PENGGUNAAN ALGORITMA *"BRANCH AND BOUND"* UNTUK MENYELESAIKAN PERSOALAN PENCARIAN JALAN (*PATH-FINDING*)

R. Aditya Satrya Wibawa (NIM. 13505064)

Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika - Institut Teknologi Bandung
Jalan Ganeca no. 10 Bandung
e-mail: if15064@students.if.itb.ac.id

ABSTRAK

Persoalan pencarian jalan (*path-finding*) adalah persoalan menjacari lintasan dari titik awal menuju titik akhir yang ditentukan dengan melewati hambatan-hambatan (*constrains*) yang ada.
Pada makalah ini akan dibahas algoritma pencari lintasan dengan menggunakan algoritma *Branch and Bound*. Algoritma Branch and Bound mirip dengan algoritma A* dan banyak digunakan pada permainan-permainan berjenis arcade atau adventure.

Kata kunci: *path-finding*, pencari lintasan, *branch and bound*, *shortest path*, lintasan terpendek.

1. PENDAHULUAN

Persoalan pencarian jalan (*path-finding*) adalah persoalan menjacari lintasan dari titik awal menuju titik akhir yang ditentukan dengan melewati hambatan-hambatan (*constrains*) yang ada.

Gambar 1. Salah satu contoh persoalan path-finding. S adalah titik awal dan F adalah titik akhir.

Pada makalah ini, akan dijelaskan cara penyelesaian persoalan *path-finding* dengan menggunakan algoritma *Branch and Bound* dengan menggunakan suatu fungsi heuristic sederhana.

Contoh kode (*code sample*) program yang dijelaskan pada makalah ini adalah dalam bahasa pemrograman Java dan dapat didownload di situs: http://students.if.itb.ac.id/~if15064/PathFinder

2. BRANCH AND BOUND [2]

2.1 Konsep Branch and Bound

Sebagaimana pada algortima runut-balik, algoritma Branch & Bound juga merupakan metode pencarian di dalam ruang solusi secara sistematis. Ruang Solusi diorganisasikan ke dalam pohon ruang status. Pembentukan pohon ruang status. Pembentukan pohon ruang status pada algoritma B&B berbeda dengan pembentukan pohon pada algoritma runut-balik. Bila pada algoritma runut-balik ruang solusi dibangun secara Depth-First Search(DFS), maka pada algoritma B&B ruang solusi dibangun dengan skema Breadth-First Search (BFS).

Pada algoritma B&B, pencarian ke simpul solusi dapat dipercepat dengan memilih simpul hidup berdasarkan nilai ongkos (cost). Setiap simpul hidup diasosiasikan dengan sebuah ongkos yang menyatakan nilai batas (bound). Pada prakteknya, nilai batas untuk setiap simpul umumnya berupa taksiran atau perkiraan.

Fungsi heuristik untuk menghitung taksiran nilai tersebut dinyatakan secara umum sebagai :

$$c(i) = f(i) + g(i)$$

yang dalam hal ini,

c(i) = ongkos untuk simpul i
f(i) = ongkos mencapai simpul i dari akar
g(i) = ongkos mencapai simpul tujuan dari simpul akar i

Nilai c digunakan untuk mengurutkan pencarian. Simpul berikutnya yang dipilih untuk diekspansi adalah simpul yang memiliki c minimum.

2.2 Algoritma Branch and Bound

Algoritma B&B jika dituliskan secara prosedural adalah

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 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 c(j), dan masukkan semua anak-anak tersebut ke dalam antrian Q.
6. Kembali ke langkah 2.

2. PENYELESAIAN PERSOALAN

3.1 Algoritma

Algoritma penyelesaian persoalan akan ditunjukkan lewat sebuah contoh sederhana berikut ini.

	1	2	3	4	5
1					
2			■		
3	S		■		F
4			■		
5					

Gambar 2. Salah satu contoh persoalan path-finding. S adalah titik awal dan F adalah titik akhir

S merupakan titik awal dan F merupakan titik akhir. Tujuan algoritma path-finding adalah menemukan jalan terpendek dari S ke F. Sebagai batasan persoalan, pergerakan hanya dapat dilakukan secara vertikal atau horizontal (tidak diagonal).

