AI-Assignment#3 (Heuristic Search)

SOLUTION

1) Consider the problem of finding a path in the grid shown below from a starting square S to goal square G. Possible moves from a square are to move up, left, right, and down exactly one square. No move may be made onto a dark square or off the edge of the grid. Mark the grid squares with the number(s) indicating when that square is expanded during each search from S to G. Assume you do not generate a node if that node's associated grid position has previously been generated. In the case of ties in evaluation function values, for siblings expand them in the precedence order up, left, right, down. In the case of ties between non-siblings, use FIFO order to expand first the node that has been in the NODES list the longest. Highlight the solution path found, if any. Use the heuristic function $h(n) = |x_n - x_g| + |y_n - y_g|$; where the grid square associated with node n is at coordinates (x_n, y_n) in the grid and the goal node G is at coordinates (x_g, y_g).

a. Greedy Best-First Search

F(n)

3	2	1	2	3	4	5	6
2	1	G	1	2	3	4	5
3	2	1	2	3	4	5	6
4	3						7
5		3	4	5	6	7	8
6	5		5	6	S		9
7	6	5	6	7	8	9	10
8	7	6	7	8	9	10	11

Search Steps & Solution Path

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

b. A* search

F(n)

3+12	2+13	1+12	2+11	3+10	4+9	5+8	6+7
2+11	1+12	0+11 G	1+10	2+9	3+8	4+7	5+6
3+10	2+11	1+10	2+9	3+8	4+7	5+6	6+5
4+9	3+10						7+4
5+8		3+4	4+3	5+2	6+1	7+2	8+3
6+7	5+6		5+2	6+1	S		9+4
7+6	6+5	5+4	6+3	7+2	8+1	9+2	10+3
8+7	7+6	6+5	7+4	8+3	9+2	10+3	11+4

Search Steps & Solution Path

		32	30	28	26	24	22
		31	29	27	25	23	20
							17
		6	5	3	1	8	14
	21		4	2	S		
	18	11	10	9	7	12	
		19	16	15	13		

c. Hill-Climbing Search

F(n)

3	2	1	2	3	4	5	6
2	1	G	1	2	3	4	5
3	2	1	2	3	4	5	6
4	3						7
5		3	4	5	6	7	8
6	5		5	6	S		9
7	6	5	6	7	8	9	10
8	7	6	7	8	9	10	11

Search Steps & Solution Path

14	15	16					
13							
12							
11		4	3	2	1		
10	9		5		S		
	8	7	6				

2) Assuming that successor states are generated in alphabetical order (in stack-based algorithms, placed on the Open List <u>in alphabetical order</u> either at the start or at the end), and ties (for priority-queue based algorithms) broken <u>in alphabetical order</u>,

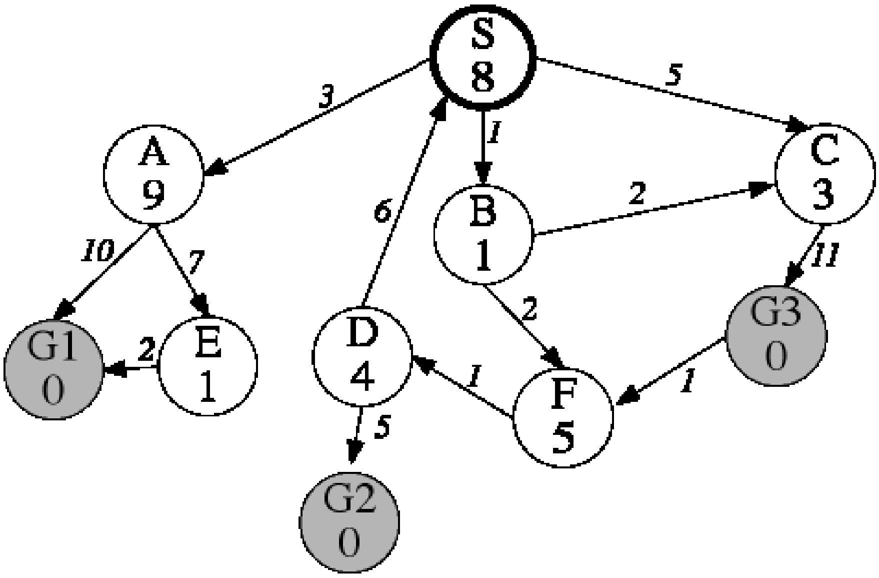
(a). ***<u>in what order are the nodes in this graph expanded by each of the following search algorithms</u>*** (tree search)? **Do not remove repeated states.**

