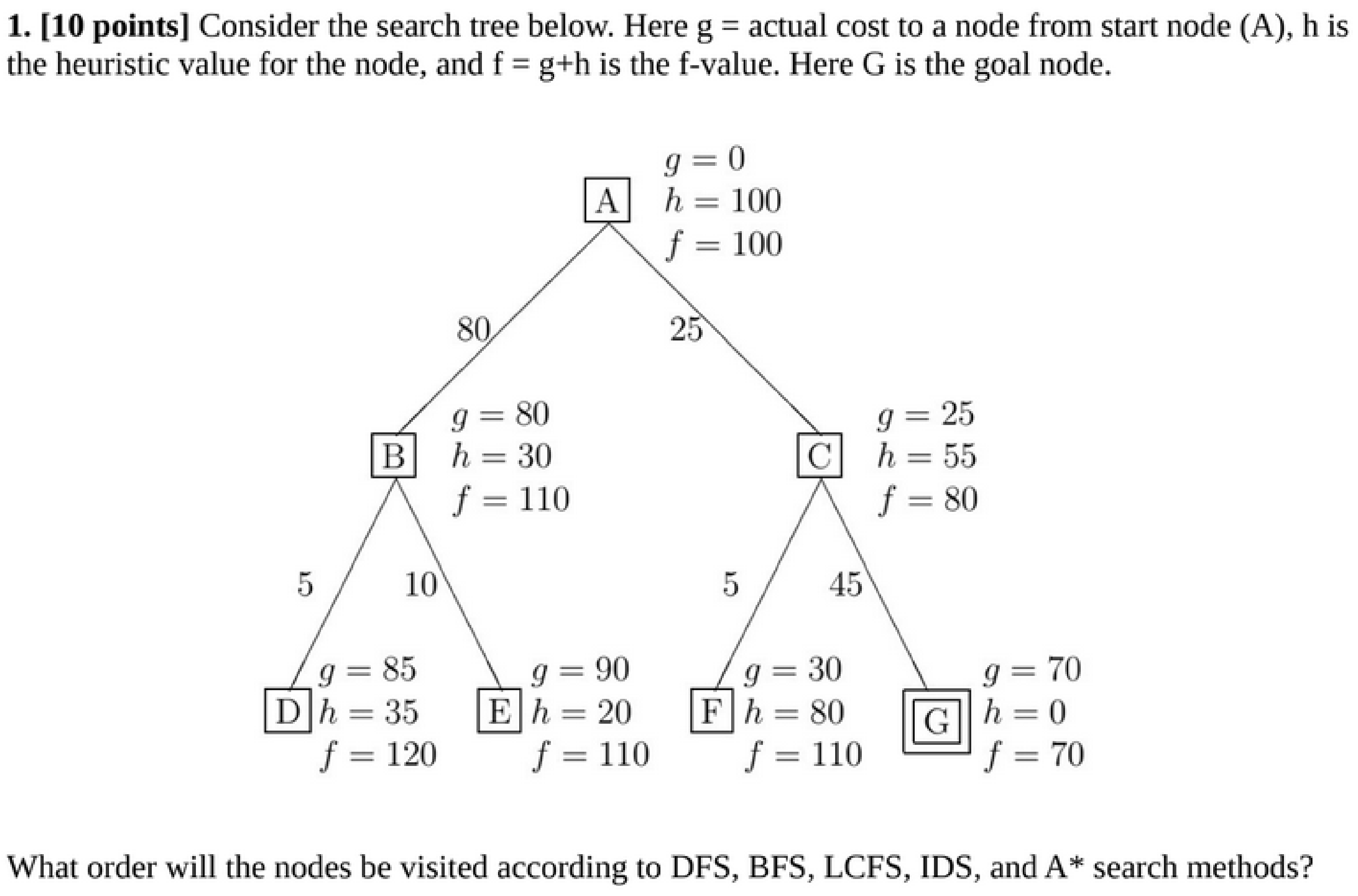
CSC425 Assignment 2

Mir Shafayat Ahmed 1910456

Question 1



1. [10 points] Consider the search tree below. Here g = actual cost to a node from start node (A), h is the heuristic value for the node, and f = g+h is the f-value. Here G is the goal node.

What order will the nodes be visited according to DFS, BFS, LCFS, IDS, and A* search methods?

