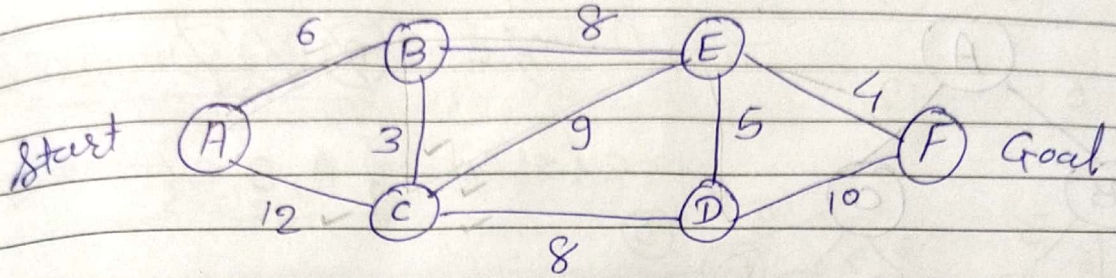
8th Feb.
AI.

UTA ID - 1001982478

Assignment - 1

* Task - 1

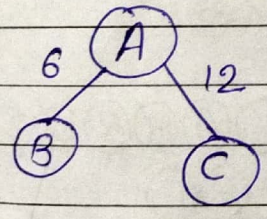


(1) Breadth first Search:- (FIFO)

step-1 ⇒ Here, starting node is (A) and goal node is (F)
⇒ In BFS, first we adding the starting node to the fringe, so,

fringe = (A)
closed set = { } = empty

step-2 ⇒ now, expand node (A). Add other node to the fringe so (A) will be pop and added to the closed set.

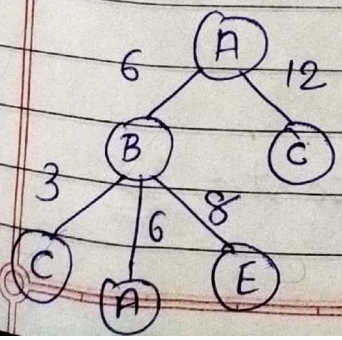


fringe = ~~(A)~~ (B) (C)

A is popped

Closed = { A }

step-3 ⇒ Now, expanding (B) node. add the successor nodes to the fringe and (B) will pop and added to the closed set.



fringe = ~~(A)~~ ~~(B)~~ (C) (E) (C) (A)

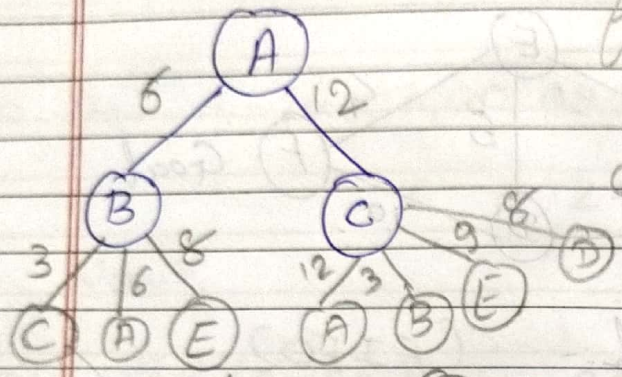
Closed = { A, B }

B is popped.

UTA ID - 1001982478

Miracle
Pg.No :
Date : / /201

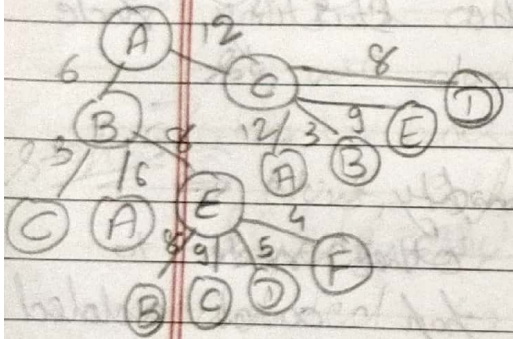
Step-4 ⇒ Expanding node (E) (C) in the graph. Add the successors for node (E) (C) and (C) (E) will be popped and added to the closed set.



fringe = (A) (B) (C) (C) (A) (E) (A) (B) (E) (D)

closed = {A, B, C} C is popped

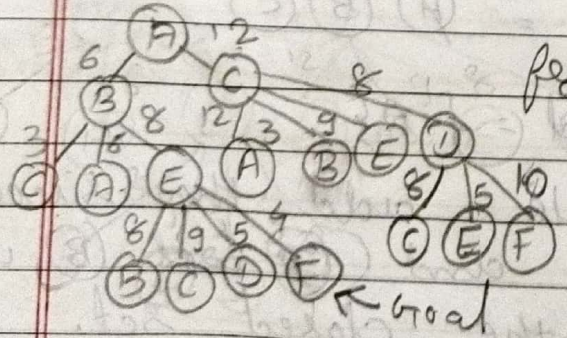
Step-5 ⇒ Expanding (E), Add successors to fringe and pop (E) node and add to closed set.



fringe = (A) (B) (C) (C) (A) (E) (A) (B) (E) (D) (B) (C) (D) (E)

closed = {A, B, C, E} E is popped.

