CS 540-1 Homework assignment #5

Due Nov 24 Jianyi Liu
Jianyi.liu@wisc.edu

Question 1: Search Algorithms [50]

Little Red Riding Hood wants to go to her grandmother's house in the forest to give her some cake. The forest is a dangerous place where the Big Bad Wolf resides. Little Red Riding Hood's mother has instructed her not to venture into the forest and to only go along the road. The following 5×5 grid shows the area that Little Red Riding Hood needs to traverse. She starts from square A (denoted by R). Her grandmother's house is located at square N (denoted by G). The forests are denoted by F (on squares F, I, M, R, U and Y). Your task is to help Little Red Riding Hood get to her grandmother's house. Assume that she can only move in four directions namely left, down, right and up. She cannot travel diagonally and will not go into the forests. **Also assume that** the successor function will cause legal moves to be examined in a **counter-clockwise** order: left; down; right; up. Note that not all of these moves may be possible for a given square (squares with forest on them will never be examined).

A R	B	C	D	E
F F	G	H	I F	J
K	L	M F	N G	O
P	Q	R F	S	T
U F	V	W	X	Y F

++

a) [10] Using **Depth-First Search**, list the squares in the order they are expanded (including the goal node if it is found). **Square A** is expanded first (hint: State B will be examined next). **Assume cycle checking is done** so that a node is not generated in the search tree if the grid position already occurs on the path from this node back to the root node (i.e., Path Checking DFS). Write down the list of states you expanded in the order they are expanded. Write down the solution path found (if any), or explain why no solution is found.

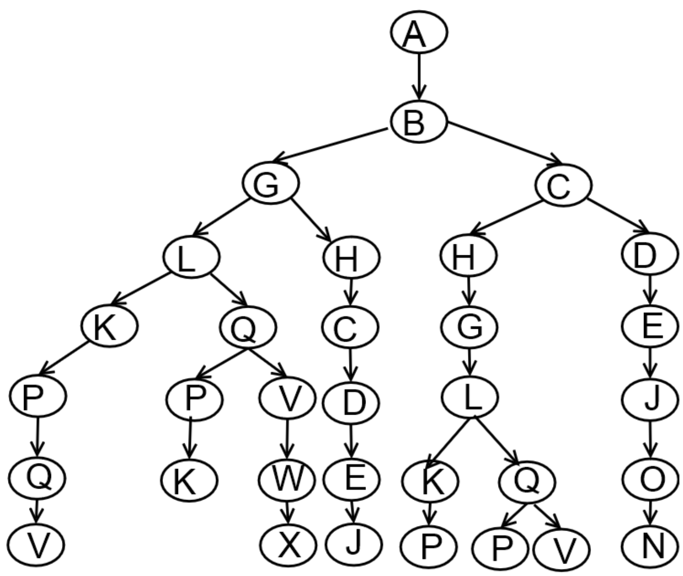
Sol: A complete list of expanded states for the Path Checking Depth First Search is depicted in the following table:

Iteration	State
1	A
2	AB
3	ABG
4	ABGL
5	ABGLK
6	ABGLKP

7	ABGLKPQ
8	ABGLKPQV
9	ABGLKPQVW
10	ABGLKPQVWX
11	ABGLKPQVWXS
12	ABGLKPQVWXST
13	ABGLKPQVWXSTO
14	ABGLKPQVWXSTON

The solution path is: A→B→G→L→K→P→Q→V→W→X→S→T→O→N

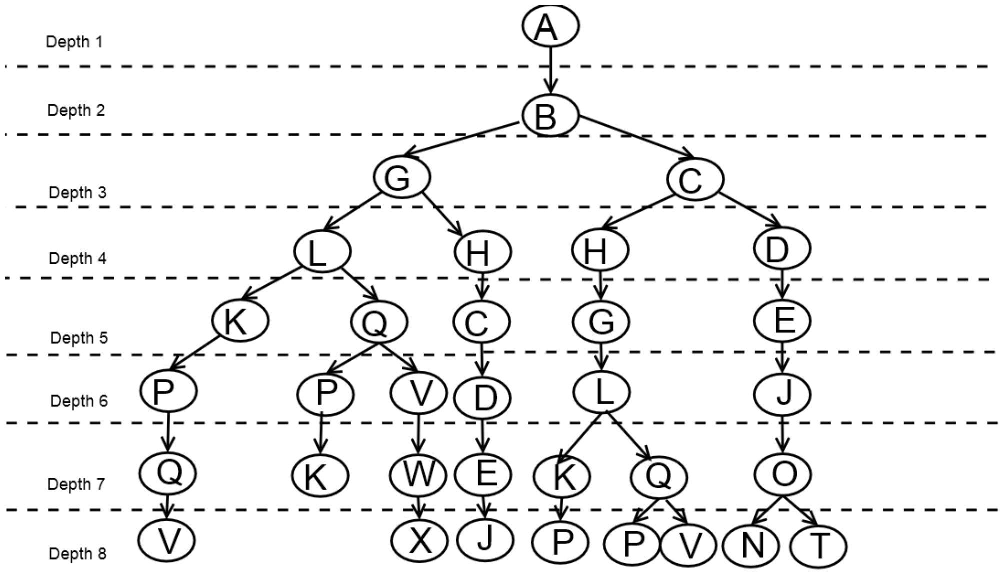
b) [10] Using **Breadth First Search** write down the list of states you expanded in the order they are expanded (until the goal node is reached). Use the same cycle checking as in the previous question.



