Pelacakan dan Penentuan Jarak Terpendek terhadap Objek dengan BFS (*Breadth First Search*) dan *Branch and Bound*

Mico (13515126)
Teknik Informatika
Sekolah Teknik Elektro dan Informatika ITB
Jl. Ganesha 10, Bandung 40132, Jawa Barat
mccuandar@gmail.com

***Abstract*—Pada zaman sekarang ini, teknologi gps memungkinkan proses pelacakan suatu objek atau barang menjadi sangat cepat. Akan tetapi, dengan semua algoritma yang ada , hasil yang didapat tidak selalu optimum. Oleh karena itu, banyak insinyur-insinyur yang melakukan penelitian pada bidang ini untuk menemukan solusi . Tujuan pelacakan objek ini adalah menemukan lintasan objek yang hilang dan menentukan jalan terdekat ke objek tersebut dengan jarak terpendek dengan menggabungkan BFS dan branch and bound.**

Keywords : Pelacakan, lintasan, objek, BFS, branch and bound

I. PENDAHULUAN

Pada zaman yang serba teknologi dan gps ini, pencarian objek hilang dapat dilakukan dengan berbagai macam cara. Dengan gps, kita dapat melacak dimana posisi objek yang telah kita integrasikan dengan gps dan jarak terpendek kita ke objek tersebut.

Permasalahan pencarian jarak terpendek direpresentasikan dengan sebuah papan permainan di mana terdapat dua titik, yaitu titik mulai dan titik selesai, dan papan memiliki sejumlah penghalang sehingga kita harus mencari jarak terpendek yang dapat dilalui dari titik mulai menuju titik selesai. Sebelum mulai memaparkan proses pencarian solusi, terlebih dahulu penulis memberikan sedikit gambaran mengenai algoritma *BranchandBound*.

Pencarian solusi pada algortima *Branch and Bound* (B&B) menggunakan skema pencarian *Breadth First Search* (BFS). Dengan metode *BFS*, simpul akar pada pohon pencarian dibangkitkan pertama kali, kemudian semua simpul anakanaknya, kemudian *successor* dari setiap simpul anak, dan seterusnya.

Algoritma B&B meminimumkan pembentukan pohon pada skema BFS. Pencarian ke simpul solusi dipercepat dengan memilih simpul hidup berdasarkan nilai ongkos (cost). Setiap simpul hidup diasosiasikan dengan sebuah ongkos yang menyatakan nilai batas (bound). Nilai batas yang dipergunakan dalam pembuatan aplikasi shortest path finder ini adalah panjang lintasan dari sebuah simpul ke simpul solusi terdekat.

II. ANALISA PENCARIAN OBJEK

A. *Pencarian Objek*

Suatu waktu sepeda motor kita dicuri, lalu pencuri membawa motor kita ke luar kota. Didalam motor kita sudah terpasang alat pengaman yang akan memancarkan signal selama motor itu hidup. Atau saat dompet kita tiba-tiba hilang , terlepas dari dugaan apakah dicuri atau tidak. Di dalam dompet kita telah kita pasang alat serupa dan aktif selama ada daya pada alat tersebut.

Alat tersebut tidak mengirim signal yang berupa lokasi atau koordinat dari objek kita yang hilang. Tetapi hanya mendeteksi kearah mana objek tersebut menghadap menurut empat arah mata angin: utara, selatan, barat, dan timur, dan memancarkannya. Alat hanya memancarkan satu kali saja sebelum terjadi terjadi perubahan berikutnya.

Jalan-jalan di kota itu vertikal atau horizontal dengan simpangan-simpangan pada posisi bilangan bulat. Data yang terekam terakhir dari objek itu adalah posisi objek terakhir sebelum hilang (kita sebut pada posisis awal), kemudian sejak itu terekam sederetan sinyal-sinyal arah hadap objek tersebut hingga kemudian berhenti entah di mana.

B. *Batasan Pelacakan Objek*

Dalam analisi penyelesaian masalah pencarian objek dengan algoritma BFS (*Breadth First Search*) terdapat batasan-batasan yang harus dipenuhi sebelumnya. Yang pertama objek tersebut harus memiliki alat yang dapat mendeteksi kearah mana objek itu menghadat menurut empat arah mata angin: utara, selatan, barat, dan timur, dan memancarkannya.