Step-6 Expanding (D), Add successors to fringe and pop (D) node and add to closed set.



fringe = (A) (B) (C) (C) (A) (E) (A) (B) (E) (D) (B) (C) (D) (F)
(C) (E) (F)

Goal

closed = {A, B, C, E, D} D is popped.

⇒ The no. of nodes expands to reach the goal = {A, B, C, E, D}

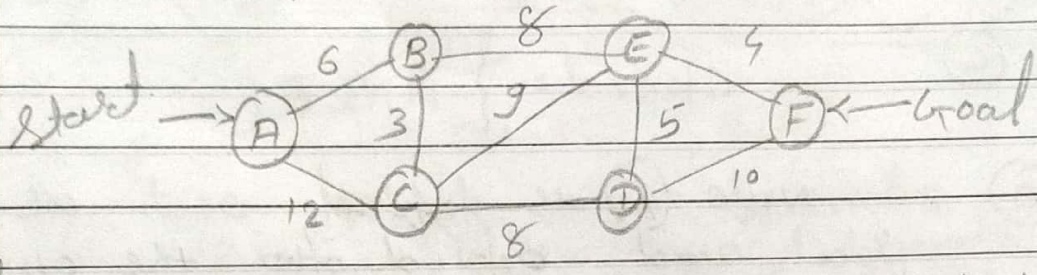
⇒ (A) →6 (B) →8 (E) →4 (F) cost = 18

UTA ID - 1001982478

depth

Miracle
Pg.No :
Date : / /201

(b) depth - first search :-

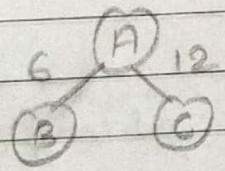


Step-1 Adding (A) to the fringe as starting node of graph.

fringe = [A]

closed : {} → empty

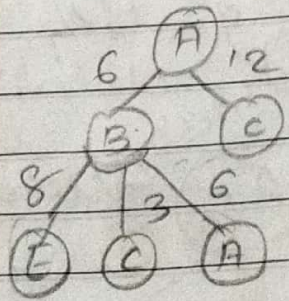
Step-2 Expanding (A) node and adding successors of (A) to the fringe and pop out (A) and add to closed set



fringe = [B C]

closed = {A}

Step-3 Expanding (B) node and adding successors to the fringe and pop out (B) and adding to closed set.

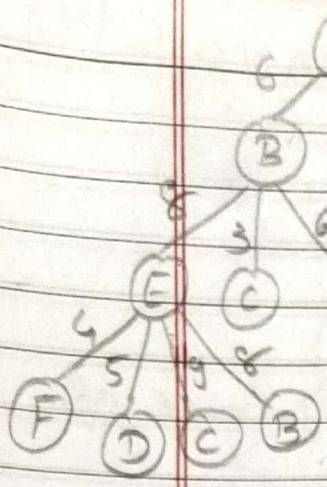


fringe = [E C A C]

Closed = {A, B}

Step-4 Expanding (E) node and adding successors to the fringe and pop out (E) and adding to closed set

UTA ID - 1001982478



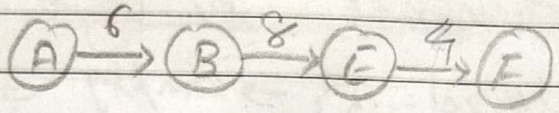
Goal

fringe = F D C B C A C

closed = {A, B, E}

∴ Node (F) is taken out as goal and added to the closed set {A, B, E, F}

∴ The no. of nodes expanded to reach the goal



∴ The total cost to reach the goal = 6 + 8 + 4 = 18

(c) Iterative - Deepening search:- → depth + level

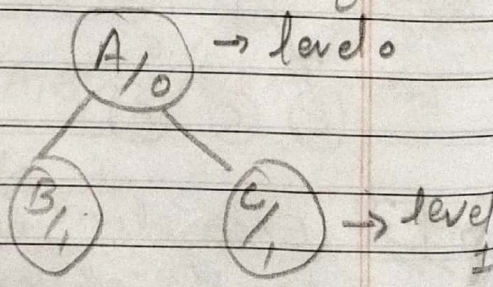
depth-1 Adding starting node (A) to fringe

fringe = A/0 | A/0 → level 0.

closed = { } → empty

depth-2 Expanding node (A) and adding the successor to the fringe and closing node (A)

