



LABORATORIUM PEMBELAJARAN ILMU KOMPUTER
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA

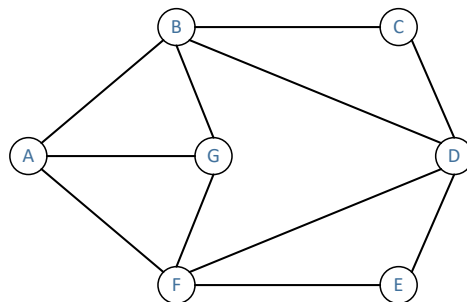
BAB : INFORMED SEARCH
NAMA : REZA AZZUBAIR WIJONARKO
NIM : 155150200111182
TANGGAL : 05/04/2017
JENIS : LATIHAN
ASISTEN : - ANNISA FITRIANI NUR
- RISKI PUSPA DEWI D. P..

ACC

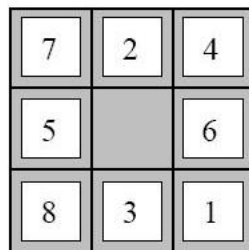
A. DEFINISI MASALAH

1. Deskripsikan dan tentukan nilai $h(n)$ dan $g(n)$ dari permasalahan berikut

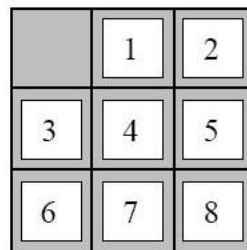
a. Graph problem, dimana titik A adalah state awal dan titik D adalah goal.



b. 8-puzzle problem

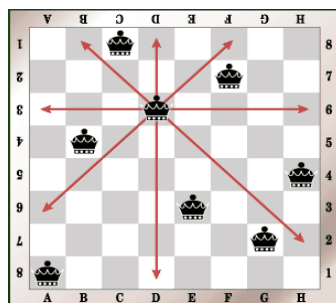


start



goal

c. 8-queen problem



2. Selesaikan permasalahan permainan tic tac toe berikut menggunakan algoritma MiniMax dan Alpha Beta Pruning

	X	
O	X	X
	O	O

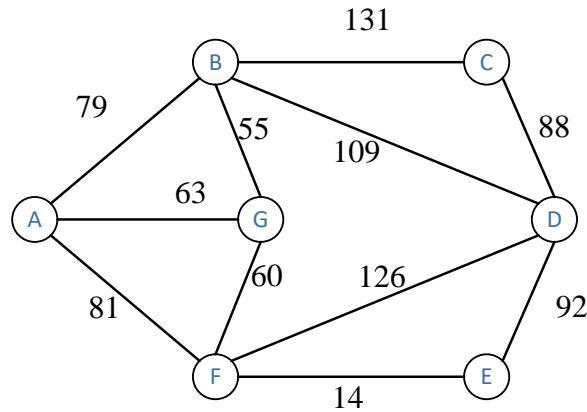
- a. Tentukan dahulu aturannya !
 - b. Selesaikan menggunakan algoritma MiniMax dan Alpha Beta Prunning
 - c. Jika kedua pemain, baik X dan O sama-sama menerapkan algoritma Alpha Beta Prunning, siapakah yang menang?
3. Sesuai dengan graph beserta informasi $g(n)$ & $h(n)$ yang telah anda definisikan pada jawaban soal nomor 1 a, carilah rute dari kota A ke kota D secara manual menggunakan :
 - a. Greedy BFS
 - b. A*

B. JAWAB

1.

a) Graph problem

Deskripsi masalah: mencari jarak terpendek antara titik A dan titik D.
 $g(n)$: nilai perpindahannya antar kota. Besarannya dapat dilihat dalam gambar berikut



$h(n)$: jarak perkiraan dari kota D ke kota-kota lain. Besarannya dapat dilihat dalam bagan berikut

A = 180

B = 90

C = 89

D = 0

E = 110

F = 79

G = 100

b) 8-puzzle problem

Deskripsi masalah: mengurutkan semua angka sehingga sama dengan goal state-nya.

$g(n)$: nilai perpindahannya. Besarannya adalah 1 untuk tiap perpindahan.

