Date
Page

Assignment 1. a

Name : Ruchita Suresh Rasal

Roll No : 53

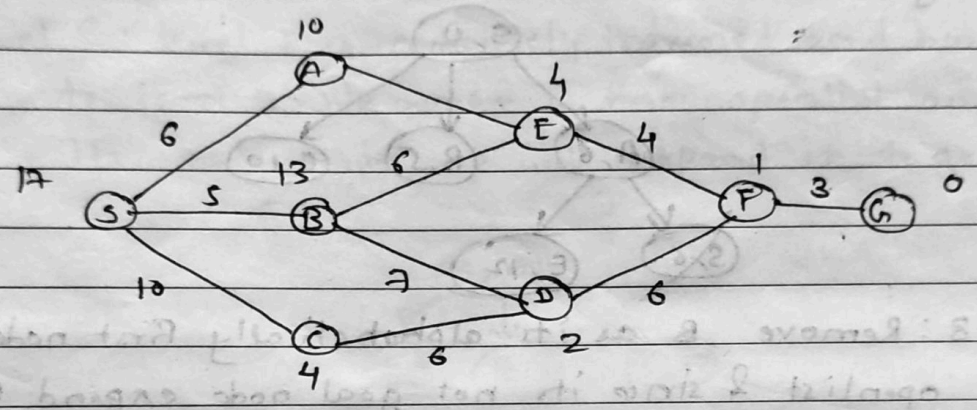
class : B. E. I. T.

Sem : VII

Sub : IS Lab

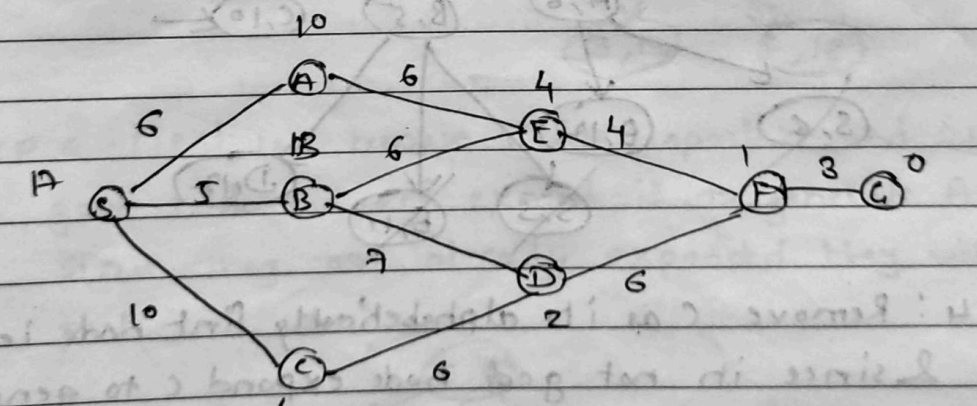
DoP	DoC	marks	sign

Q. 1. Consider following definition of state space for some arbitary problem. The number mentioned against the edges in cost to be incurred in moving from one node to other in any direction. The number in red font mentioned against the node is the heuristic function value.



Using above data solve following problems:

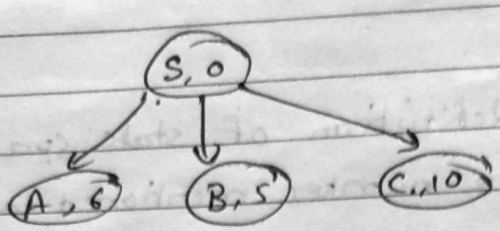
Q. 1.1. Apply BFS on above graph.



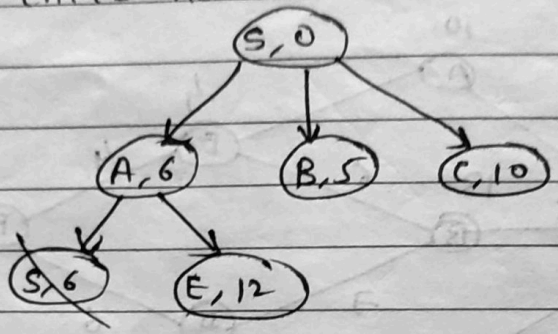
BFS we will show steps in Graph Search implementation of this algorithm on above problem.

Step 0 : Put initial node S into openlist path cost for it is 0.

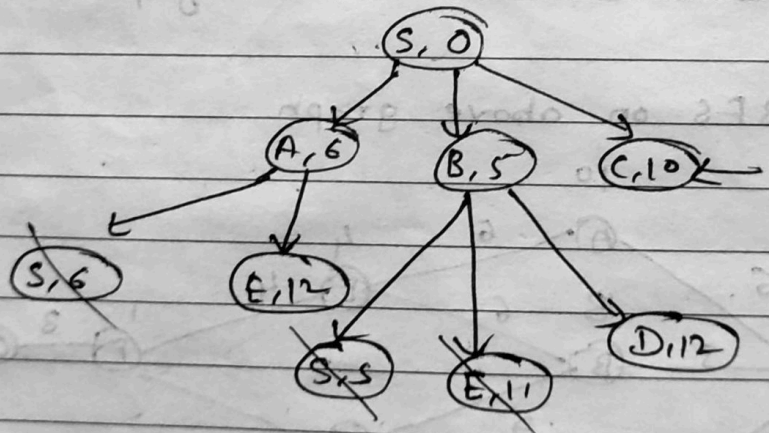
Step 1 : Remove S from openlist & since its not goal node expand it to generate its child nodes A, B, C.



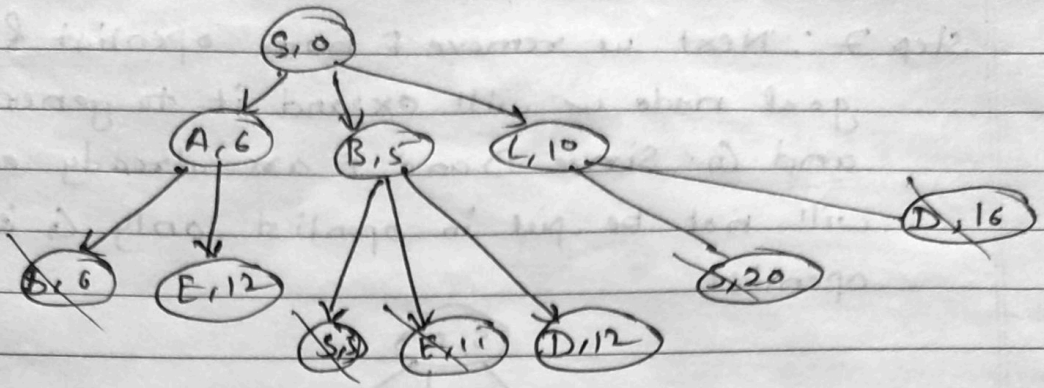
Step 2 : Remove A as its alphabetically first node in openlist and since its not goal node expand A to generate child node S and E



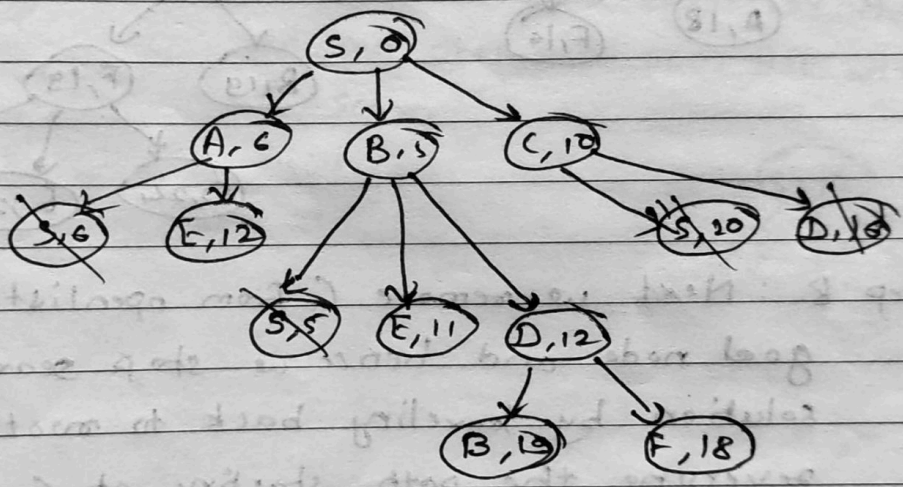
Step 3 : Remove B as its alphabetically first node in openlist & since its not goal node expand B to generate child nodes S, E and D.