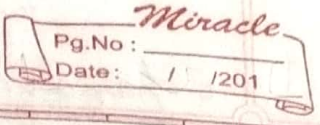
fringe = A/0 B/1 C/1 |

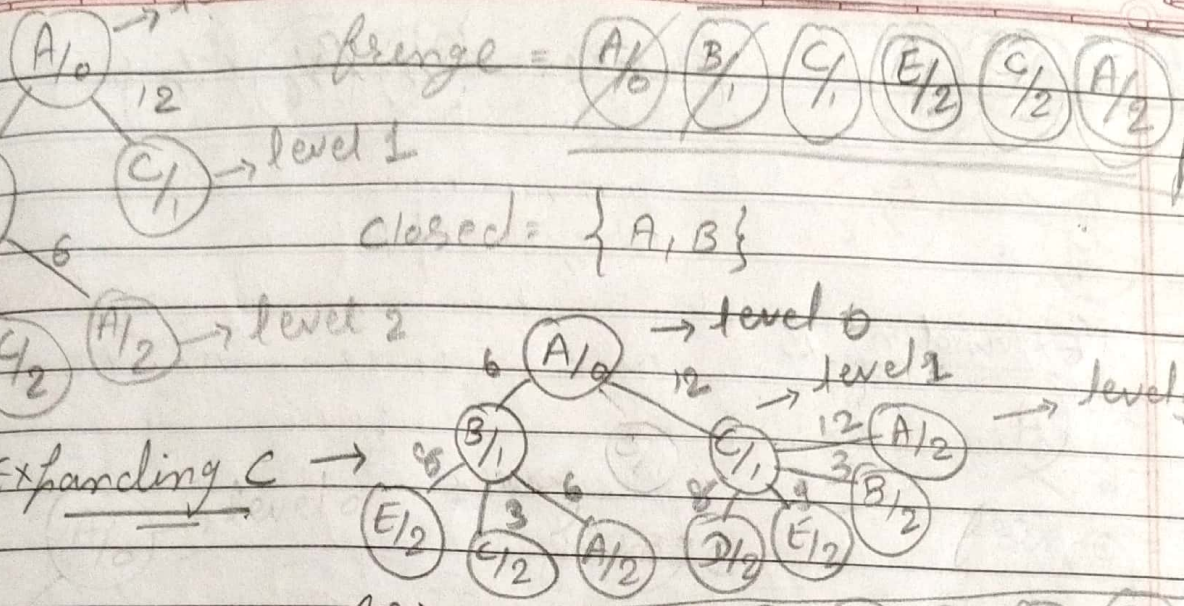


closed = {A}

Expanding B

UTA ID - 1001982478





fringe = A_0 B_1 C_1 E_2 C_2 A_2

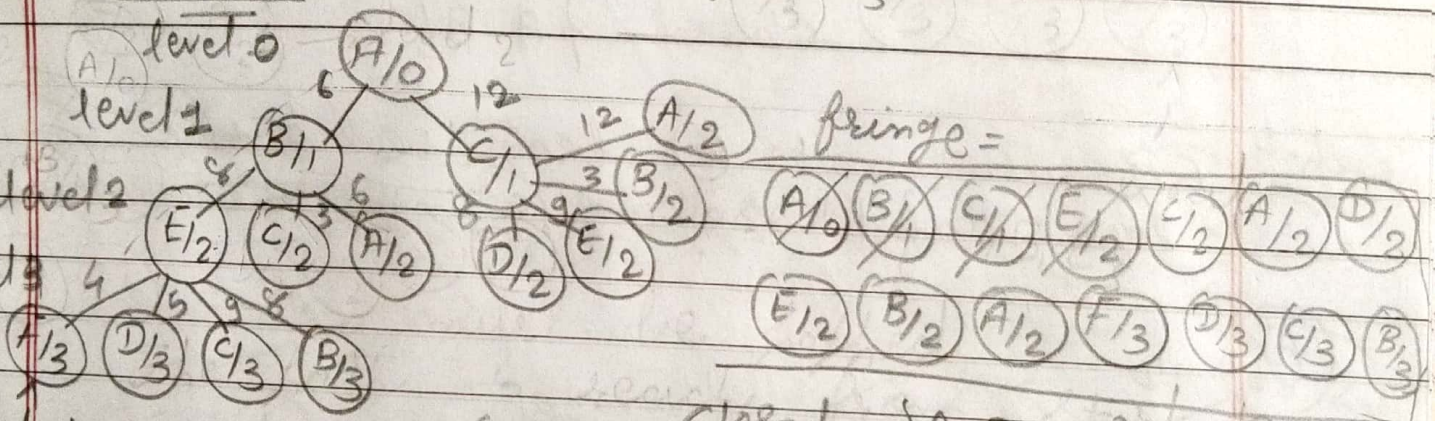
closed = {A, B}

Step-4 Expanding C →

fringe = A_0 B_1 C_1 E_2 C_2 A_2 D_2 E_2 B_2 A_2

Step-5 Expanding E

closed = {A, B, C}



fringe = A_0 B_1 C_1 E_2 C_2 A_2 D_2 E_2 B_2 A_2 F_3 D_3 C_3 B_3

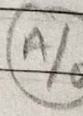
closed = {A, B, C, E}

goal

Path: A_0 →(6) B_1 →(8) E_2 →(4) F_3 cost = 18

(b) Uniform cost search → cost.

