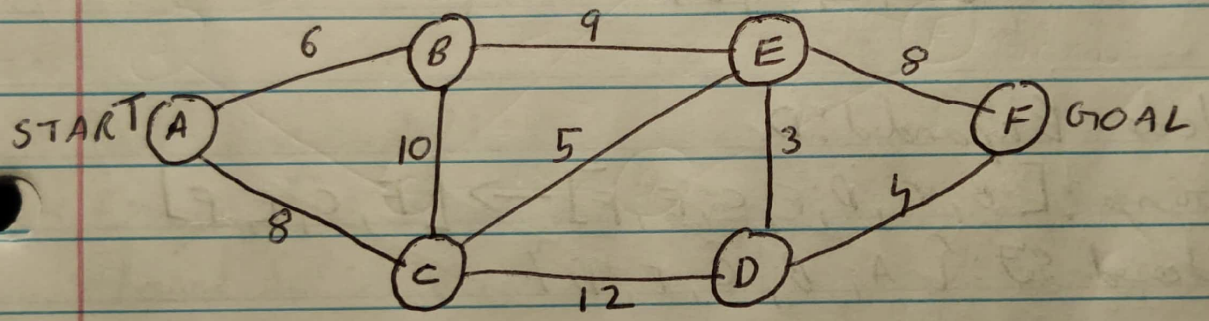
ASSIGNMENT-1

ANGAD MANJUNATHA
1001718335.

Task 1
Programming assignment

Task 2



; Breadth First Search (BFS)

Node Expanded: A
Fringe: ~~[B, C]~~ [B, C]
Closed Set: {A}

Node Expanded: B
Fringe: [C, E, C, A]
closed set: {A, B}

Node Expanded : C
Fringe : [A, E, C, A, A, B, E, D] → [E, A, B, E, D]
Closed Set : {A, B, C}

Node Expanded : E
Fringe : [B, C, D, F, A, B, E, D] → [D, F, A, B, E, D]
Closed Set : {A, B, C, E}
Fringe : [A, B, E, D, B, C, D, F] → [D, B, C, D, F]
Closed Set {A, B, C, E}

Node Expanded : D
Fringe : [B, C, D, F, C, E, F] → [F, C, E, F]
Closed Set : {A, B, C, E, D}

Node Expanded : F
Fringe : [C, E, F, E, D]
Closed Set : {A, B, C, E, D, F}

We have reached our goal so we stop here and reconstruct the path

start (A) → (B) → (E) → (D) → (F) GOAL

Total cost - 22

ii Depth First Search (DFS)

Node Expanded : A
Fringe : [B, C]
closed Set : [A]

Node Expanded : B
Fringe : [E, D, C, A, C]
closed Set : [A, B]

Node Expanded : E
Fringe : [F, D, C, B, D, C, A, C]
closed Set : [A, B, E]

Node Expanded : F
Fringe : [D, C, B, D, C, A, C]
closed Set : [A, B, E, F]

Goal state reached

Start (A) → (B) → (E) → (F) GOAL
Total cost — 23

iii Iterative Deepening Search (IDS)

It 0 Node Expanded: A
Fringe : [ ]
closed set: { }

It 1 Node Expanded: A
Fringe : [B, C]
closed Set: { A }

It 2 Node Expanded: B, C
Fringe: [E, C, A, B, E, D]
closed Set: { A, B, C }

It 3 Node Expanded: E, D
Fringe: [A, C, [illegible], F, D, B, C, F, F, E, C]
closed set: { A, B, C, F }

Path - A -> B -> C -> F
Total cost - 6 + 9 + 8 = 23

iv Uniform Cost Search

Step 1. Fringe = [A]
closed set: [ ]

Step 2. Pop A.
Fring = [B, C]
closed set: [A]

Step 3 Pop B
Fringe = [C, A, E, C]
closed set: [A, B]

Step 4 Pop C
Fring [A, E, E, C, A B, D]
closed set: [A, B, C]

Step 5 Pop A, bw A is present, so no sucessors added

Step 6. Pop E
Fringe [E, C, A, D, B, C, D, F, B]
closed set: [A, B, C, E]

Step 7. Pop E, C, A, all of them are present, so no sucessors added.

Step 8 Pop D
Fringe [B, C, E, F, D, F, B, C]
closed Set {A, B, C, E, D}

Step 9. Pop B, C, E, all are present, so no successor added.

Step 10. Pop F
~~Exit~~ Our goal node is reached and it satisfies our condition.

