Penerapan Algoritma BFS, DFS, DLS dan IDS dalam Pencarian Solusi *Water Jug Problem*

Nursyamsiah Pertiwi[1], Esty Hutami Dewi Lubis[2], Lafrania Taufik[3]

Laboratorium Ilmu dan Rekayasa Komputasi
Departemen Teknik Informatika, Institut Teknologi Bandung
Jl. Ganesha 10, Bandung

E-mail : if14059@students.if.itb.ac.id[1], if14071@students.if.itb.ac.id[2],
if14097@students.if.itb.ac.id[3]

Abstrak

Dalam dunia Informatika banyak masalah klasik yang menjadi pembahasan di masa sekarang antara lain TSP (*Travelling Salesman Problem*), Graf Colouring, N-Queen dan sebagainya. Kali ini, penulis membahas mengenai W*aterJug Problem*. Water Jug Problem merupakan suatu contoh masalah yang membutuhkan konversi situasi menjadi situasi lain yang diinginkan dengan menggunakan sekumpulan operasi tertentu. Biasanya permasalahan ini ditemukan dalam bidang *artificial intelligence*.

Dalam pencarian solusi untuk Water Jug Problem ini, penulis memakai algoritma BFS, DFS, DLS dan IDS yang dirasakan cocok dalam penyelesaian masalah ini.

***Kata kunci**: DFS, BFS,DLS, IDS, Water Jug Problem*

1. Pendahuluan

Algoritma BFS, DFS, DLS, dan IDS, adalah algoritma yang digunakan dalam penelusuran (mengunjungi) graf. Keempat algoritma ini dikategorikan ke dalam Blind Search atau Exhautives Search. Karena dengan keempat algoritmanya semua simpul harus dikunjungi, dalam kasus terburuk.

Water Jug problem adalah masalah yeng membutuhkan konversi situasi menjadi situasi lain yang diinginkan dengan menggunakan sekumpulan operasi tertentu. Dan masalah ini dapat di selesaikan dengan werepresentasikan semua kemungkinan hasil dalam sebuah pohon. Maka masalh ini dapat dikategorikan sebagai masalah yang membutuhkan penelusuran graf. Oleh karena itu penulis berpendapat bahwa masalah ini dapat diselesaikan dengan baik dengan menggunakan keempat algoritma tersebut (BFS, DFS, DLS, dan IDS).

1.1 Tujuan Penulisan

1. Menyelesaikan masalah Water Jug dengan algoritma penelusuran graf
2. Mengenalkan algoritma DLS dan IDS sebagai alternatif penyelesaian persoalan yang menyangkut penelusuran graf.

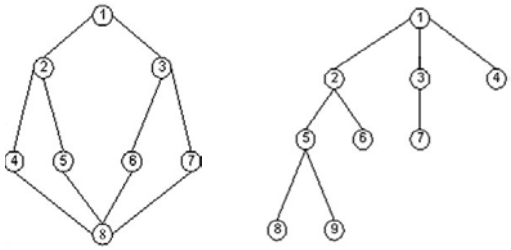
2. Algoritma

Berikut adalah penjelasan singkat dari algoritma-algoritma yang akan digunakan untuk mendapatkan penyelesaian terbaik dari *Water Jug Problem*

2.1. BFS (Breadth First Search)

Misalkan terdapat graf/pohon dengan n buah simpul dan v merupakan simpul awal penelusuran maka algoritma BFS adalah sebagai berikut:

1. Kunjungi simpul *v*
2. Kunjungi semua simpul yang bertetangga dengan simpul *v* terlebih dahulu.
3. Kunjungi simpul yang belum dikunjungi dan bertetangga dengan simpul-simpul yang tadi dikunjungi, demikian seterusnya
4. Jika graf berbentuk pohor berakar, maka semua simpul pada aras *d* dikunjungi lebih dahulu sebelum mengunjungi simpul-simpul pada aras *d* + 1.



Contoh.

(a) (b)

Gambar 2.1. Dua buah graf yang dikunjungi

Untuk graf pada gambar 2.1(a), bila simpul awal adalah 1 maka urutan dikunjunginya adalah 1, 2, 3, 4, 5, 6, 7, 8.
Sedangkan untuk graf pada gambar 2.1(b), bila simpul awal juga 1 maka urutan dikunjunginya adalah 1, 2, 3, 4, 5, 6, 7, 8, 9.