Step-1 Starting with node (A) and adding to the fringe.



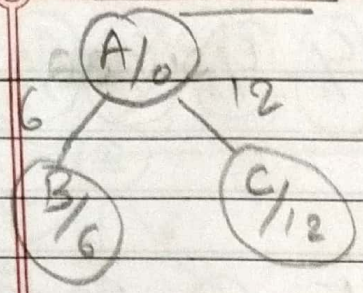
fringe = A_0

closed = {} → empty

UTA ID - 1001982478

Expanding A :-

Step-2

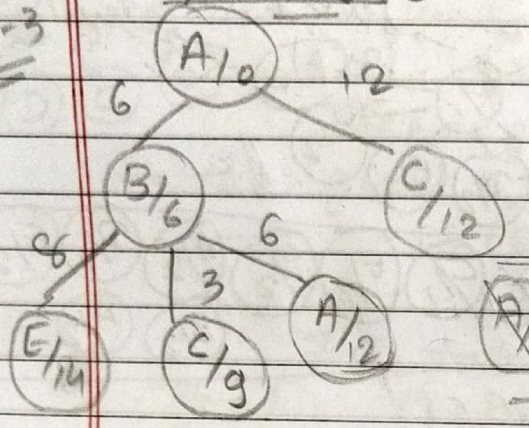


fringe = A/0 B/6 C/12

closed = {A}

Expanding B

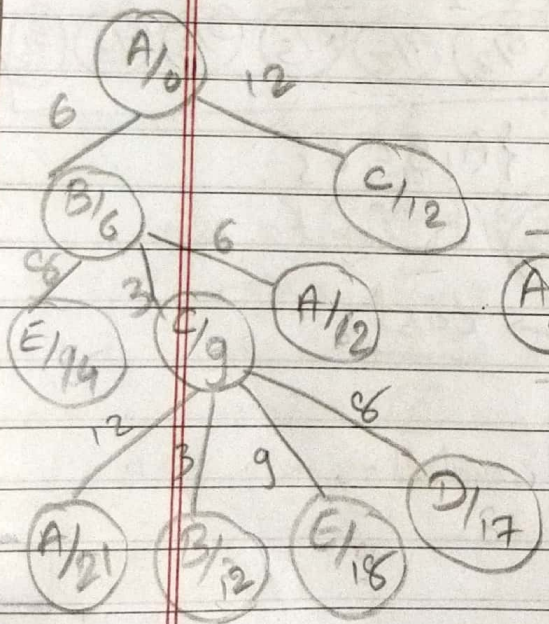
Step-3



fringe = A/0 B/6 C/9 C/12 A/12 E/14

closed : {A, B}

Step-4 Expanding node (C) because B to C is less and adding C into the closed set



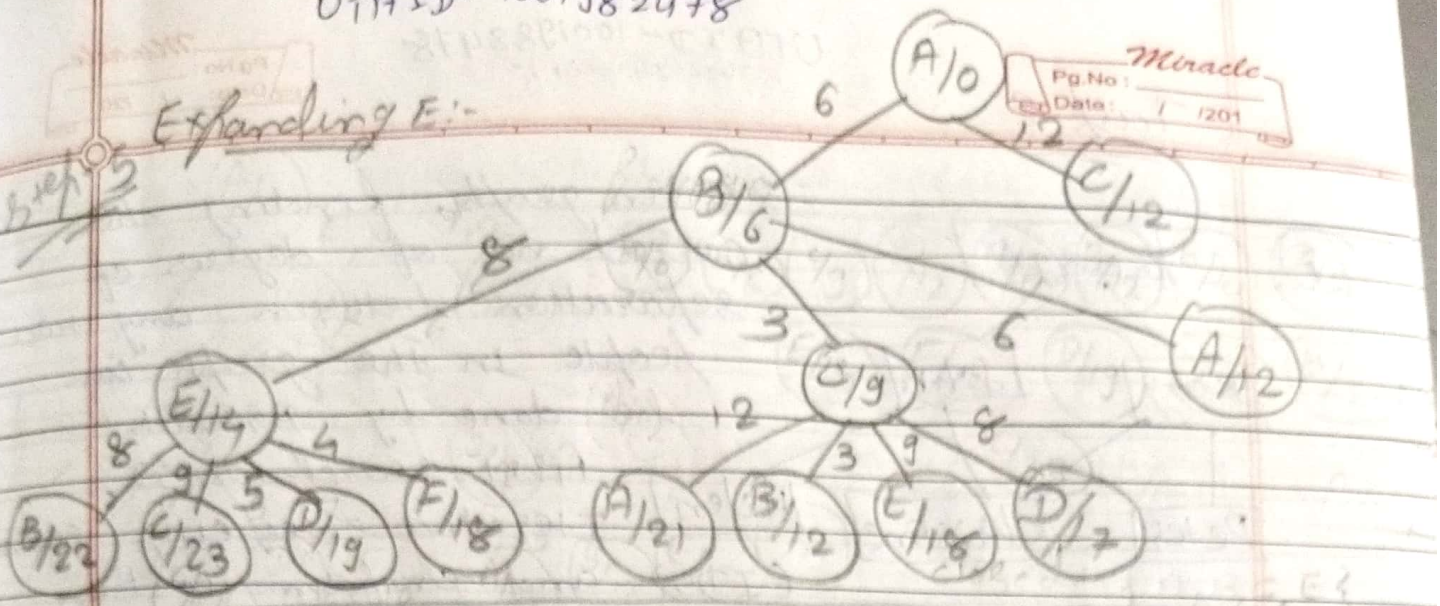
fringe = A/0 B/6 C/9 C/12 A/12 B/12 E/14 D/17 E/18 A/21