Also for each,

(b). ***<u>which goal state is the founded goal state?</u>***

(c). ***<u>what is the cost of the path found?</u>***

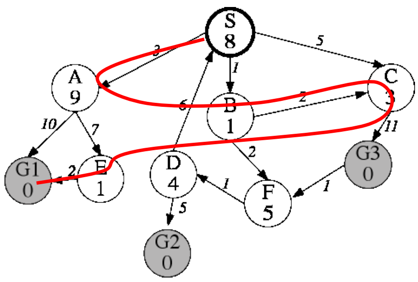
(d). ***<u>Is the founded path optimal?</u>***



a. Breadth-First Search

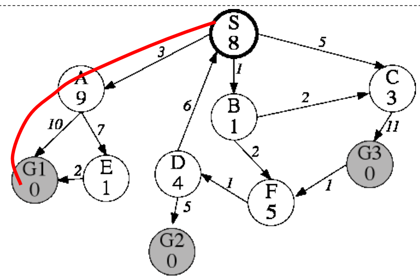
Expanded Nodes: ***SABCEG1***
Founded Goal State: ***G1***
Founded Cost Path: ***SAG1 (13)***
Is It An Optimal Path? **No**

The breadth-first search is only optimal when the shallowest solution is optimal.



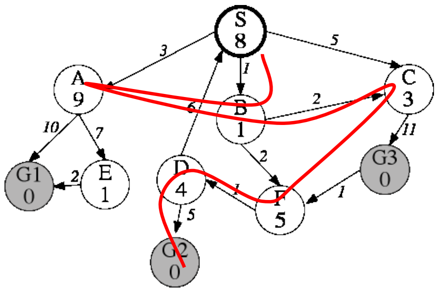
b. Depth First Search

Expanded Nodes: ***SAEG1***
Founded Goal State: ***G1***
Founded Cost Path: ***SAG1 (13)***
Is It An Optimal Path? **No**



c. Uniform Cost Search

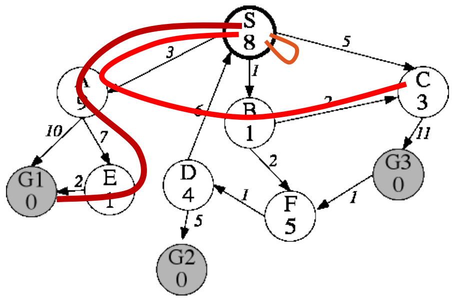
Expanded Nodes: ***SBACFDG2***
Founded Goal State: ***G2***
Founded Cost Path: ***SBFDG2 (9)***
Is It An Optimal Path? **Yes**



d. Iterative Deepening

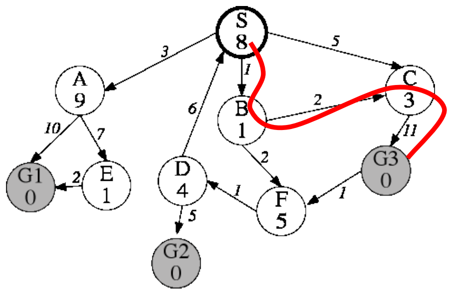
Expanded Nodes: **0:** ***S*** **1:** ***SABC*** **2:** ***SAEG1***
Founded Goal State: ***G1***
Founded Cost Path: ***SAG1 (13)***
Is It An Optimal Path? **No**

ID is optimal only when breadth-first search is optimal.