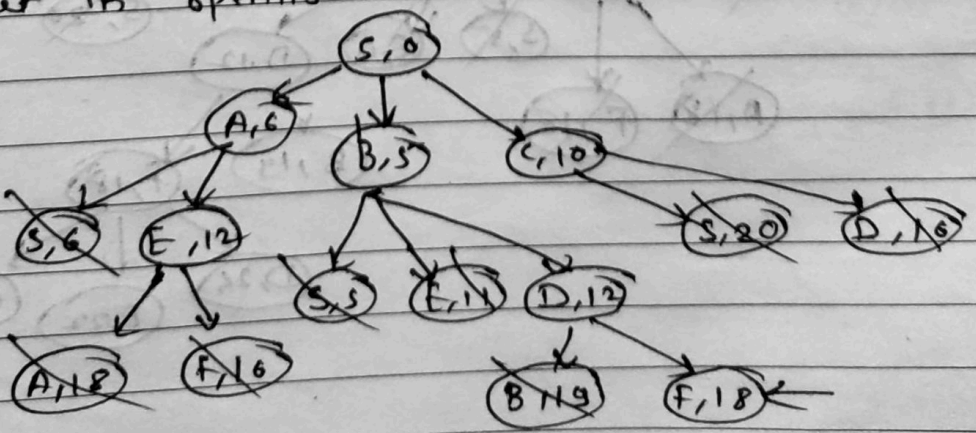
Step 4 : Remove C as its alphabetically first node in openlist & since its not goal node expand C to generate child nodes S and D. Since they are already in openlist dont put them in openlist



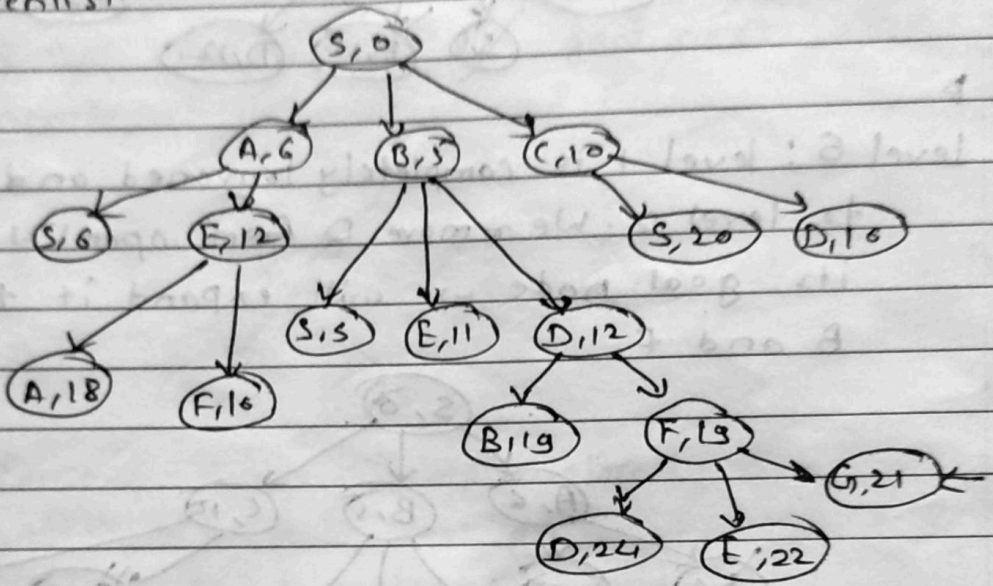
level 5 : level 1 is completely traversed and hence we move to level 2. We remove D from openlist and since its goal node we will expand it to generat B and F



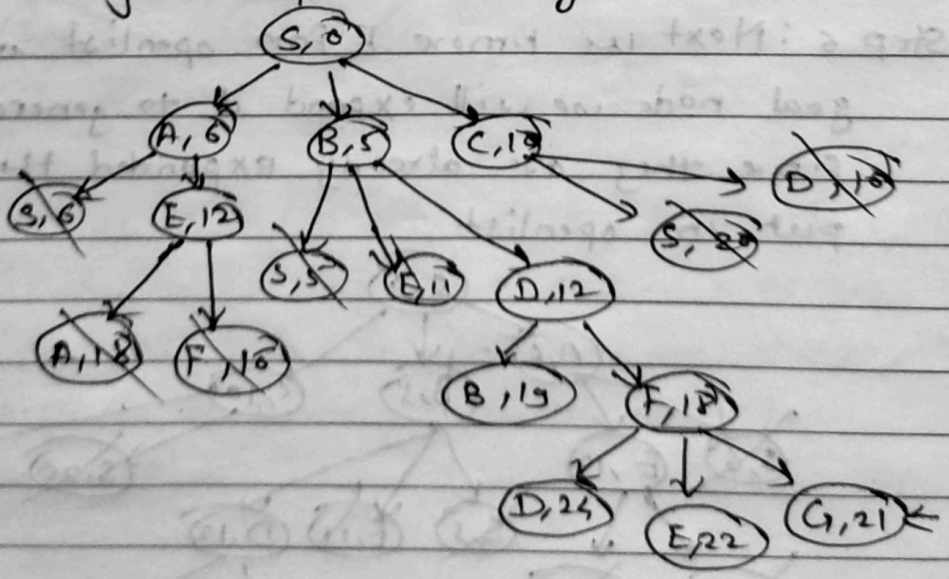
Step 6 : Next we remove E from openlist and since it goal node we will expand it to generate A and F since they are already expanded they will not be put in openlist



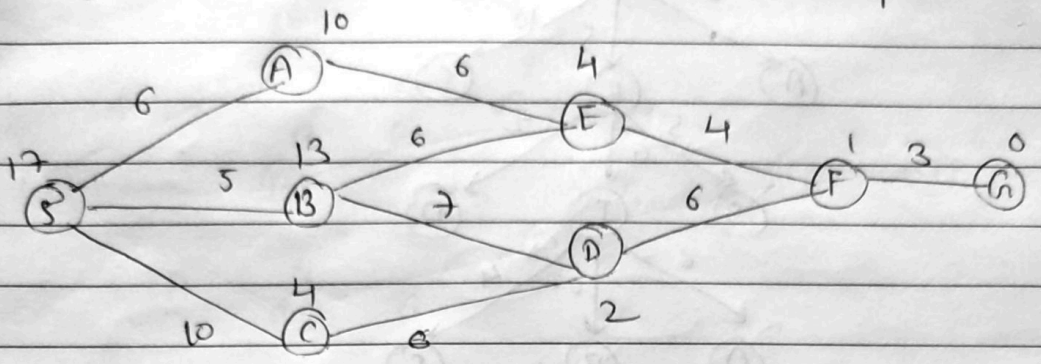
Step 7 : Next we remove F from openlist & since it goal node we will expand it to generate D & and G. Since D and E are already expanded they will not be put in openlist only G is send to openlist



Step 8 : Next we remove G from openlist, G is the goal node and hence we stop search and return solution by traveling back to root node and reversing the path starting at G.



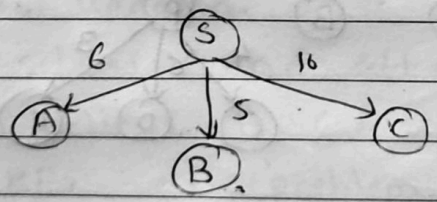
Q. 1.3 Apply uniform cost search on the problem above



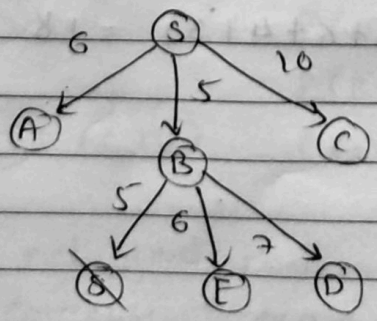
→ Here, we will maintain a priority queue the same as BFS with cost of path as its priority, lower the cost higher is the priority.

