D. Grissom CS430

Homework 1 – Search Chapter 3

Due: See online for exact due date

Name: **Gregory Montilla**

Directions: You must complete this assignment alone, but can consult with others on high-level details.

1.) Fill in the space and time complexities (in terms of the following variables) and indicate if an algorithm is complete and/optimal for each of the algorithms listed in the chart.

b = branching factor
m = maximum depth of tree
g = depth of goal solution

	BFS	DFS	Iterative Deepening	UCS	A*
Time	b^g	b^m	b^g	$b^{C/e}$	g
Space	b^g	bm	bm	$b^{C/e}$	g
Optimal?	Yes	No	Yes	Yes	Yes
Complete?	Yes	No	Yes	Yes	Yes

2.) For question 1 above, explain the rational for the time and space complexities for both BFS and DFS in your own terms (essentially, I want to make sure you really understand these concepts):

a) BFS Time: It is $O(b^g)$ because at worst case scenario the solution is at most *g.*

b) BFS Space: It is also $O(b^g)$ because at worst case scenario the solution is at the end of the fringe. For every fringe there is b^g nodes.

c) DFS Time: It is $O(b^m)$ because at worst case scenario, the solution is at the bottom of the tree which is *m*.

d) DFS Space: It is $O(bm)$ because on every fringe there is b * m nodes.

For the remaining problems, use each search strategy to find the goal (node G; start from node S). You CANNOT just state the answer. For each problem below, a chart which represents what nodes/paths are on the FRINGE at each iteration/step of the search algorithm can be found. ***For each iteration in problems 3-6, indicate each path and its corresponding cost, as indicated in the graph.***

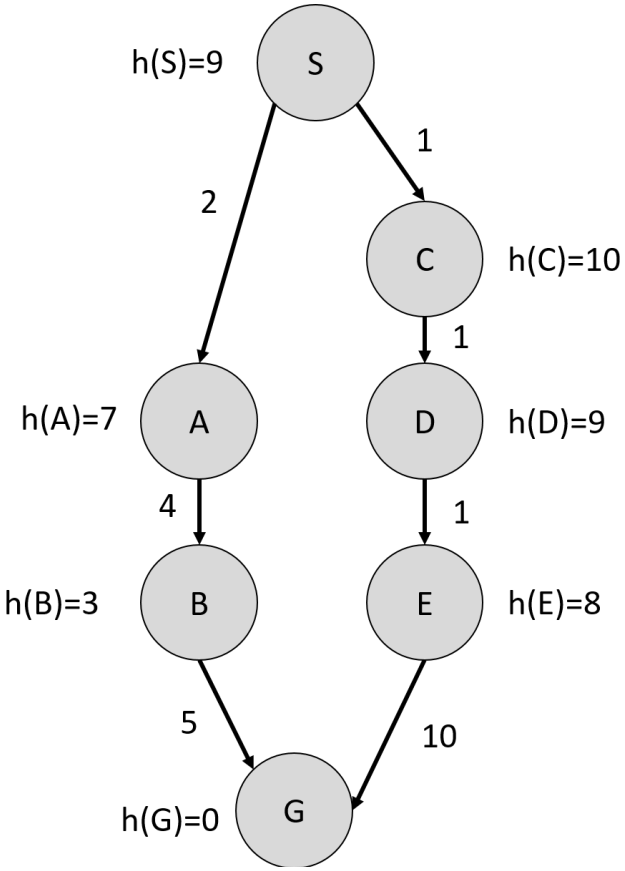
Once you have expanded and written all the nodes/paths on the fringe for an iteration, draw a box around the node/path that will be removed from the fringe at the next iteration. Any time you encounter a goal state, circle it (even if it remains on the fringe, circle it each iteration).

HINT1: You will know you are done when you have a node that is both circled and boxed.

HINT2: Don't try to read my mind on the number of rows or columns given…you may not need them all.

HINT3: I've filled out the first iteration for you. Since the path "S" is the only node on the fringe to begin with, it is chosen to be removed (hence the box) and will not appear in iteration 2.

If you want to represent the next node A, which comes from S, write it as "SA" in the path field below. If you want to represent the node B, which came through A and S, write it as "SAB" in the path field below, etc.



3.) **Breadth First Search (BFS)**. When expanding, expand from left to right (i.e., if a child has two nodes, add the one on the left to the fringe first). Although you will need to use a standard queue and set (to keep track of visited states), you do not need to draw them here for points/correctness, but you may want to draw them on scrap paper below to help.

	Nodes/Paths on Fringe									
Iteration #	Path	depth	Path	depth	Path	depth	Path	depth	Path	depth
1	S	0								
2	~~SA~~	~~1~~	SC	1						
3	~~SC~~	~~1~~	SAB	2						
4	~~SAB~~	~~2~~	SCD	2						
5	~~SCD~~	~~2~~	SABG	3						
6	~~SABG~~	~~3~~	SCDE	4						
7										
8										
9										
10										
11										
12										
13										
14										

How many nodes were expanded (i.e., how many iterations did it take? 6
What was the path computed by this search strategy? SABG
Was the computed path optimal? YES

4.) **Depth First Search (DFS)**. When expanding, expand from left to right (i.e., if a child has two nodes, add the one on the left to the fringe first). Although you will need to use a stack and set (to keep track of visited states), you do not need to draw them here for points/correctness, but you may want to draw them on scrap paper below to help.

	Nodes/Paths on Fringe									
Iteration #	Path	depth	Path	depth	Path	depth	Path	depth	Path	depth
1	S	0								
2	SA		SC	2						
3	SA	1	SCD	3						
4	SA	1	SCDE	4						
5	SA	1	SCDEG	5						
6										
7										
8										
9										
10										
11										
12										
13										
14										

How many nodes were expanded (i.e., how many iterations did it take? 4
What was the path computed by this search strategy? SCDEG
Was the computed path optimal? NO

5.) **Uniform Cost Search (UCS)**. When expanding, expand from left to right (i.e., if a child has two nodes, add the one on the left to the fringe first). Although you will need to use a priority queue and set (to keep track of visited states), you do not need to draw them here for points/correctness, but you may want to draw them on scrap paper below to help.

	Nodes/Paths on Fringe									
Iteration #	Path	g(n)	Path	g(n)	Path	g(n)	Path	g(n)	Path	g(n)
1	S	0								
2	SC	1	SA	2						
3	SCD	2	SA	2						
4	SA	2	SCDE	3						
5	SCDE	3	SAB	6						
6	SAB	6	SCDEG	13						
7	SABG	11	SCDEG	13						
8										
9										
10										
11										
12										
13										
14										

How many nodes were expanded (i.e., how many iterations did it take? 7
What was the path computed by this search strategy? SABG
Was the computed path optimal? YES

6.) **A* Search (A*)**. When expanding, expand from left to right (i.e., if a child has two nodes, add the one on the left to the fringe first). Although you will need to use a priority queue and set (to keep track of visited states), you do not need to draw them here for points/correctness, but you may want to draw them on scrap paper below to help.

	Nodes/Paths on Fringe									
Iteration #	Path	f(n)	Path	f(n)	Path	f(n)	Path	f(n)	Path	f(n)
1	S	9								
2	SA	9	SC	11						
3	SAB	7	SC	11						
4	SABG	8	SC	11						
5										
6										
7										
8										
9										
10										
11										
12										
13										
14										

How many nodes were expanded (i.e., how many iterations did it take? 4
What was the path computed by this search strategy? SABG
Was the computed path optimal? YES