2.2. DFS (Depth First Search)

Misalkan terdapat graf/pohon dengan n buah simpul dan v merupakan simpul awal penelusuran maka algoritma DFS adalah sebagai berikut:

1. Kunjungi simpul *v*,
2. Kunjungi simpul *w* yang bertetangga dengan simpul *v*.
3. Ulangi DFS mulai dari simpul w.
4. Ketika mencapai simpul *u* sedemikian sehingga semua simpul yang bertetangga dengannya telah dikunjungi, pencarian dirunut-balik (*backtrack*) ke simpul terakhir yang dikunjungi sebelumnya dan mempunyai simpul *w* yang belum dikunjungi.
5. Pencarian berakhir bila tidak ada lagi simpul yang belum dikunjungi yang dapat dicapai dari simpul yang telah dikunjungi

Contoh:
Untuk graf pada gambar 2.1(a), bila simpul awal adalah 1 maka urutan dikunjunginya adalah 1, 2, 4, 8, 5, 3, 6, 7.
Sedangkan untuk graf pada gambar 2.1(b), bila simpul awal juga 1 maka urutan dikunjunginya adalah 1, 2, 5, 8, 9, 6, 3, 7, 4.

2.3. DLS (Depth Limited Search)

Pada dasarnya, algoritma DLS sama dengan algoritma DFS, hanya saja dalam permasalahan penelusuran graf, sebelumnya ditentukan terlebih dahulu batas maksimum level yang dikunjungi.

Misalkan terdapat graf/pohon dengan n buah simpul dan v merupakan simpul awal penelusuran maka algoritma DFS adalah sebagai berikut:

1. Tentukan batas kedalaman pohon yang akan dikunjungi.
2. Kunjungi simpul *v*.
3. Kunjungi simpul *w* yang bertetangga dengan simpul *v,* yang berada di kedalaman pohon <= batas.
4. Ulangi DLS mulai dari simpul w.
5. Ketika mencapai simpul *u* sedemikian sehingga semua simpul yang bertetangga dengannya telah dikunjungi, pencarian dirunut-balik (*backtrack*) ke simpul terakhir yang dikunjungi sebelumnya dan mempunyai simpul *w* yang belum dikunjungi.
6. Pencarian berakhir bila tidak ada lagi simpul yang belum dikunjungi yang dapat dicapai dari simpul yang telah dikunjungi dalam kedalaman ponon <= batas.

Contoh:
Untuk graf pada gambar 2.1(a), bila simpul awal adalah 1 dan batas kedalaman adalah 3 maka urutan dikunjunginya adalah 1, 2, 4, 8, 5, 3, 6,.
Sedangkan untuk graf pada gambar 2.1(b), bila simpul awal juga 1 maka urutan dikunjunginya adalah 1, 2, 5, 6, 3, 7, 4.

2.4. IDS (Iterarive Deepening Search)

Misalkan terdapat graf/pohon dengan n buah simpul dan v merupakan simpul awal penelusuran maka algoritma BFS adalah sebagai berikut:

1. Kita mulai kunjungan pohon dari simpul di level 1
2. Kunjungi simpul *v*
3. Kunjungi semua simpul yang bertetangga dengan simpul *v* terlebih dahulu, yang berada di level 1.
4. Jika hasil belum ditemukan, maka kita mulai pencarian dari simpul awal, ke semua tetangganya yang berada di simpul 2 (level +1).
5. Begitu terus berulang hingga hasil didpatkan

3. Pencarian Solusi *Water Jug Problem*

3.1 Deskripsi Umum

Terkadang apa yang kita punya, tidak sesuai dengan apa yang kita harapkan. Begitu pun dalam Water Jug Problem. Kita mempunyai (misalkan) sebuah gelas dengan kapasitas 4 liter dan sebuah gelas yang lain dengan kapasitas (misalkan) 3 liter. Kedua gelas tidak memiliki skala ukuran dan dalam keadaan kosong.

Kita ingin mendapatkan air sebanyak 2 liter dalam gelas yang berkapasitas 4 liter dan tidak boleh mendapatkannya dengan menggunakan prediksi sendiri. Selain itu, kita harus mengikuti suatu operasi-operasi tertentu, seperti dalam tabel berikut.