Step 1: Start node and check if we have reached any destination node, i.e. No thus continue

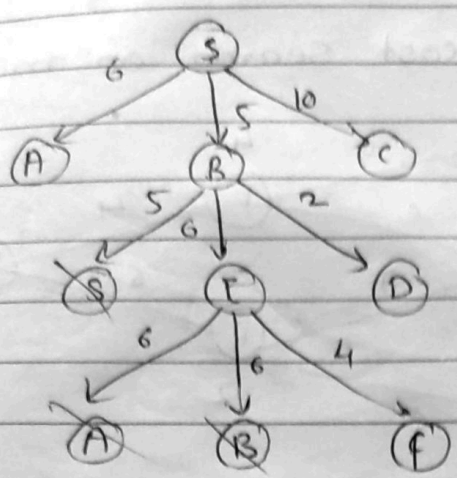
Step 2: From S node we can reached A, B & C select the cheapest path first & further expand ie. B



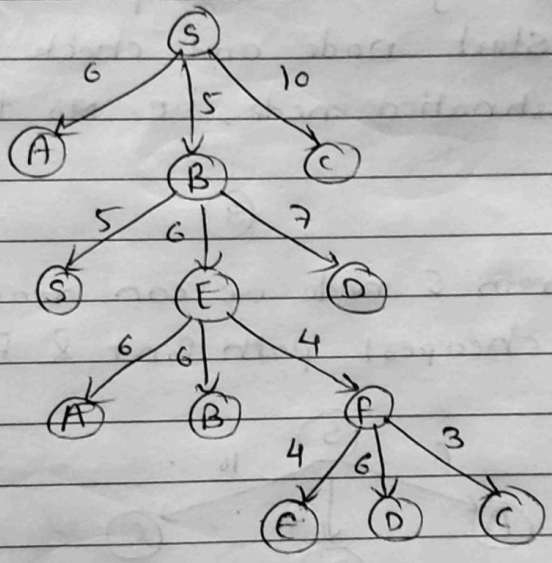
step 3: From B node we can reached S, E & D



Step 4: Select cheapest path for further expansion ie. E from E node we can reached A, B, F.

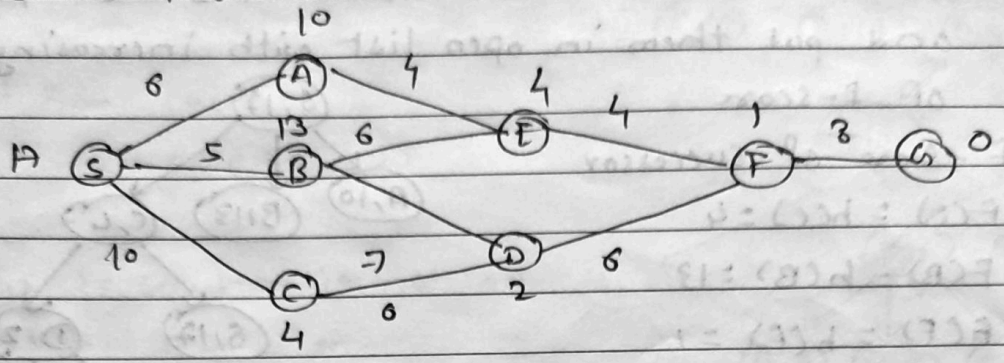


Step 5 : from f node we can reached E, D & F



Hence solution is S → B → E → F → G with path cost (5+6+4+3) = 18

Q.1.4. Apply Best first search & clearly show all steps using search tree.



Initialization : compute f-score for s and put it in the openlist

F-score of s : $f(s) = h(s) = 17$

S,17 ←

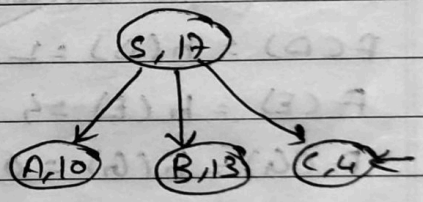
Step 1 : Remove s from openlist and since its not goal node expand it. For each successor A, B, C compute f-score and put them in open list with increasing order of f-score.

F-score of Successors

$f(A) = h(A) = 10$

$f(B) = h(B) = 13$

$f(C) = h(C) = 10$

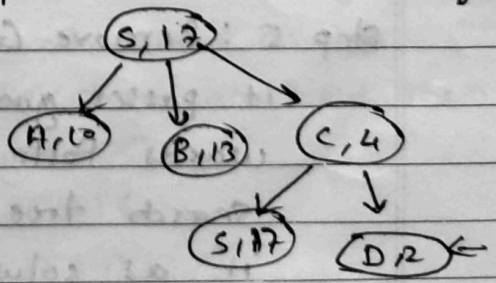


Step 2 : Remove C from openlist and since its not goal node expand it. For each successor S, D compute f-score and put them in open list with increasing order of f-score

F-score of successors

$f(S) = h(S) = 17$

$f(D) = h(D) = 2$



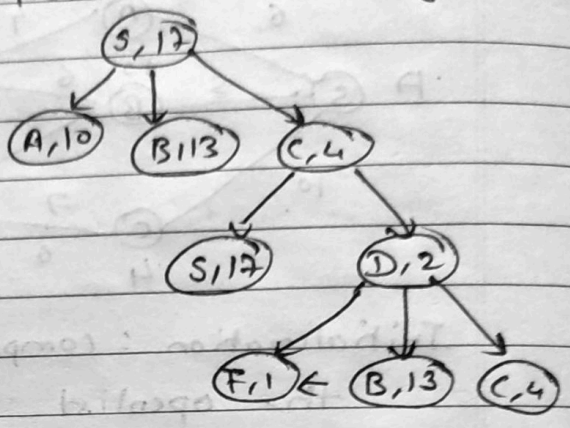
Step 3 : Remove D from openlist & since its goal node expand it. For each successor C, B if compute f-score and put them in open list with increasing order of f-score

F-score of successor

$F(C) = h(C) = 4$

$F(B) = h(B) = 13$

$F(F) = h(F) = 1$



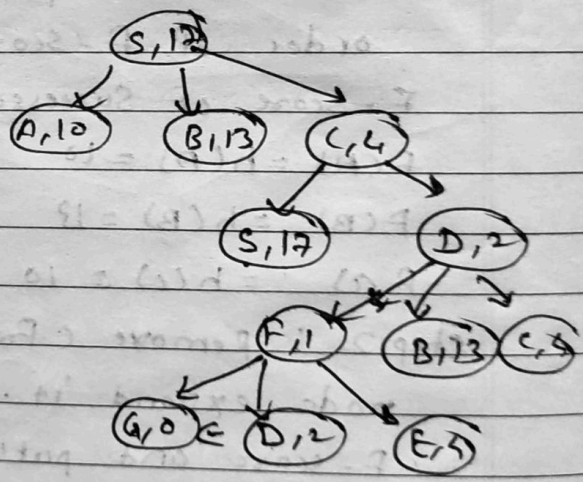
Step 4 : Remove F from openlist and since its not goal node expond it. For each successor D, E, G compute f-score and put them in openlist with increasing order of f-score

F-score of successor

$F(D) = h(D) = 2$

$F(E) = h(E) = 4$

$F(G) = h(G) = 0$



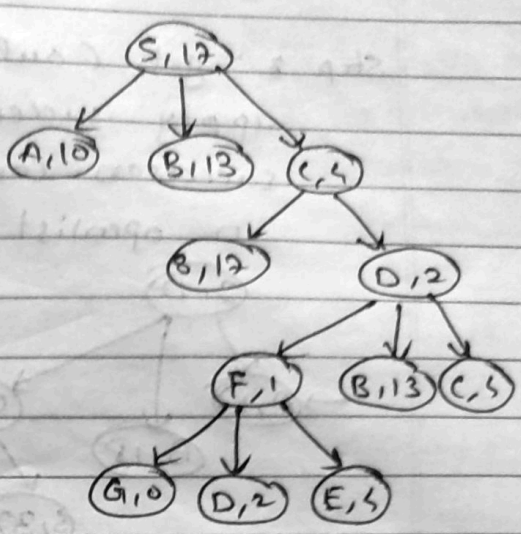
Step 5 : Remove G from openlist and since its goal node it passes goal test. stop search and traverse backwards following parent pointers to root node of search tree, reverse the path traversed return it as solution to problem.

Solution is

S → C → D → F → G with

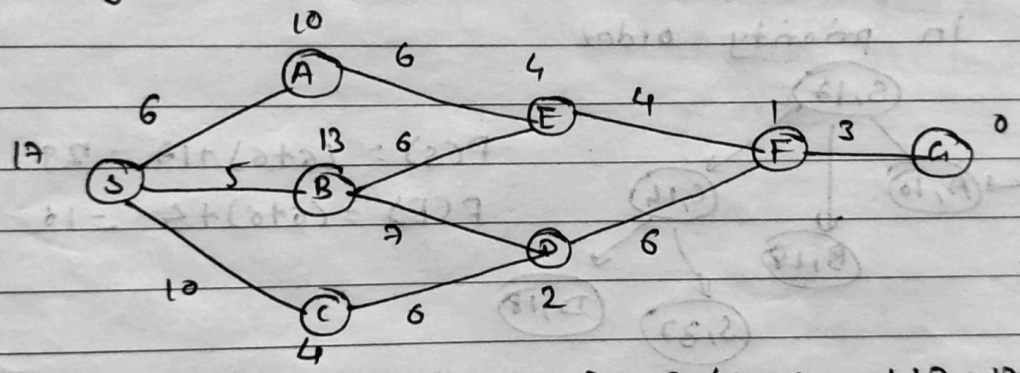
solution cost =

10 + 6 + 6 + 3 = 25



This is solution is not an optimal solution.

