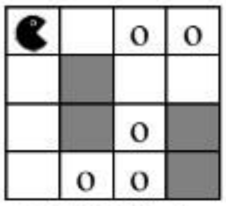
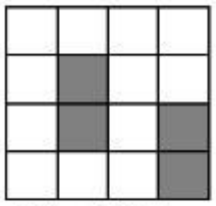
QUESTION 1:

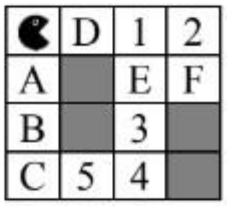
Given below the configurations of simplified Pacman game without ghosts. The objective of the game is to eat all of the dots placed in the maze. The Pacman can move in four directions (up, down, left, and right). Whenever Pacman enters in a cell with the dot it immediately consumes it. The Grey cells in the maze are blocked.


Initial State

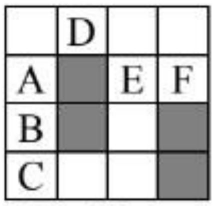

Goal State

Part a.

Execute the state space search algorithms on the following maze where, numbers represent food cells and alphabets represents empty cells. You can build the tree with current position of Pacman instead of drawing complete states. You should create the nodes from left to right in alphabetical order.

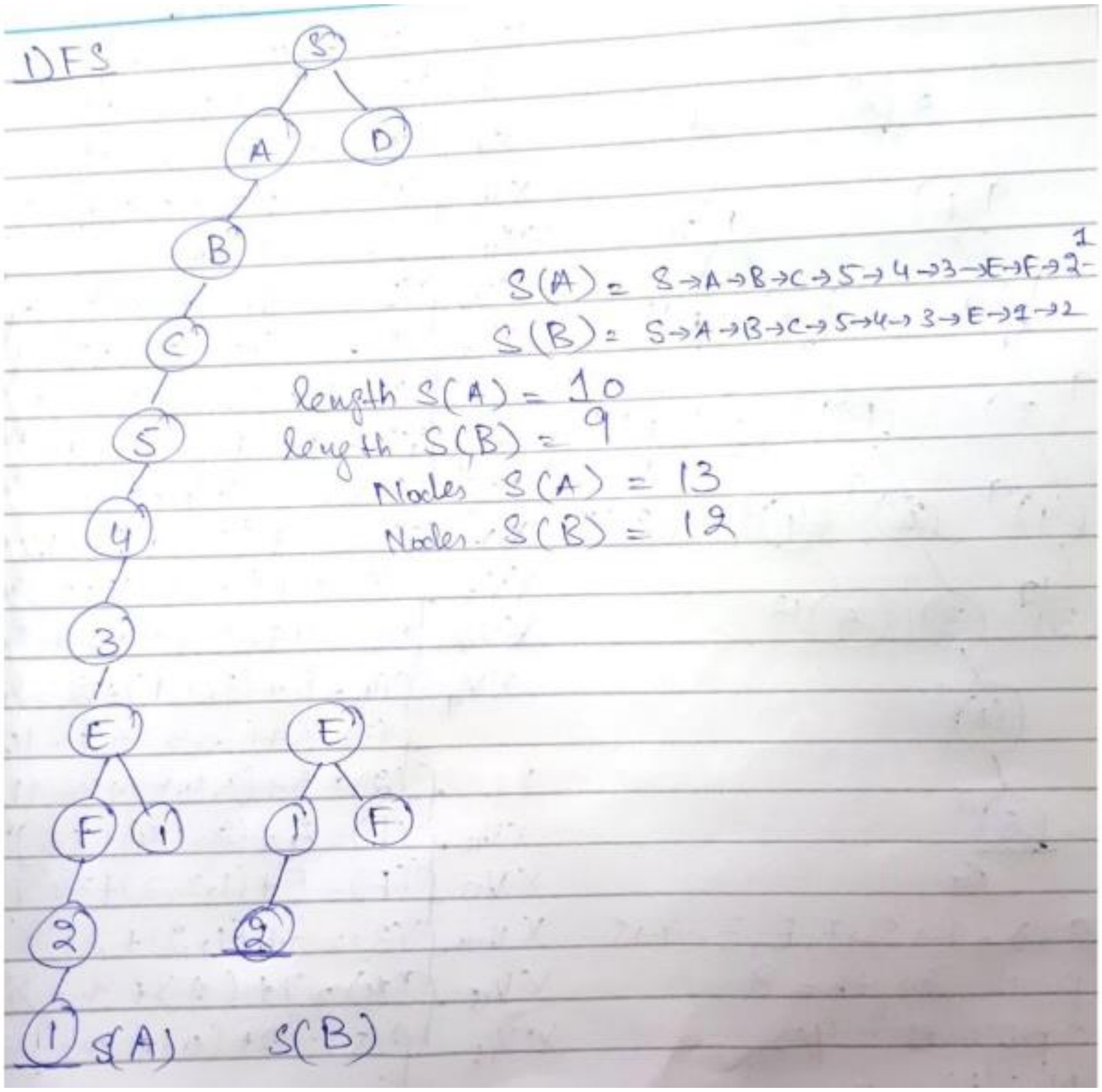


Initial State



Goal State

1. Depth First Search (DFS)