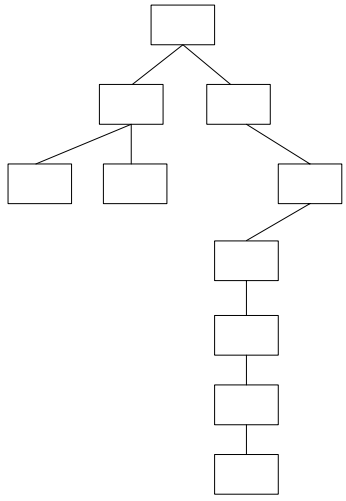
Ket:
x adalah gelas dengan kapasitas 4 liter
y adalah gelas dengan kapasitas 3 liter

1.	(x,y) If x < 4	→	(4,y)	Isi penuh gelas 4 galon
2.	(x,y) If y < 3	→	(x,3)	Isi penuh gelas 3 galon
3.	(x,y) If x > 0	→	(x-d,y)	Buang sebagian air dari gelas 4 galon
4.	(x,y) If y > 0	→	(x,y-d)	Buang sebagian air dari galon ukuran 3 galon
5.	(x,y) If x > 0	→	(0,y)	Kosongkan gelas 4 galon

6.	(x,y) If y > 0	→	(x,0)	Kosongkan gelas 3 galon
7.	(x,y) If x+y ≥4 and y > 0	→	(4,y-(4-x))	Tuangkan air dari gelas 3 galon ke gelas 4 galon sampai gelas 4 galon penuh
8.	(x,y) If x+y ≥3 and x > 0	→	(x-(3-y),3)	Tuangkan air dari gelas 4 galon ke gelas 3 galon sampai gelas 3 galon penuh
9.	(x,y) If x+y ≤4 and y > 0	→	(x+y,0)	Tuangkan seluruh air dari gelas 3 galon ke gelas 4 galon
10.	(x,y) If x+y ≤3 and x > 0	→	(0,x+y)	Tuangkan seluruh air dari gelas 4 galon ke gelas 3 galon
11.	(0,2)	→	(2,0)	Tuangkan 2 galon air dari gelas 3 galon ke gelas 4 galon
12.	(2,y)	→	(0,y)	Buang 2 galon dalam gelas 4 galon sampai habis.

Tabel 3.1

3.2 Pencarian Solusi dengan menggunakan Algoritma BFS



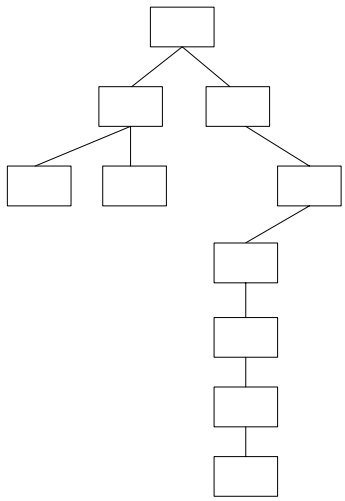
Gambar 3.2. Pohon Ruang Status untuk *Water Jug Problem*

Langkah-langkah yang dilakukan dalam pencarian solusi menggunakan algoritma BFS adalah
1. Kunjungi simpul pertama (0,0)
2. Cek apakah sesuai dengan keinginan. Jika true, maka masalah selesai. Jika false, kunjungi simpul tetangga (4,0)
3. Jika simpul tetangga sudah dikunjungi semuanya, maka bentuk anak dari seluruh simpul dengan memperhatikan operasi-operasi yang terdapat dalam Tabel 3.1
4. Ulangi langkah 1 hingga 3.

Berikut pseudocode untuk algoritma BFS:

```
procedure BFS (input v: Point)
kamus
w: point
q:  antrian

algoritma
  buatAntrian(q); {buat antrian kosong}
write(v)
dikunjungi[v]  ←  true  {array  untuk
menampung simpul yang sudah dikunjungi}

masukAntrian(q,v)

{kunjungi  semua  simpul  graf  selama
antrian belum kosong}

while not AntrianKosong(q) do
  hapusAntrian(q,v)
  for w←1 to n do
    if A[v,w] = 1 then
      if not dikunjungi[w] then
        write(w)
        masukAntrian(q,w)
      endif
    endif
  endfor
endwhile
```

3.3. Pencarian Solusi dengan menggunakan algoritma DFS



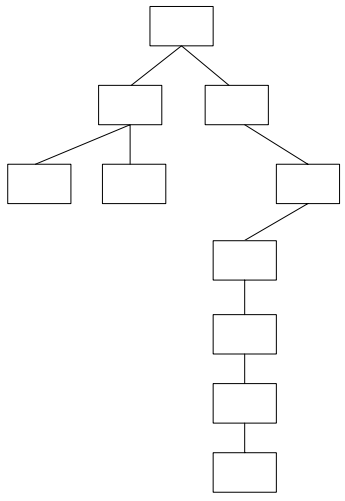
Gambar3.3. Pohon DFS

Langkah-langkah yang dilakukan dalam pencarian solusi menggunakan algoritma DFS adalah