$h(n)$: jumlah tiles yang salah posisi atau jarak Manhattan angka-angka yang masih salah posisi.

c) 8-queen problem

Deskripsi masalah: mengatur semua catur queens sehingga tidak ada catur-catur queens yang sama dalam satu baris horizontal, vertikal, maupun diagonal.

$g(n)$: nilai perpindahannya/peletakannya. Besarannya adalah 1 untuk tiap perpindahan/peletakan.

$h(n)$: posisi-posisi catur queen yang tidak sehorizontal, severtikal, dan sediagonal.

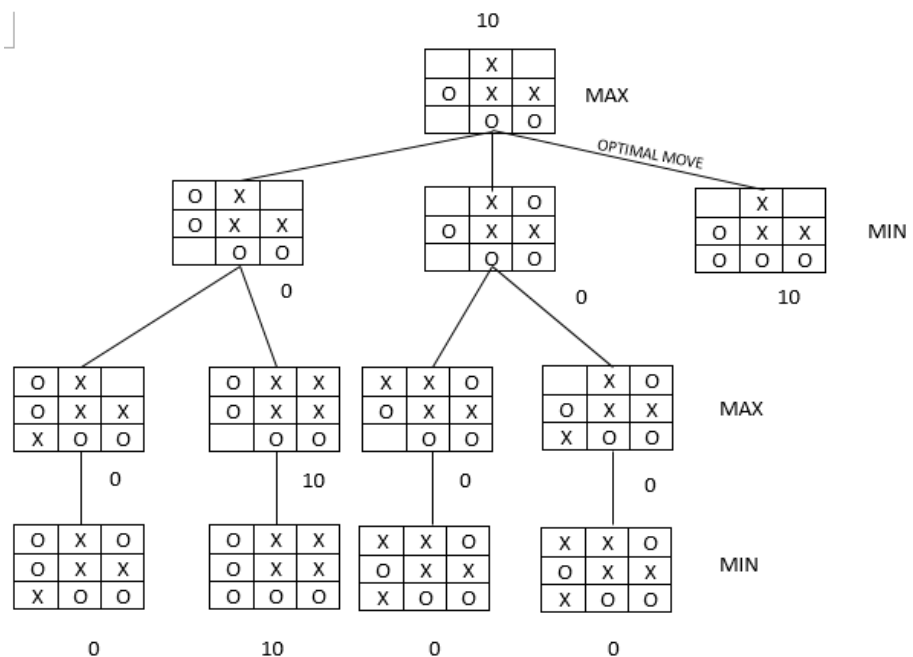
2. Penyelesaian permasalahan permainan tic-tac-toe menggunakan algoritma MiniMax dan Alpha-Beta Pruning.

- a) Aturan permainan: permainan dilakukan antara dua orang/agen. Salah satu bermain sebagai “O” dan “X”. Tujuan dari permainan (goal state) adalah ketika salah satu simbol mengisi kotak-kotak yang berukuran 3x3 dengan 3 simbol yang sama secara horizontal, vertikal, atau pun diagonal. Pemain yang menggunakan simbol “O” bermain duluan (urutan bisa terserah para pemain). Setelah pemain dengan simbol “O” (atau simbol “X”) mengisi salah satu kotaknya, pemain lain mengisi salah satu kotak. Kedua pemain saling bergantian hingga salah satu menang atau semua kotak sudah terisi penuh (draw).
- b) Penyelesaian tic-tac-toe dengan intial state

