Date 1
Page

Assignment 1 Part A

Name :- Namra Javid Palte

Roll no :- 41

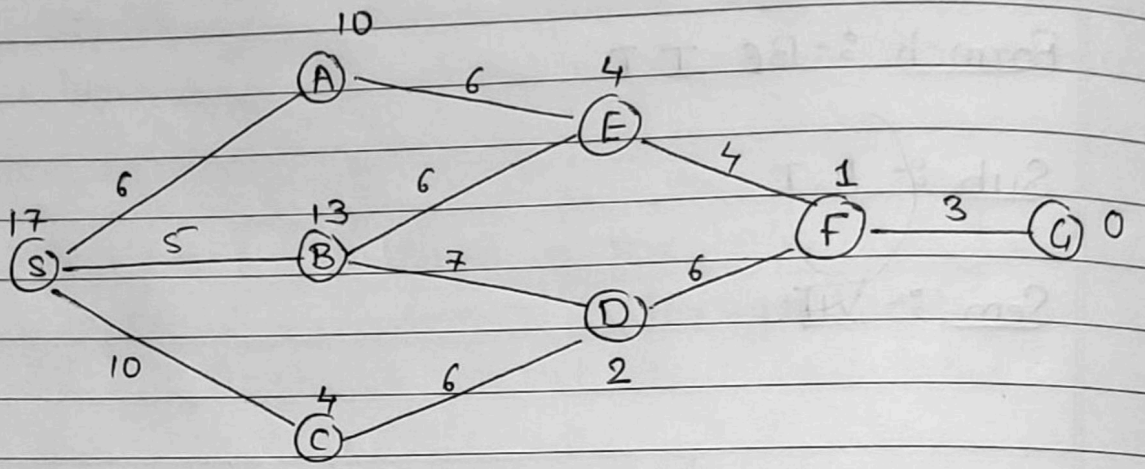
Branch :- BE I.T.

Sub :- A.I.

Sem :- VII

DOP	DOC	marks	Sign

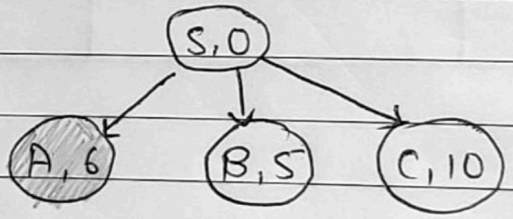
Q.1) Consider following definition of state space for some arbitrary problem. The no. mentioned against the edges is cost to be incurred in moving from one node to other in any direction. The no. in red font mentioned against the node is the hueristic function value.



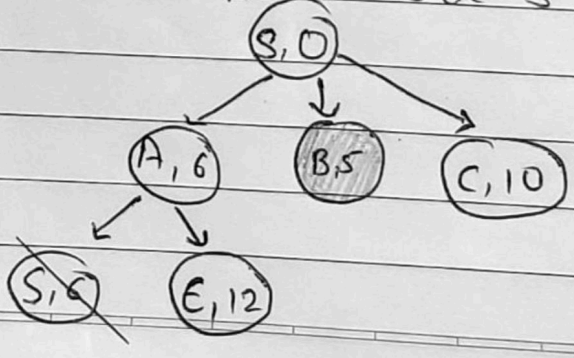
→ Apply BFS on above graph

BFS :- step 0 : put initial node S into openlist path cost for it is 0.

step 1 : Remove S from ~~into~~ openlist & since its not goal node expand it generate its child node A, B, C

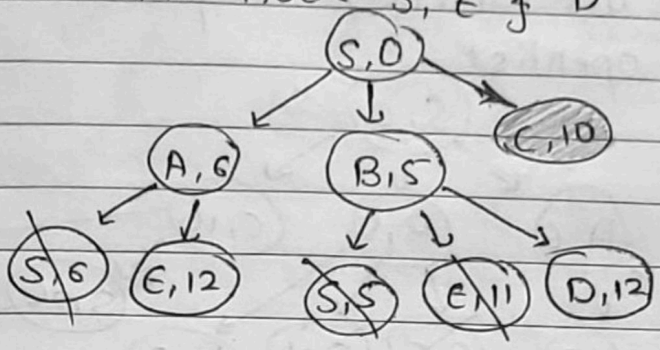


step 2 :- Remove A as its alphabetically first node in Openlist and since its not goal node expand A to generate child node S and E.

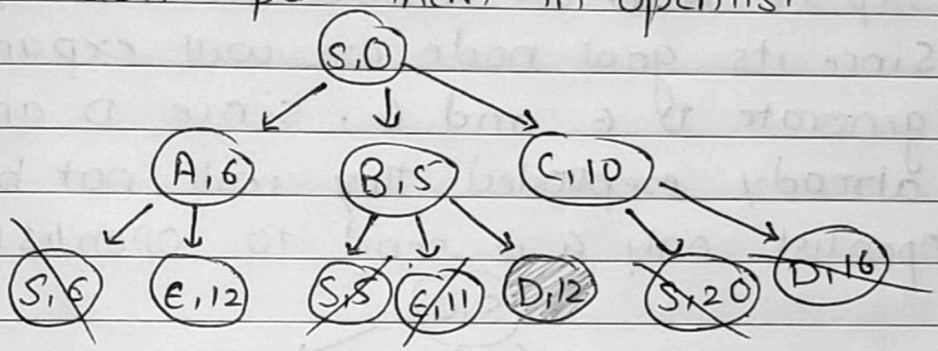


Date: 3
Page

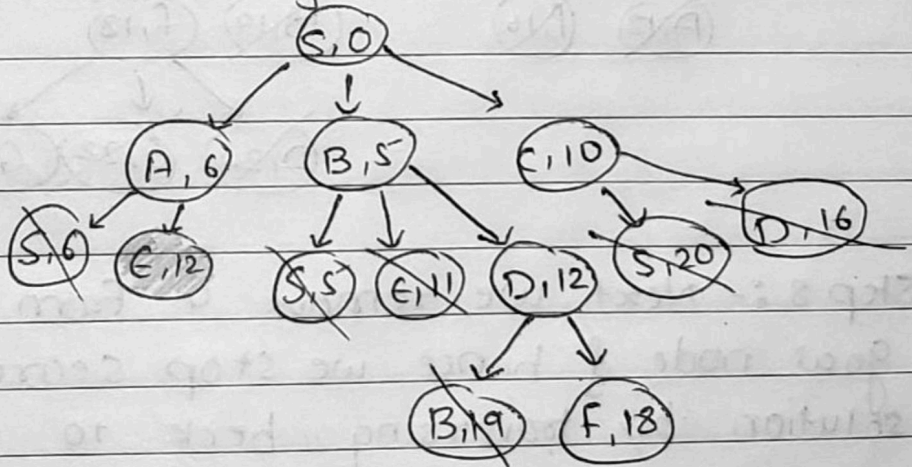
Step 3 :- Remove B as alphabetically first node in openlist & since it not goal node expand B to generate child node S, E & D