$S(A) = S \to A \to B \to C \to 5 \to 4 \to 3 \to E \to F \to 2 - 1$

$S(B) = S \to A \to B \to C \to 5 \to 4 \to 3 \to E \to 1 \to 2$

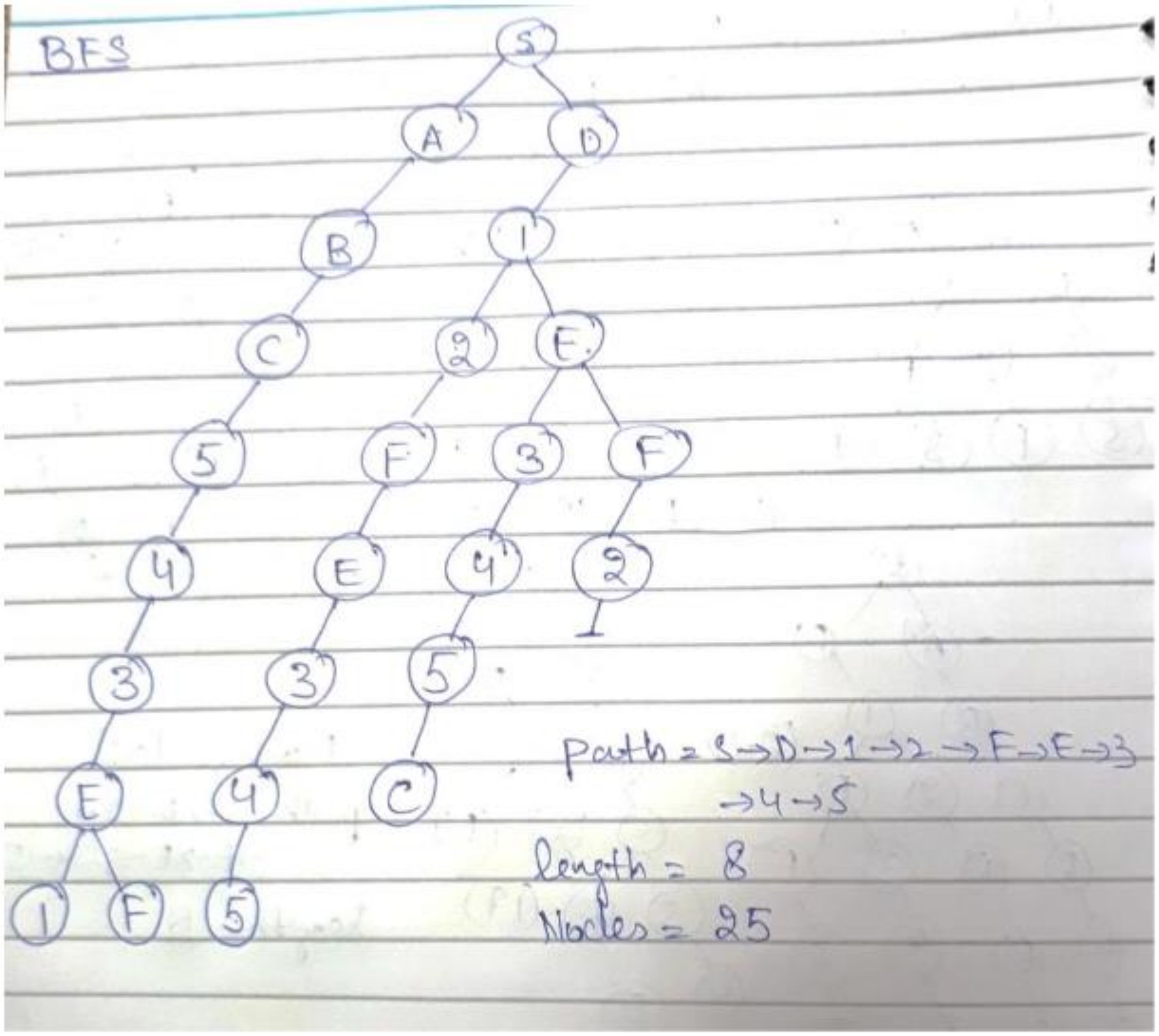
length $S(A) = 10$

length $S(B) = 9$

Nodes $S(A) = 13$

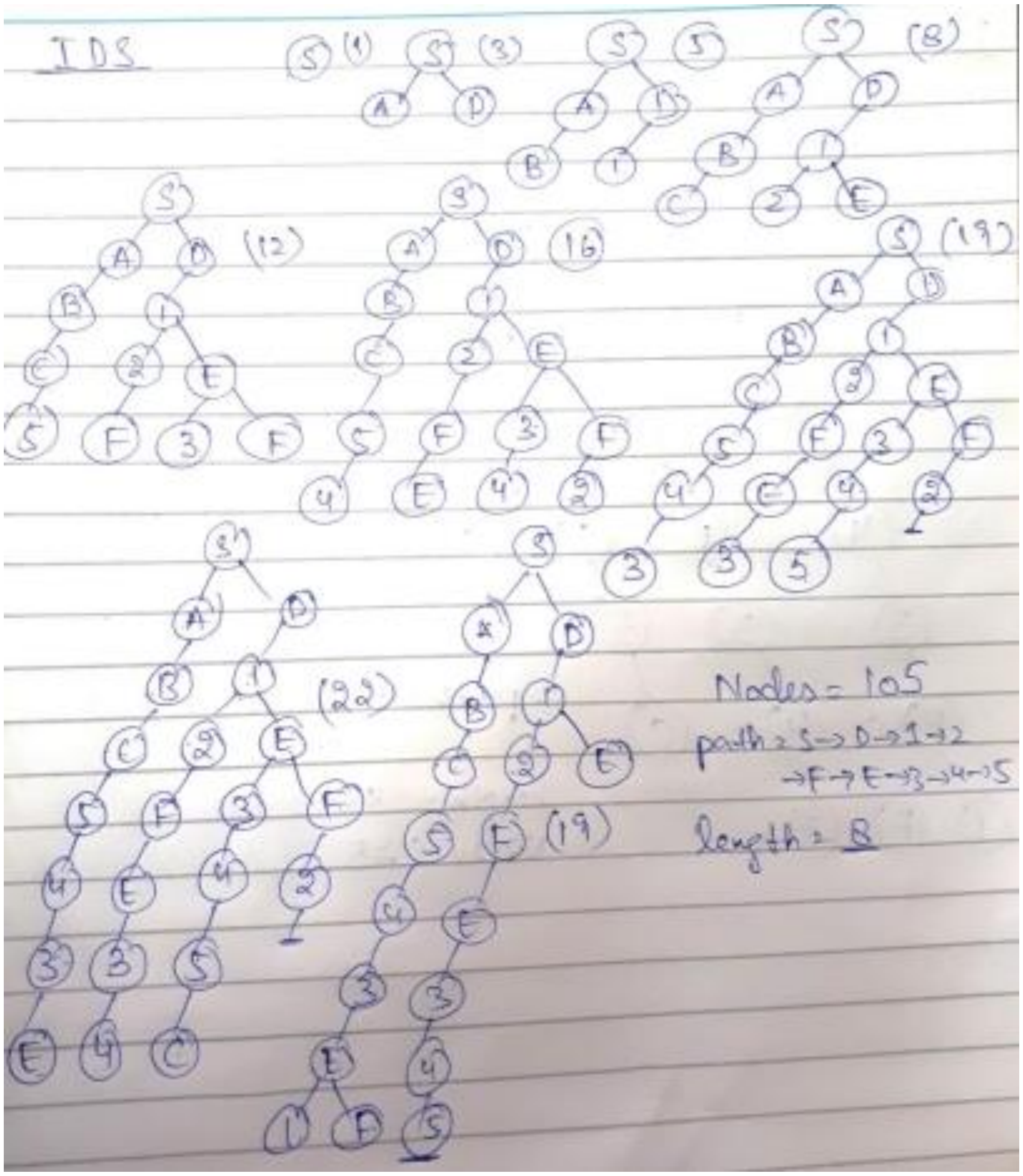
Nodes $S(B) = 12$

2. Breadth First Search (BFS)



Path = S→D→1→2→F→E→3
→4→5
length = 8
Nodes = 25

3. Iterative Deepening Search (IDS)

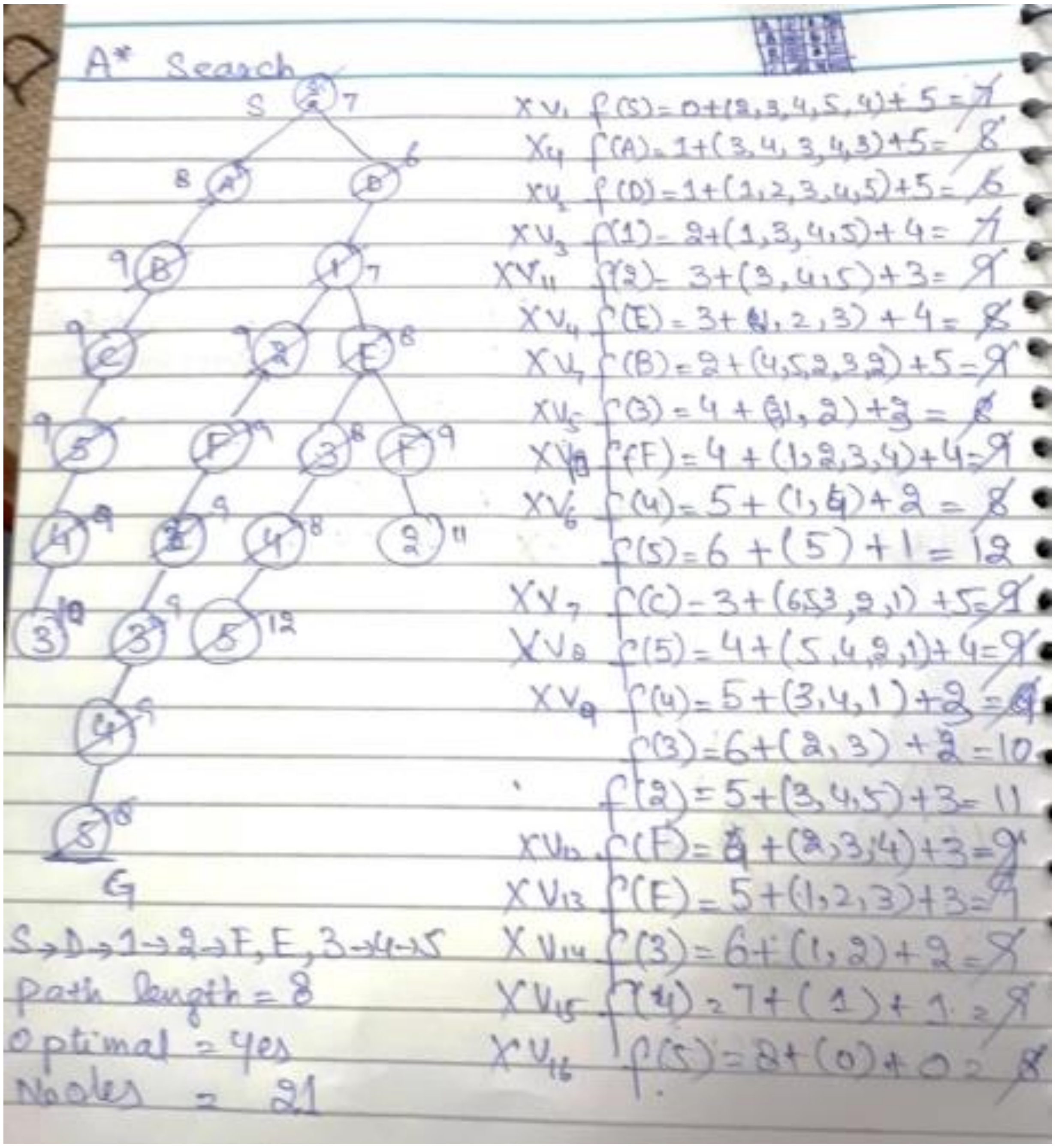


Nodes = 105
path = S→D→1→2
→F→E→3→4→5
length = 8

Part b. Apply A* search on the maze provided in Part a. and build the search tree with the heuristic function provided below. Clearly indicate the order in which each state is expanded with open list (frontier) and closed list (visited nodes) updates.

h(n) = Calculate Manhattan distance from current position to each dot and select the minimum distance + Number of remaining dots

Note: Manhattan distance will be the number of horizontal and vertical moves without considering the blocked cells.