Figure 1: Tree

DFS

A –> B –> D –> E –> C –> F –> G

BFS

A –> B –> C –> D –> E –> F –> G

LCFS

A –> C –> F –> G

IDS

A –> [A –> B –> C] –> [A –> B –> D –> E –> C –> F –> G]

A*

A –> C –> G

Question 2

2. [20 points] Suppose two robots are positioned in an 8 x 8 maze (see figure 1 below) and they want to meet in the same cell. The black cells in the figure are blocked; one cannot move into that cell. In every step, each robots can move one cell up, down, left, right, or not move at all. Suppose each move costs 1. Also the robots do not alternate turns, they move simultaneously. For example, if the robots are initially at the locations as shown in figure 1 below and chooses to move up and left respectively, at step 1, they will end up at the locations shown in the figure 2.

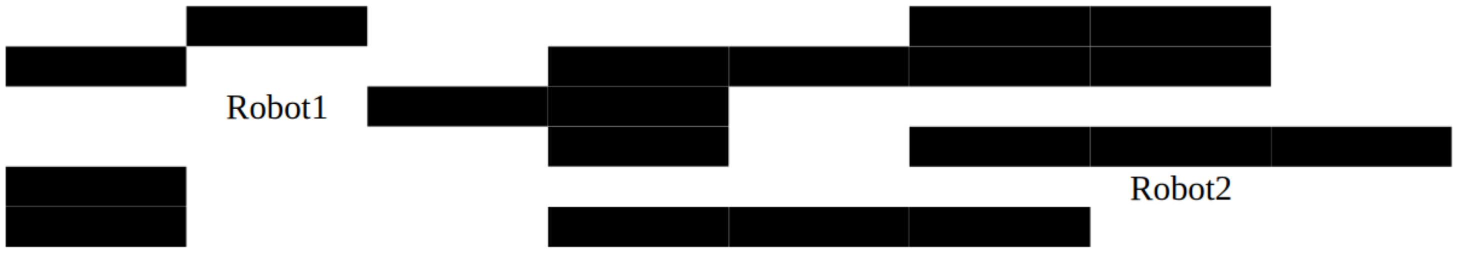


figure 1: Initial position of Robot1 and Robot2

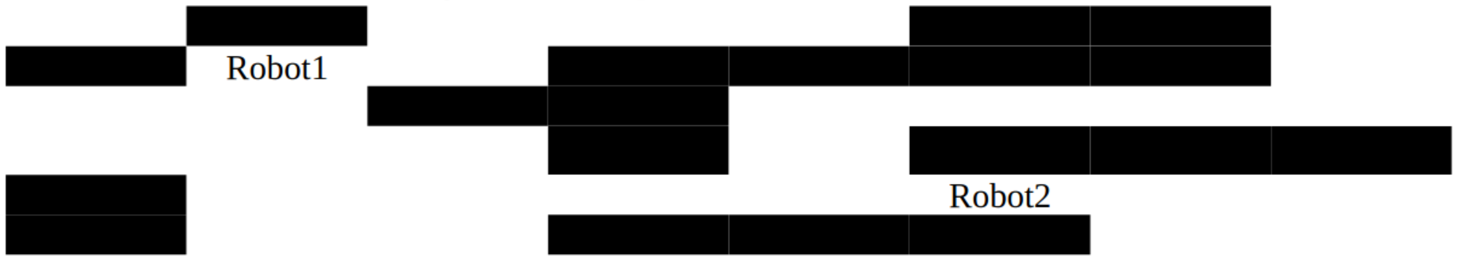


figure 2: Position of the robots after step 1

a. [6 points] How would you formulate it as a single agent search problem -
i. How would you represent the states?
ii. What are the possible moves/actions from a state? (Fix an order of the moves to be explored.)
iii. How would you represent the start state in the above example and its children/successor states based of the order of the moves?
iv. How many legal states are there for this particular instance of the problem?
v. How would you test whether the state you have reached is a goal state?
vi. What would be an admissible heuristic for this problem? Justify why it is admissible.

b. [4 points] Which of the following problems will be able to find the optimal solution to the problem – BFS, DFS, LCFS, and A*? Why or why not.

c. [10 points] Show the execution of the above algorithms up to 5 steps: in each step show which path is selected from the frontier, and the contents of the frontier at the end of the step.

a

i

Each state can be represented as the posititon of each robots combined.
e.g, the initial state in figure 1 can be written as :

`{[3][1],[5][6]}`
State of figure 2 can be :
`{[2][1],[5][5]}`
Template : {[row(1)][col(1)], [row(2)][col(2)]}

ii

The next state down from any state would be the new posititons of the robots if they take any of the actions.
Since the number of actions that can be taken is 5 and the robots are moving simutaneously, there will be 5*5 = 25 states next.
The order of the choice of steps can be LL, LU, LR, LD, LN, UL, UU, . . . , NN

iii

Start state = `{[3][1],[5][6]}`
Children states:
LL `{[3][0],[5][5]}` | LU `{[3][0],[4][5]}` | LR `{[3][0],[5][7]}` | . . . | NN `{[3][1],[5][6]}`

iv

64-19 = 45 Legal states

v

If both the row and column for the robots are the same. e.g `{[3][1],[3][1]}`

vi

For the best case of the two robots, there cannot be any illegal states between them. Therefore the manhattan distance can be an admissible heuristic since in the best case `h(n) = c(n)` but if there are any illegal states in between, `h(n) < c(n)`. So it can be an admissible heuristic.