Langkah pertama, jadikan titik (1,1) sebagai simpul akar dari pohon ruang penyelesaian. Selanjutnya, bangkitkan anak-anaknya, yaitu titik-titik yang berada di atas (2,1), kanan (3,2), dan bawahnya (4,1) (titik di sebelah kiri tidak ada).

Sebagai catatan, pembangkitan selalu dimulai dari sisi atas, kanan, bawah, dan kiri secara berurutan.

Untuk menghitung cost tiap simpul, terlebih dahulu kita *set* cost akar adalah 0 dan menentukan fungsi heuristic. Fungsi heuristic yang akan digunakan adalah

$g(i) = x_{finish} - x_i + y_{finish} - y_i$

dengan
x_{finish} = nomor baris titik finish
y_{finish} = nomor kolom titik finish
x_i = nomor baris titik simpul i
yi = nomor kolom titik simpul i

Dengan persamaan

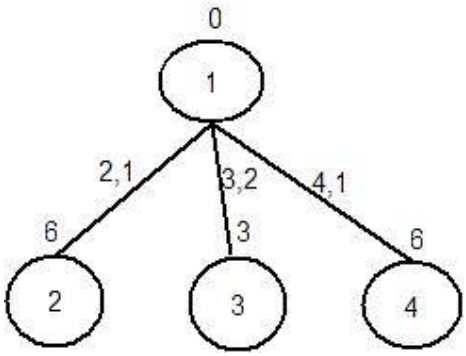
$c(i) = f(i) + g(i)$

dapat ditentukan cost untuk simpul i

$c(i) = f(i) + |x_{finish} - x_i| + |y_{finish} - y_i|$

dapat dihitung
c(2) = 0 + |3 – 2| + |5 – 1| = 6
c(3) = 0 + |3 – 3| + |5 – 2| = 3
c(4) = 0 + |3 – 2| + |5 – 1| = 6

Karena simpul 3 yang memiliki cost terkecil, maka simpul 3 yang pertama kali dibangkitkan. Akan dihasilkan pohon ruang penyelesaian sebagai berikut.



Gambar 3. Pohon ruang penyelesaian

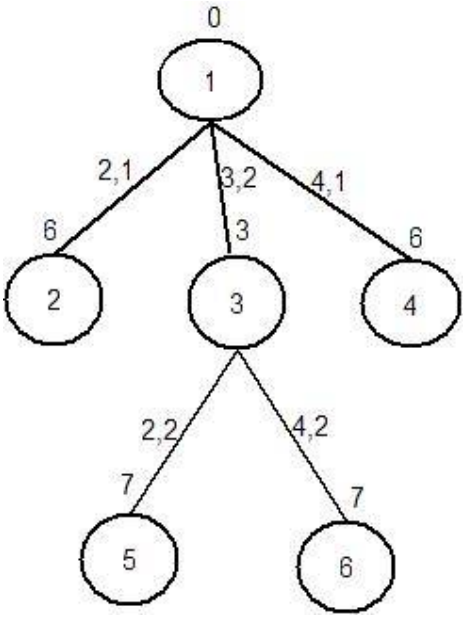
	1	2	3	4	5
1					
2	2(1)		■		
3	1	3(1)	■		F
4	4(1)		■		
5					

Gambar 4. Gambar pada matriks yang berseuaian

keterangan:
= simpul hidup
= simpul mati
n(m) = simpul yang bersesuaian simpul n anak dari m

Nomor simpul merupakan urutan dibangkitkannya simpul. Nomor pada sisi menunjukkan lokasi titik yang bersesuaian dengan simpul. Nomor yang terletak pada bagian atas simpul merupakan nilai cost untuk simpul tersebut.

Selanjutnya akan dibangkitkan anak-anak simpul 3. Titik-titik yang bersesuaian adalah (2,2) dan (4,2). Titik (3,1) dan (3,3) tidak dibangkitkan karena (3,1) sudah dibangkitkan dan (3,3) merupakan constrain yang tidak bisa dilewati. Dengan perhitungan cost yang sama seperti sebelumya, akan dihasilkan pohon ruang penyelesaian sebagai berikut.