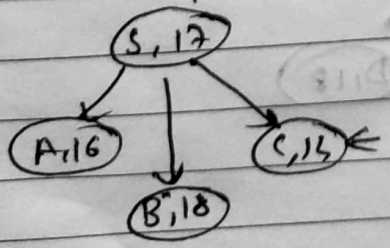
Q. 1.5 Apply A* algorithm and clearly show all the steps using search tree.



Step 1 : Calculate f-score for S (f(S) = 0+17 = 17) put S in open list open list

→ S, 17

Step 2 : get S out of openlist and since it not goal apply successor function to get A, B and C as its successor. compute f-score for them and put them in openlist in priority order

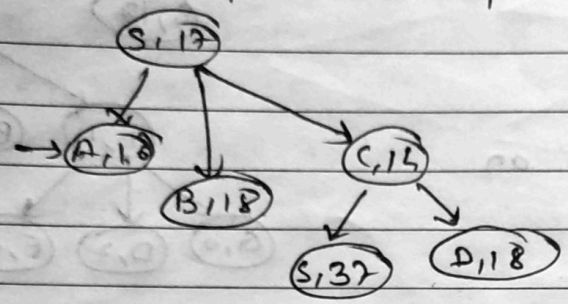


f(A) = 6 + 10 = 16

f(B) = 5 + 13 = 18

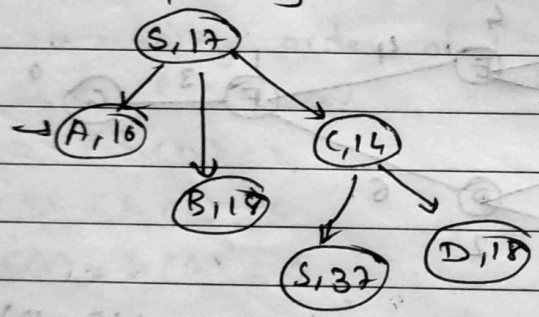
f(C) = 10 + 4 = 14

Step 3 : get C out of open list and since it not goal apply successor function to get S, D as its successor. Compute f-score for them and put them in openlist in priority order



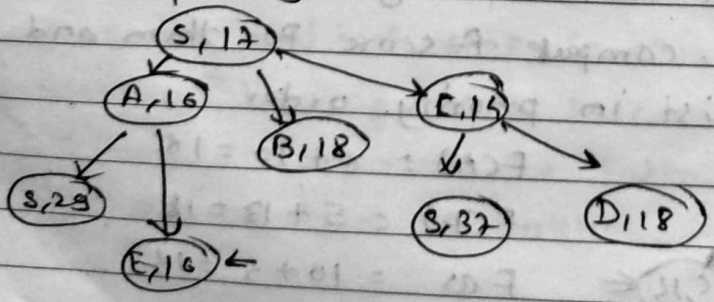
F(S) = (6+6) + 17 = 29
F(E) = (6+6) + 4 = 16

Step 4 : get A out of open list and since its not goal apply successor function to get S, D as its successor. Compute f-score for them and put them in openlist in priority order

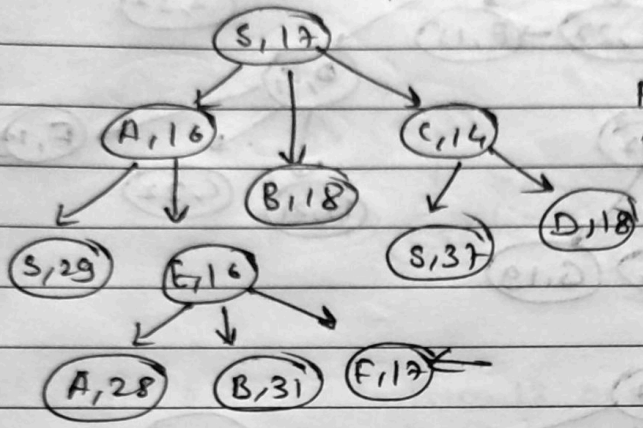


F(S) = (6+6) + 17 = 29
F(E) = (6+6) + 4 = 16

Step 5 : get A out of open list and since its goal apply successor function to get S, E as its successor compute f-score for them and put them in openlist in priority order

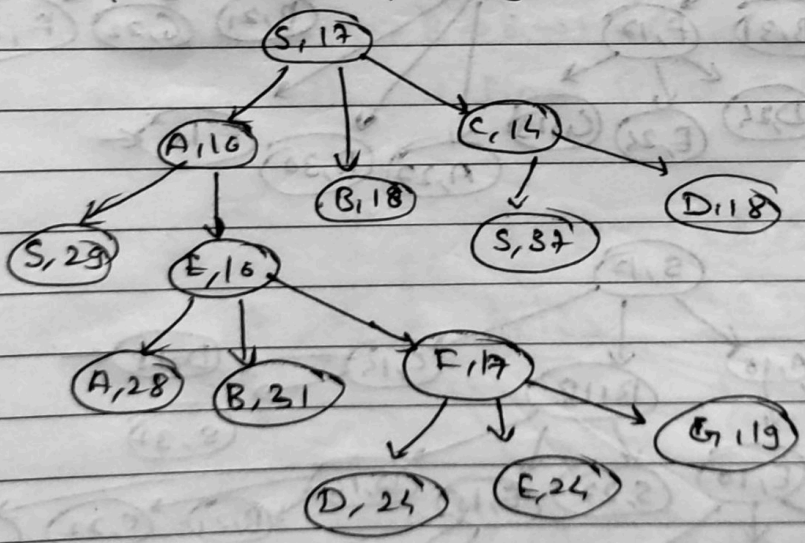


Step 6 : get E out of open list and since its not goal apply successor function to get A, B and F as its successors. Compute f-score for them & put them in openlist in priority orde

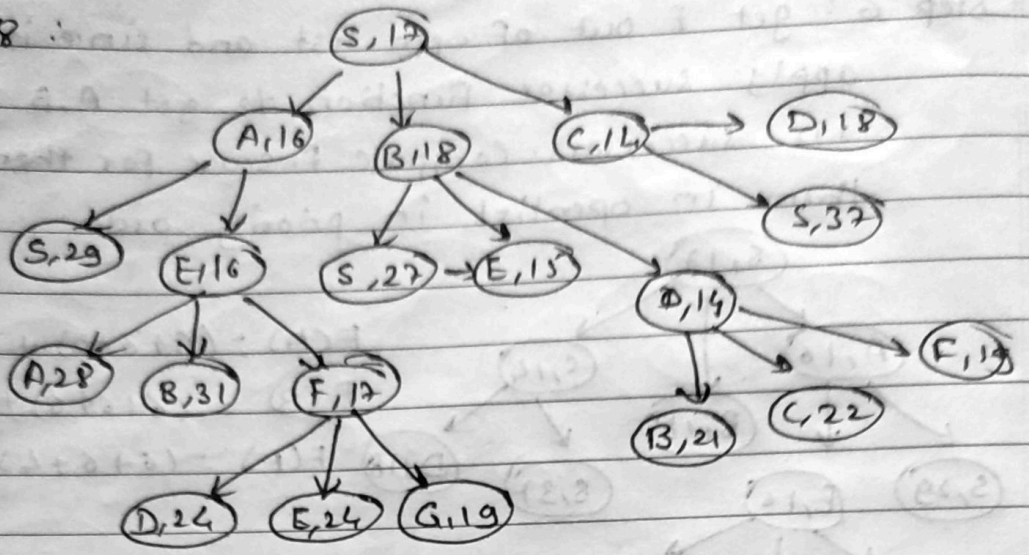


F(A) = (6+6+6) + 10 = 28
F(B) = (6+6+6) + 12 = 30
F(F) = (6+6+4) + 1 = 17

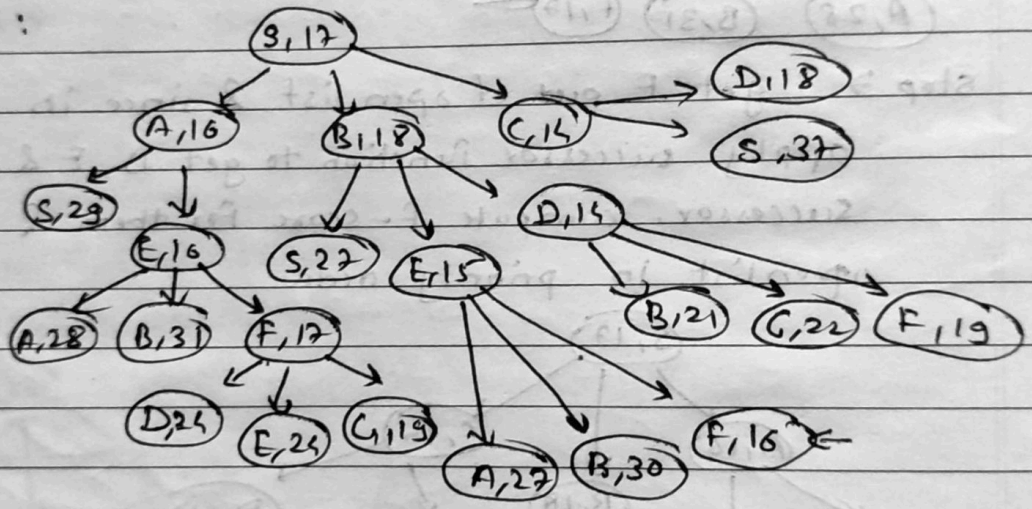
step 7 : get F out of open list & since it not goal apply successor function to get D, E & G as it successor. compute f-score for them & put in openlist in priority orde



