Faculty of Engineering and Technology
Electrical and Computer Engineering Department

Artificial Intelligence
(ENCS3340)

HomeWork #1

Prepared by:

Nour Naji– 1190270

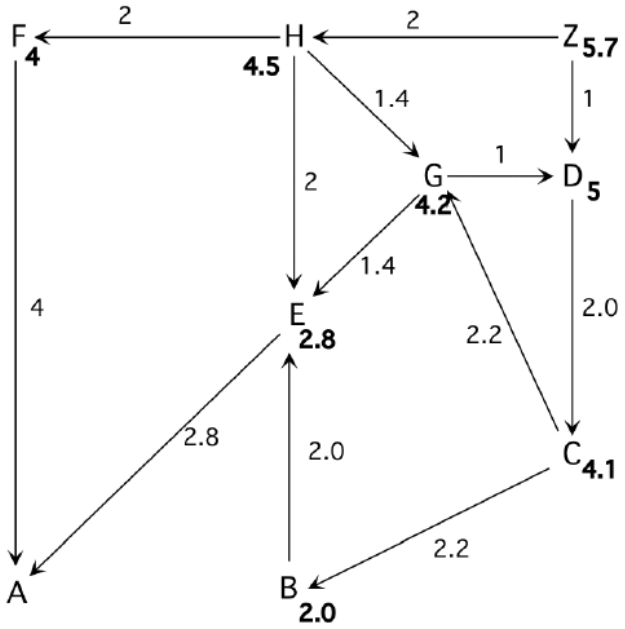
Supervised by: Dr. Yazan Abu Farha

Section: 3

Date: 28 April 2022

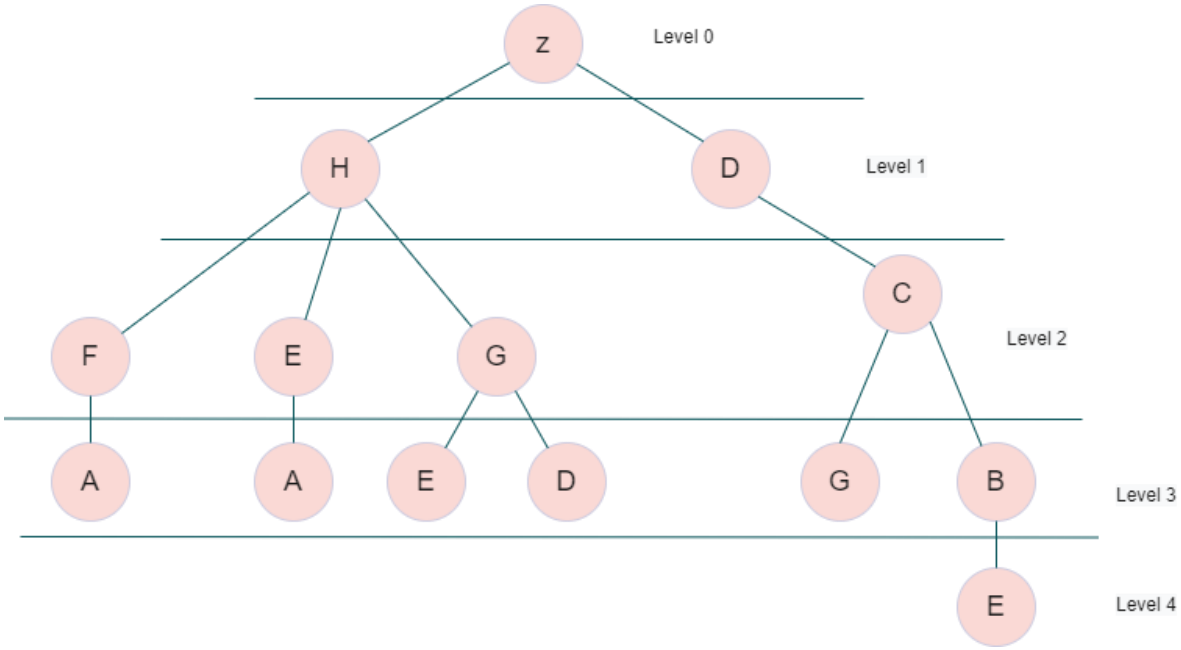
1. Question 1:

→ The start node is Z and the goal node is A.