closed : {A, B, C}

UTA ID- 1001982478

Step 2

Expanding E:-

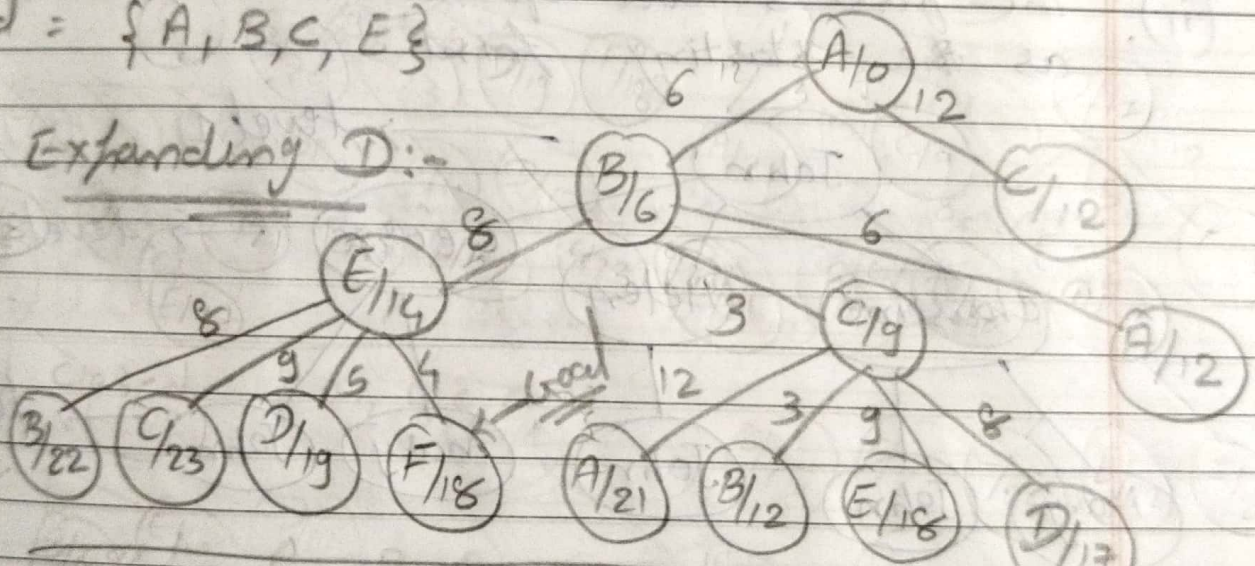


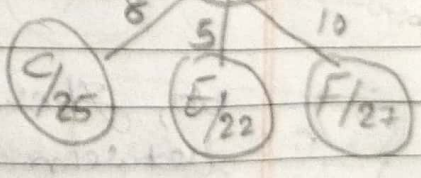
fringe = A/0 B/6 C/9 C/12 A/12 B/12 E/14 D/17 E/18 F/18 D/19 A/21 B/22 C/23

closed = {A, B, C, E}

Step 3

Expanding D:-





fringe = A/0 B/6 C/9 C/12 A/12 B/12 E/14 D/17 E/18 F/18 D/19 A/21 B/22 E/22 C/23 C/25 F/27