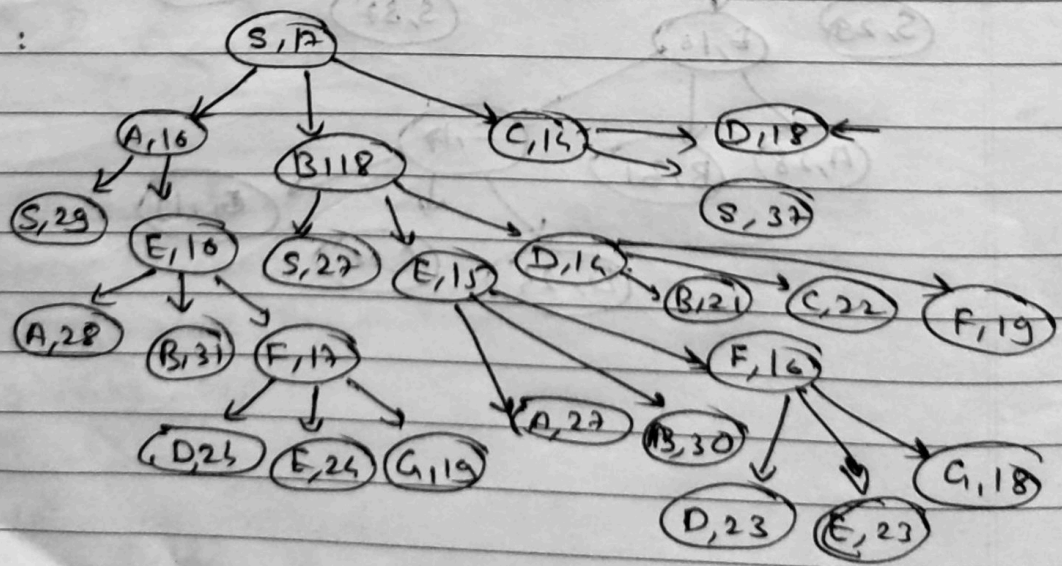
Step 8 :



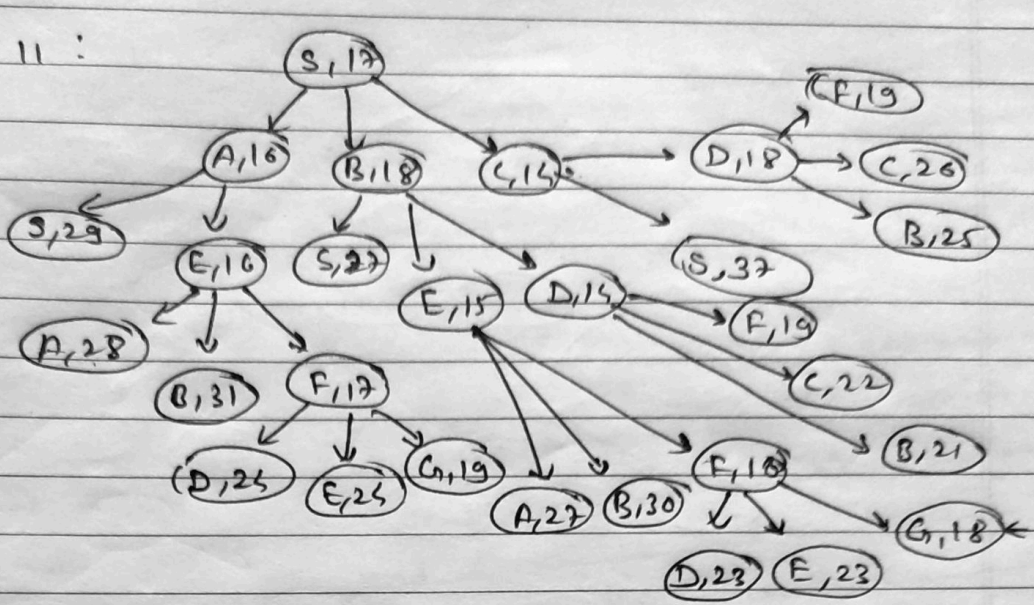
Step 9 :



Step 10 :

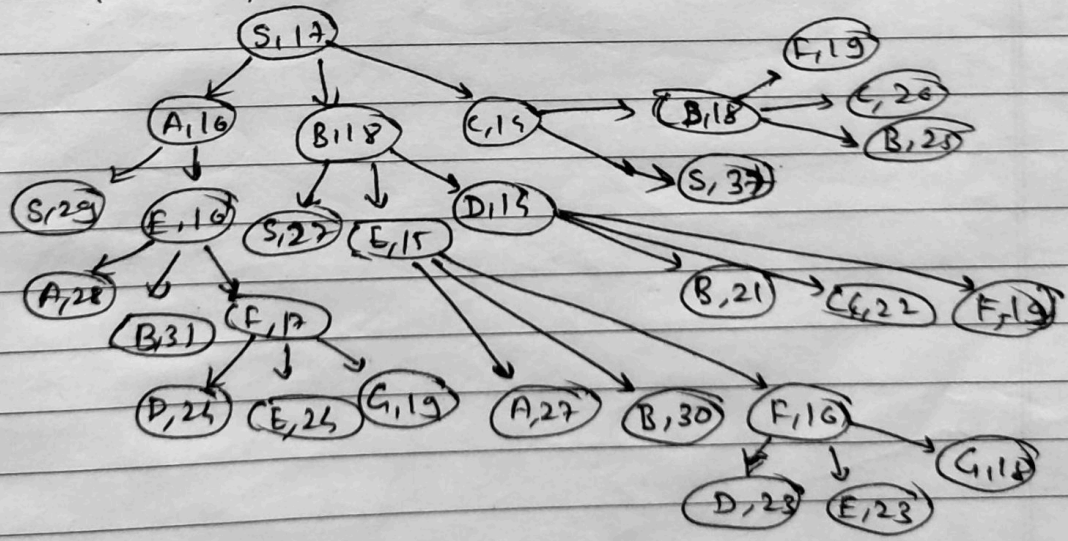


Step 11 :

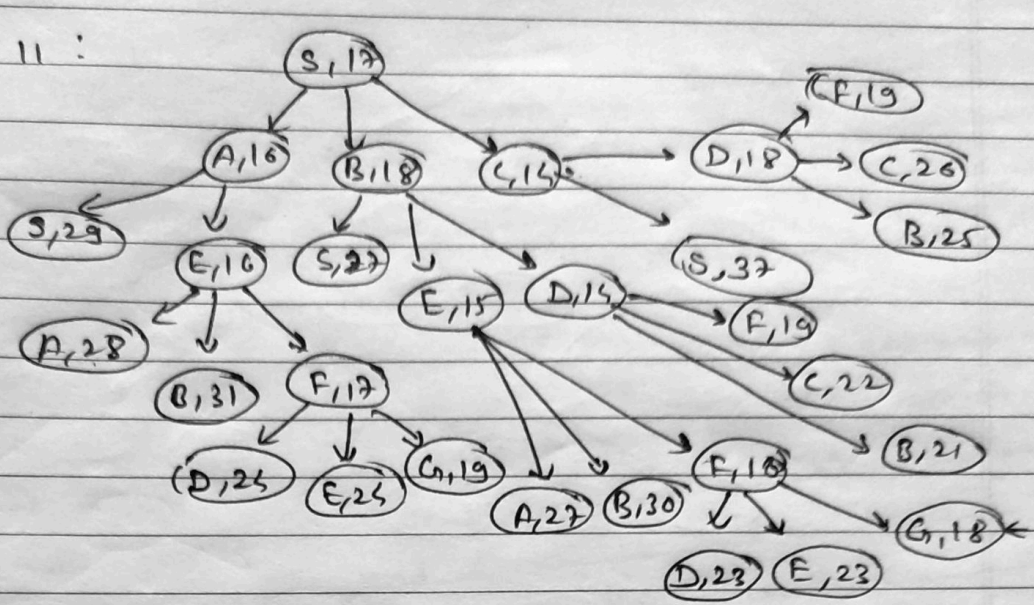


Step 12: Remove G, from 18 open list, since G is goal node goal test results in true and we will find solution & return it. We traverse backwards towards the root from G, 18 and reverse the path we traverse to get solution