A* Search

XV_1 $f(S) = 0 + (2,3,4,5,4) + 5 = \cancel{7}$

XV_4 $f(A) = 1 + (3,4,3,4,3) + 5 = \cancel{8}$

XV_2 $f(D) = 1 + (1,2,3,4,5) + 5 = \cancel{6}$

XV_3 $f(1) = 2 + (1,3,4,5) + 4 = \cancel{7}$

XV_{11} $f(2) = 3 + (3,4,5) + 3 = \cancel{9}$

XV_4 $f(E) = 3 + (1,2,3) + 4 = \cancel{8}$

XV_7 $f(B) = 2 + (4,5,2,3,2) + 5 = \cancel{9}$

XV_5 $f(3) = 4 + (1,2) + 2 = \cancel{8}$

XV_{10} $f(F) = 4 + (1,2,3,4) + 4 = \cancel{9}$

XV_6 $f(4) = 5 + (1,4) + 2 = \cancel{8}$

$f(5) = 6 + (5) + 1 = 12$

XV_7 $f(C) = 3 + (6,5,3,2,1) + 5 = \cancel{9}$

XV_8 $f(5) = 4 + (5,4,2,1) + 4 = \cancel{9}$

XV_9 $f(4) = 5 + (3,4,1) + 2 = \cancel{9}$

$f(3) = 6 + (2,3) + 2 = 10$

$f(2) = 5 + (3,4,5) + 3 = 11$

XV_{12} $f(F) = 4 + (2,3,4) + 3 = \cancel{9}$

XV_{13} $f(E) = 5 + (1,2,3) + 3 = \cancel{9}$

XV_{14} $f(3) = 6 + (1,2) + 2 = \cancel{8}$

XV_{15} $f(4) = 7 + (1) + 1 = \cancel{9}$

XV_{16} $f(5) = 8 + (0) + 0 = \cancel{8}$

S → D → 1 → 2 → F, E, 3 → 4 → 5

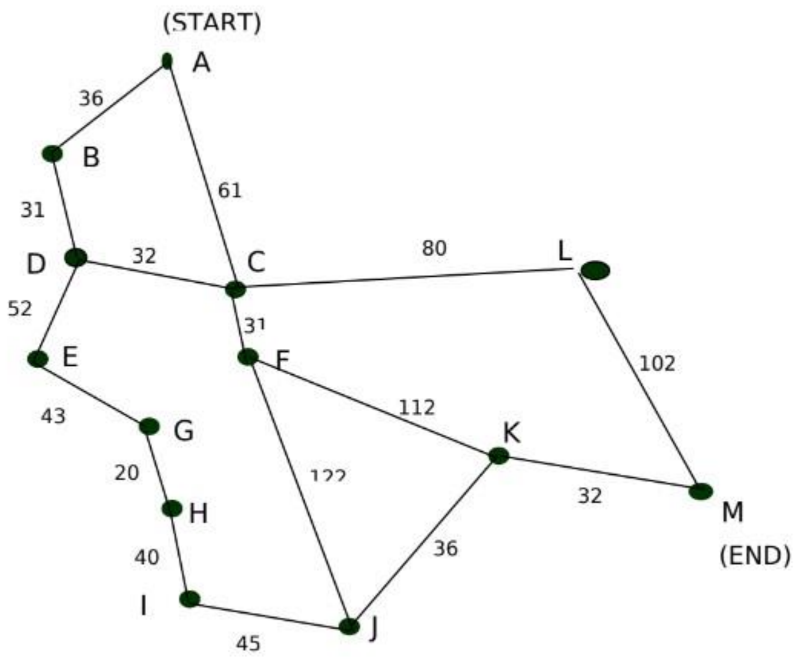
path length = 8

Optimal = yes

Nodes = 21

QUESTION 2:

Consider the following map.



Using the A* algorithm avoiding repeated states, work out a route from town A to town M.

Use the following cost functions.

- g (n) = The cost of each move as the distance between each town (shown on map).
- h (n) = The Straight Line Distance between any town and town M. These distances are given in the table below.