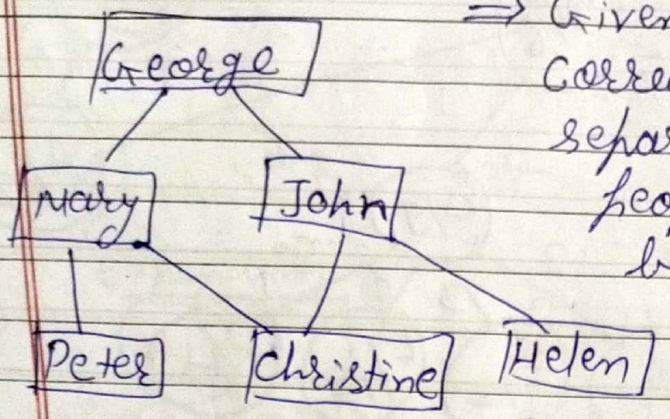
Goal

closed = {A, B, C, E, D}

Path = A/0 —6→ B/6 —8→ E/14 —4→ F/18 cost = 18

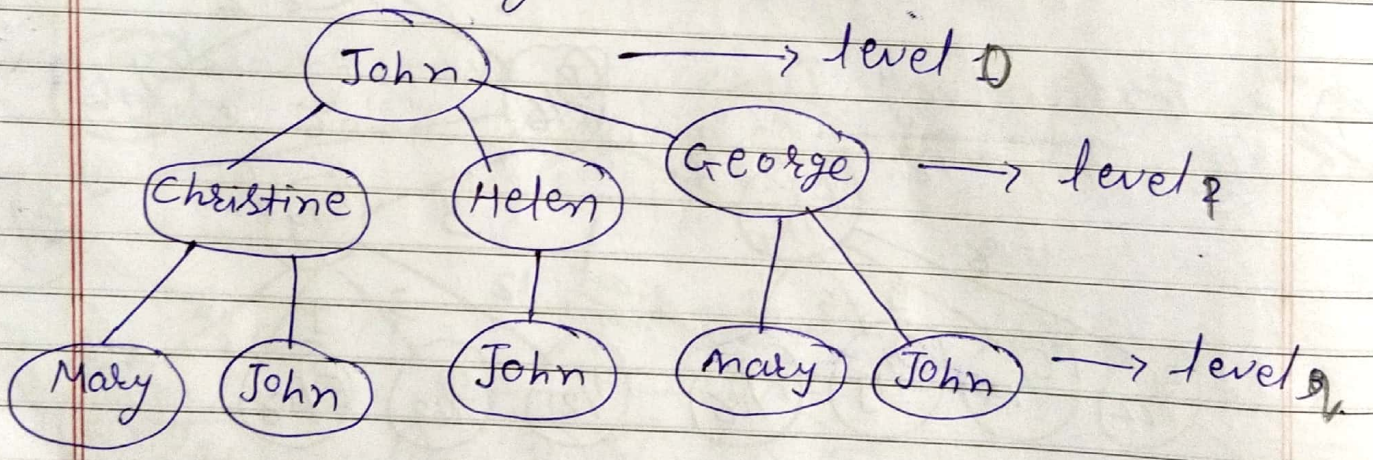
UTA ID - 1001982478

Task-2

(i). 

⇒ Given graph, finding the correct no. of degrees of separation between any two people in the graph can be done by Breadth-first search (BFS), Iterative-deepening search (IDS) and Uniform cost search (UCS) but not depth first search as it can be lead to an infinite loop.

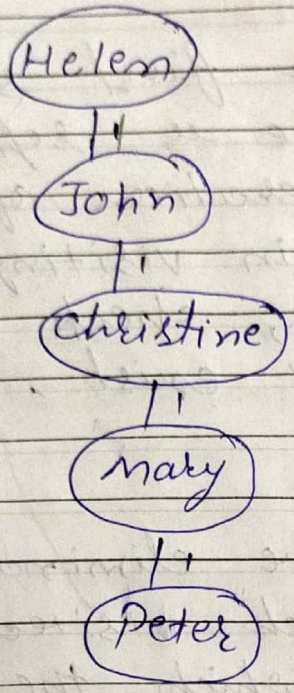
(ii). The first 3-level of search tree, with John as the starting point



(iii). No, There is a none one to one correspondence between nodes in search tree and vertices in SNCR.

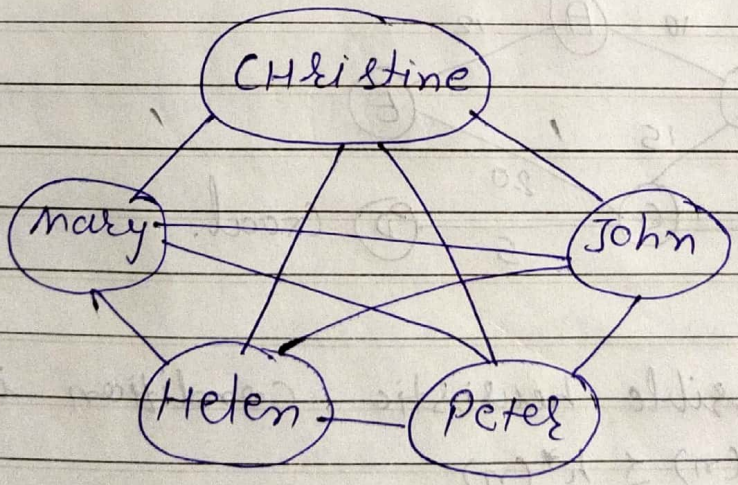
⇒ Because, George, Mary, John, Christine has multiple connection with nodes but Peter and Helen doesn't satisfy the one-to-one correspondence between the nodes.