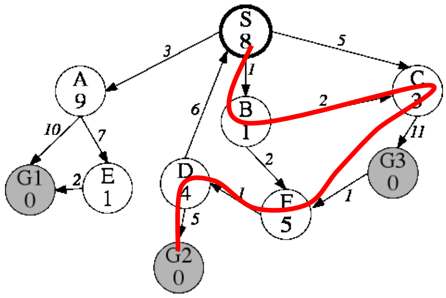
e. Greedy Best-First Search

Expanded Nodes: ***SBCG3***
Founded Goal State: ***G3***
Founded Cost Path: ***SBCG3 (14)***
Is It An Optimal Path? **No**



*f. A**

Expanded Nodes: ***SBCFDG2***
Founded Goal State: ***G2***
Founded Cost Path: ***SBFDG2 (9)***
Is It An Optimal Path? **Yes**



*g. IDA**

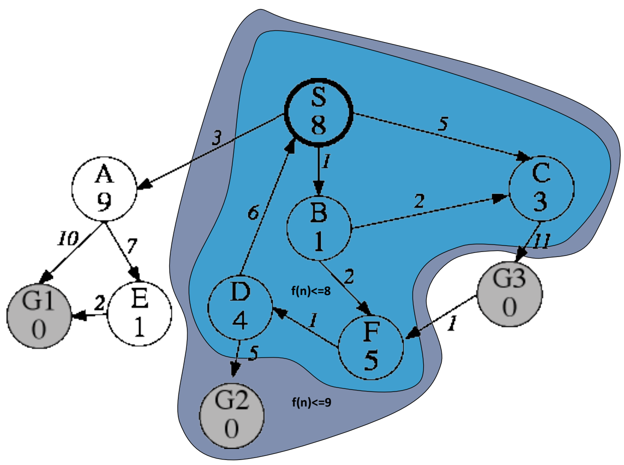
Expanded Nodes:

Cutoff=8 → SBCFD

Cutoff=9 → SBCFDG2

Founded Goal State: ***G2***
Founded Cost Path: ***SBFDG2 (9)***
Is It An Optimal Path? **Yes**

IDA is the depth-first search within f-limit.*



3) If h1(s) and h2(s) are both admissible heuristic functions, which of the following are also admissible:

h3(s) = h1(s) + h2(s)

h3(s) = | h1(s) - h2(s) |

h3(s) = max(h1(s), h2(s))

h3(s) = min(h1(s), h2(s))

Be sure to explain your answers. For those cases where h3(s) is not admissible, show a counter example. Which of the above four combinations do you feel is the best one? Why?

Admissible means it must give an answer that is less than or equal to the real answer.

h3(s) = h1(s) + h2(s)

Not Admissible, Suppose the true cost is 10. h1 and h2 are perfect classifiers, so both predict 10. Therefore h3
= 10 + 10 which is 20 which is more than the actual cost, so inadmissible.

h3(s) = | h1(s) - h2(s) |

Admissible, if h1(s) and h2(s) are admissible, the absolute difference between them is less than or equal h1(s) and h2(s).

h3(s) = max(h1(s), h2(s))

Admissible, if h1(s) and h2(s) are admissible, choosing one of them is still admissible.

h3(s) = min(h1(s), h2(s))

Admissible, if h1(s) and h2(s) are admissible, choosing one of them is still admissible.

The best one is ***h3(s) = max(h1(s), h2(s))*** since it always return the maximum value, you know that the higher values of admissible heuristic functions, the better.

4) Prove each of the following statements whether it is <u>true</u> or <u>false</u>:

a. Breadth-first search is a special case of uniform-cost search.

When all step are equal, $g(n) \propto depth(n)$, so uniform-cost search reproduces breadth-first search. Therefore, it is <u>true</u>.

b. Breadth-first search, depth-first search, and uniform-cost search are special cases of best-first search.

Breadth-first search is best-first search with $f(n) = depth(n)$; depth-first search is best first search with $f(n) = -depth(n)$; uniform-cost search is best-first search with $f(n) = g(n)$. Therefore, it is <u>true</u>.

c. Uniform-cost search is a special case of A* search.

Uniform-cost search is A* search with $h(n) = 0$. Therefore, it is <u>true</u>.