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

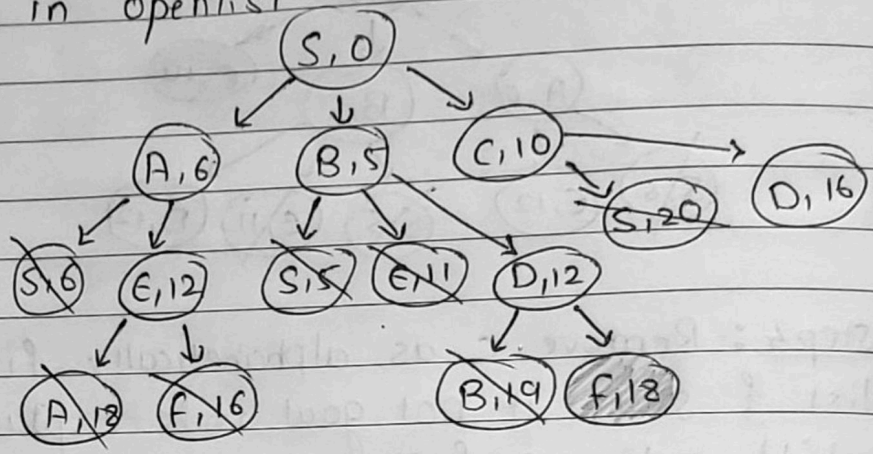


Step 5 :- Level 1 is completely traversed and hence we move to level 2. We remove D from openlist and openlist & since it goal node we will expand it to generate B & F.

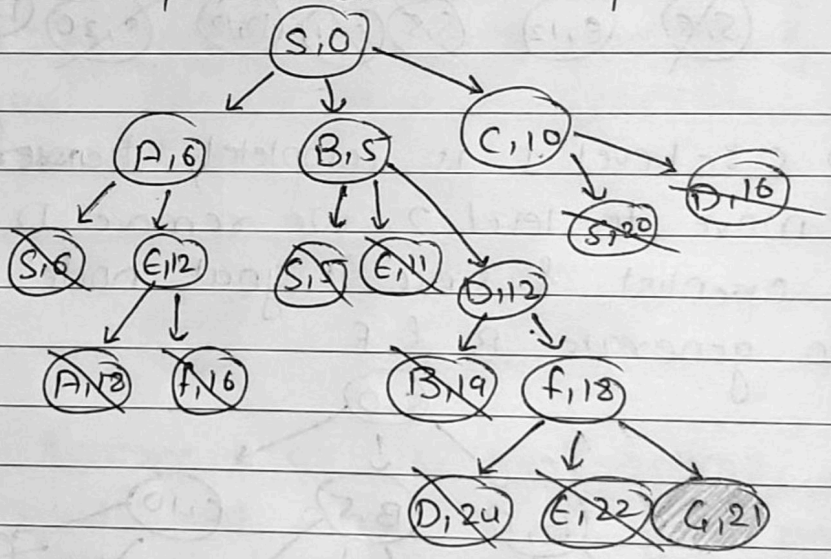


Date 4
Page

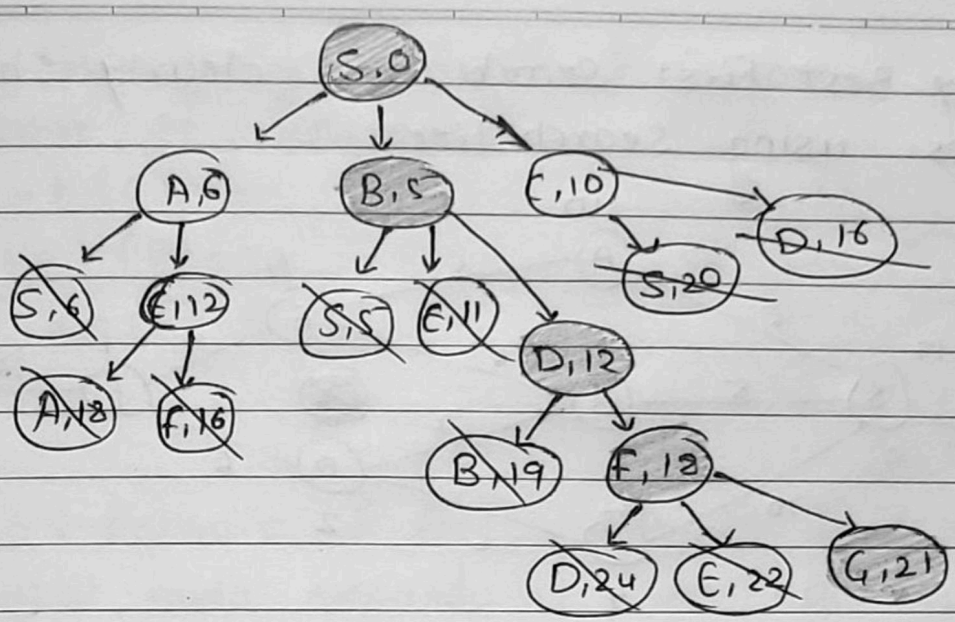
step 6 :- Next we remove E from openlist & since it goal node we will expand it to generate A & F. Since they are already expanded they will not be put in openlist



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

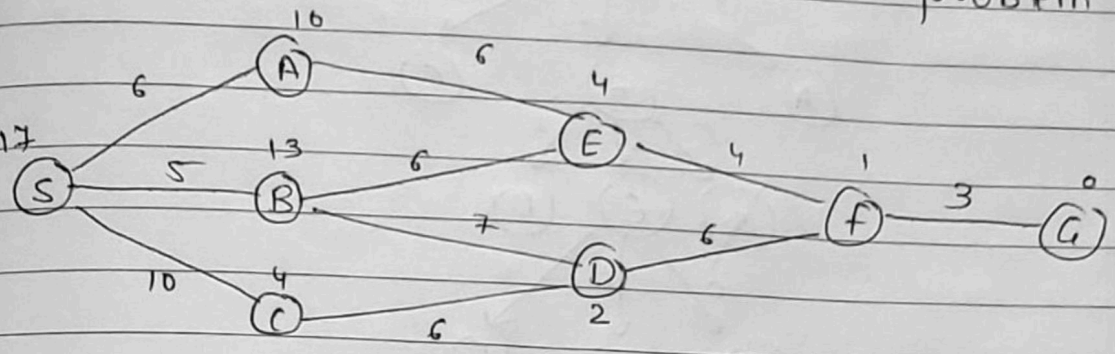


Step 8 :- Next we remove G from openlist. G is goal node & hence we stop search and return solution by traversing back to root node & reversing the path starting at G.