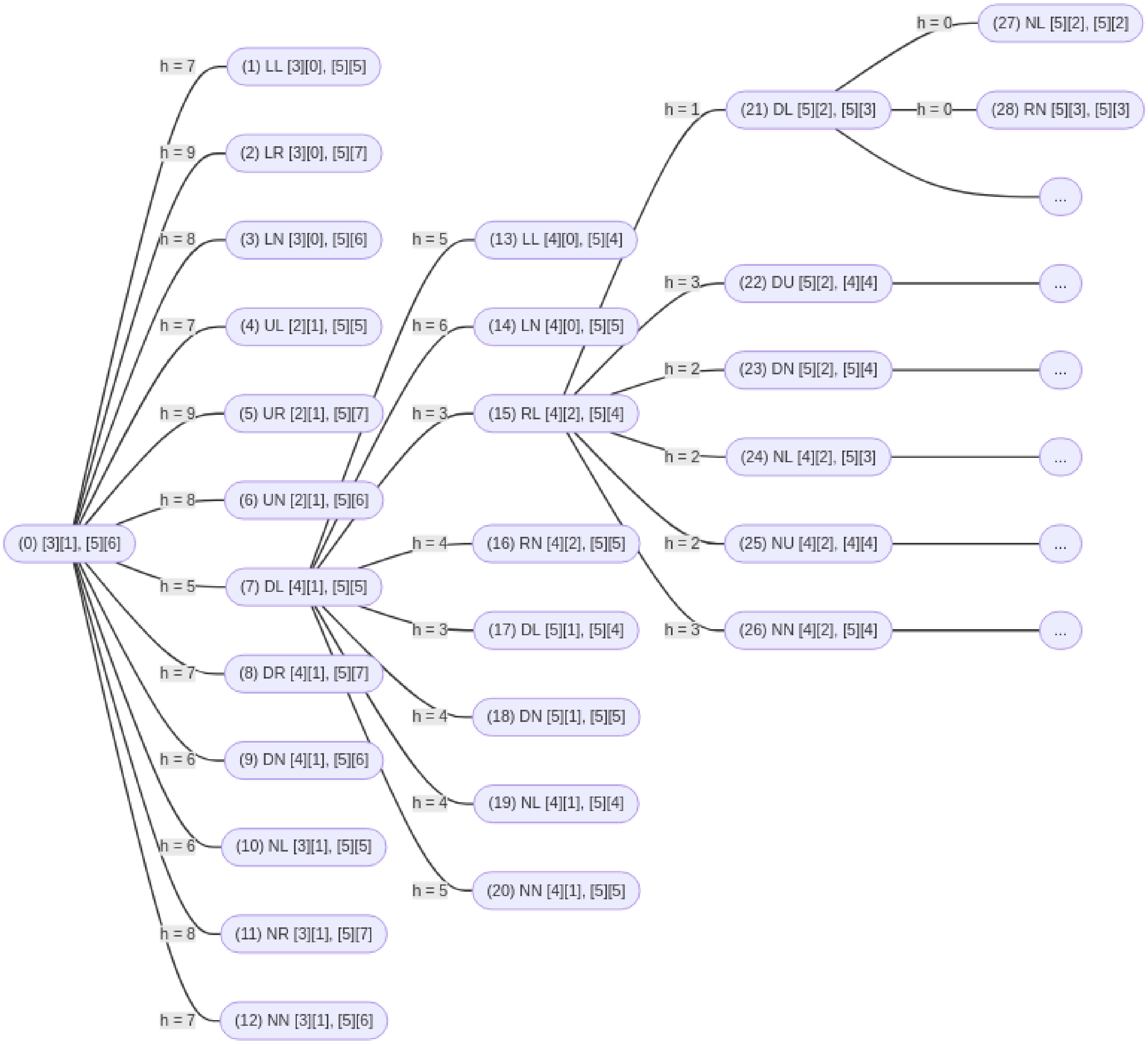
b

Pure BFS and DFS can both find the solution but there is a high likelyhood that they will arrive at the solution much later.

Since LCFS does not depend on heuristics, and the cost for all steps are 1, LCFS behaves like BFS in this case but goes one step deeper since goal test is done only after a node is selected for expansion.