Hence solution is S → B → E → F → G with path cost 18. This is optimal path cost Final search Tree After A* will be:

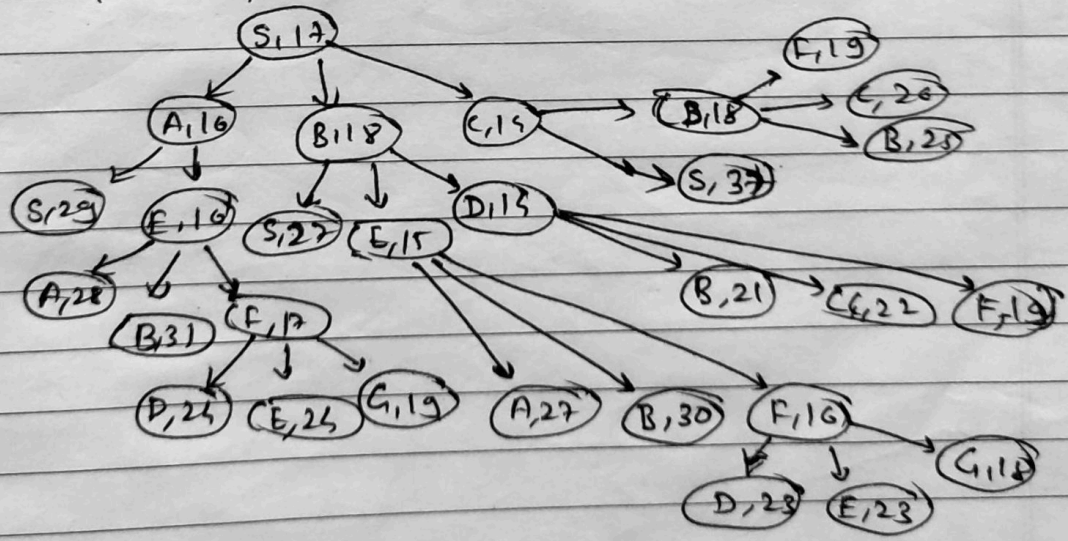


Step 11 :

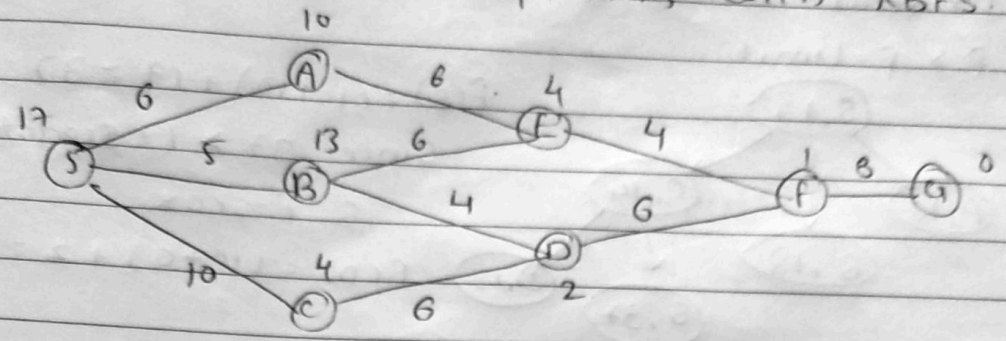


Step 12: Remove G, from 18 open list, since G is goal node goal test results in true and we will find solution & return it. We traverse backwards towards the root from G, 18 and reverse the path we traverse to get solution

Hence solution is S → B → E → F → G with path cost 18. This is optimal path cost Final search Tree After A* will be:



① Extra Question same problem with RBFS.



→ Step 0 : Call RBFS with S as initial node and f-limit as infinity.

Step 1 : RBFS (problem, S, infinity)

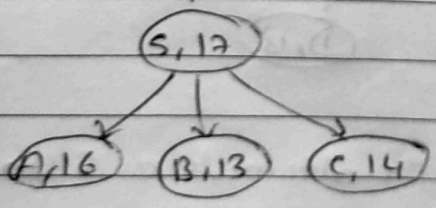
a) Success S is not goal node, ~~other~~ Generate its Successor A, B & C compute f-score for them.

b) if new f-score is > than earlier score (those visited earlier will have f-score computed or f-score initialised first time then f-score for that successor is updated.

c) Best is set as C and alternative As A

d) Update f-limit to C-f score = 14

e) Call RBFS with problem definition C and min (14, 16).



$f(A) = g(A) + h(A) = 16$

$f(B) = g(B) + h(B) = 18$

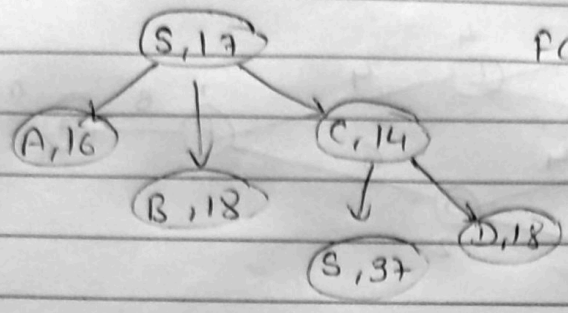
$f(C) = g(C) + h(C) = 14$

e.a) Since C is not goal node. Generate its successor S and D compute f-score for them.

e.b) Update f-score of S and D as 37 & 18 resp

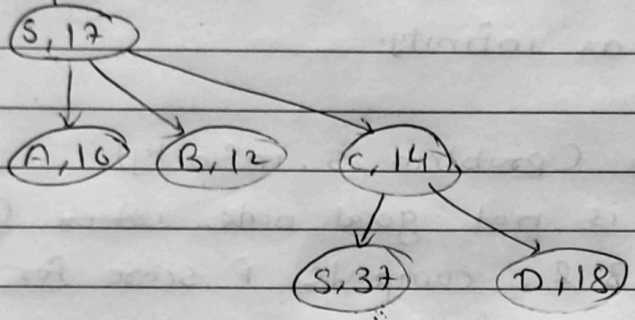
e.c) best is set as D and alternates as S.

e.d) return failure & best f = 18 since best f > f-limit

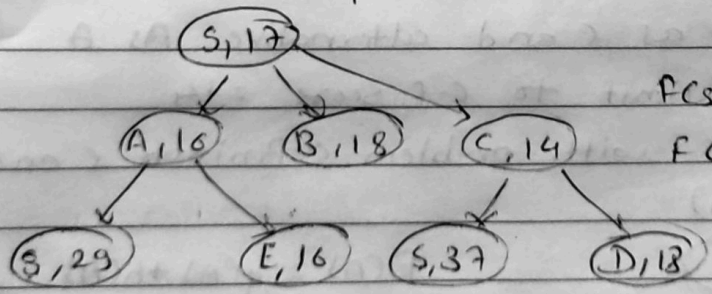


f(S) = (10+10) + 17 = 37
Since 37 > 17 update S f-score
f(D) = (10+6) + 2 = 18

f) update CF score as 18 returned from call RBFS with problem definition, C and min(14,16)



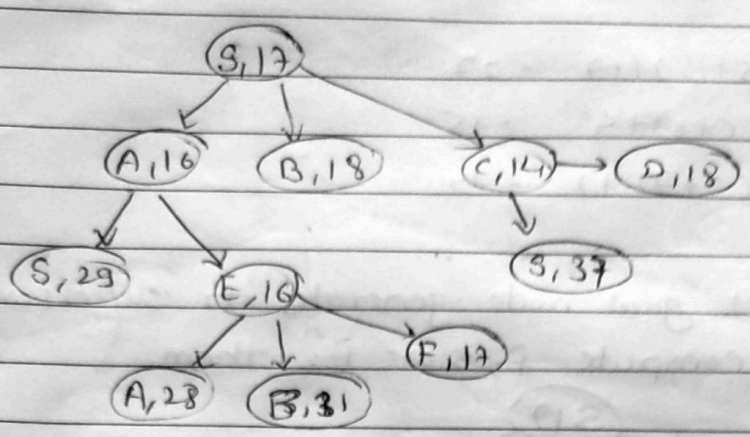
g) Now A is best node and Ca as alternate node. Since A is not goal node. Generate its successor S and E compute f-score for them.