8.1.2) Apply Depth Limited DFS with I = 3. for this algorithm to be complete what should be the minimum value of I.

Q.13) Apply Uniform cost search on the problem above.

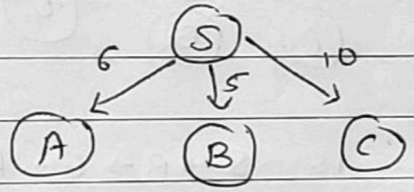


→ Here we will maintain a priority queue the same as BFS with cost of path as its priority, lower the cost higher is the priority.

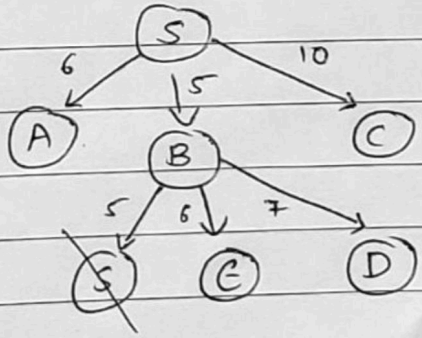
step 1 :- start node and check if we have reached any destination node, i.e. NO thus continue.

(S)

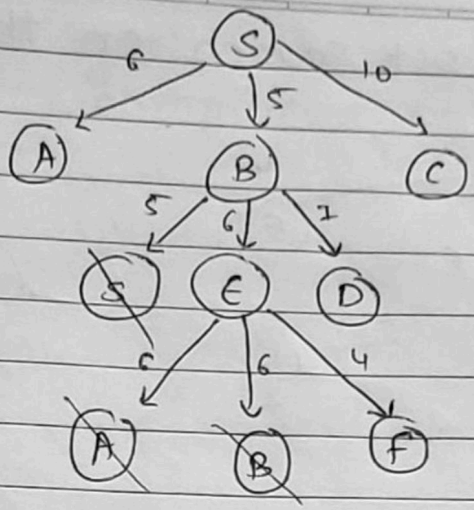
step 2 :- from S node we can reached A, B & C select the cheapest path first & further expand. i.e. B



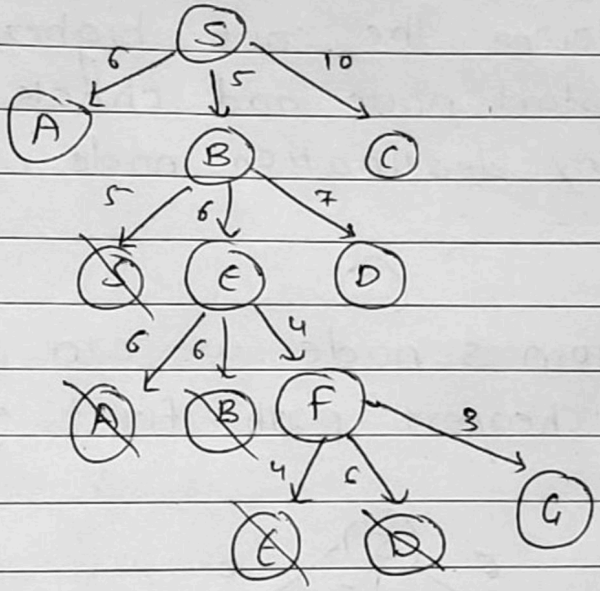
step 3 :- from B node we can reached S, E, & D



Step 4 :- Select the cheapest path for futher expansion i.e. E : from E node we can reached A, B, F

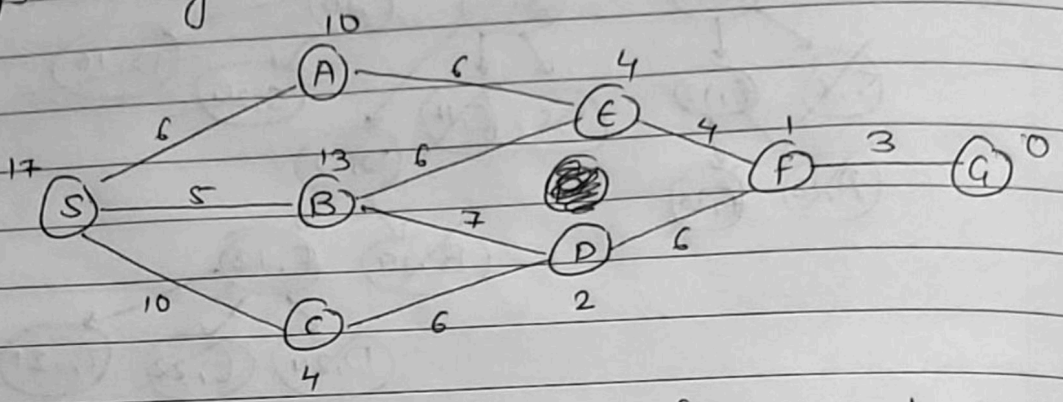


Step 5 :- From f node we can reached E, D & G



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

Q-1.4) Apply Best first search and clearly show all the steps using search tree.



→ Initialization: compute f-score for S and put it in the openlist.

f-score S : $f(s) = h(s) = 17$ (S, 17)

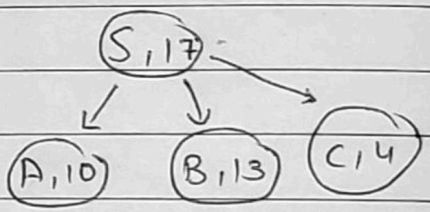
Step 1 :- Remove S from openlist and since its not goal node expand it. for each successor A, B, C compute f-score and put them in open list with increasing order of f-score.

f-score of successor

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

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

$f(C) = h(C) = 4$

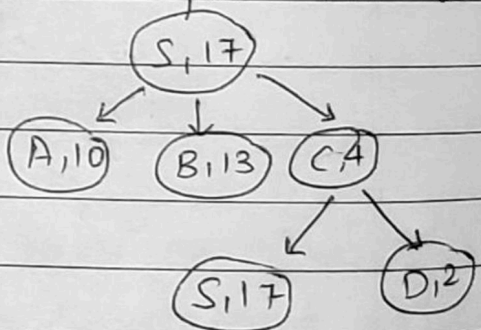


Step 2 : Remove C from openlist and since its not goal node expand it. for each successor S, D compute f-score and put them it open list with increasing order of f-score.