Provide the search tree for your solution and indicate the order in which you expanded the nodes. Finally, state the route you would take and the cost of that route.

Straight Line Distance to M

A	223
B	222
C	166

E	165
F	136
G	122

I	100
J	60
K	32

M	0

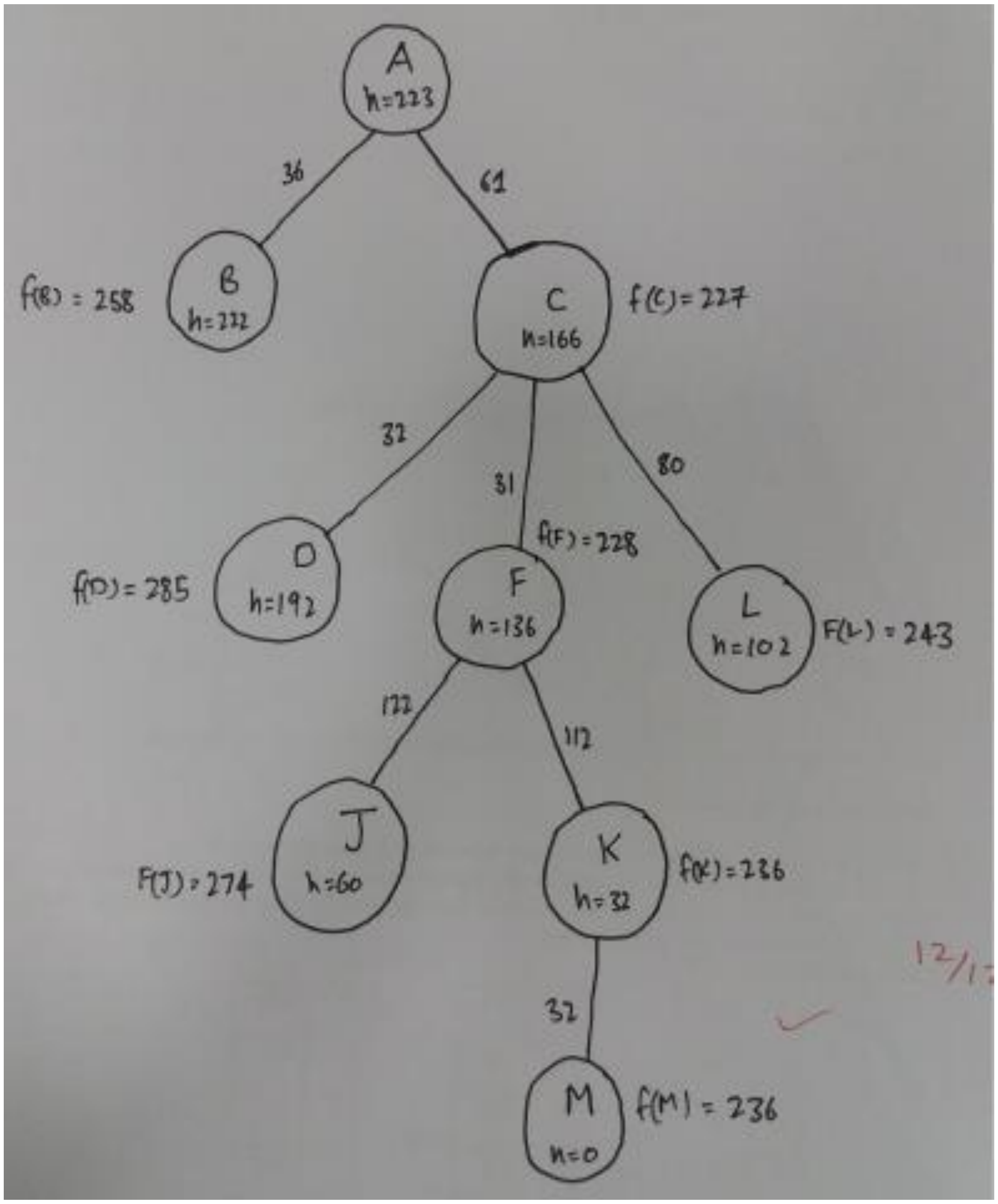
D	192

H	111

L	102

In your answer provide the following:

i) The search tree that is produced, showing the cost function at each node



ii) State the order in which the nodes were expanded.

ACFKM

iii) State the route that is taken, and give the total cost.

