxN yang telah dibentuk. Dalam kasus ini program akan melakukan random data untuk menyimpan huruf – huruf dalam matriks. Namun hal ini bias juga dilakukan dengan memasukan file .txt ke dalam matriks.

```
public Boggle(int N, PrefixST st){
  this.N  = N;
  this.st = st;
  visited = new boolean[N][N];
  board = new char[N][N];
  for (int i = 0; i < N; i++)
    for (int j = 0; j < N; j++)
      board[i][j] =
(char)(Math.random() * 26 + 'a');
}
```

Setelah objek matriks diciptakan dan semua baris dan kolom pada matriks diisi dengan huruf – huruf yang telah diacak, buat fungsi DFS yang digunakan untuk mencari semua kemungkinan kata yang dapat dibentuk dari matriks yang telah dibuat

```
private void dfs(String prefix, int
i, int j{
  if (i<0 || j<0 || i>=N || j>=N)
      return;

// kondisi yang menjaga suatu kotak
tidak dikunjungi lebih dari sekali
  if (visited[i][j]) return;

// agar backtracking tidak mengulang
state yang sudah dijalankan
  if (!st.containsPrefix(prefix))
      return;

// kondisi untuk mengeset kolom dan
baris yang sudah dikunjungi menjadi
true
  visited[i][j] = true;

//tambahkan kata yang baru
  prefix = prefix + board[i][j];
  if (st.contains(prefix))
    System.out.println(prefix);

//menentukan semua tetangga
for (int ii=-1; ii<=1; ii++)
  for (int jj=-1; jj<=1; jj++)
    dfs(prefix, i+ii, j+jj);

visited[i][j] = false;
}
```

Fungsi DFS di atas digunakan untuk memperoleh semua kata yang mungkin dibentuk di dalam matriks serta memberikan tanda jika suatu kotak pada matriks telah dikunjungi.

Setelah itu tampilkan semua kata yang mungkin didapat dari matriks yang telah dibentuk

```
public void showWords() {
  for (int i = 0; i < N; i++)
    for (int j = 0; j < N; j++)
      dfs("", i, j);
}
```

Berikut main program yang dapat digunakan untuk menyelesaikan permainan Boggle

```
public static void main(String[]args)
{
  int N = Integer.parseInt(args[0]);

  // read in the list of words
  PrefixST st = new PrefixST();

while(!StdIn.isEmpty())
    st.add(StdIn.readString());
System.err.println("Done reading
dictionary");
```

```
  Boggle boggle = new Boggle(N, st);
  System.out.println(boggle);
  boggle.showWords();
}
```

Dengan program ini maka permainan Boggle dapat diselesaikan dengan sempurna, tanpa ada satu kata pun dalam kamus yang akan terlewat. Karena semua kemungkinan kata yang dapat dibentuk dari matriks telah dicoba satu per satu.

Namun meskipun kita mencoba semua kemungkinan kata satu persatu namun algoritma ini jauh lebih mangkus dibandingkan dengan algoritma brute force maupun algoritma BFS. Pada bab berikutnya akan dibahas mengapa algoritma DFS yang digunakan.

4. ANALISIS PEMILIHAN ALGORITMA DFS

4.1 Mengapa Algoritma DFS

Algoritma DFS merupakan salah satu algoritma yang paling mangkus dibandingkan dengan algoritma lainnya terutama untuk mencari lintasan dengan kedalaman yang paling maksimum. Artinya jawaban yang dicari mungkin ditemukan pada simpul dengan kedalaman maksimum. Dalam permainan Boggle ini kata yang dicari dimungkinkan berada di simpul terdalam, karena panjang kata yang mungkin terdapat dalam kamus tidak bias kita tentukan. Maka tentu saja algoritma DFS adalah algoritma yang paling cocok dengan permainan Boggle ini. Sehingga tidak disarankan memilih algoritma BFS maupun brute force.

4.2 Mengapa bukan Algoritma BFS

Algoritma BFS memang dalam kasus normal biasa disamakan dengan algoritma DFS terutama untuk pencarian lintasan di antara dua simpul ataupun mencari lintasan terpendek dari suatu graph. Bahkan untuk beberapa kasus algoritma BFS bisa lebih mangkus dibandingkan dengan algoritma DFS.

Namun pada kasus ini algoritma BFS sangat tidak disarankan. Karena pada permainan Boggle jawaban yang diharapkan ada lebih dari satu sehingga sangat kecil kemungkinannya kita akan mendapatkan jawaban pada simpul terdalam. Atau dalam kasus ini kata yang dibentuk hanya terdiri dari dua atau tiga huruf.