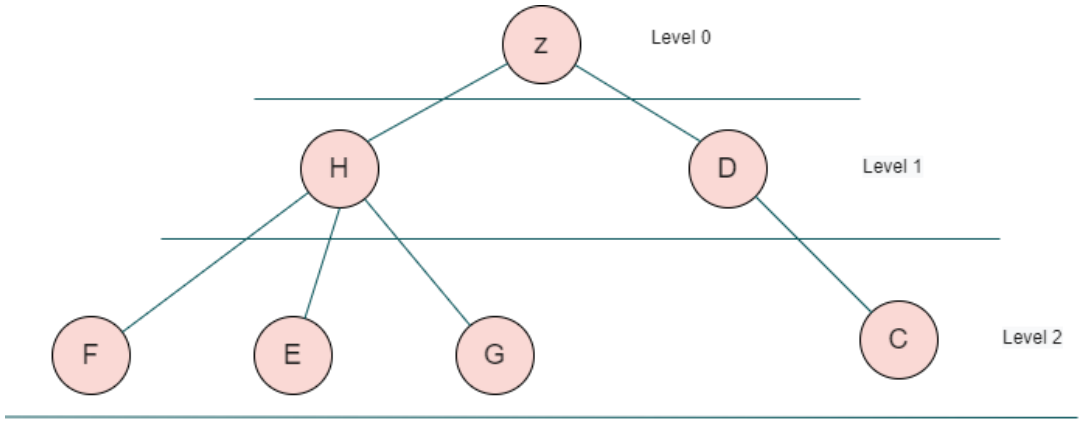


Iterative Deeping Search (IDS):

The levels of the given graph are described below: (**from left to right**)



Depth 2 means ➔ apply the Depth First Search (DFS) in two levels, at each level the node is taken from left to right.