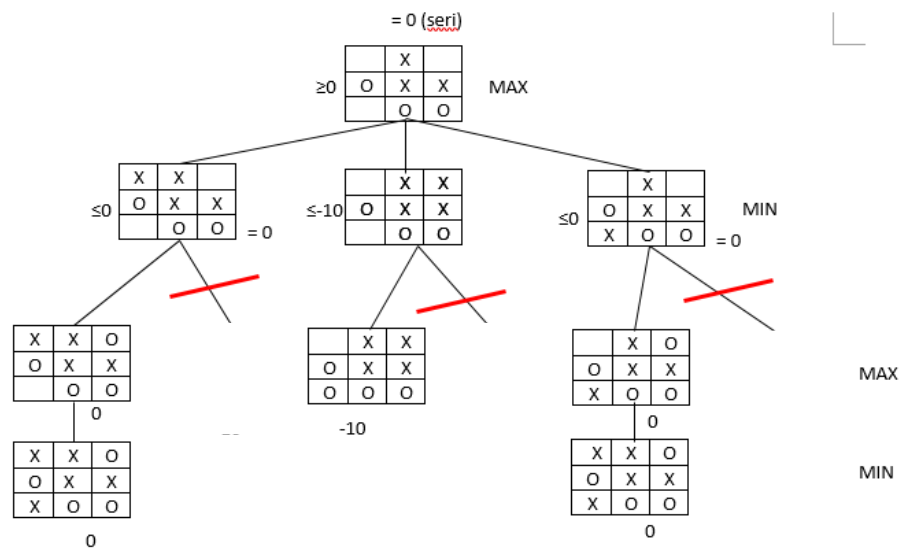
	X	
O	X	X
	O	O

misal bila menang, draw, atau kalah, skor yang didapatkan berturut-turut +10, 0, dan -10, berikut penyelesaiannya menggunakan algoritma

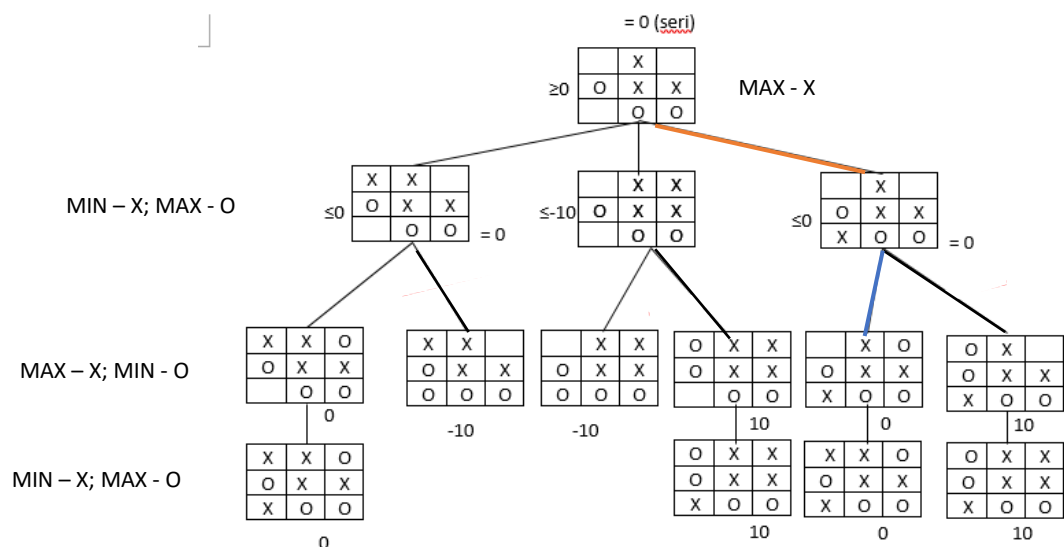
- MiniMax (agen/pemain sebagai “O”)



- Alpha-Beta Pruning (agen/pemain sebagai “X”)