F-score of Successor

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

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



Step 3 : Remove D from openlist and since its not goal node expand it. for each successor C, B, F compute f-score and put them in open list

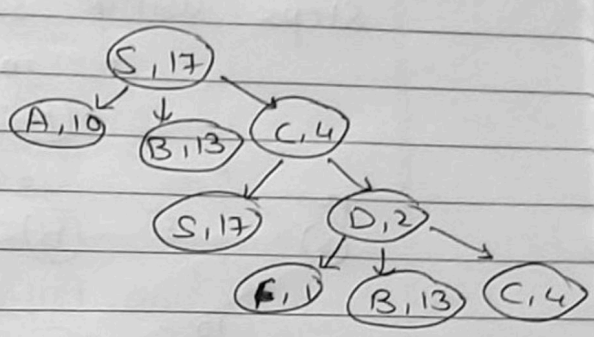
with increasing order of f-score.

f-score of successors

$f(C) = h(C) = 4$

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

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



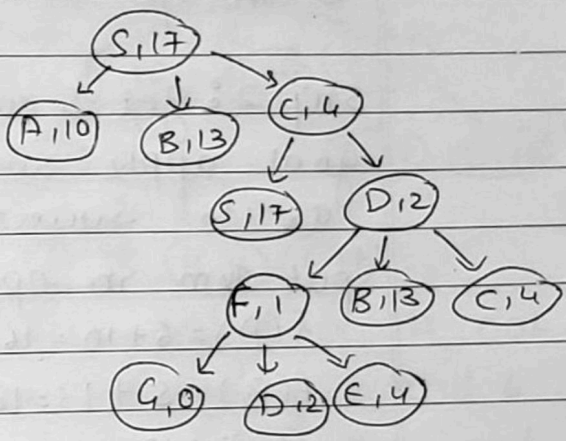
step 4:- Remove f from openlist and since is not goal node expand it for each successor D, E, G compute f-score and put them in open list with increasing order of f-score

f-score of successors

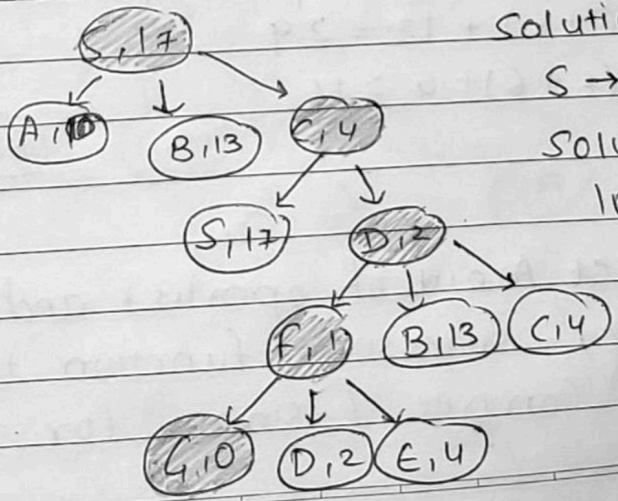
$f(D) = h(D) = 2$

$f(E) = h(E) = 4$

$f(G) = h(G) = 0$



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

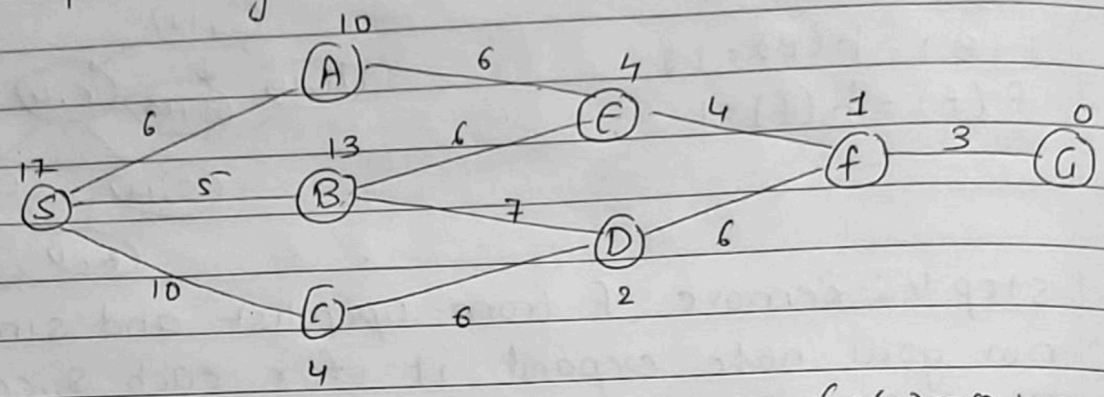


solution is

$S \rightarrow C \rightarrow D \rightarrow F \rightarrow G$ with solution cost = $10 + 6 + 6 + 3 = 25$

This is solution not an optimal solution.

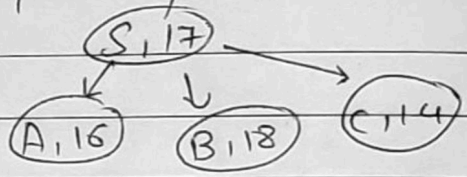
Q.1.5) Apply A* algorithm and clearly show all the steps using search tree.