1. Masukkan simpul akar (0, 0) ke dalam antrian Q, jika simpul akar adalah simpul tujuan (goal node) maka solusi ditemukan. Stop.
2. Jika Q kosong, tidak ada solusi. Stop.
3. Cek apakah sesuai dengan keinginan (2,0). Jika true maka solusi ketemu. Jika false maka bangkitkan simpul tetangga dan ulang (2).

```
procedure DFS (input v: Point)
kamus
w: point
q:  antrian

algoritma
write(v)
dikunjungi[v]  ←  true  {array  untuk
menampung simpul yang sudah dikunjungi}

for w←1 to n do
    if A[v,w] = 1 then {simpul v dan
simpul w bertetangga}
      if not dikunjungi[w] then
        DFS(w)
      endif
    endif
endfor
```

3.4. Pencarian Solusi dengan menggunakan algoritma DLS dan IDS



Langkah-langkah yang dilakukan dalam pencarian solusi menggunakan algoritma DLS adalah

1. Tentukan kedalaman simpul yang dikunjungi (n), misalkan n=5.
2. Masukkan simpul akar (0, 0) ke dalam antrian q, jika simpul akar adalah simpul tujuan (goal node) maka solusi ditemukan. Stop.
3. Jika S kosong, tidak ada solusi. Stop.
4. Cek apakah sesuai dengan keinginan (2,0). Jika true maka solusi ketemu. Jika false maka bangkitkan simpul tetangga, dan ulang (2), hingga kedalaman simpul yang dikunjungi <= 5 .

```
procedure DLS (input v: Point, level:
integer)

kamus
w: point
q: antrian

algoritma
write(v)
dikunjungi[v]  ←  true  {array  untuk
menampung simpul yang sudah dikunjungi}
level ← 1

while not level > 5 do
   { kunjungi semua simpul di level 1,
    panggl algoritma DFS}
  if A[v,w] = 1 then {simpul v dan
simpul w bertetangga}
      if not dikunjungi[w] then
        DLS(w, level)
      endif
  endif
  level ← level +1
endwhile
```

Langkah-langkah yang dilakukan dalam pencarian solusi menggunakan algoritma IDS adalah

1. Mulai pencarian dari level pertama pohon yang akan dikunjungi
2. Lalu panggil procedure DLS untuk setiap level
3. Ulang terus sampai didapat hasil

```
procedure IDS (input v: Point)

kamus
n : integer
w: point
s : stack

algoritma
write(v)

for n ← 1 to infinite do
  DLS (v,n)
endfor
```

Jumlah galon dalam gelas 4 galon	**Jumlah galon dalam gelas 3 galon**	**Aturan yang dilakukan**
0	0	-

0	3	2
3	0	9
3	3	2
4	2	7
0	2	5 atau 12
2	0	9 atau 11

Gambar 3.4. Suatu solusi untuk Water Jug Problem

4. Kesimpulan

Untuk hampir semua masalah pencarian dalam persoalan pencarian pada graf, maka algoritma IDS, selalu menunjukkan hasil yang lebih optimum . namun keempat algoritma ini, memiliki kelebihan dan kekurangan masing-masing.

Berikut terdapat tabel perbandingan kompleksitas berdasarkan, ruang dan waktu, juga perbandingan keoptimalan dan keselesaian dari keempat algoritma.

Keterangan tabel :
b : branching factor
l : level/kedalaman pohon, dimana solusi ditemukan
m : kedalaman maksimum pohon yang ditelusuri
d : batas kedalaman

Kriteria	**BFS**	**DFS**	**DLS**	**IDS**
Waktu	b^l	b^m	b^d	b^l
Ruang	b^l	bm	bd	bl
Optimal?	Ya	Tidak	Tidak	Ya
Selesai?	Ya	Tidak	Ya, If d>=l	Ya

Daftar Pustaka

1. Munir, Rinaldi. "*Strategi Algoritmik*". Laboratorium Ilmu Rekayasa dan Komputasi. Institut Teknologi Bandung .
2. **http://www.ai-depot.com**
3. "*Pengntar AI*", Universitas Gunadarma