(iv). A SNG with exactly 5 people where atleast two people has 4 degree of seperation between them considering.



⇒ So, the degree of separation between Helen and Peter is 4//.

(v). A SNG with exactly 5 people and each having 1 degree of separation can be done by christine, Mary, John, Helen, Peter.



⇒ So, all the nodes have 1 degree of separation between them as they are directly connected to each other.

UTA ID - 1001982478

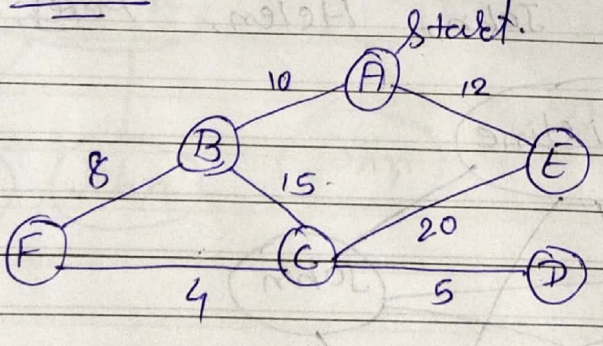
(vi). Given problem statment we know that a single node is 1KB and 1 million KB is equal to 1GB.

⇒ So, we can use Breadth-first search (BFS) using Graph search where any visited node is separated and reduce the generation of duplicate nodes and stored in visiting list. If it's visited again, check that list and if it's already exist, the node is terminated.

⇒ Duplicate nodes can be eliminated by using BFS. in which visited and unvisited nodes by which the memory capacity will not exceed 1GB memory.

⇒ So, Convert the search tree into Graph tree.

* Task-3



⇒ For Admissible heuristic, condition is

$h(n) \leq h^*(n)$

↓ assumed cost

→ Actual cost.

UTA ID - 1001982478

From the graph,

$h^*(A) = 10+8+4+5 = 27$
$h^*(B) = 8+4+5 = 17$
$h^*(C) = 5$
$h^*(D) = 0$ (Goal).
$h^*(E) = 20+5 = 25$
$h^*(F) = 4+5 = 9$

⇒ Heuristic 1:- $\because h(n) \leq h^*(n)$

$h(A) = 5$	$h^*(A) = 27$	~~Not satisfied~~
$h(B) = 20$	$h^*(B) = 17$	Not satisfied
$h(C) = 15$	$h^*(C) = 5$	Not satisfied
$h(D) = 0$	$h^*(D) = 0$	
$h(E) = 10$	$h^*(E) = 25$	~~Not satisfied~~
$h(F) = 0$	$h^*(F) = 9$	

⇒ In order to make it heuristic admissible the value of $h(C)$ must be less than or equal to 5 and $h(B)$ and 17.

⇒ Heuristic 2:-

$h(A) = 40$	$h^*(A) = 27$	Not satisfied.
$h(B) = 40$	$h^*(B) = 17$	Not satisfied
$h(C) = 40$	$h^*(C) = 5$	Not satisfied.
$h(D) = 40$	$h^*(D) = 0$	
$h(E) = 40$	$h^*(E) = 25$	
$h(F) = 40$	$h^*(F) = 9$	

⇒ We need to modify $h(A), h(B), h(C), h(D), h(E), h(F)$ to make it heuristic admissible value.

Miracle
Pg.No :
Date : / /201

→ Heuristic :- 3

$h(A) = 10$	$h^*(A) = 27$
$h(B) = 15$	$h^*(B) = 17$
$h(C) = 0$	$h^*(C) = 5$
$h(D) = 0$	$h^*(D) = 0$
$h(E) = 25$	$h^*(E) = 25$
$h(F) = 5$	$h^*(F) = 9$

