CS 440/ ECE 448: Introduction to Artificial Intelligence Spring 2009

Problem Set 1 Solutions

1. [Search Spaces]

(a) Depth-First Search

Node	Queue
-	A
A	$B\ C$
B	$D\ F\ C$
D	$J\ K\ F\ C$
J	$K\ F\ C$
K	$S\ T\ F\ C$
S	$T\ F\ C$
T	$F\ C$

(b) Best-First Search ($\varphi_j = h_j$)

Node	Queue
-	A_{12}
A	$B_8\ C_{10}$
B	$C_{10}\ D_{11}\ F_{19}$
C	$H_3\ D_{11}\ G_{12}\ F_{19}$
H	$Q_0\ D_{11}\ G_{12}\ R_{17}\ F_{19}$
Q	$D_{11}\ G_{12}\ R_{17}\ F_{19}$

(c) Uniform-Cost Search ($\varphi_j = g_j$)

Node	Queue
-	A_0
A	$C_8\ B_{10}$
C	$B_{10}\ G_{11}\ H_{15}$
B	$G_{11}\ D_{15}\ F_{15}\ H_{15}$
G	$P_{14}\ D_{15}\ F_{15}\ H_{15}\ N_{19}$
P	$D_{15}\ F_{15}\ H_{15}\ N_{19}$
D	$F_{15}\ H_{15}\ J_{19}\ N_{19}\ K_{25}$
F	$H_{15}\ J_{19}\ N_{19}\ M_{20}\ L_{23}\ K_{25}$
H	$J_{19}\ N_{19}\ M_{20}\ Q_{20}\ L_{23}\ K_{25}\ R_{25}$
J	$N_{19}\ M_{20}\ Q_{20}\ L_{23}\ K_{25}\ R_{25}$
N	$M_{20}\ Q_{20}\ L_{23}\ K_{25}\ R_{25}\ W_{26}\ V_{28}$
M	$Q_{20}\ L_{23}\ K_{25}\ R_{25}\ W_{26}\ V_{28}$
Q	$L_{23}\ K_{25}\ R_{25}\ W_{26}\ V_{28}$

(d) Hill-Climbing

Node	Queue
-	A
A	B
B	D
D	J
J	-
FAIL	

(e) A* ($\varphi_j = g_j + h_j$)

Node	Queue
-	A_{12}
A	$B_{18}\ C_{18}$
B	$C_{18}\ D_{26}\ F_{34}$
C	$H_{18}\ G_{23}\ D_{26}\ F_{34}$
H	$Q_{20}\ G_{23}\ D_{26}\ F_{34}\ R_{42}$
Q	$G_{23}\ D_{26}\ F_{34}\ R_{42}$

(f) The shortest path from A to E is ($A \rightarrow D \rightarrow C \rightarrow B \rightarrow E$). The φ matrix is given below. Each distance number has a subscript indicating which node in the previous column the arrow points to.

	0	1	2	3	4
A	0	-	-	-	-
B	-	9_A	7_C	5_C	-
C	-	6_A	4_D	-	-
D	-	3_A	-	-	-
E	-	10_A	8_D	-	7_B

2. [Search Space Formulation]

   (a) Suppose we consider each land mass to be a node in a graph and each bridge to be an edge. We can label the graph as in Figure 1.

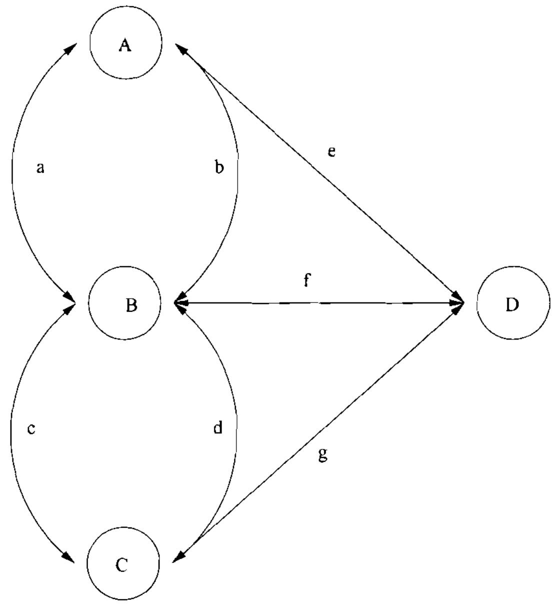

Figure 1: The Seven Bridges of Konigsberg as a graph

   A state can be represented by an ordered set of edges traversed. The initial state is thus represented by the empty set and a final state is an ordered set of the edges that when traversed will return you to your starting node.

   (b) An edge in our search space represents a legal traversal of one edge in the graph from our current location. That is, the edge in the graph must be connected to our current location and not yet have been traversed. For two states to be connected by an edges their ordered sets of edges must be identical except that the child state must be have exactly one more edge at the end of its set.

   (c) There are no reasonable heuristics for this problem. This is because the optimal path must traverse every edge and at every level of the search tree, every node has exactly the same number of edges in its ordered set. Thus all goal states will be the same distance from the root of the search tree, so there is no shortest path to a goal state.

   The general problem is to find an Eulerian cycle in the graph. It can be proved by a theorem that a graph has an Eulerian cycle if and only if each node has an even number of outgoing edges. However, this is not a reasonable heuristic because it only indicates whether or not a solution exists for the current problem, not how to find it.

   (d) By the theorem stated in part (c), this graph has no solution because not all nodes have an even number of outgoing edges.

3. [Triangle Peg Solitaire]

(a) Each state is a set of numbered pegs remaining on the board. Two are states are connected by an edge if there is a legal move from the arrangement of pegs in the parent node to the arrangement of pegs in the child state. The initial state every hole filled with a peg except at 5 (4 for part (d)). All final states will have exactly one peg remaining on the board.

(b) An implementation in MATLAB is included in the file hw1.m

(c) From the program, we obtain the following output:

```
Move peg 14 to 5
Move peg 7 to 9
Move peg 12 to 14
Move peg 3 to 8
Move peg 2 to 7
Move peg 11 to 4
Move peg 4 to 13
Move peg 10 to 3
Move peg 1 to 6
Move peg 14 to 12
Move peg 6 to 13
Move peg 12 to 14
Move peg 15 to 13
```

The final state has a lone peg at 13. From this initial state it is not possible to find a different final state.

(d) From the program (modified to change the initial state), we obtain the following output:

```
Move peg 6 to 4
Move peg 1 to 6
Move peg 4 to 1
Move peg 10 to 3
Move peg 1 to 6
Move peg 14 to 5
Move peg 6 to 4
Move peg 7 to 2
Move peg 12 to 14
Move peg 15 to 13
Move peg 13 to 4
Move peg 2 to 7
Move peg 11 to 4
```

For this run of the program, the final state has a lone peg at 4. By using a different search method or processing the child nodes in a different order it is possible to reach final states with a lone peg at either 6 or 13.