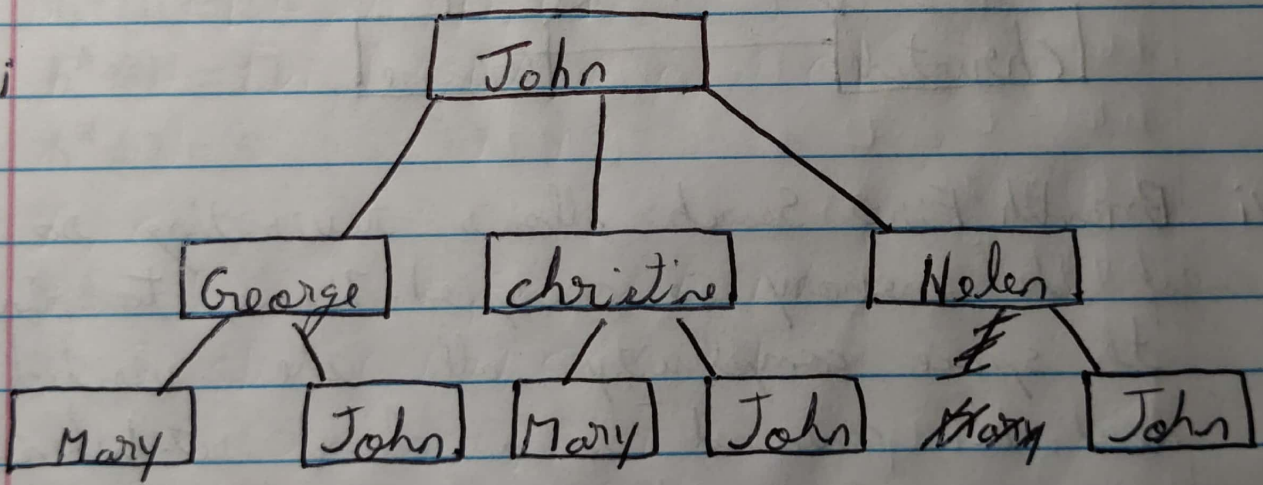
~~Path is~~ A → B → E → F
~~Total~~ Cost 6 + 9 + 8 = 23

Path : A → C → E → D → F
Total cost: 8 + 5 + 3 + 4 = 20

Task-3

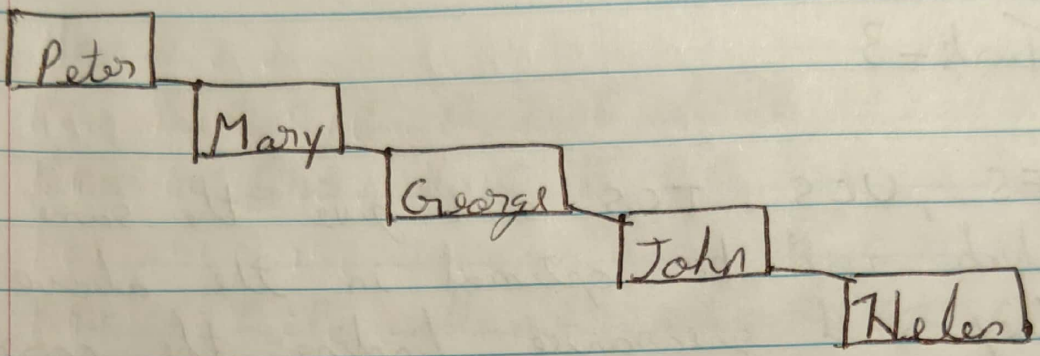
i BFS, UCS, ICS will give the same solution which will be optimal in the above scenario. They will guarantee finding the correct number of degrees of separation between any two people in the graph.
Whereas DFS on the other hand, is known for going into infinite loops. Hence may not give the correct degrees of freedom.

ii

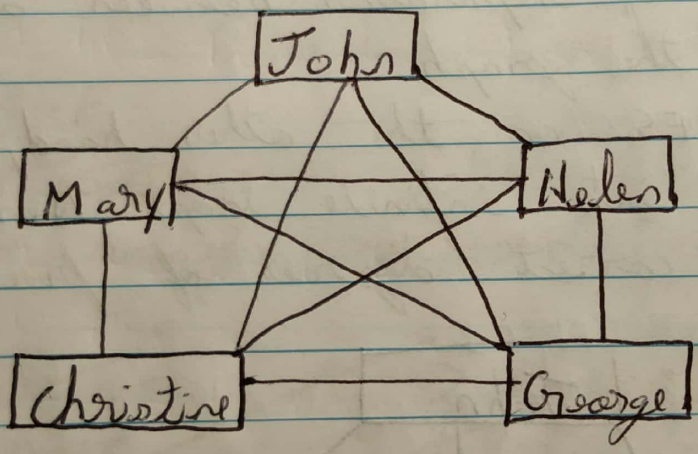


iii From the above search tree, we can see that there is no one-to-one correspondence between nodes in the tree and the vertices in the SNG. As we can see, the resulted search tree has many repeated states but SNG has a single vertex for the same.

iv

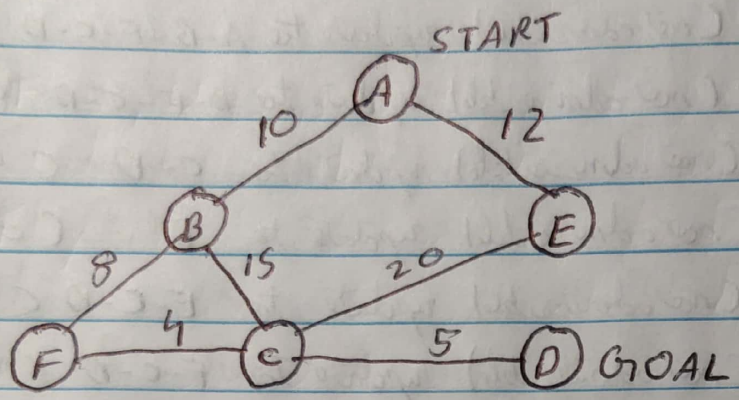


v.



vi Breadth First Search allows revisiting so most of the memory is occupied. Hence, to minimize the space complexity, all the visited node must be contained in a closed set. Whenever a new node needs to be added to the fringe it should be checked with closed set. This method allows less memory consumption not exceeding 1GB for SNO of 1 million.