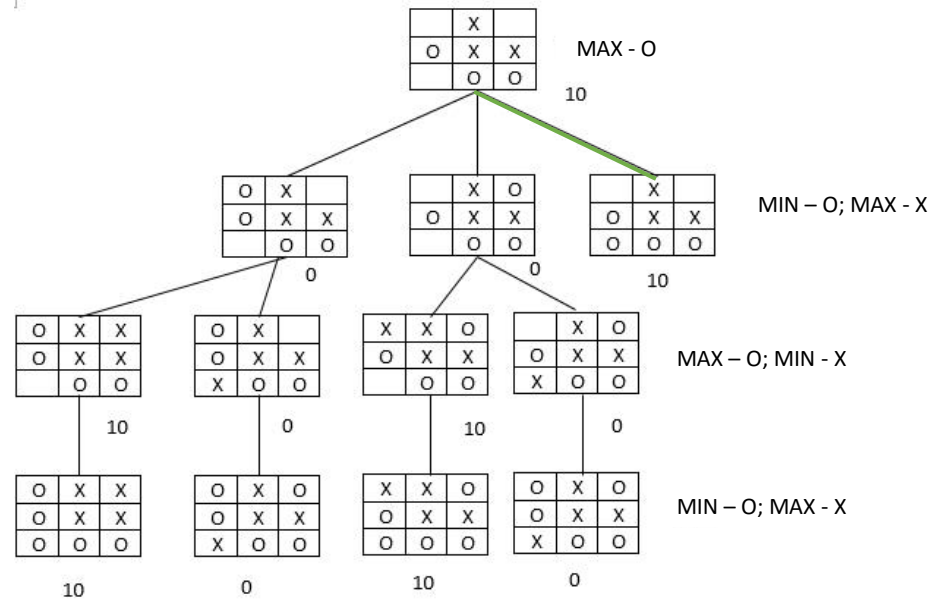


- c) Menurut kami, bila kedua pemain menerapkan alpha-beta pruning, pemenang bergantung pada siapa yang bermain terlebih dahulu dan initial state-nya. Bila pemain “X” bermain terlebih dahulu, keduanya akan seri. Berikut pembuktiannya bila pemain “X” main lebih dahulu (+10 bila pemain “X” menang; -10 bila pemain “O” menang):



Pertama, pemain “X” akan memilih cabang yang paling kanan karena hasil algoritma a-b-nya menunjukkan bahwa yang paling menguntungkan adalah cabang paling kanan. Kedua, pemain “O” akan memilih cabang yang kiri agar pemain “O” tidak kalah (bila pemain “O” mengambil cabang kanan, pemain “X” akan menang sehingga pemain “O” akan kalah). Kemudian, karena tidak ada langkah lagi yang bisa diambil, semua pemain “X” akan mengisi kotak terakhir sehingga mengakibatkan keduanya seri.

Namun, lain lagi bila pemain “O” yang bermain lebih dahulu, pemain “O” yang akan menang. Berikut pembuktiannya:



Dari hasil algoritma di atas, pemain “O” akan memilih jalur yang paling kanan agar menang.

3. Pencarian rute terpendek menggunakan

a) Greedy best-first search

Prinsip kerja algoritma greedy best-first search: mengunjungi titik yang memiliki nilai heuristik ($h(n)$) paling kecil dibanding titik-titik lain; bisa mengunjungi titik yang terlewat selama nilai heuristik titik itu paling kecil dibandingkan nilai heuristik pada titik-titik lain.

$h(n)$: jarak perkiraan dari kota D ke kota-kota lain. Besarannya dapat dilihat dalam bagan berikut