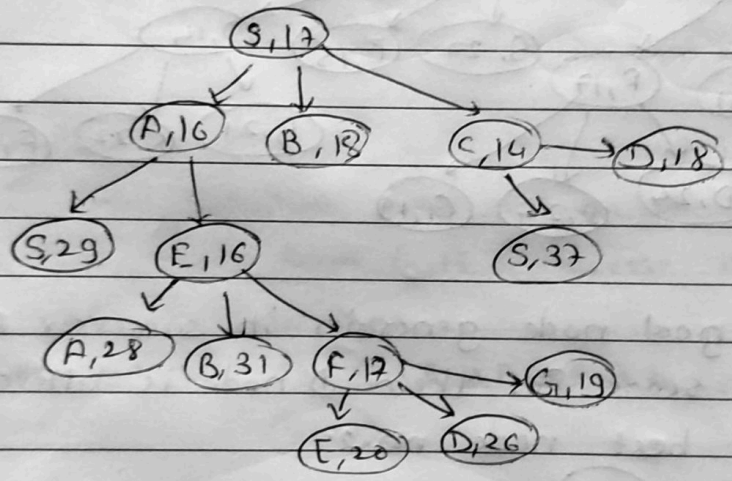
f(S) = (6+6) + 17 = 29
f(E) = (6+6) + 4 = 16

h) Now best is set as E & alternate as S since E is not goal node generates its successor A, B, F compute f-score for them.

f(A) = (6+6+6) + 10 = 28
f(B) = (6+6+6) + 13 = 31
f(F) = (6+6+4) + 1 = 17



i) Best set is F and alternate is F is not goal node. Generate its successor E, D & G compute f-score for them.

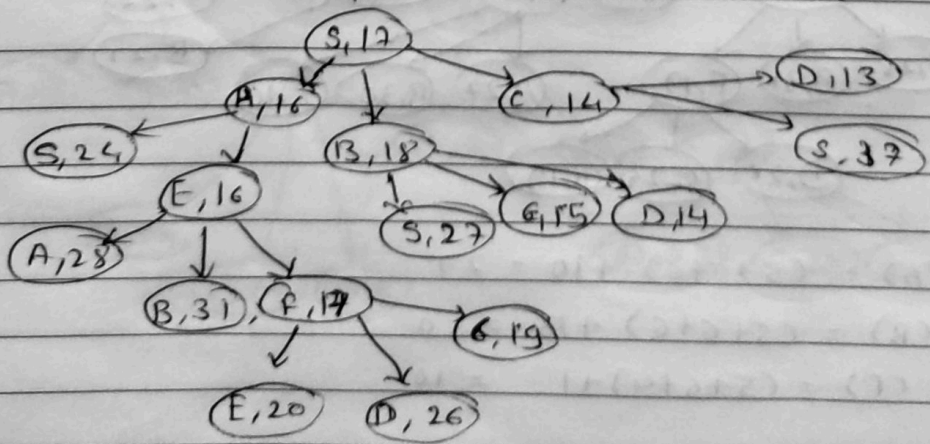


$f(G) = (6+6+4+3)+0$
$= 19$

$f(D) = (6+6+6+6)+2$
$= 26$

$f(E) = (4+4+4+4)+4$
$= 20$

j) Now B is Best node and A is alternate node since B is not a goal node generate its successor E, D, S & compute f-score for them

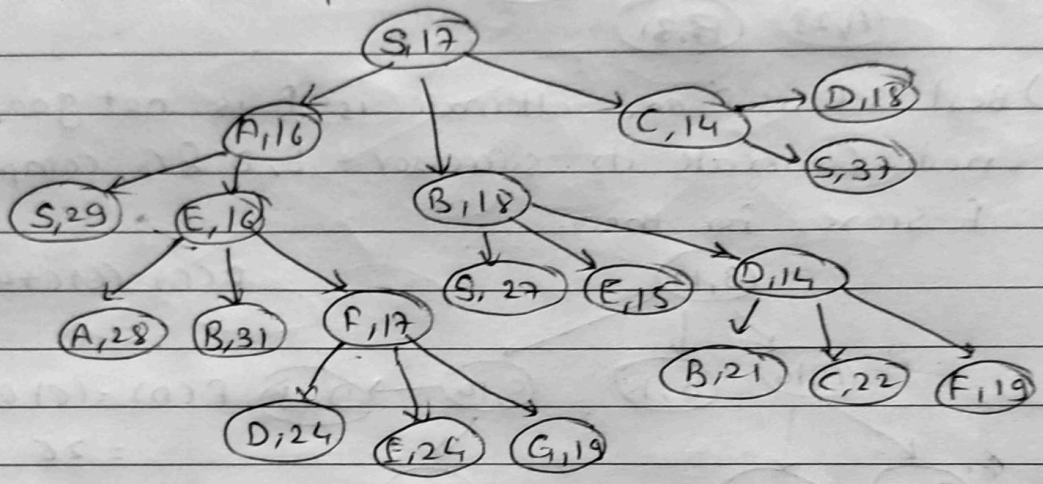


$f(s) = (5+5)+19 = 27$

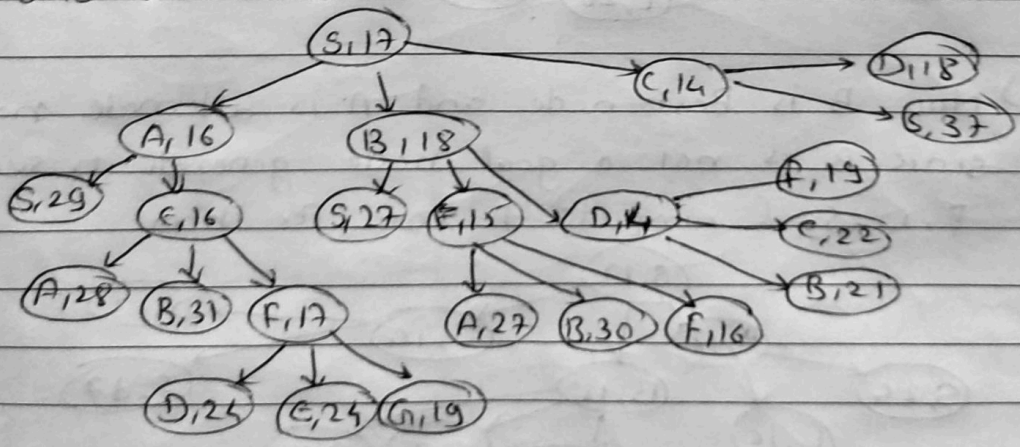
$f(E) = (6+6)+4 = 15$

$f(D) = (5+7)+2 = 14$

k) D is not goal node generates its successor B, C & f and compute f-score for them



L) E is not goal node generates its successor A, B, F compute f-score for them D node is alternate node and E is best node now.