Task-4



Heuristic values can be obtained by the shortest path from one state to goal state.

$h^*(A) = 27$ $h^*(D) = 0$

$h^*(B) = 17$ $h^*(E) = 25$

$h^*(C) = 5$ $h^*(F) = 9$

When the values do not exceed, the heuristic is said to be admissible

Heuristic 1

$h(A) = 5 \leq 27$ (admissible)

$h(B) = 20 \leq 17$ (not admissible) update to B-F-C-D (25)

$h(C) = 15 \leq 5$ (not admissible) update to C-D (5)

$h(D) = 0 \leq 0$ (admissible)

$h(E) = 10 \leq 25$ (admissible)

$h(F) = 0 \leq 9$ (admissible)

Heuristic 2

h(A) = 40 ≤ 27 (not admissible) update to A-B-F-C-D (27)
h(B) = 40 ≤ 20 (not admissible) update to B-F-C-D (20)
h(C) = 40 ≤ 15 (not admissible) update to C-D (5)
h(D) = 40 ≤ 0 (not admissible) update to D (0)
h(E) = 40 ≤ 25 (not admissible) update to E-C-D (25)
h(F) = 40 ≤ 9 (not admissible) update to F-C-D (9)

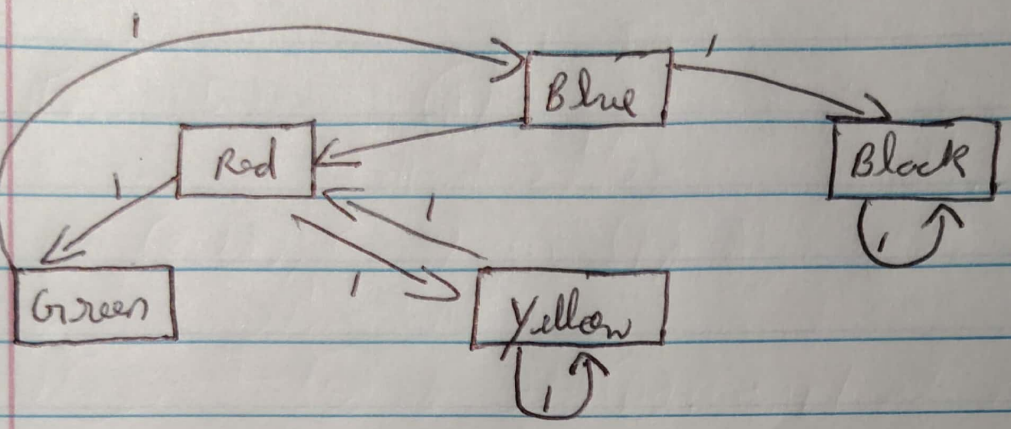
Heuristic 3

h(A) = 10 ≤ 27 (admissible)
h(B) = 15 ≤ 20 (admissible)
h(C) = 0 ≤ 15 (admissible)
h(D) = 0 ≤ 0 (admissible)
h(E) = 25 ≤ 25 (admissible)
h(F) = 5 ≤ 9 (admissible)

Heuristic 4

h(A) = 0 ≤ 27 (admissible)
h(B) = 0 ≤ 20 (admissible)
h(C) = 0 ≤ 15 (admissible)
h(D) = 0 ≤ 0 (admissible)
h(E) = 0 ≤ 25 (admissible)
h(F) = 0 ≤ 9 (admissible)

Task - 5



h (Red) = 3 ⇒ Red → Green → Blue → Black

h (Green) = 2 ⇒ Green → Blue → Black

h (yellow) = 4 ⇒ yellow → Red → Green → Blue → Black

h (black) = 0

h (blue) = 1 ⇒ Blue → Black.

Task - 5

Figure 5

If source and destination are diagonally connected i.e (2,2), (3,3). Greedy Algorithm is a better choice than A*

In the rest of the cases, both A* and Greedy algorithm will work the same.

Therefore, Greedy Search will perform better than or same as A*.

Figure 6

Case 1 Consider the source (2,1) and destination (2,3) Greedy Search will run for infinite time and A* on the other hand will complete in finite time. Therefore A* is better

Case 2. Consider the Source (2,3) and destination (3,4) A* Search will expand more nodes than Greedy Search.

Therefore Greedy Search is better.

Case 3 Consider the Source (0,2) and destination (0,5) Greedy and A* Search will perform the same.

Hence,
Greedy Search performs sometimes better, sometimes worse and sometimes the same as A*, depending on the start and end states.