A = 180

B = 90

C = 89

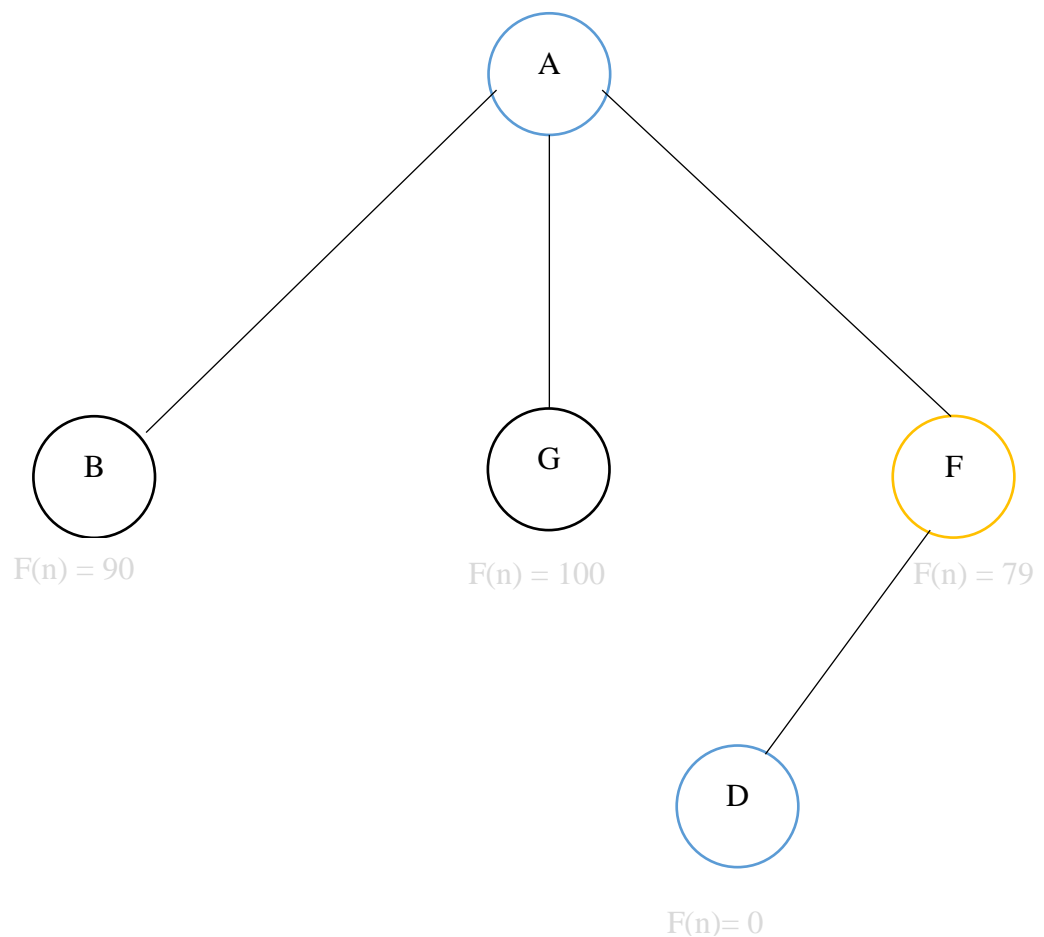
D = 0

E = 110

F = 79

G = 100

$F(n)$: total jarak yang ditempuh = $h(n)$.



Rute yang dipilih dari A: F – D

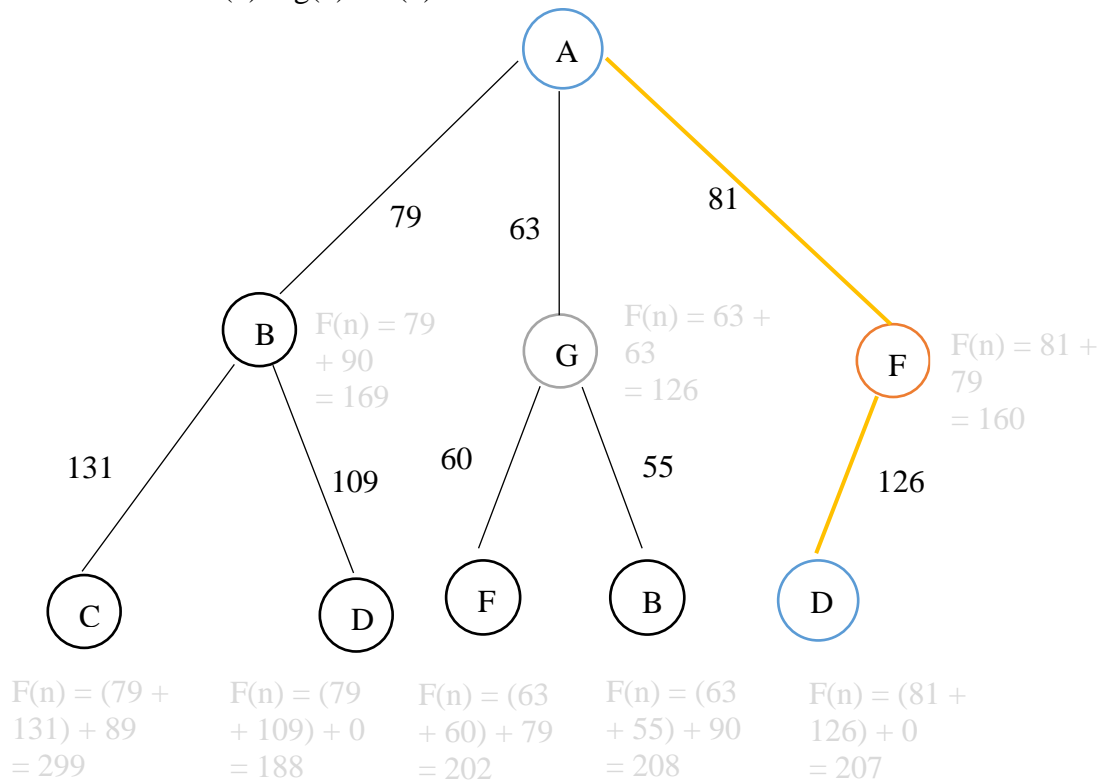
b) A* (A-star)

Prinsip kerja algoritma A*: mengunjungi titik yang memiliki nilai heuristik ($h(n)$) dan bobot riil ($g(n)$) paling kecil dibanding titik-titik lain; bisa mengunjungi titik yang terlewat selama nilai heuristik titik itu paling kecil dibandingkan nilai heuristik pada titik-titik lain. Nilai bobot akan bertambah seiring dengan banyaknya titik yang dikunjungi, sedangkan nilai heuristik akan tetap bergantung pada titik yang akan dituju.