→ step 1: calcwate f-score for S (f(S)=0+17=17) put S in openlist (S, 17)

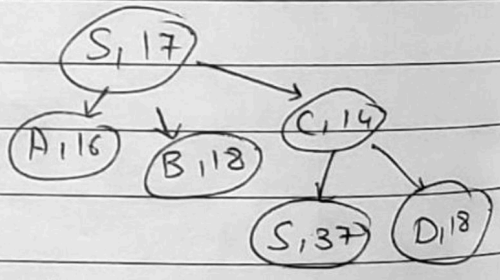
step 2: get S out of openlist and since its not goal apply successor function to get A, B, C as its successor. Compute f-score for them & put them in openlist in priority order.

f(A) = 6+10 = 16
f(B) = 5+13 = 18
f(C) = 10+4 = 14



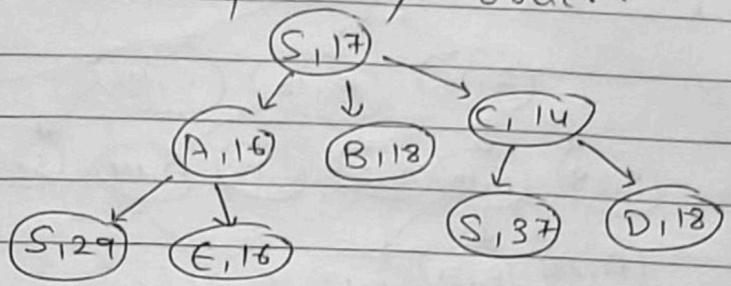
step 3:- get C out of open list & since its not goal apply successor function to get S, D as its successor. Compute f-score for them and put them in openlist in priority order.

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



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

in openlist in priority order.

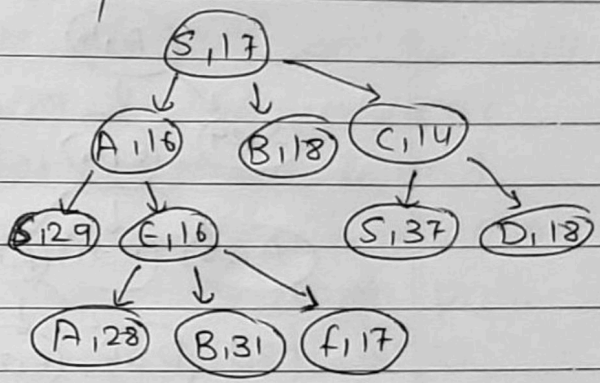


step 5 :- get E out of openlist and since its not goal apply successor function to get A, B and F as its successors. Compute f-score for them and put them in openlist in priority order.

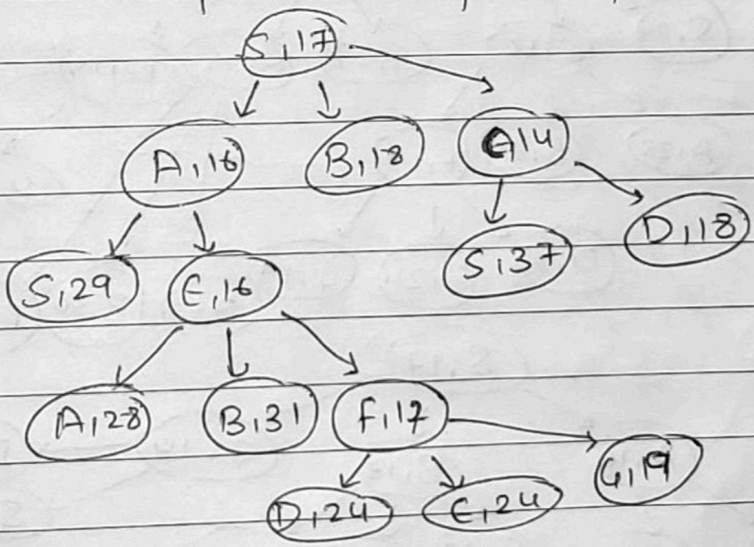
$f(A) = (6+6+6)+10 = 28$

$f(B) = (6+6+6)+13 = 31$

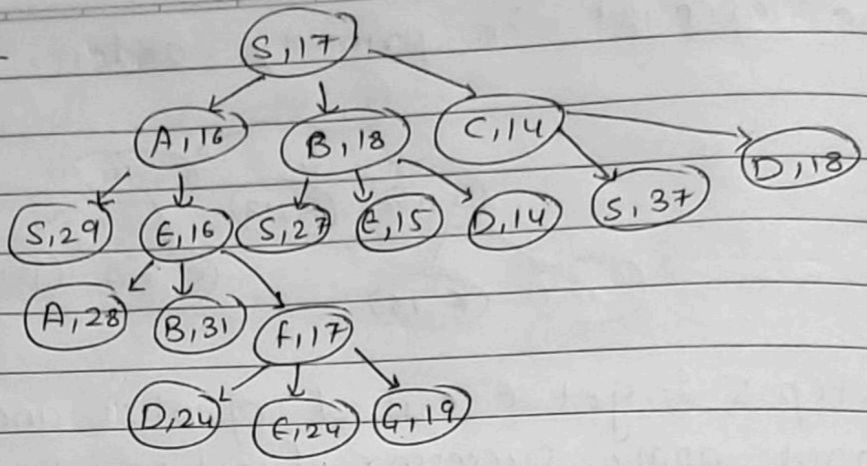
$f(F) = (6+6+4)+1 = 17$



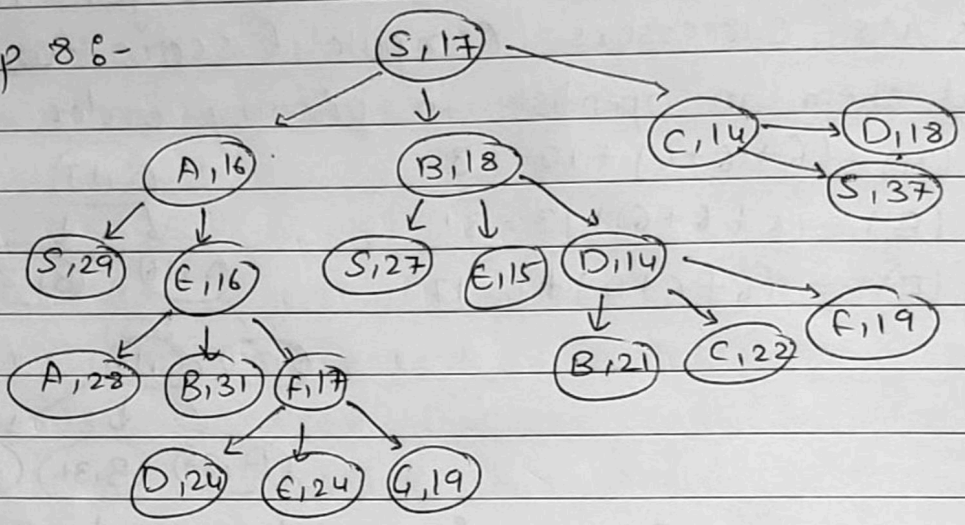
step 6 :- get f out of open list and since its not goal apply succesor function to get D, E and G as it successors. Compute f-score for them and put them in openlist in priority order.