Yang kedua jalan-jalan di dalam kota tersebut dibatasi, yaitu hanya vertical atau horizontal dengan simpangan-simpangan pada posisi bilangan bulat.

Alat yang terpasang di dalam objek memberikan data yang di asumsikan menjadi file teks bernama arah.txt yang berisi data dalam format sebagai berikut :

Baris pertama file berisi dua bilangan integer yang memiliki range 1 sampai 100 yang dipisahkan oleh karakter spasi. Masing-masing menyatakan baris dan kolom dari peta kota. Misalnya X = 60 dan Y = 50.

Pada setiap baris dari X terdapat deretan karakter sebanyak Y yang berisi karakter '0' , '.' dan '*'. Yang akan menjelaskan bagian-bagian peta.

Pada baris ke X+1 terdapat bilangan integer N antara 1 sampai 9999 yang menyatakan berapa banyak signal arah objek yang direkam.

Masing-masing N baris berikutnya berisi char U yang berarti Utara, T yang berarti Timur, S yang berarti Selatan dan B yang berarti Barat dan tidak ada 2 signal yang berurtan sama.

C. *Penentuan Jarak*

Penentuan jarak dapat dilakukan dengan *Reduce Cost Matrix.* Jika seandainya dalam kotak 50x50, objek yang hilang berada di rentang kotak 5x5 dari jarak awal, maka dengan RCM dapat kita tentukan jarak terdekat untuk menuju objek tersebut dan kembali lagi ke posisi semula (dalam kasus titik semula adalah tujuan akhir kita setelah menemukan barang yang hilang).

Misalnya : objek yang hilang adalah motor kita, dan radius motor kita adalah 5x5 dalam peta. Dan ketika di masukkan ke matriks menghasilkan :

$$\begin{bmatrix} \infty & 187 & 96 & 184 & 106 \\ 187 & \infty & 119 & 205 & 293 \\ 96 & 119 & \infty & 280 & 202 \\ 184 & 205 & 280 & \infty & 174 \\ 106 & 184 & 202 & 174 & \infty \end{bmatrix}$$

Lakukan reduksi baris dan kolom sehingga setiap kolom dan baris mengandung bilangan 0 sehingga menjadi :

$$\begin{bmatrix} \infty & 68 & 0 & 20 & 10 \\ 68 & \infty & 0 & 18 & 174 \\ 0 & 0 & \infty & 116 & 106 \\ 10 & 8 & 106 & \infty & 0 \\ 0 & 164 & 96 & 0 & \infty \end{bmatrix}$$

Selanjutnya, proses reduksi ini akan menghasilkan nilai batas simpul akar atau ˆ*(), yang didapat dari penjumlahan semua elemen pengurang tadi. Jadi, *( )= 96+119+96+174+106+23+68 = 682. Disini berarti telah dibangkitkan pohon ruang status yang baru berisi satu buah simpul dengan bobot 682.

Selanjutnya kita hitung cost dari simpul-simpul yang ada :

1. Simpul 2; lintasan 1,2

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & 0 & 18 & 174 \\ 0 & \infty & \infty & 116 & 106 \\ 10 & \infty & 106 & \infty & 0 \\ 0 & \infty & 96 & 0 & \infty \end{bmatrix}$$

   *(2)= 682 + 68 + 0 = 750

2. Simpul 3; lintasan 1,3

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 68 & \infty & \infty & 18 & 174 \\ \infty & 0 & \infty & 116 & 106 \\ 10 & 8 & \infty & \infty & 0 \\ 0 & 164 & \infty & 0 & \infty \end{bmatrix} = \begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 50 & \infty & \infty & 0 & 156 \\ \infty & 0 & \infty & 116 & 106 \\ 10 & 8 & \infty & \infty & 0 \\ 0 & 164 & \infty & 0 & \infty \end{bmatrix}$$

   Matriks di atas diperoleh dari pengurangan baris ke 2 oleh 18.