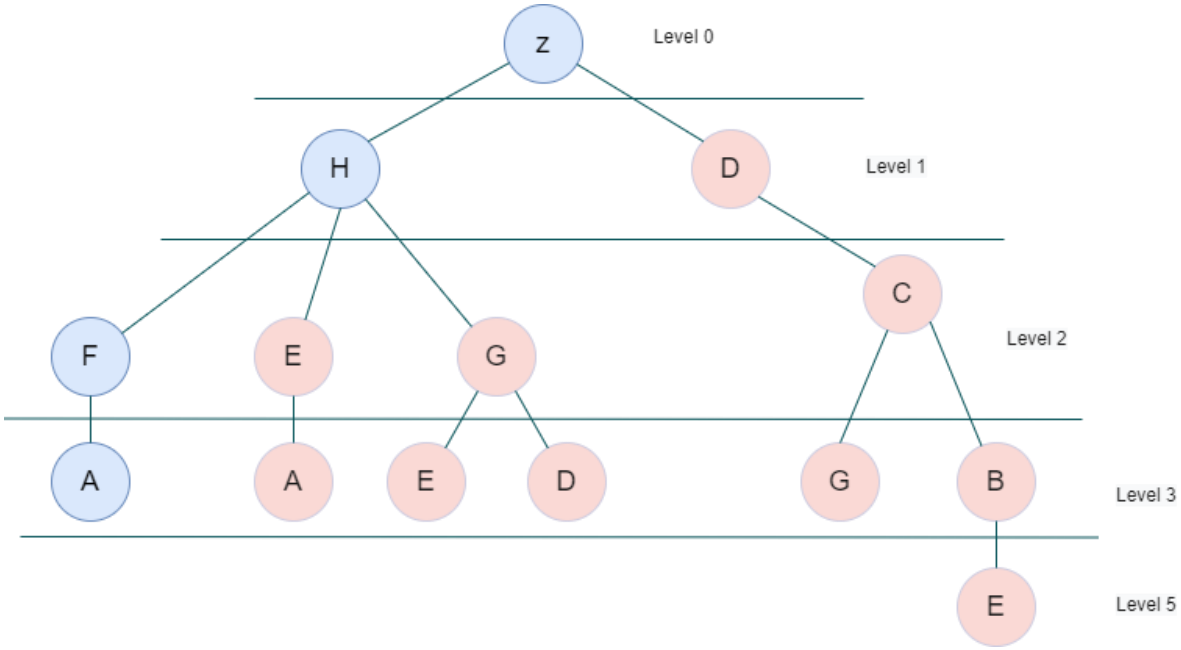


✓ **Visit node : Z => H => F => E => G => D => C.**

The path is as follows: Z, then H, which is the first level and has a depth of 1, so another node is taken from left to right, which is F, which has a depth of 2, so return to H, take E, return to H, take G, return to H, return to Z, return to D, return to D, return to Z.

- The goal is A, and it does not exist in the above path ➔ increment the level by 2.

The result is as follow:



✓ **path: Z => H => F => A.**

Heuristic:

State	Heuristic: h(n)
A	0
B	2
C	4.1
D	5
E	2.8
F	4
G	4.2
H	4.5
Z	5.4

A Heuristic function is an evaluation function that estimates the cost of getting from one place to another. It is a way to inform the search about the direction to a goal.

A heuristic h(n) is admissible if for every node n, $h(n) \leq h^*(n)$ where $h^*(n)$ is the true cost to reach the goal state from n. An admissible heuristic never overestimates the cost to reach the goal.

→ **Z node:**

heuristic h(Z) = 5.7

h*(Z) => true cost from Z to Goal:

- ZDCBEA = 10
- ZDCGEA = 9.4
- ZHFA = 8
- ZHEA = 6.8
- ZHGEA = 7.6
- ZHGDCBEA = 13.4

$h(Z) \leq h^*(Z)$, so h(Z) is admissible.

→ **D node:**

h(D) = 5

h*(D) => true cost from paths:

- DCGEA = 8.4
- DBCEA = 9

$h(D) \leq h^*(D)$, so h(D) is admissible

→ **F node:**

h(F) = 4

h*(F) => true cost from paths:

- FA = 4

h(F) ≤ h*(F), so h(F) is admissible

→ **H node:**

h(H) = 4.5

h*(F) => true cost from paths:

- HEA = 4.8
- HGEA = 5.6

h(H) ≤ h*(H), so h(H) is admissible

→ **C node:**

h(C) = 4.1

h*(C) => true cost from paths:

- CBEA = 7
- CGEA = 6.4

h(C) ≤ h*(C), so h(C) is admissible

→ **B node:**

h(B) = 2

h*(B) => true cost from paths:

- BEA = 4.8

h(B) ≤ h*(B), so h(B) is admissible

→ **E node:**

h(E) = 2.8

h*(E) => true cost from paths:

- EA = 2.8

h(E) ≤ h*(E), so h(E) is admissible

→ **G node:**

h(G) = 4.2

h*(G) => true cost from paths:

- GEA = 4.2

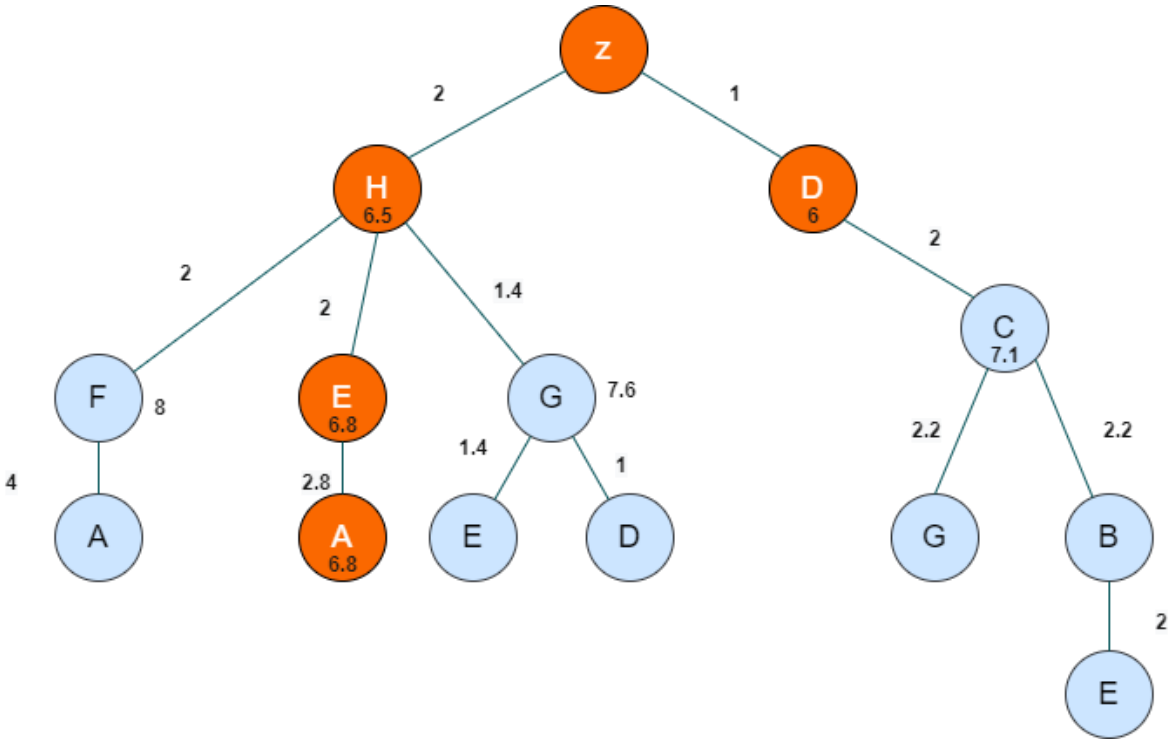
h(G) ≤ h*(G), so h(G) is admissible

The heuristic h(n) is admissible since ➔ h(goal) = 0

for each node the value of h(n) ≤ h*(n), so the result that the heuristic function h(n) is admissible.

A* Search

To apply the A* search for the graph, the steps are:



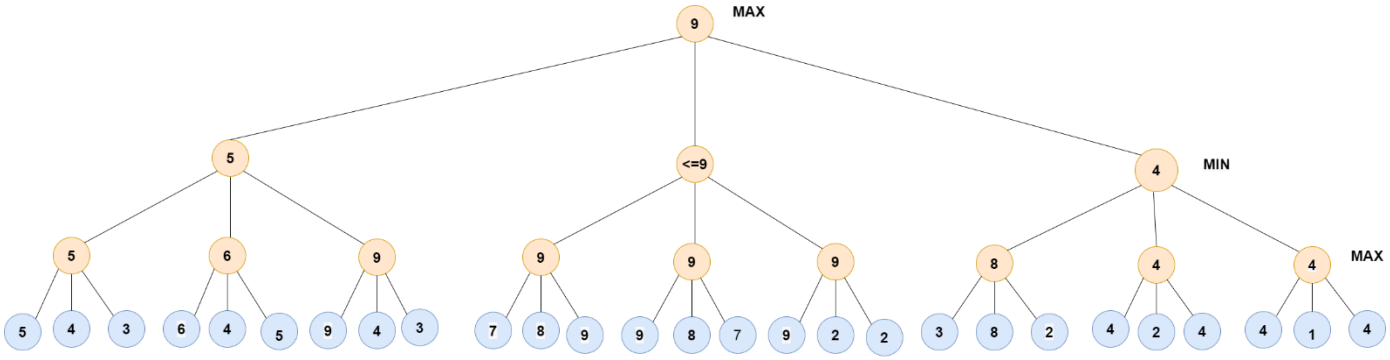
The numbers in the line denote the cost of moving from one node to another, while the numbers inside the node denote the value of f(n), which is dependent on path cost g(n) and the value of the heuristic function h(n) ➔ f(n)= g(n) + h(n)