Selain $h(n)$ dan $F(n)$, terdapat satu fungsi tambahan:

$g(n)$: jarak real yang ditempuh. Bisa diakumulasi dari jarak real sebelumnya yang sudah ditempuh.

$$F(n) = g(n) + h(n).$$



Rute yang dipilih dari A: F – D



LABORATORIUM PEMBELAJARAN ILMU KOMPUTER
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA

BAB : INFORMED SEARCH
NAMA : REZA AZZUBAIR WIJONARKO
NIM : 155150200111182
TANGGAL : 05/04/2017
JENIS : TUGAS
ASISTEN : - ANNISA FITRIANI NUR
- RISKI PUSPA DEWI D. P..

ACC

A. DEFINISI MASALAH

Buatlah program (“untuk soal latihan praktikum nomor 3 b” atau “ untuk contoh masalah maze/labirin diatas dengan menggunakan algoritma A* dengan nilai $f(n) = g(n) + h(n)$, misal $g(n)$ adalah tingkat kedalaman penelusuran, dan $h(n)$ adalah jarak Manhattan atau Euclidian”) dengan memodifikasi program pada contoh program diatas.

B. JAWAB

Node.java

```
1 package Tugas;
2 public class Node {
3     double f,g;
4     final double h;
5     String nama;
6     Node pre;
7     Edge[] tetangga;
8
9     public Node( String nama, double h) {
10         this.h = h;
11         this.nama = nama;
12         g=0;
13         f=0;
14     }
15
16     public void setTetangga(Edge[] tetangga) {
17         this.tetangga = tetangga;
18     }
19
20     @Override
21     public String toString(){
22         return nama;
23     }
24 }
```

Edge.java

```
1 package Tugas;
2 public class Edge {
3     final double gn;
4     final Node tujuan;
5     public Edge( Node tujuan, double g) {
6         this.gn = g;
7         this.tujuan = tujuan;
8     }
9 }
```

aStar.java

```
1 package Tugas;
2 import java.util.*;
3 public class aStar {
4
5     LinkedList<Node> rute = new LinkedList();
6     double finalCost=0;
7
8     public void start(Node awal, Node target) {
9         LinkedList<Node> Explored = new LinkedList();
10        PriorityQueue<Node> queue = new
11        PriorityQueue<>(20, (Node x, Node y) -> {
12            if (x.f > y.f) {
13                return 1;
14            }
15        });
16    }
```

14	} else if (x.f < y.f) {
15	return -1;
16	} else {
17	return 0;
18	}
19	} //override compare method
20);
21	
22	queue.add(awal);
23	boolean ketemu = false;
24	while ((!queue.isEmpty()) && (!ketemu)) {
25	
26	Node curr = queue.poll();
27	Explored.add(curr);
28	if (curr.nama.equals(target.nama)) {
29	ketemu = true;
30	makePath(curr);
31	break;
32	}
33	
34	for (Edge i : curr.tetangga) {
35	Node next = i.tujuan;
36	double cost = i.gn;
37	double g = cost + curr.g;
38	double f =g + next.h;
39	System.out.println("-----
40	-----");
41	System.out.println(curr+" -> "+next+" =
42	"+g+" + "+next.h+"\t= "+" "+f);
43	
44	if (Explored.contains(next) && f >=
45	next.f) {
46	} else if (!queue.contains(next) f <=
47	next.f) {
48	Node temp=next;
49	next.pre = curr;
50	next.g = g;
51	next.f = f;
52	if (queue.contains(temp)) {
53	queue.remove(temp);
54	}
55	queue.add(next);
56	}
57	}
58	}
59	}
60	
61	private void makePath(Node target) {
62	rute.push(target);
63	Node path=target;
64	while (path.pre != null) {
65	rute.push(path.pre);
66	path = path.pre;
67	}
68	}
69	
70	public void getPath() {
71	System.out.println("-----
72	-----");

```

73     Node pop;
74     System.out.print("Path = [ ");
75     while (!rute.getLast().equals(rute.peek())) {
76         pop = rute.pop();
77         finalCost+=pop.f;
78         System.out.print(pop + " (" +pop.f+") -> ");
79     }
80     pop=rute.pop();
81     finalCost+=pop.f;
82     System.out.print( pop+ " (" +pop.f+") ]\n");
83     System.out.println("Final Cost (f(n)) total =
84 "+finalCost);
85     System.out.println("-----
86 -----");
87 }
88 }