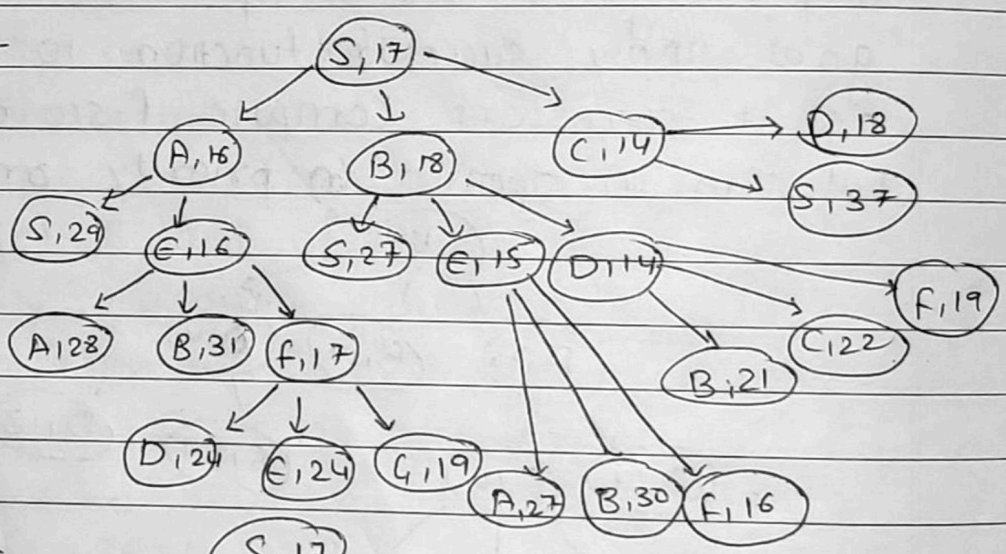
step 7 :-



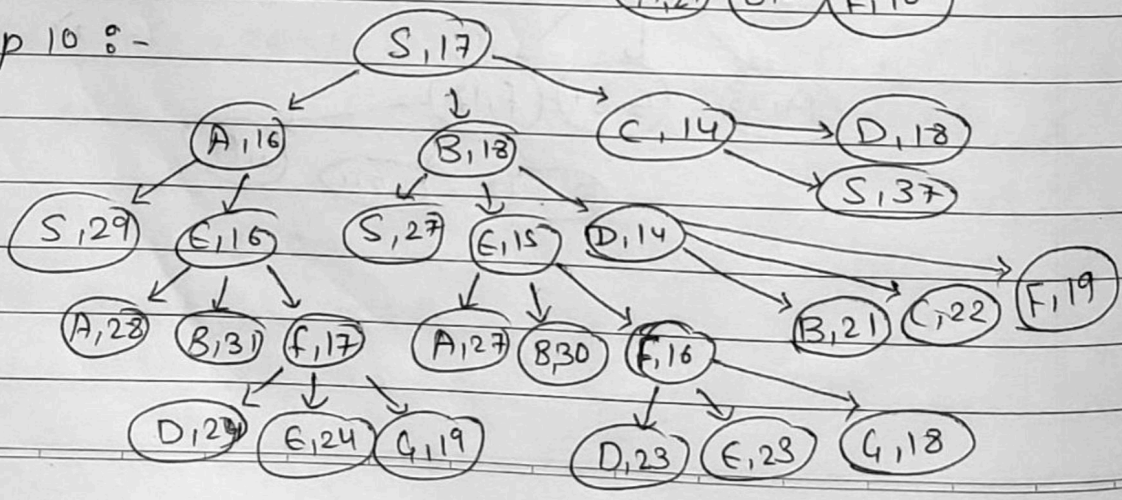
step 8 :-



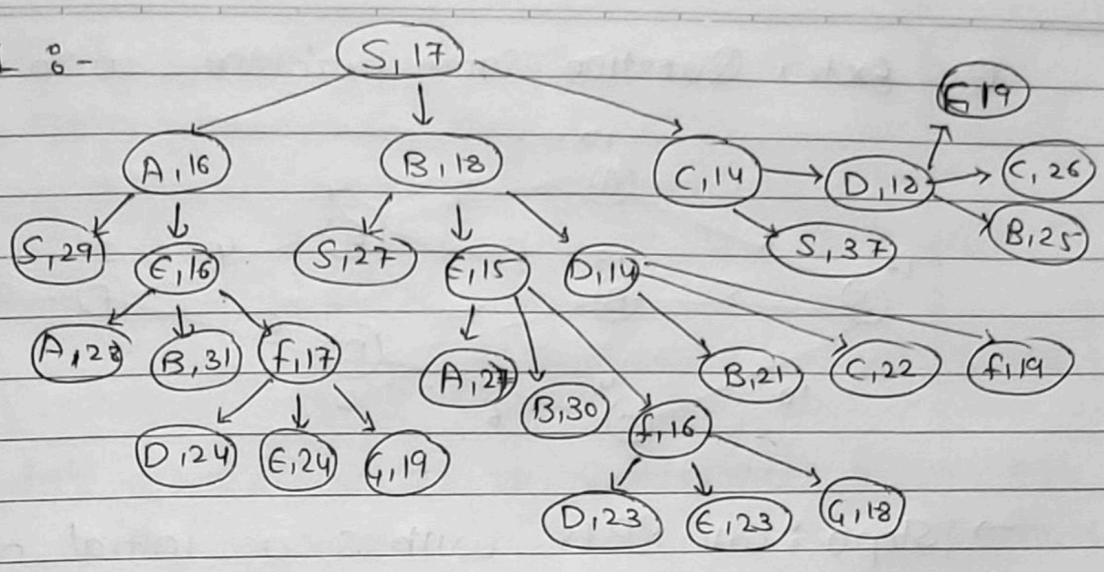
Step 9 :-



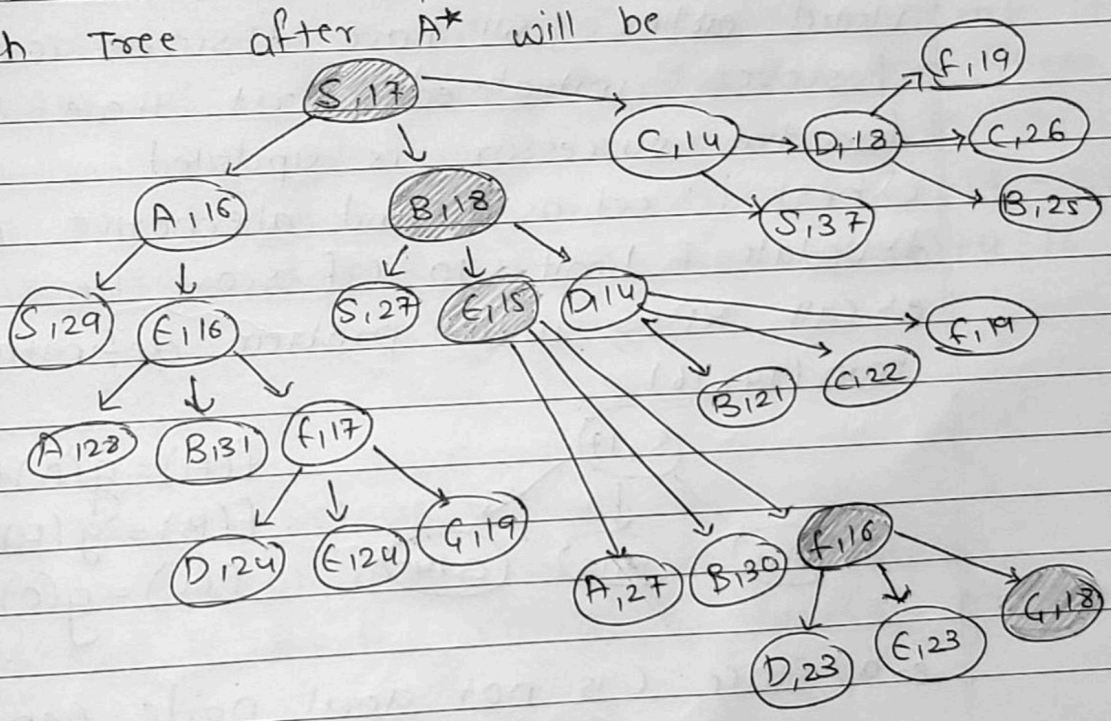
step 10 :-



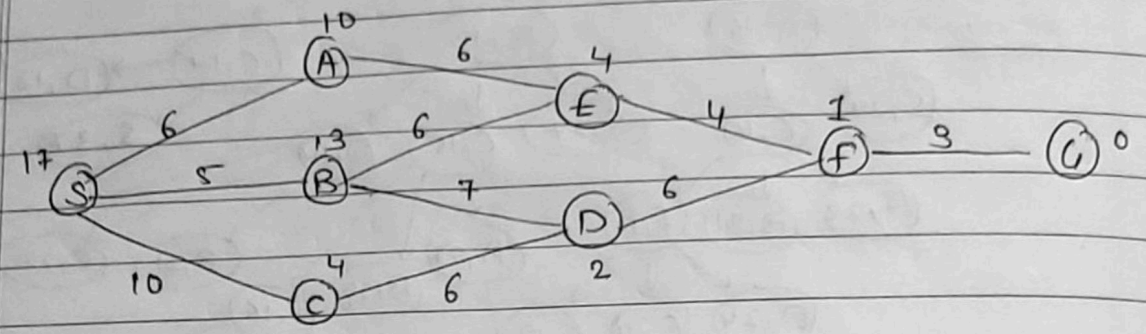
step 11 :-



step 12 :- Remove G, 18 from open list, since G is goal node goal test results in true and we will find the solution and return it. We traverse backwards towards the root from G, 18 and reverse the path we traverse to get the solution. Hence Solution is S → B → E → f → G with path cost 18. This is the optimal path cost final search Tree after A* will be



