Chonglin Zhang
DUE 9-17-2023
CSCE625

Problem 1 (Written; 5 pts): Consider depth first search. When the goal is 20 and the node is visited (taken out of node list), (1) what are the nodes that remain in the node list? (list them in the correct order). Also (2) which nodes have been visited until then, in what order?

1) Remain node in the node list: 21,11,22,23,3,6,12,24,25,13,26,27,7,14,28,29,13,30,31
2) Visited Node: 1,2,4,8,16,17,9,18,19,5,10,20

Problem 2 (Written; 5 pts): Consider breadth first search. When the goal is 12 and the node is visited (taken out of node list), (1) what are the nodes that remain in the node list? (list them in the correct order). Also (2) which nodes have been visited until then, in what order?

1) Remain node in the node list:
   13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31
2) Visited Node: 1,2,3,4,5,6,7,8,9,10,11,12

Problem 3 (Written; 5 pts): Why is the space complexity of BFS O(b^(d+1)), not O(b^d), where b is the branching factor and d is the goal depth?

In BFS, we need to track all the nodes at each step until reach the step d. This will get the time complexity O(b^d). However, the space complexity of BFS O(b^(d+1)). Here is reason: by the time you reach the end of this goal depth d, and then all of these nodes at depth d+1 are stored in the node list.

Problem 4 (Written; 5 pts): Can depth limited search become incomplete in the case of the finite search tree above? If so, give an example (use Figure 1). If not, explain why not (use Figure 1).

It depends on the depth limit l and depth of solution d. If l bigger and equal to d, the depth limit search will become complete because it finds the goal in the limit depth. Using figure 1 as an example: Goal is 4, depth limit l = 3 and depth of solution d is 2. It will always find the goal when l >= d. Otherwise, DLS becomes incomplete when l < d because there are no way to get to the depth of solution d.

Problem 5 (Written; 5 pts): Consider iterative deepening search. When the goal is 9 , how many nodes are visited before reaching that node? Hint: Include repeated visits in the count. Include the visit to the goal 9 in the count.

Depth limit = 0: 1
Depth limit = 1: 1,2,3
Depth limit = 2: 1,2,4,5,3,6,7
Depth limit = 3: 1,2,4,8,9,5,10,11,3,6,12,13,7,14,15
Find the goal 9 when the depth limit = 3. The total visited nodes are 5 when the goal is 9.

Problem 6 (Written; 5 pts): For the problem shown in Fig. 2, show that the heuristic is admissible ($h(n) \leq h*(n)$ for all n). Actual cost from node to node are shown as edge labels. The heuristic function value for each node is shown in a spearate table to the right. Note: You have to compute $h*(n)$ for each n and compare to the h(n) table.

h*(A) = 60 (A->B->F->C->E->G) > h(A) = 55
h*(B) = 40 (B->F->C->E->G) > h(B) = 35
h*(C) = 20 (C->E->G) > h(C) = 17
h*(D) = 70 (D->F->C->E->G) > h(D) = 66
h*(E) = 10 (E->G) > h(E) = 9
h*(F) = 30 (F->C->E->G) > h(F) = 27
h*(G) = 0 (already goal)

As the h(n) and h*(n) show above, heuristic is admissible because all h(n) are smaller and equal to h*(n)

Problem 7 (Written; 15 pts): Manually conduct greedy best-first search on the graph below (Fig. 2), with initial node A and goal node G . Show: 1. Node list content at each iteration. 2. Node visit order 3. Solution path 4. Cost of the final solution.

Initial node A -> goal node G

Visit step 1: Node list
    (A): h = 55, path = (A)
-> visit order 1: (A): h = 55, path = (A)

Visit step 2: Node list
    (E): h = 9: path = (A E)
    (B): h = 35: path = (A B)
    (D): h = 66: path = (A D)
-> visit order 2: (E): h = 9, path = (A E)

Visit step 3: Node list
    (G): h = 0: path = (A E G)
    (B): h = 35: path = (A B)
    (D): h = 66: path = (A D)

-> visit order 3: (G): h = 0, path = (A E G)

Node visit order: A-> E-> G
Solution path: A->E->G
Cost of the final solution: 60 + 10 = 70

Problem 8 (Written; 15 pts): (1) Repeat the problem right above with A∗ search. (2) In addition, show the f(n) value for all nodes expanded (you need this to sort them in the node list). (3)Which one gives a lower cost solution: Greedy best-first or A∗?

Node list:
    (A): f = 0 + 55 = 55, path = (A)
-> visit: (A): f = 55, path = (A)

Node list:
    (B): f = 20 + 35 = 55, path = (A B)
    (E): f = 60 + 9 = 69, path = (A E)
    (D): f = 10 + 66 = 76, path = (A D)
-> visit: (B): f = 55, path = (A B)

Node list:
    (B): f = 20 + 35 = 55, path = (A B)
    (E): f = 60 + 9 = 69, path = (A E)
    (D): f = 10 + 66 = 76, path = (A D)
-> visit: (B): f = 55, path = (A B)

Node list:
    (F): f = 20 + 10 + 27 = 57, path = (A B F)
    (C): f = 20 + 30 + 17 = 67, path = (A B C)
    (E): f = 60 + 9 = 69, path = (A E)
    (D): f = 10 + 66 = 76, path = (A D)
    (E): f = 20 + 50 + 9 = 79, path = (A B E)
-> visit: (F): f = 55, path = (A B F)

Node list:
    (C): f = 20 + 10 + 10 + 17= 57, path = (A B F C)
    (C): f = 20 + 30 + 17 = 67, path = (A B C)
    (E): f = 60 + 9 = 69, path = (A E)
    (D): f = 10 + 66 = 76, path = (A D)
    (G): f = 20 + 10 + 45 + 0 = 75, path = (A B F G)
    (E): f = 20 + 50 + 9 = 79, path = (A B E)
-> visit: (C): f = 57, path = (A B F C)

Node list:

(E): f = 20 + 10 + 10 + 10 + 9 = 59, path = (A B F C E)
(C): f = 20 + 30 + 17 = 67, path = (A B C)
(E): f = 60 + 9 = 69, path = (A E)
(D): f = 10 + 66 = 76, path = (A D)
(G): f = 20 + 10 + 45 + 0 = 75, path = (A B F G)
(E): f = 20 + 50 + 9 = 79, path = (A B E)
(G): f = 20 + 10 + 10 + 40 = 80, path = (A B F C G)

-> visit: (E): f = 56, path = (A B F C E)

Node list:

(E): f = 20 + 10 + 10 + 10 + 10 = 60, path = (A B F C E G)
(C): f = 20 + 30 + 17 = 67, path = (A B C)
(E): f = 60 + 9 = 69, path = (A E)
(D): f = 10 + 66 = 76, path = (A D)
(G): f = 20 + 10 + 45 + 0 = 75, path = (A B F G)
(E): f = 20 + 50 + 9 = 79, path = (A B E)
(G): f = 20 + 10 + 10 + 40 = 80, path = (A B F C G)

-> visit: (G): f = 60, path = (A B F C E G)

Node visit order: (A B F C E G)
Solution path: (A B F C E G)
Cost of the final solution: 60

The cost solution of Greedy best-first is 70, and the solution of A* is 60. Therefore, the lowest cost solution is A*