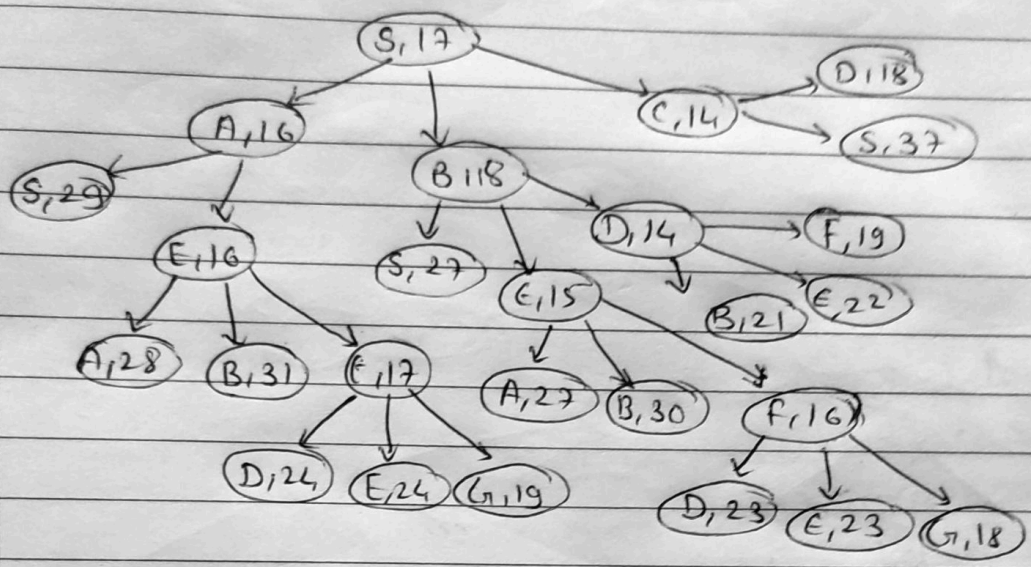


$f(A) = (5+6+6)+10 = 27$

$f(B) = (5+6+6)+13 = 30$

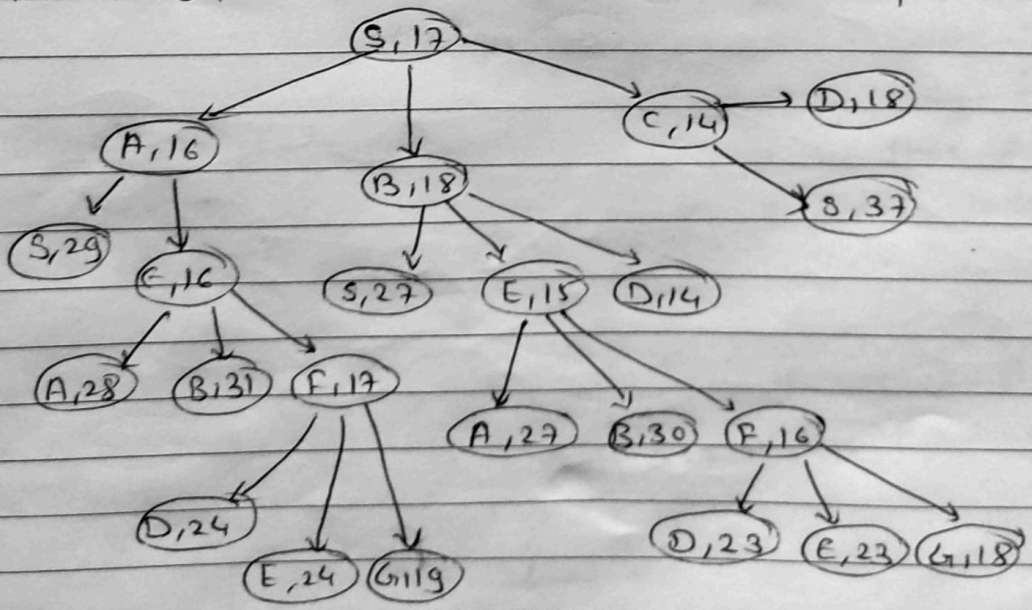
$f(F) = (5+6+4)+1 = 10$

m) F is not goal node. Generate its successor D, E, & G compute F-score of them.



Since G is goal node, goal test result in true & we will find the solution and return it we traverse backwards towards root from G,18 & reverse the path we traverse to get the solution

Hence solution is S → B → E → f → 4 with path cost 8.



Q.2. Consider following instance of 8 puzzle problem:

8	7	6
2	1	5
3	4	-

Initial configuration

-	8	7
2	1	6
3	4	5

Goal configuration

Consider Hueristic functions defined below:
h1 : Misplaced tiles count except space
h2 : Correctly placed tiles count except space
h3 : Sum of Manhattan distance between current and correct position of all times except space.

Answer the following questions:

a) In the 8 puzzle problm we are connected with getting to goal configuration within least number of steps. All moves are thus equally costly. Define g(n) in you own words. What will be the cost of 6 step solution to some arbitary 8 puzzle instance?

→

The lowest path cost g(n) can be the cost to search the goal configuration in least steps
In our case, we can reach the final configuration in at least 4 moves : UP, UP, LEFT, LEFT
Since all the moves are equally costly. we compute g(n) as

$g(n) = 1+1+1+1$

$g(n) = 4$

Consider the following arbitary 8 puzzle instance which gives solution in 6 steps

8	7	6
2	1	5
-	3	4

The solution can be represented as:

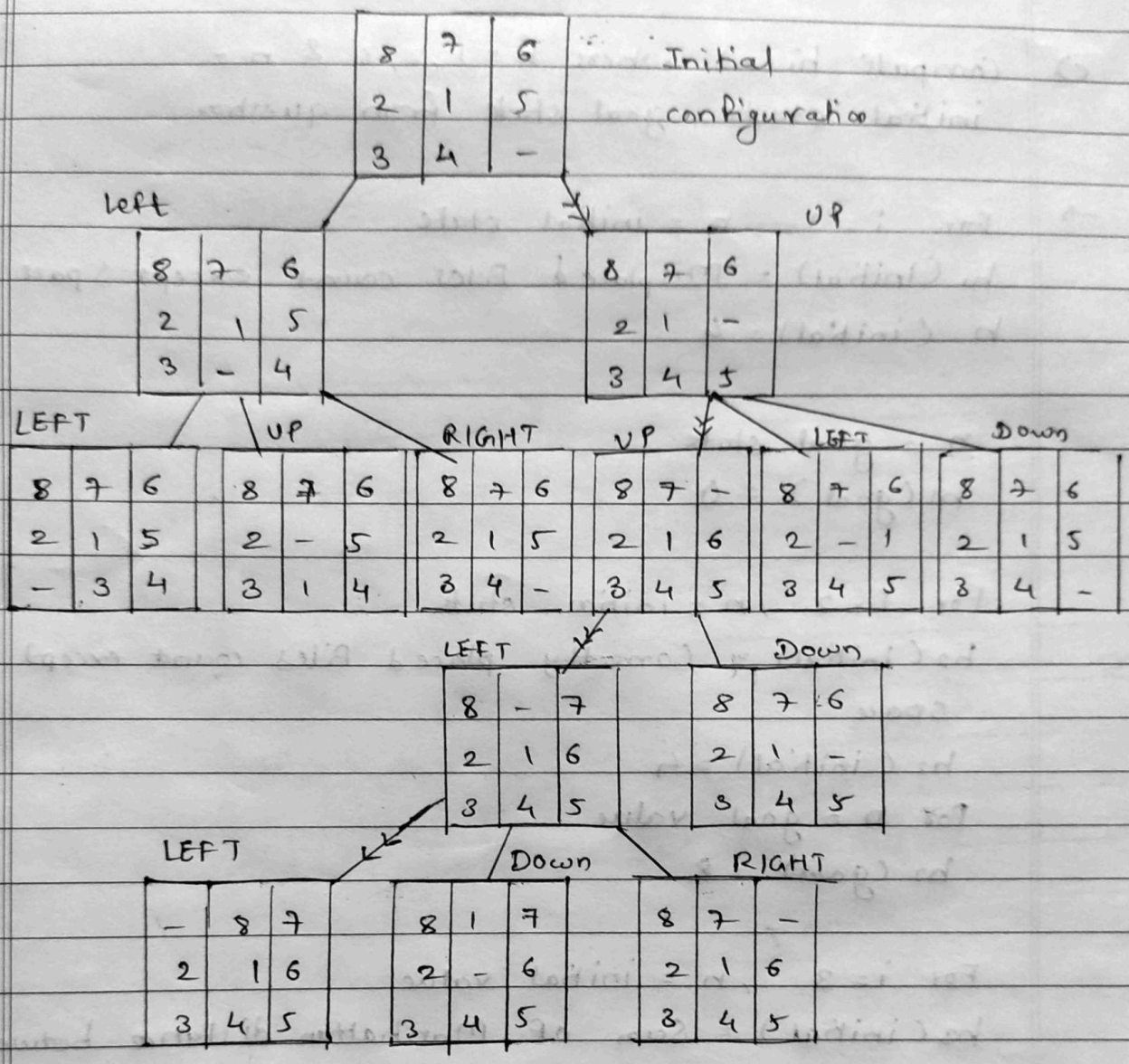
{{8,7,6}, {2,1,5}, {-,3,4}} → {{8,7,6}, {4,5}, {3,-,4}} → {{8,7,6}, {2,15}, {3,4,-}} → {{8,7,6}, {2,1,-}, {3,4,5}} → {{8,7,-}, {2,16}, {3,4,5}} → {{8,-,7}, {2,1,6}, {3,4,5}} → {{-,8,7}, {2,1,6}, {3,4,5}}

Since all the moves are equally costly, the cost would be

$$g(n) = 6$$

C. Draw exhaustive state space tree of depth limited to 4 for instance of 8 puzzle problem in the question.

Ans:



Final configuration

e) Compute $h_i(n)$ where $i = 1, 2, 3$ & $n =$ initial state, goal state from question

→ For $i = 1$, n = initial state

h_1(initial) = misplaced tiles count except space

h_1(initial) = 4

n = goal state

h_1(goal) = 0

For $i = 2$, n = initial state

h_2(initial) = correctly placed tiles count except space

h_2(initial) = 4

For n = goal value

h_2(goal) = 8

For $i = 3$, n = initial value

h_3(initial) = Sum of Manhattan distance between current and correct position of all tiles except space

h_3(initial) = 0 + 0 + 0 + 0 + 1 + 1 + 1 + 1

= 4

for n = goal state

h_3(goal) = 0