CSCE 625 Homework 1

Search

Problem 1:

Visited Node	Node List	Visited List
	[1]	
1	[2, 3]	[1]
2	[4, 5, 3]	[1, 2]
4	[8, 9, 5, 3]	[1, 2, 4]
8	[16, 17, 9, 5, 3]	[1, 2, 4, 8]
16	[17, 9, 5, 3]	[1, 2, 4, 8, 16]
17	[9, 5, 3]	[1, 2, 4, 8, 16, 17]
9	[18, 19, 5, 3]	[1, 2, 4, 8, 16, 17, 9]
18	[19, 5, 3]	[1, 2, 4, 8, 16, 17, 9, 18]

(1) The nodes that remain in the node list are [19, 5, 3]
(2) The nodes that are visited until 18 is reached in order are [1, 2, 4, 8, 16, 17, 9, 18]

Problem 2:

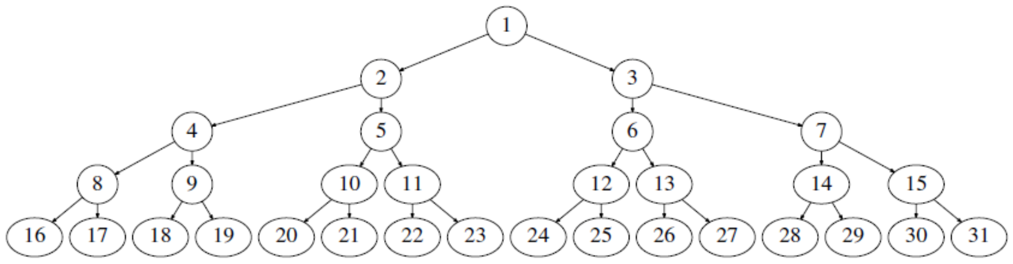
Visited node	Node List	Visited List
	[1]	
1	[2, 3]	[1]
2	[3, 4, 5]	[1, 2]
3	[4, 5, 6, 7]	[1, 2, 3]
4	[5, 6, 7, 8, 9]	[1, 2, 3, 4]
5	[6, 7, 8, 9, 10, 11]	[1, 2, 3, 4, 5]
6	[7, 8, 9, 10, 11, 12, 13]	[1, 2, 3, 4, 5, 6]

(1) The nodes that remain in the node list are [7, 8, 9, 10, 11, 12, 13]
(2) The nodes that are visited until 6 is reached in order are [1, 2, 3, 4, 5, 6]

Problem 3:

In BFS, we store the nodes at d+1 depths in the node list when searching for a number that is at d depth. Even if we have b^d nodes at d depth, we need to explore d+1 depth too, which has b^(d+1) nodes. So, the space complexity is O(b^(d+1)) instead of O(b^d).

Problem 4:
Yes, the Depth Limited Search can become incomplete in the case of a finite tree below.



In the above figure, if the goal that we need to search is 19 which is in d = 4 and we have a limit at L = 2. In that case, we will be iterating over the following nodes only

L	Nodes visited
0	1
1	[1, 2, 3]
2	[1, 2, 4, 5, 3, 6, 7]

In this case, we would never encounter 19 (which is our goal). Hence DLS can be limited if the depth (d) at which the goal lies is more than the limit set (L).

Problem 5:

Depth	Nodes Visited
d = 0	[1]
d = 1	[1, 2, 3]
d = 2	[1, 2, 4, 5, 3, 6, 7]
d = 3	[1, 2, 4, 8, 9, 5, 10, 11, 3, 6, 12]

On adding all the nodes that are visited, the total number of nodes that are visited until we reach 12, without including the visit to 12 in the count, is **21**.

Problem 6:

Node	Heuristic Value	Actual Value
G	0	0
F	45	h*(F) = FG = 60
E	28	h*(E) = EG = 100
B	70	h*(B) = BG = 150
C	72	h*(C) = CF + h*(F) = 30 + 60 = 90
D	57	h*(D) = DC + h*(C) = 50 + 90 = 140
A	80	h*(A) = AC + h*(C) = 20 + 90 = 110

Hence, in all the cases, h(n) ≤ h*(n) for all n.

Problem 7:

In Greedy-best first search, we always select the node that seems to be closest to the goal according to the heuristic function without taking into account the entire path cost.

1. Initialization:
    a. Node list content: [A]
    b. Node visit order: [A]

2. Iteration 1:
    a. Expand node A
    b. Node list content: [D, B, C] (h(n) values of B, C and D are 70, 72 and 57 respectively)
    c. Node visit order: [A]

3. Iteration 2:
    a. Expand node D
    b. Node list content: [E, B, C] (because h(n) values of B, C and E are 70, 72 and 28 respectively)
    c. Node visit order: [A, D]

4. Iteration 3:
    a. Expand node E
    b. Node list content: [G, F, B, C] (because h(n) values of F and G are 45 and 0 respectively)
    c. Node visit order: [A, D, E]

5. Termination: Since node G is reached in Iteration 3, we can terminate the search
    a. Node list content: [F, B, C]
    b. Node visit order: [A, D, E, G]

Node visit order: A -> D -> E -> G
The solution path: ADEG
Cost of the final solution: AD + DE + EG = 50 + 70 + 100 = 220

Problem 8:

f(n) = h(n) + h*(n) where h(n) = Heuristic Value and h*(n) = Actual Value

For A* Search:
1. Iteration 1:
    a. Expand node A.
    b. Generate successors: B, C, D.
    c. Calculate f(n) = g(n) + h(n) for each successor.
    d. f(B) = g(B) + h(B) = 10 + 70 = 80.
    e. f(C) = g(C) + h(C) = 20 + 72 = 92.
    f. f(D) = g(D) + h(D) = 50 + 57 = 107.
    g. Node to expand: B with f(B) = 80.

2. Iteration 2:
    a. Expand node B.
    b. Generate successors: E, G.
    c. Calculate f(n) = g(n) + h(n) for each successor.
    d. f(E) = g(E) + h(E) = 100 + 28 = 128.
    e. f(G) = g(G) + h(G) = 150 + 0 = 150.
    f. Node to expand: E with f(E) = 128.

3. Iteration 3:
    a. Expand node E.
    b. Generate successors: F, G.
    c. Calculate f(n) = g(n) + h(n) for each successor.
    d. f(F) = g(F) + h(F) = 80 + 60 = 140.
    e. f(G) = g(G) + h(G) = 100 + 0 = 100.
    f. Node to expand: G with f(G) = 100.

So, the A* search yields a solution path A -> B -> E -> G with a cost of 10 + 100 + 100 = 210.

Comparing the two paths:

Greedy Best-First Search: A -> D -> E -> G with a cost of 50 + 70 + 100 = 220
A* Search: A -> B -> E -> G with a cost of 10 + 100 + 100 = 210.

In this scenario, A* Search gives a lower cost solution compared to Greedy Best-First Search.