Expanded states level	State
0	A
1	B
2	GC
3	LHHD
4	KQCGE
5	PPVDLJ
6	QKWEKQO
7	VXJPPVN

The solution path is: A→B→C→D→E→J→O→N.

c) [10] Using **Iterative Deepening Search**, draw the trees built at each depth until a solution is reached. Use the same cycle checking as in the previous questions.



List of expanded states	State
1	A
2	AB
3	ABGC
4	ABGLHCHD
5	ABGLKQHCCHGDE
6	ABGLKPQPVHCDCHGDE
7	ABGLKPQQPKVWHCDECHGLKQDEJC
8	ABGLKPQVQPLVWXHCDEJCHGLKPQPVDEJON

d) [10] Consider the heuristic function $h(n)$ which is the Manhattan distance between a given square and the goal square. Is $h(n)$ an admissible heuristic?
Sol:

The definition of an admissible heuristic function to be one that never over-estimates the true cost from any given state to the goal state.
The best possible situation we can place ourselves in is one where our current state has a clear shot to the goal; in this case, the heuristic would be equivalent to the true cost.
A worse situation is where the Little Red Riding Hood have to avoid the forest that is directly on her path to the goal, the heuristic under-estimates the true cost (the heuristic cost would be equivalent to charging straight through the forest).
So, the heuristic function $h(n)$ is admissible.

e) [10] Perform A* search using the heuristic function *h*(*n*). In the case of ties, expand states in alphabetical order. List each square in the order they are added to the OPEN list, and mark it with *f*(*n*) = *g*(*n*)+*h*(*n*) (show f, g and h separately). When expanded (including square N), label a state with a number indicating when it was expanded (square A should be marked 1). Highlight the solution path found (if any), or explain why no solution is found.

Sol: The process followed to reach the solution is shown in the following table.

The final solution path found is: ABCDGHLEJKON

Step	OPEN	*f*(*n*) = *g*(*n*)+*h*(*n*)	*g*(*n*)	*h*(*n*)	Expanded
1	A	5	0	5	
2					A
3	B	5	1	4	
4					B
5	G	5	2	3	
6	C	5	2	3	
7					C
8	H	5	3	2	
9	D	5	3	2	
10					D
11	E	7	4	3	
12					G
13	L	5	3	2	
14					H
15					L
16	K	7	4	3	
17	Q	7	4	3	
18					E
19	J	7	5	2	
20					J
21	O	7	6	1	
22					K
23	P	9	5	4	
24					0
25	N	7	7	0	
26	T	9	7	2	
27					N

Question 2: Programming Part: A* Search [50]

Heuristic function

For this problem, you are required to try different heuristic functions to see their effects on A* algorithm.

The first heuristic function is

$$H_1(s) = 8 - \max(n1; n2; n3)$$

where n1, n2, n3 are number of blocks in the central square. This formula is obtained by relaxing the constraints and allowing the blocks to switch remotely.

The second one is

$$h_2(s) = 0$$

As a result, the A* algorithm with this heuristic will degenerate into BFS.
The last one should be designed by yourself. You should write down your heuristic function h3 and verify its correctness in WrittenPart.pdf. We are happy to see if you can raise a better heuristic function, but since it is an open question, so we do not require your heuristic function to beat h1. As long as you can show the validity of your heuristics and incorporate it into the code sketch correctly, then you will get full scores for this part.

Sol:

$h_3(x) = \max(\frac{n1}{8}, \frac{n2}{8}, \frac{n3}{8})$ represent the max possibility of the element (i, where i = 1, 2, 3) in the counter square. With the more possibility it appears in the counter square, the less block we have to remove.