```

Main.java

```

1  package Tugas;
2  public class Main {
3
4      public static void main(String[] args) {
5          System.out.println("-----
6          -----");
7          System.out.println("                A-STAR
8  SEARCH");
9          System.out.println("-----
10         -----");
11         String[] nama={"A","B","C","D","E","F","G"};
12         double[] hn={150,90,75,0,88,118,138};
13         Node n[]=new Node[nama.length];
14         for (int i = 0; i < n.length; i++) {
15             n[i]=new Node(nama[i],hn[i]);
16         }
17
18         n[0].setTetangga(new Edge[]{
19             new Edge(n[1], 79),
20             new Edge(n[6], 63),
21             new Edge(n[5], 81)});
22         n[1].setTetangga(new Edge[]{
23             new Edge(n[0], 79),
24             new Edge(n[6], 55),
25             new Edge(n[2], 131),
26             new Edge(n[3], 109),});
27         n[2].setTetangga( new Edge[]{
28             new Edge(n[1], 131),
29             new Edge(n[3], 88)});
30         n[3].setTetangga( new Edge[]{
31             new Edge(n[2], 88),
32             new Edge(n[4], 92),
33             new Edge(n[1], 109),
34             new Edge(n[5], 126)});
35         n[4].setTetangga( new Edge[]{
36             new Edge(n[5], 148),
37             new Edge(n[3], 92)});
38         n[5].setTetangga(new Edge[]{

```

```

39         new Edge(n[4], 148),
40         new Edge(n[0], 81),
41         new Edge(n[3], 126),
42         new Edge(n[6], 60));
43     n[6].setTetangga(new Edge[] {
44         new Edge(n[0], 63),
45         new Edge(n[1], 55),
46         new Edge(n[5], 60));
47
48     aStar search = new aStar();
49     search.start(n[0], n[3]);
50     search.getPath();
51 }
52 }

```

Penjelasan

Node.java

- Class node berisi informasi sebuah kota :nama, nilai , tetangga dan node predesesor untuk mempermudah tracing
- Baris 9-14: constructor dengan parameter String nama dan double h untuk menginisialisasi object Node
- Baris 16-18: method setTetangga berfungsi untuk mengeset nilai tetangga sesuai dengan parameter

Edge.java

- Class Edge tidak memiliki method, tetapi berfungsi sebagai penghubung antar-Node dan menyimpan path cost (g(n)) aStar.java