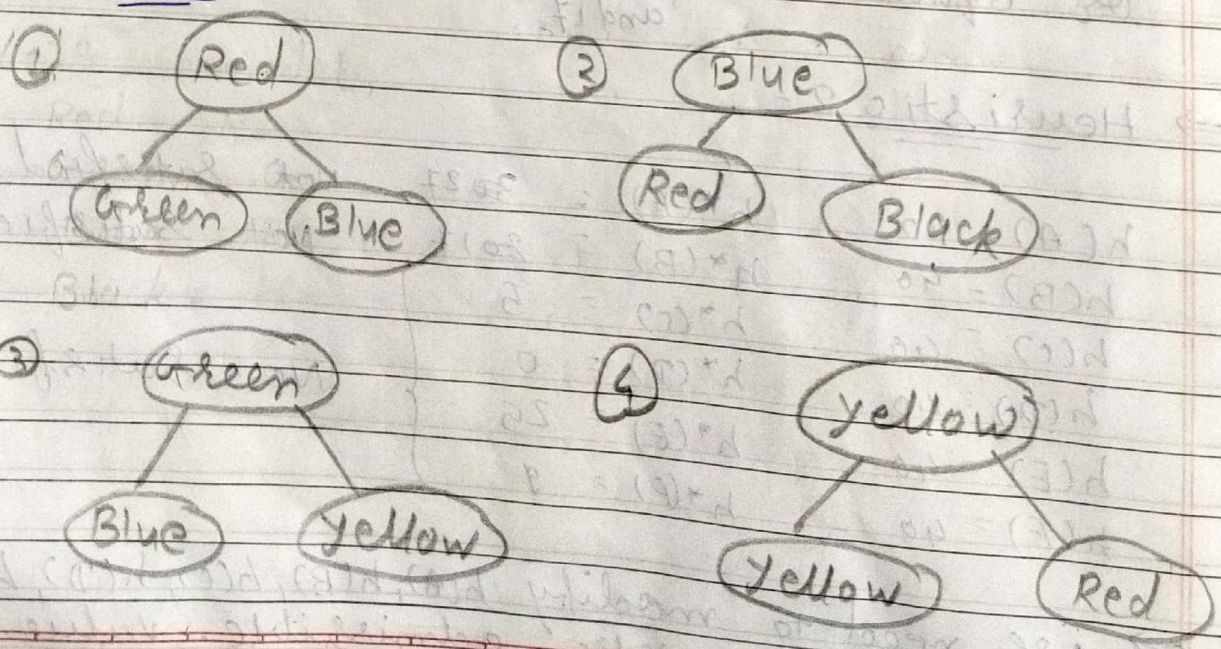
It is admissible heuristic value.

→ Heuristic 4:-

$h(A) = 0$	$h^*(A) = 27$
$h(B) = 0$	$h^*(B) = 17$
$h(C) = 0$	$h^*(C) = 5$
$h(D) = 0$	$h^*(D) = 0$
$h(E) = 0$	$h^*(E) = 25$
$h(F) = 0$	$h^*(F) = 9$

It is admissible heuristic value.

* Task - 4



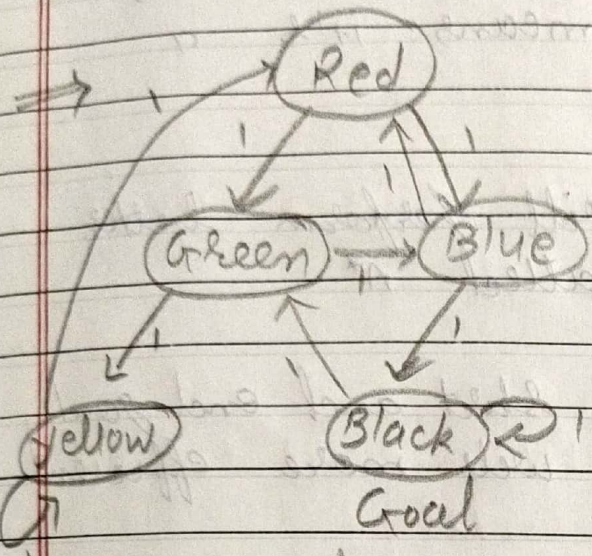
UTA ID - 1001982478

Miracle
Pg.No:
Date: / /201

(5) Black

Green Black

combine all states



Given 5 states colors, cost to move from one state to another state is 1

The goal is Black and the best admissible heuristics,

h(Red) = 2
h(Green) = 2
h(Blue) = 1
h(yellow) = 3
h(Black) = 0 (Goal)

* Task - 5

⇒ In this greedy search algorithm, h(n) is the Euclidean distance from n to the destination.

→ A* algorithm is f(n).

∴ f(n) = g(n) + h(n)

Miracle
Pg.No :
Date : / /201

→ where, h(n) is euclidian distance.
g(n) is minimum " between nodes.

⇒ For fig 4:- There are no broken links in this nodes. That means it's a continuous graph.

⇒ So, that greedy search will perform better than the other. It's called A*.

⇒ So, it's depending on the start and end goal. So, sometimes Greedy will more efficient than A*.

⇒ for example, (0,0) to (8,8) A* will visit more nodes than Greedy.

⇒ For fig 5:- There are some missing links between the nodes.

→ So, that, Greedy search performs sometimes better, sometimes worse and sometimes the same as A*.

→ for example,

① Greedy performs better than A*
(0,2) to (3,4) nodes.

② A* performs better than Greedy
(5,5) to (5,6) towns.

③ Performs Equal (Both)
(8,0) to (8,8) towns