   *(3)= 682 + 0 + 18 = 700

3. Simpul 4; lintasan 1,4

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 68 & \infty & 0 & \infty & 174 \\ 0 & 0 & \infty & \infty & 106 \\ \infty & 8 & 106 & \infty & 0 \\ 0 & 164 & 96 & \infty & \infty \end{bmatrix}$$

   *(4)= 682 + 20 + 0 = 702

4. Simpul 5; lintasan 1,5

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 68 & \infty & 0 & 18 & \infty \\ 0 & 0 & \infty & 116 & \infty \\ 10 & 8 & 106 & \infty & \infty \\ \infty & 164 & 96 & 0 & \infty \end{bmatrix} = \begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 68 & \infty & 0 & 18 & \infty \\ 0 & 0 & \infty & 116 & \infty \\ 2 & 2 & 98 & \infty & \infty \\ \infty & 164 & 96 & 0 & \infty \end{bmatrix}$$

   Matriks di atas diperoleh dari pengurangan baris ke 4 oleh 8.

   *(5)= 682 + 10 + 8 = 700

Selanjutnya, pilih simpul yang memiliki nilai batas terkecil, dalam hal ini terdapat 2simpul yaitu simpul 3 dan simpul 5. Pertama-tama, kita ekspansi simpul 3 terlebih dahulu :

5. Simpul 6; lintasan 1,3,2

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & 0 & 156 \\ \infty & \infty & \infty & \infty & \infty \\ 10 & \infty & \infty & \infty & 0 \\ 0 & \infty & \infty & 0 & \infty \end{bmatrix}$$

   *(6)= 700 + 0 + 0 = 700

6. Simpul7;lintasan 1,3,4

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 50 & \infty & \infty & \infty & 156 \\ \infty & \infty & \infty & \infty & \infty \\ \infty & 8 & \infty & \infty & 0 \\ 0 & 164 & \infty & \infty & \infty \end{bmatrix} = \begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & \infty & \infty & 106 \\ \infty & \infty & \infty & \infty & \infty \\ \infty & 8 & \infty & \infty & 0 \\ 0 & 164 & \infty & \infty & \infty \end{bmatrix}$$

   Matriks di atas diperoleh dari pengurangan baris ke 2 oleh 50.

   *(7)= 700 + 116 + 50 = 866

7. Simpul 8; lintasan 1,3,5

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 50 & \infty & \infty & 0 & \infty \\ \infty & \infty & \infty & \infty & \infty \\ 10 & 8 & \infty & \infty & \infty \\ \infty & 164 & \infty & 0 & \infty \end{bmatrix} = \begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 50 & \infty & \infty & 0 & \infty \\ \infty & \infty & \infty & \infty & \infty \\ 2 & 0 & \infty & \infty & \infty \\ \infty & 164 & \infty & 0 & \infty \end{bmatrix}$$

   *(8)= 700 + 106 + 8 = 814

8. Simpul 9; lintasan 1,5,2

∞	∞	∞	∞	∞		∞	∞	∞	∞	∞
∞	∞	0	18	∞		∞	∞	0	0	∞
0	∞	∞	116	∞	=	0	∞	∞	98	∞
2	∞	98	∞	∞		2	∞	98	∞	∞
∞	∞	∞	∞	∞		∞	∞	∞	∞	∞

Matriks di atas diperoleh dari pengurangan kolom ke 4 oleh 18.
*(9)= 700 + 164 + 18 = 882

9. Simpul 10; lintasan 1,5,3

∞	∞	∞	∞	∞		∞	∞	∞	∞
68	∞	∞	18	∞		50	∞	∞	0	∞
∞	0	∞	116	∞	=	∞	0	∞	116	∞
2	2	∞	∞	∞		0	0	∞	∞	∞
∞	∞	∞	∞	∞		∞	∞	∞	∞	∞

Matriks di atas diperoleh dari pengurangan baris ke 2 oleh 18 dan baris ke 4 oleh 2.
*(10)= 700 + 96 + 20 = 816

10. Simpul11; lintasan 1,5,4

∞	∞	∞	∞	∞		∞	∞	∞	∞	∞
68	∞	0	∞	∞		68	∞	0	∞	∞
0	0	∞	∞	∞	=	0	0	∞	∞	∞
∞	2	98	∞	∞		∞	0	96	∞	∞
∞	∞	∞	∞	∞		∞	∞	∞	∞	∞

Matriks di atas diperoleh dari pengurangan baris ke 2 oleh 18 dan baris ke 4 oleh 2.
*(10)= 700 + 96 + 20 = 816