Q 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)

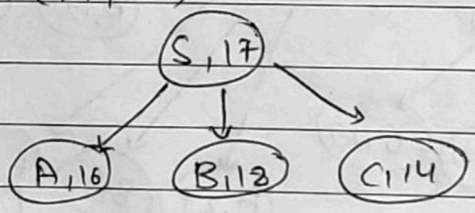
a) since S is not goal node. Generate its successors A, B and C compute f-score for them.

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)



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

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

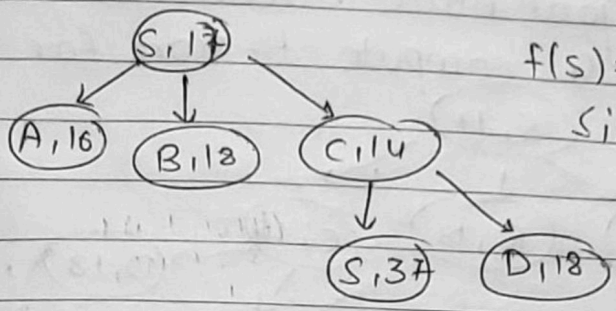
$f(C) = g(C) + h(C) = 14$

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

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

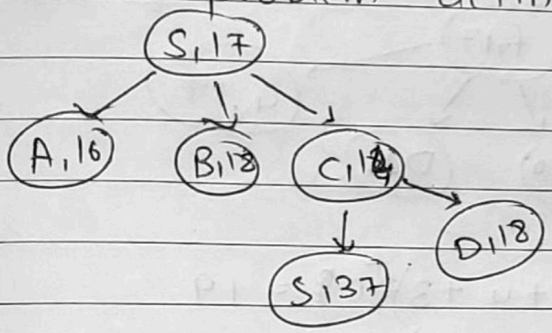
e. c) best is set as D and alternates as S

e.d) return failure and best. f=18 since best. f > f_limit.