Gambar 5. Pohon ruang penyelesaian

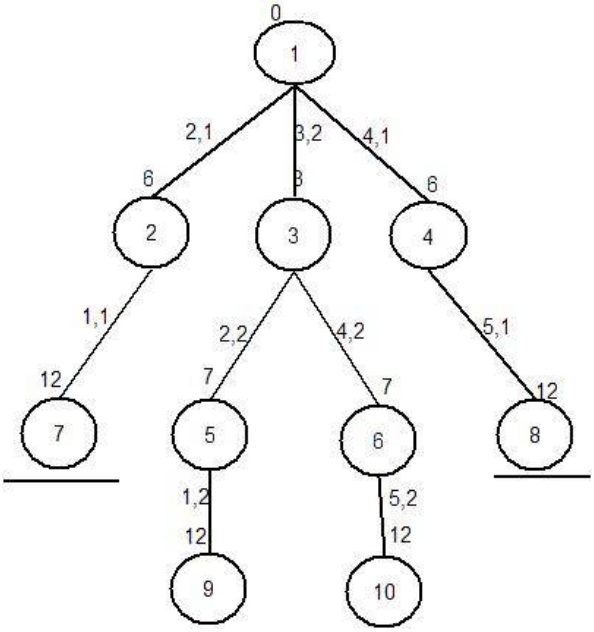
	1	2	3	4	5
1					
2	**2(1)**	5(3)			
3	1	3(1)			F
4	4(1)	6(3)			
5					

Gambar 6. Gambar pada matriks yang berseuaian

Karena simpul 2 merupakan simpul hidup yang memiliki nilai cost yang paling kecil, maka simpul 2 akan dibangkitkan anak-anaknya. Simpul 2 dibangkitkan lebih dulu daripada simpul 4 karena simpul 2 terletak lebih kiri.

Simpul 2 akan hanya membangkitkan satu anak yaitu simpul 7 yang bersesuaian dengan titik (1,1). Simpul yang bersesuaian dengan titik (2,2), (3,1) tidak dibangkitkan simpul-simpul tersebut sudah pernah dibangkitkan. Berikut ini pohon ruang penyelesaian yang digasilkan. Simpul 2 akan membangkitkan simpul anak dengan cost 12. Karena simpul hidup dengan cost terkecil sekarang adalah simpul 4, maka akan dibangkitkan anak-anak simpul 4 yang akan menghasilkan satu simpul anak dengan cost 12.

Dengan membangkitkan simpul-simpul anak dari simpul 5 dan 6 secara berurutan, akan didapat bahwa simpul 7 dan 8 merupakan simpul daun dan bukan merupakan solusi, sehingga simpul 7 dan 8 akan "dibunuh". Pada gambar, ini ditandai dengan garis horizontal di bawah simpul. Pohon ruang penyelesaiannya adalah sebagai berikut.