11. Simpul 12; lintasan 1,3,2,4

∞	∞	∞	∞	∞
∞	∞	∞	∞	∞
∞	∞	∞	∞	∞
∞	∞	∞	∞	0
0	∞	∞	∞	∞

*(12)= 700 + 18 + 0 = 718

12. Simpul 13; lintasan 1,3,2,5

∞	∞	∞	∞	∞		∞	∞	∞	∞	∞
∞	∞	∞	∞	∞		∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	=	∞	∞	∞	∞	∞
10	∞	∞	∞	∞		0	∞	∞	∞	∞
∞	∞	∞	0	∞		∞	∞	∞	0	∞

*(13)= 700 + 174 + 10 = 884

Dari gambar di atas, maka kita dapat menyimpulkan bahwa rute dengan *cost* minimum, dalam hal ini berarti rute terpendek, adalah melalui simpul 1-3-6-12-14-1.

D. Dasar Teori

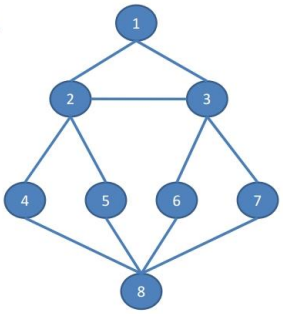
Breadth-first search adalah algoritma yang melakukan pencarian secara melebar yang mengunjungi simpul secara preorder yaitu mengunjungi suatu simpul kemudian mengunjungi semua simpul yang bertetangga dengan simpul tersebut terlebih dahulu. Selanjutnya, simpul yang belum dikunjungi dan bertetangga dengan simpulsimpul yang tadi dikunjungi , demikian seterusnya. Jika graf berbentuk pohon berakar, maka semua simpul pada aras d dikunjungi lebih dahulu sebelum simpul-simpul pad aras d+1.

Algortiam ini memerlukan sebuah antrian q untuk menyimpan simpul yang telah dikunjungi. Simpul-simpul ini diperlukan sebagai acuan untuk mengunjungi simpul-simpul yang bertetanggan dengannya. Tiap simpul yang telah dikunjungi masuk kedalam antrian hanya satu kali saja.

Pencarian Melebar (BFS)

- Traversal dimulai dari simpul v.
- Algoritma:

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.



RN-MLK/IF2211/2013 4

BFS: Struktur Data

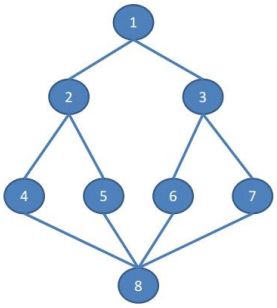
1. Matriks ketetanggaan A = [aij] yang berukuran nxn,
   a_{ij}= 1, jika simpul i dan simpul j bertetangga,
   a_{ij}= 0, jika simpul i dan simpul j tidak bertetangga.
2. Antrian q untuk menyimpan simpul yang telah dikunjungi.
3. Tabel boolean yang bernama dikunjungi

```
dikunjungi : array[1..n] of boolean
```

dikunjungi[i] = true jika simpul i sudah dikunjungi
dikunjungi[i] = false jika simpul i belum dikunjungi

RN-MLK/IF2211/2013 5

BFS: Ilustrasi



Iterasi	V	Q	dikunjungi							
			1	2	3	4	5	6	7	8
Inisialisasi	1	{1}	T	F	F	F	F	F	F	F
Iterasi 1	1	{2,3}	T	T	T	F	F	F	F	F
Iterasi 2	2	{3,4,5}	T	T	T	T	T	F	F	F
Iterasi 3	3	{4,5,6,7}	T	T	T	T	T	T	T	F
Iterasi 4	4	{5,6,7,8}	T	T	T	T	T	T	T	T
Iterasi 5	5	{6,7,8}	T	T	T	T	T	T	T	T
Iterasi 6	6	{7,8}	T	T	T	T	T	T	T	T
Iterasi 7	7	{8}	T	T	T	T	T	T	T	T
Iterasi 8	8	{}	T	T	T	T	T	T	T	T

RN-MLK/IF2211/2013 7

Metode Branch and Bound adalah sebuah teknik algoritma yang secara khusus mempelajari bagaimana caranya memperkecil Search Tree menjadi sekecil mungkin. Sesuai dengan namanya, metode ini terdiri dari 2 langkah yaitu :

- Branch yang artinya membangun semua cabang tree yang mungkin menuju solusi.
- Bound yang artinya menghitung node mana yang merupakan active node (E-node) dan node mana yang merupakan dead node (D-node) dengan menggunakan syarat batas constraint (kendala).