The following is the path: **Z => H => E => A**, with a total cost of 6.8.

Because no approach with a cost less than 6.8 can reach the objective node A, this solution is the best. As a result, the solution above is the least-cost approach that achieves **optimality**.

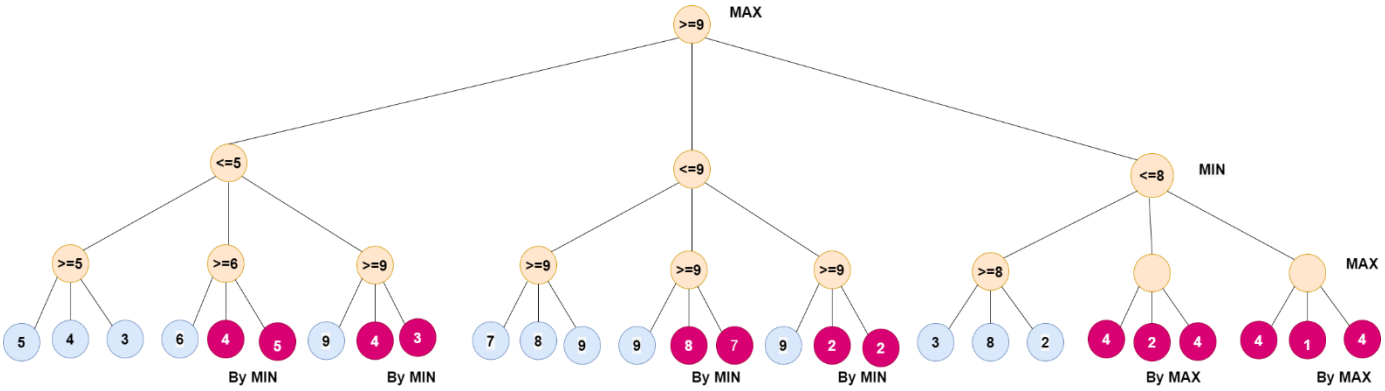
Question 2:

2.1 MinMax Search:



→ Nodes in Orange are the results of applying this search.

2.2 MinMax Search with Alpha-Beta Purning:



→ Note that after applying MinMax Search with alpha-beta pruning, the nodes in red are not visited nodes.

Question 3:

Global and local searching techniques:

The goal of both local and global search approaches is to find a solution(s) that optimize the objective function.

In global searching algorithms the goal is to always find the correct and optimal solution with the lowest cost path if one exists, whereas in local searching, the goal is to locate the answer only if one exists (care about the result not the process or the way used).

Local search reduces complexity at the cost of possible suboptimal solutions. Local search is good for:

1. problems with memory constraints, Local search algorithms keep only one current state in memory (or a fixed number of states, if using algorithms like local beam search and genetic algorithm) , However, path-constructing algorithms usually take exponential order of memory for large search trees.
2. problems with changes in state space, for instance, online search.
3. Local search sacrifices completeness for greater efficiency by ordering partial solutions by some heuristic predicting how close a partial solution is to a complete one. Beam search is a greedy algorithm.

Global searching approaches are employed for issues with a minimal number of variables, where computing time is not critical, and the value of finding the true global solution is very high. It's good for scanning the whole input search space and getting near to (or exactly discovering) the function's extrema.