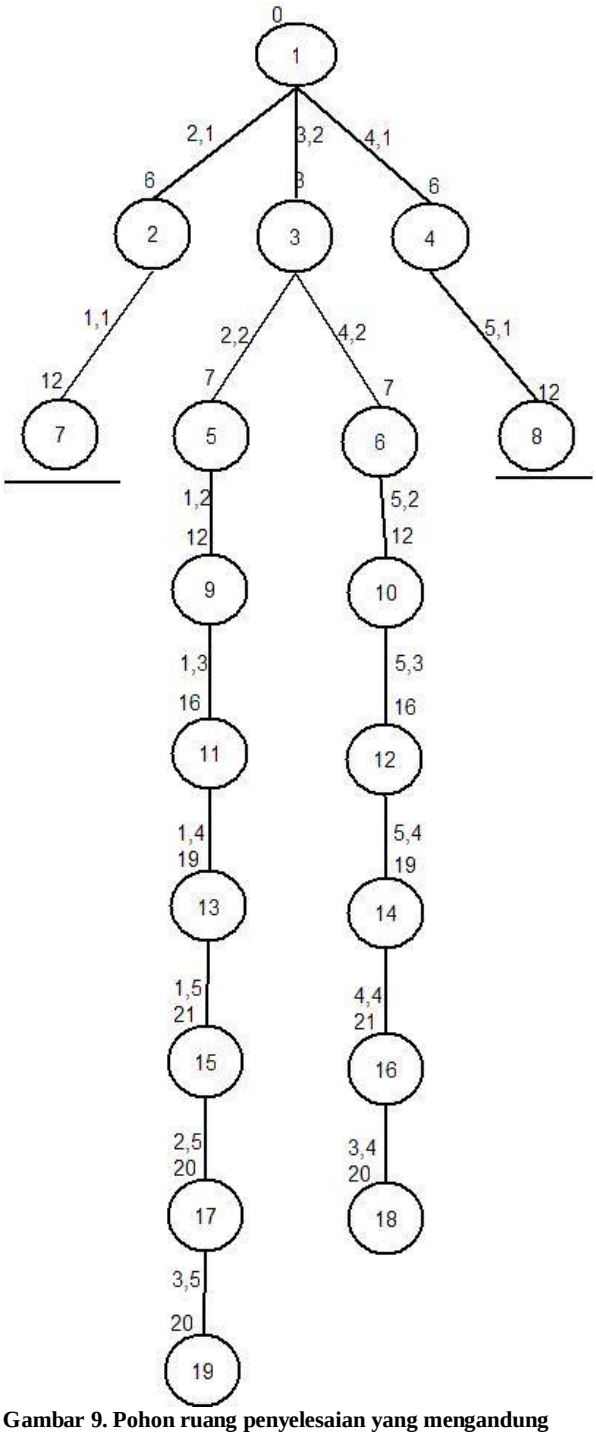


Gambar 7. Pohon ruang penyelesaian

	1	2	3	4	5
1	7(2)	9(5)			
2	2(1)	5(3)			
3	1	3(1)			F
4	4(1)	6(3)			
5	8(4)	10(6)			

Gambar 8. Gambar pada matriks yang berseuaian

Dengan meneruskan pembangkitan simpul-simpul anak, pada akhirnya aka didapatkan pohon ruang penyelesaian sebagai berikut.



Gambar 9. Pohon ruang penyelesaian yang mengandung solusi

	1	2	3	4	5
1	7(2)	9(5)	11(9)	13(11)	15(13)
2	2(1)	5(3)			17(15)
3	1	3(1)		18(16)	19(17)
4	4(1)	6(3)		16(14)	
5	8(4)	10(6)	12(10)	14(12)	

Gambar 8. Gambar pada matriks yang berseuaian

Untuk mengetahui lintasan penyelesaian, dilakukan runut dari belakang, atau dari simpul anak ke orang tuanya, yaitu simpul 19 – 17 – 15 – 13 – 11 – 9 – 5 – 3 – 1. Dengan membalik urutan di atas dan menuliskan titik-titik yang bersesuaian, dapat ditemukan lintasan solusi dari titik awal ke titik akhir, yaitu titik (3,1) , (3,2), (2,1), (1,2), (1,3), (1,4), (1,5), (2,5), dan (3,5).

3.2 Pseudo-Code

Algoritma

```
procedure findPath()
  while not found and ada simpul hidup
      node n1 = findLeastCost();
      generateChilds(n1);

function findLeastCost() : node
{mengembalikan node dengan cost terkecil}

procedure generateChilds(node n)
{membangkitkan simpul anak n}
```

4. UJI SIMULASI

Algoritma ini akan diuji coba pada kasus yang lebih rumit. yaitu pada matriks ukurang 21 x 51 dengan hambatan-habatan yang lebih banyak. Pada bagian ini hanya akan diperlihatkan hasil programnya (PathFinder 1.0). Program tersebut dibuat oleh penulis untuk keperluan pengujian.

Berikut ini adalah matriks yang diberikan.

Gambar 9. Screenshot program

Tanda ‘x’ pada gambar merupakan hambatan yang tidak dapat dilalui. Tanda ‘S’ merupakan titik awal dan tanda ‘F’ merupakan titik akhir. Hasil dari program tersebut adalah seperti di bawah ini.

Gambar 9. Screenshot program (solusi)

Tanda ‘*’ menyatakan lintasan penyelesaian. Jika tidak terdapat penyelesaian, maka program akan mencetak ‘Tidak ada solusi’.

REFERENSI

[1] Lester, Patrick, “A* Pathfinding for Beginners”, http://www.policyalmanac.org/games/aStarTutorial.htm, 2005. Waktu akses : 22 Mei 2007 pukul 14.00

[2] Munir, Rinaldi, “Diktat Kuliah IF2251 Startegi Algoritmik”, 2007.