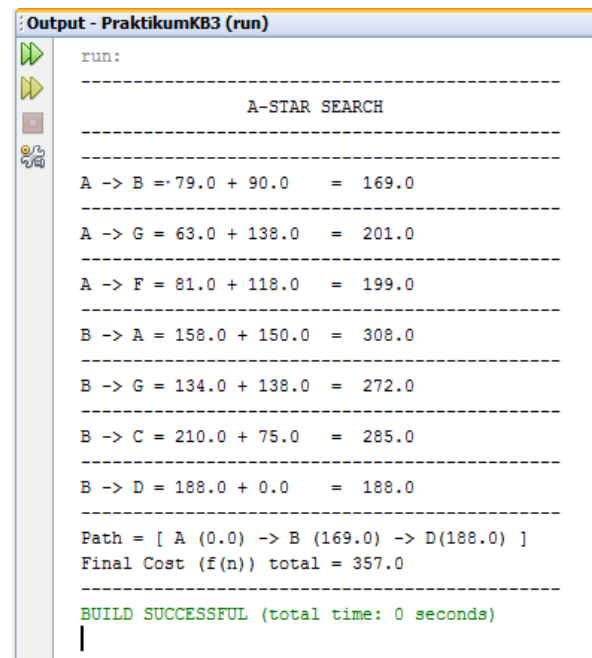
aStar.java

- Class aStar merupakan class tempat jantung program berada
- Baris 8-59: method void start memulai proses search dengan membuat list Explored dan queue (antrean). Pada queue isi antriannya yang diatur sesuai nilai f node dari terkecil. Pencarian dilakukan dengan perulangan yang akan berhenti jika nilai variable ketemu true atau antrian queue kosong. Dalam memasukkan node ke antrian, node dicek nilai f-nya dan dicek apakah sudah diexplore. Jika nama node sekarang sama dengan nama target jalankan method makePath untuk menyimpan rute dan ganti nilai variable ketemu jadi true.
- Baris 61-68: Method makePath berparameter State terakhir dari pencarian. Lalu method akan melakukan push state tersebut dan predecessor nya secara berurutan sampai ke state yang tidak memiliki predecessor (initial state) ke linkedlist rute.
- Baris 70-87: Method getPath digunakan untuk mencetak rute dengan mencetak nilai pop dari linkedlist route dan cost-nya. Method getPath digunakan untuk mencetak rute dengan mencetak nilai pop dari linkedlist route dan cost-nya.

Main method

- Main method diawali dengan menginisialisasi nilai kota dan mengisi nilai tetangganya dan menjalankan method start dari class aStar

Screenshot:



```
Output - PraktikumKB3 (run)
run:
-----
                        A-STAR SEARCH
-----
A -> B = 79.0 + 90.0    = 169.0
-----
A -> G = 63.0 + 138.0   = 201.0
-----
A -> F = 81.0 + 118.0   = 199.0
-----
B -> A = 158.0 + 150.0  = 308.0
-----
B -> G = 134.0 + 138.0  = 272.0
-----
B -> C = 210.0 + 75.0   = 285.0
-----
B -> D = 188.0 + 0.0    = 188.0
-----
Path = [ A (0.0) -> B (169.0) -> D(188.0) ]
Final Cost (f(n)) total = 357.0
-----
BUILD SUCCESSFUL (total time: 0 seconds)
|
```

C. KESIMPULAN

1. Definisikan konsep dasar metode informed search menurut pemahaman Anda!
Konsep dasar metode informed search sama seperti dengan uninformed search yakni mencari solusi terbaik dari sebuah permasalahan dalam keadaan sekarang, tetapi juga mempertimbangkan fungsi heuristik suatu permasalahan itu. Fungsi heuristik adalah informasi tambahan yang dimiliki oleh pelaku algoritma (agen) agar dapat memecahkan masalah yang dihadapinya dengan sebaik-baiknya. Fungsi heuristik bisa ditentukan secara statis oleh pembuat algoritma atau desainer program dan juga ditentukan secara dinamis oleh si agen dengan mempertimbangkan hasil olahan percepts-nya.
2. Sebutkan kelebihan & kekurangan metode informed search!
Kebanyakan algoritma yang menggunakan metode informed search dapat memberi hasil yang efektif dan, bila fungsi heuristiknya bagus, dapat menyelesaikan masalah dalam kompleksitas waktu dan ruang memory yang cukup rendah. Namun, karena metode informed search bergantung pada keakuratan fungsi heuristiknya, dalam kasus terburuk (worst case) kompleksitas waktu dan ruang memory-nya bisa sebesar 2^n yang berarti kompleksitasnya sangat tinggi.
3. Jelaskan bentuk permasalahan yang hanya dapat diselesaikan menggunakan metode informed search!
Metode informed search sangat cocok digunakan untuk menyelesaikan permasalahan-permasalahan yang environment-nya sudah diketahui sebelumnya sehingga dari keadaan environment itu dapat diciptakan fungsi heuristik menuju solusinya. Contoh, ada permasalahan untuk mencari jarak terpendek dari kota A ke kota B. Seandainya lokasi kota-kota dan jarak antarkotanya sudah diketahui, kita dapat membuat fungsi heuristik sehingga kita dapat dengan akurat menentukan jarak yang terpendek dari kota A ke kota B menggunakan fungsi heuristik itu.