A* will find the optimal path since the (admissible) heuristic of the Manhattan Distance between the two robots can guide the path finding into choosing `f(n)` whose values are lowest first.

c



(0) [3][1], [5][6]
h = 7
(1) LL [3][0], [5][5]
h = 9
(2) LR [3][0], [5][7]
h = 8
(3) LN [3][0], [5][6]
h = 7
(4) UL [2][1], [5][5]
h = 9
(5) UR [2][1], [5][7]
h = 8
(6) UN [2][1], [5][6]
h = 5
(7) DL [4][1], [5][5]
h = 7
(8) DR [4][1], [5][7]
h = 6
(9) DN [4][1], [5][6]
h = 6
(10) NL [3][1], [5][5]
h = 8
(11) NR [3][1], [5][7]
h = 7
(12) NN [3][1], [5][6]
h = 5
(13) LL [4][0], [5][4]
h = 6
(14) LN [4][0], [5][5]
h = 3
(15) RL [4][2], [5][4]
h = 4
(16) RN [4][2], [5][5]
h = 3
(17) DL [5][1], [5][4]
h = 4
(18) DN [5][1], [5][5]
h = 4
(19) NL [4][1], [5][4]
h = 5
(20) NN [4][1], [5][5]
h = 1
(21) DL [5][2], [5][3]
h = 3
(22) DU [5][2], [4][4]
h = 2
(23) DN [5][2], [5][4]
h = 2
(24) NL [4][2], [5][3]
h = 2
(25) NU [4][2], [4][4]
h = 3
(26) NN [4][2], [5][4]
h = 0
(27) NL [5][2], [5][2]
h = 0
(28) RN [5][3], [5][3]
...

BFS

Step 1:

```
0 --> 1
0 --> 2
.
0 --> 7
.
0 --> 12
```

Step 2:

```
0 --> 1 --> child1(1)
0 --> 1 --> child2(1)
.
0 --> 7 --> 13
0 --> 7 --> 14
0 --> 7 --> 15
.
0 --> 12 --> childn(12)
```

Step 3:

```
0 --> 1 --> child1(1) --> child1(child1(1))
0 --> 1 --> child1(1) --> childn(child1(1))
.
0 --> 7 --> 15 --> 21
.
0 --> 12 --> childn(12) --> childn(childn(12))
```

DFS

Step 1:

```
0 --> 1 --> child1(1) --> ... --> parent(leaf1) -->leaf1
```

Step 2:

```
0 --> 1 --> child1(1) --> ... --> parent(leaf1) -->leaf2
```

Step 3:

```
0 --> 1 --> child1(1) --> ... --> parent(leaf1) -->leaf3
```

LCFS

Same as BFS

A*

Step 1:

Frontier:
0, f = 0 + 7 = 7

Step 2:

Frontier:
`0 -->` 7, f = 1 + 5 = 6
.
`0 -->` 1, f = 1 + 7 = 8
.
`0 -->` 12, f = 1 + 7 = 8 .

`0 --> 5`, f = 1 + 9 = 10

Step 3:

Frontier:
`0 --> 7 --> 15`, f = 2 + 3 = 4
`0 --> 7 --> 17`, f = 2 + 3 = 4
.
`0 --> 7 --> 14`, f = 2 + 6 = 8
`0 --> 1`, f = 1 + 7 = 8
.
`0 --> 12`, f = 1 + 7 = 8 .
`0 --> 5`, f = 1 + 9 = 10

Step 4:

Frontier:
`0 --> 7 --> 15 --> 21`, f = 3 + 1 = 4
.
`0 --> 7 --> 15 --> 26`, f = 3 + 3 = 6
`0 --> 7 --> 17`, f = 2 + 3 = 4
.
`0 --> 7 --> 14`, f = 2 + 6 = 8
`0 --> 1`, f = 1 + 7 = 8
.
`0 --> 12`, f = 1 + 7 = 8 .
`0 --> 5`, f = 1 + 9 = 10

Step 5:

Frontier:
`0 --> 7 --> 15 --> 21 --> [27]`, f = 4 + 0 = 4
`0 --> 7 --> 15 --> 21 --> [28]`, f = 4 + 0 = 4
.
`0 --> 7 --> 15 --> 26`, f = 3 + 3 = 6
`0 --> 7 --> 17`, f = 2 + 3 = 4
.
`0 --> 7 --> 14`, f = 2 + 6 = 8
`0 --> 1`, f = 1 + 7 = 8
.
`0 --> 12`, f = 1 + 7 = 8 .
`0 --> 5`, f = 1 + 9 = 10

Goal Node Found, No other path with `f(n) < 4`