Algoritma Branch & Bound

- B&B: BFS + least cost seach
  - BFS murni: Simpul berikutnya yang akan diekspansi berdasarkan urutan pembangkitannya (FIFO)
- B&B:
  - Setiap simpul diberi sebuah nilai cost: $\hat{c}(i)$ = nilai taksiran lintasan termurah dari simpul status i ke simpul status tujuan.
  - Simpul berikutnya yang akan diekspansi **tidak lagi** berdasarkan urutan pembangkitannya, tetapi simpul yang memiliki *cost* yang paling kecil (least cost search) – pada kasus minimasi.

IF2211 B&B/MLK&RN 4

Algoritma Branch & Bound (2)

1. Masukkan simpul akar ke dalam antrian Q. Jika simpul akar adalah simpul solusi (goal node), maka solusi telah ditemukan. Stop.
2. Jika Q kosong, tidak ada solusi. Stop.
3. Jika Q tidak kosong, pilih dari antrian Q simpul i yang mempunyai $\hat{c}(i)$ paling kecil. Jika terdapat beberapa simpul i yang memenuhi, pilih satu secara sembarang.
4. Jika simpul i adalah simpul solusi, berarti solusi sudah ditemukan, stop. Jika simpul i bukan simpul solusi, maka bangkitkan semua anak-anaknya. Jika i tidak mempunyai anak, kembali ke langkah 2.
5. Untuk setiap anak j dari simpul i, hitung $\hat{c}(j)$, dan masukkan semua anak-anak tersebut ke dalam Q.
6. Kembali ke langkah 2.

III. Implementasi dan Analisis

A. *Implementasi*

Dalam implementasinya kita perlu definisikan type posisi berikut :

-type posisi=record x, y: integer end;

Breadth adalah array dari elemen bertype posisi ini.

Selanjutnya kita perlu prosedur untuk memindahkan posisi bertanda "*" ke dalam array breadth.

```
nb := 0;
    for i := 1 to nrow do
        for j := 1 to ncol do
            if peta[i,j] = '*' then
                begin
                inc(nb);
                breadth[nb].x := i;
                breadth[nb].y := j;
                peta[i,j] := '.';
        { clearkan posisi tsb}
end;
```

Lalu untuk setiap titik breadth[b] dilakukan penandaan kembali posisi selanjutnya sesuai dengan arah yang diberikan. Misalnya untuk arah Timur maka

```
posNext := breadth[b];
inc(posNext.x);
while Jalan(posNext) do
    begin
    peta[posNext.r,posNext.x] := '*';
    inc(posNext.x);
end;
```

Untuk arah Utara maka lakukan dengan

```
posNext := breadth[b];
dec(posNext.y);
while Jalan(posNext) do
    begin
    peta[posNext.y,posNext.x] := '*';
    dec(posNext.y);
end;
```

Untuk arah Barat maka lakukan dengan

```
posNext := breadth[b];
inc(posNext.x);
while Jalan(posNext) do
    begin
    peta[posNext.r,posNext.x] := '*';
    dec(posNext.x);
end;
```

Untuk arah Selatan maka lakukan dengan

```
posNext := breadth[b];
dec(posNext.y);
while Jalan(posNext) do
    begin
    peta[posNext.y,posNext.x] := '*';
    inc(posNext.y);
end;
```

Untuk mencek keadaan sedang jalan atau tidak maka kita membutuhkan method Jalan yang memeriksanya.

```
function Jalan(posNext: posisi):
boolean;
```

```
begin
....
{ kalau keluar peta maka false }
{ kalau tembok maka false }
{ kalau tidak maka true }
....
end;
```

Setelah semua titik pada breadth diproses maka breadh dikosongkan kembali dan diisi oleh titik-titik pada peta yang bertanda "*" kembali (dengan memanggil procedure di paling atas).