ACFKM

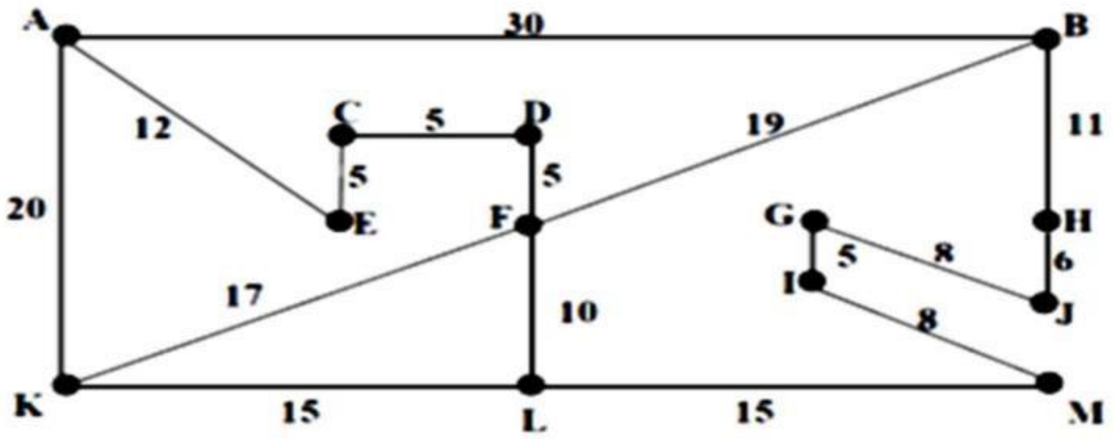
Explain the relationship between the A* algorithm and the Uniform Cost Search algorithm?

A*: f(n) = g(n) + h(n)
Uniform Cost Search = f(n) = g(n)

Relation:
When all step costs are equal, breadth-first search is optimal because it always expands the shallowest unexpanded node. By a simple extension, we can find an algorithm that is optimal with any step-cost function. Instead of expanding the shallowest node, uniform-cost search expands the node n with the lowest path cost g(n).

QUESTION 3:

Consider the following map (not drawn to scale)



Part a.

Use A* algorithm to work out a route from town A to town M. Use the following cost functions g(n)

= Total cost of reaching from town A to town n (step cost of each move is given on the map)

h(n) = Straight line distance between town n and town M These costs are given in the table below. h(n) can be used as an estimated distance to M.

Town	Distance
A	56
B	22
C	30
D	29
E	29
F	30
G	14

Town	Distance
H	10
I	8
J	5
K	30
L	15
M	0

i) Show the order of nodes expanded by A*.

EKDABCLM

ii) What path/route would be found by A* using this heuristic function?

A -> B -> F -> L -> M

Part b.

Repeat the above question for the following heuristic function.

Town	Distance
A	80
B	10
C	50
D	20
E	10
F	30
G	60

Town	Distance
H	30
I	20
J	50
K	60
L	20
M	0

i) Show the order of nodes expanded by A*.

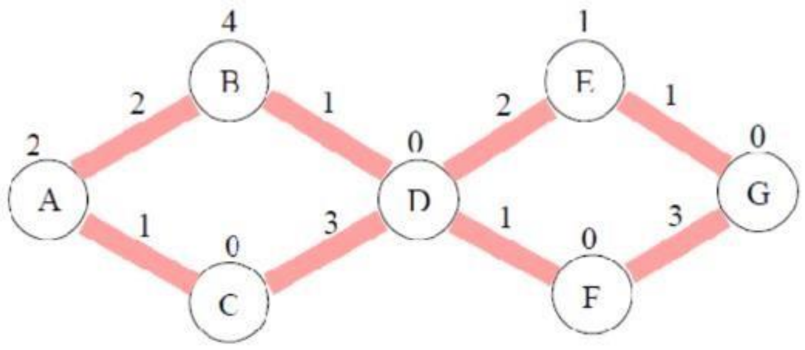
EBLHCIDFM

ii) What path/route would be found by A* using this heuristic function?

A -> E -> B -> F -> M

QUESTION 4:

The following is a state-space search graph of a problem with nodes labeled using alphabets, edges are the thick shaded lines. The number above each edge giving the transition cost (e.g., cost(C,D) = 3) of the corresponding action and the number above each node is its heuristic value of that node (e.g., h(A) = 2). The node labeled A is the initial state and that labeled as G is the goal state.



i. List, in order, the nodes that will be expanded if A* algorithm is used to find a path from the initial state to the goal state.