Jika kita lihat lagi bagaimana cara BFS melakukan pencarian, BFS akan memeriksa terlebih dahulu simpul yang bertetanggaan dengan simpul akar sehingga dalam permainan Boggle jika terdapat huruf :

H	E		
P	L		

Jika metode BFS yang diterapkan maka jika pencarian dimulai dari huruf H berikutnya BFS akan memeriksa satu per satu huruf mulai dari E kemudian melakukan backtrack ke huruf H dan memeriksa simpul tetangga dari H yang lain yaitu huruf P kemudian huruf L. Dengan demikian kita tidak akan pernah bisa mendapatkan kata HELP dengan metode BFS.

Sedangkan dengan metode DFS yang melakukan pencarian secara mendalam kita dimungkinkan untuk memeriksa huruf H pertama kemudian ke simpul berikutnya E lalu L dan yang terakhir P. Hal ini dimungkinkan karena DFS mencoba semua kemungkinan sampai simpul daun terlebih dahulu sehingga sepanjang apapun kata yang ingin dibentuk kita dimungkinkan mendapatkannya.

Berikut contoh permainan Boggle

B	U	H	F
F	A	A	G
E	R	A	E
T	S	E	R

Dan berikut semua kata yang dihasilkan pada permainan Boggle dengan matriks 4x4

1. **gastrea** (5)	2. **barest** (4)	3. **easter** (4)
9. **rarest** (4)	10. **raster** (4)	11. **reseau** (4)
17. **trefah** (4)	18. **agars** (3)	19. **areae** (3)
25. **erase** (3)	26. **ester** (3)	27. **farer** (3)
33. **frere** (3)	34. **frets** (3)	35. **fubar** (3)
41. **hares** (3)	42. **harts** (3)	43. **ragas** (3)
49. **resee** (3)	50. **reset** (3)	51. **sager** (3)
57. **agar** (2)	58. **agas** (2)	59. **agee** (2)
65. **arts** (2)	66. **asea** (2)	67. **baas** (2)
73. **east** (2)	74. **eras** (2)	75. **erst** (2)
81. **fets** (2)	82. **frae** (2)	83. **frag** (2)
89. **gear** (2)	90. **gees** (2)	91. **haaf** (2)
97. **hart** (2)	98. **raga** (2)	99. **rage** (2)
105. **rets** (2)	106. **saga** (2)	107. **sage** (2)
113. **tear** (2)	114. **tree** (2)	115. **tref** (2)
121. **age** (1)	122. **aha** (1)	123. **are** (1)
129. **bar** (1)	130. **ear** (1)	131. **eau** (1)
137. **far** (1)	138. **fer** (1)	139. **fes** (1)
145. **gee** (1)	146. **hae** (1)	147. **hag** (1)
153. **ref** (1)	154. **reg** (1)	155. **res** (1)
161. **ser** (1)	162. **set** (1)	163. **tea** (1)

5. KESIMPULAN

Penggunaan algoritma Depth First Search untuk menyelesaikan permainan Boggle ini sangat tepat karena dengan menggunakan algoritma DFS semua kata yang mungkin dibentuk pada matriks of char dapat diperoleh. Ditambah lagi karena kompleksitas algoritma DFS sendiri yang cukup baik yaitu $O(b^m)$ maka untuk menyelesaikan permainan Boggle ini hanya dibutuhkan kompleksitas algoritma $O(N^2 . b^N)$ karena pengulangan yang harus dilakukan sebanyak N^2 kali pengulangan.

6. UCAPAN TERIMA KASIH

Pertama – tama saya mau mengucapkan terima kasih kepada Tuhan Yang Maha Esa karena atas berkat dan rahmatnya makalah ini berhasil diselesaikan dengan baik. Selain itu saya juga mau mengucapkan terima kasih kepada Ir. Rinaldi Munir, M.T. dan Dr. Nur Ulfa Maulidevi, S.T., M.Sc selaku dosen mata kuliah IF 2211 Strategi Algoritma tahun ajaran 2014/2015, serta tak lupa saya berterima kasih kepada orang tua dan teman – teman yang selalu mendukung.

Akhir kata saya memohon maaf jika ada kesalahan penulisan dalam makalah ini dan saya berharap makalah ini dapat digunakan sebaik – baiknya dan bisa diperbaiki ke depannya.

REFERENSI

[1] http://www.wordplays.com/boggle
[2] Open Data Structures - Section 12.3.2 Depth-First-Search
[3] Depth-First Explanation and Example
[4] http://www.comscigate.com /cs/IntroSedgewick/40/adt/47/st/Boggle.java .html
[5] Descripsi dari *Dictionary of Algorithms and Data Structures*

REFERENSI GAMBAR

https://codersupremo.files.wordpress.com/2013/11/rec2.gif
http://s3.amazonaws.com/answer-board image/fe137445-ce69-4469-9e4d-ac2132dde584.png

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, 2 Mei 2015

Gerry Kastogi
13513011