Setelah arah masukan terakhir maka peta langsung dituliskan

Contohnya , peta masukan berikut ini

```
.........
.0.0..000.
.0..0.....
.0..0.....
....0.....
*.....00.
.........
```

Breadth hanya berisi 1 titik, yaitu (6.1). Jika kemudian masukan arah adalah Timur maka breadth kemudian akan berisi titik-titik di samping kanan tanda bintang yaitu posisi-posisi kosong sampai ketemu tembok. Sekarang peta menjadi sbb.

```
.........
.0.0..000.
.0..0.....
.0..0.....
....0.....
.******00.
.........
```

Breadth hanya berisi 6 titik, yaitu (6.2), (6,3),...,(6,7). Jika kemudian masukan adalah Utara maka hal seperti di atas dilakukan ke arah atas untuk masing-masing tanda bintang. Dan peta menjadi sbb.

```
..*..*....
.0*0.*000.
.0**0**...
.0**0**...
.***0**...
.......00.
.........
```

Jika kemudian masukan adalah Barat maka hal seperti di atas dilakukan ke arah atas untuk masing-masing tanda bintang. Dan peta menjadi sbb.

```
*****.....
.0.0*.000.
.0*.0*....
.0*.0*....
***.0*....
.......00.
.........
```

Jika kemudian masukan adalah Utara maka hal seperti di atas dilakukan ke arah atas untuk masing-masing tanda bintang. Dan peta menjadi sbb.

```
*.*.**....
*0*0.*000.
*0*.0*....
*0*.0*....
....0.....
.......00.
.........
```

Jika kemudian masukan adalah Barat maka hal seperti di atas dilakukan ke arah atas untuk masing-masing tanda bintang. Dan peta menjadi sbb.

```
*****.....
.0.0*.000.
.0..0.....
.0..0.....
....0.....
.......00.
.........
```

Jika kemudian masukan adalah Utara maka hal seperti di atas dilakukan ke arah atas untuk masing-masing tanda bintang. Dan peta menjadi sbb.

```
....*.....
.0.0..000.
.0..0.....
.0..0.....
....0.....
.......00.
.........
```

Jika sinyal masukan sudah habis maka tanda-tanda bintang menyatakan semua kemungkinan objek itu berada saat ini.

B. Analisis

Pencarian objek yang hilang ini dapat diselesaikan dengan algoritma BFS karena persoalan ini dapat diumpamakan dengan graph traversal BFS (Breadth Firsh Search). Khusus untuk masalah ini BFS dapat diimplementasikan tanpa menggunakan queue untuk menyimpan pencabangannya (breadthnya).

Karena, dari suatu titik dan arah, kita mendapatkan

sejumlah titik berikutnya sebagai breadth dalam BFS. Karena ukuran kota yang diumpamakan dengan matrix 50x50 maka untuk kasus terburuk akan ada 49x49=2401 titik dalam breadth. Karena searching space sangat terbatas maka breadth tidak akan mengalami "peledakan".

Jadi kita menggunakan array breadth yang setiap elemennya menyimpan koordinat titik pencabangan. Dalam inisialisasi array hanya berisi satu titik yaitu posisi awal objek.

Dalam setiap sinyal arah yang dibaca maka dari setiap titik dalam array breadth ini kita temukan kembali semua kemungkinan titik berikutnya di dalam peta dan menandainya. Setelah hal tersebut dilakukan untuk semua titik dalam breadth maka titik-titik pada peta yang telah ditandai dicatat ke dalam array breadt

IV. KESIMPULAN

Kesimpulan yang didapat adalah bahwa engan algoritma BFS (*Breadth First Search)* kita dapat menemukan objek yang hilang. Namun karena dibatasi oleh n buah simpul graf maka cara di atas dapat dipakai. Akan tetapi, apabila simpulnya banyak sehingga menimbulkan peledakan apabila di turunkan semua cabangnya maka penyelesaian dapat ditambahkan dengan queue yang menampung simpul yang telah dikunjungi sehingga kita tidak harus menurunkan semua cabang dari simpul graf tersebut. Untuk Pencarian rute itu sendiri dengan *reduce cost matrix* dapat ditentukan jarak dan simpul-simpul yang dapat ditempuh dengan asumsi bahwa ketika objek telah didapatkan kembali, maka orang yang mencari akan kembali ke titik awal pencarian.

REFERENCES

[1] Munir, Rinaldi, “Diktat Kuliah IF2211 Strategi Algoritma”, Program Studi Teknik Informatika STEI ITB, 2017

[2] https://onbuble.wordpress.com/2011/05/26/6/

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, 18 Mei 2017

ttd

Mico (13515126)