   ACBDEFG

ii. What path (if any) would be found by A* algorithm in part a.

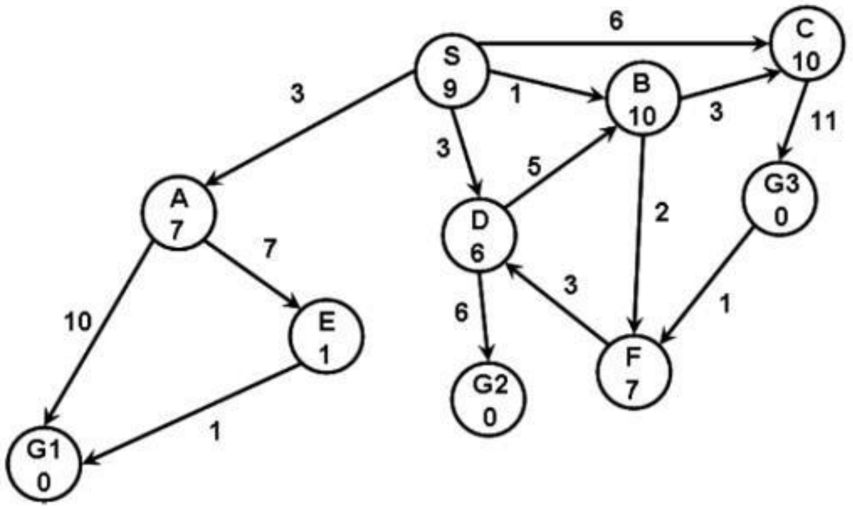
   A -> C -> D -> E -> G

iii. What would be the state (nodes in it) of frontier/queue when the goal is found by A* algorithm in part b.

   No nodes left in the frontier because all nodes have been explored to find the lowest cost path.

QUESTION 5:

For the state space search graph below, where S is the initial state and G1, G2, and G3 are goal states. Arcs are labeled with the cost of associated actions and the heuristic cost to a goal is shown inside the nodes. Mark the all goal states that might be returned by each of the following search algorithms.



DFS	**€ G1**	**€ G2**	**€ G3**
BFS	**€ G1**	**€ G2**	**€ G3**
Uniform Cost Search	**€ G1**	**€ G2**	**€ G3**
A*	**€ G1**	**€ G2**	**€ G3**

DFS: (All goal states are reached)

1) S -> A -> E -> G1
2) S -> D -> G2
3) S -> B -> C -> G3

BFS: (All goal states are reached)

Depth1: S -> A, S -> B, S -> C, S -> D
Depth2: S -> A -> E, S -> B -> F, S -> C -> G3, S -> D -> G2
Depth3: S -> A -> G1 , S -> B -> F, S -> C -> G3, S -> D -> B

Uniform Cost Search: (All goal states are reached)

1) Explore nodes with the lowest cost:
   S -> B (cost 1)
   S -> A (cost 3)

S -> D (cost 3)

2) Explore nodes with the next lowest cost:

    S -> B -> F (cost 3)
    S -> A -> E (cost 10)
    S -> D -> G2 (cost 6)
    S -> D -> B (cost 8)
    S -> C (cost 6)

3) Explore nodes with the next lowest cost:

    S -> B -> F -> D (cost 5)
    S -> D -> G2 -> G2 (cost 12)
    S -> C -> G3 (cost 17)
    S -> D -> B -> F (cost 7)
    S -> D -> B -> F -> D (cost 10)
    S -> A -> G1 (cost 13)

A*: (All goal states are reached)

1) Calculate the total estimated cost (f(n)) for each node:

    S (f(S) = 9), A (f(A) = 10), B (f(B) = 11), C (f(C) = 16), D (f(D) = 9)

2) Explore nodes with the lowest total estimated cost:

    S -> A -> E (f(E) = 11)
    S -> B (f(B) = 11)
    S -> D (f(D) = 9)

3) Explore nodes with the next lowest total estimated cost:

    S -> A -> G1 (f(G1) = 13)
    S -> D -> G2 (f(G2) = 15)
    S -> C (f(C) = 16)

4) Explore nodes with the next lowest total estimated cost:

    S -> D -> B (f(B) = 17)
    S -> B -> F (f(F) = 18)
    S -> C -> G3 (f(G3) = 27)