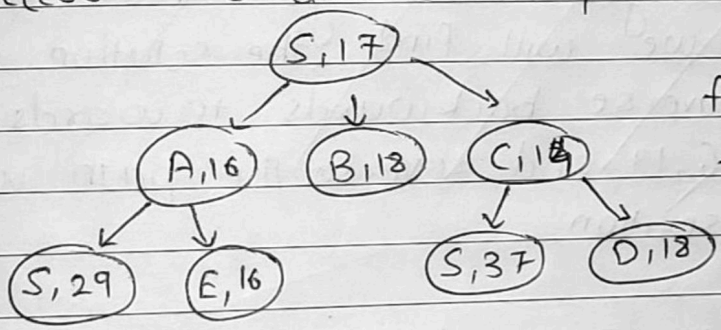


f(S) = (10+10)+17 = 37
Since 37 > 17 update S f-score
f(D) = (10+6)+2 = 18

f) update C. fscore as 18 returned from call RBfs with problem definition, C and min(14, 16)

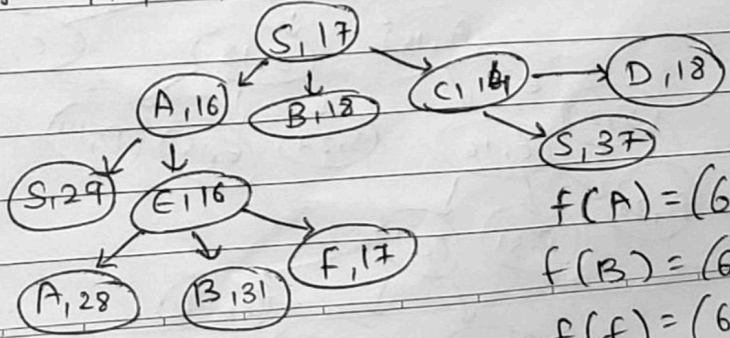


g) Now A is best node and Ca as alternate node. Since A is not goal node. Generate its successor S and E compute f-score for them.



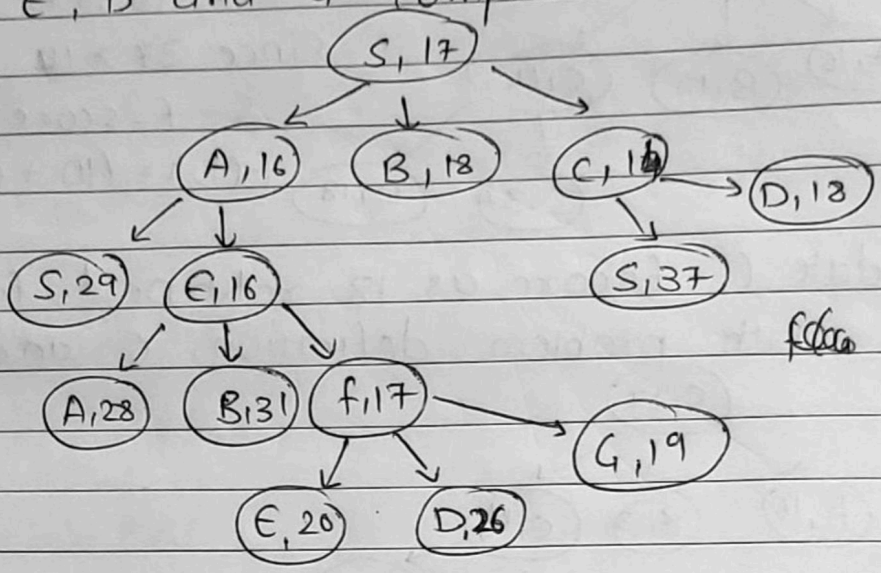
f(S) = (6+6)+17 = 29
f(E) = (6+6)+4 = 16

h) Now best is set as E and 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 A. Since f is not goal node. Generates its successor E, D and G compute f-score for them.



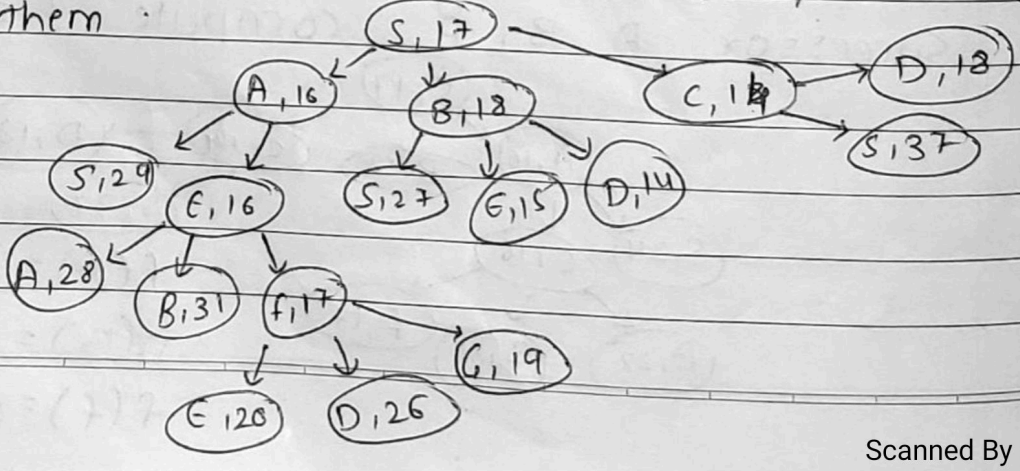
$f(G) = (6+6+4+3)+0 = 19$

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

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

Since G is goal node, goal test result in true and we will find the solution and return it. We traverse backwards towards the root from G,19 and reverse the path we traverse to get the solution.

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

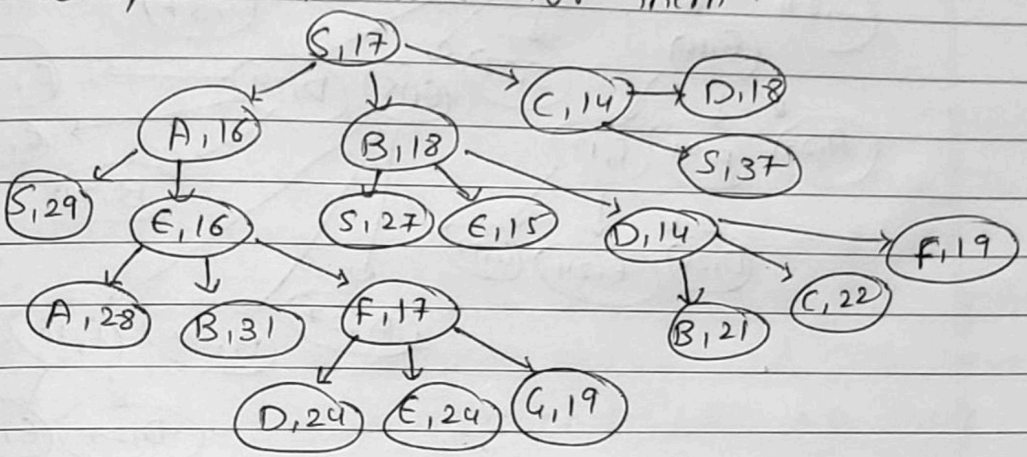


$f(s) = (5+5)+17 = 27$

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

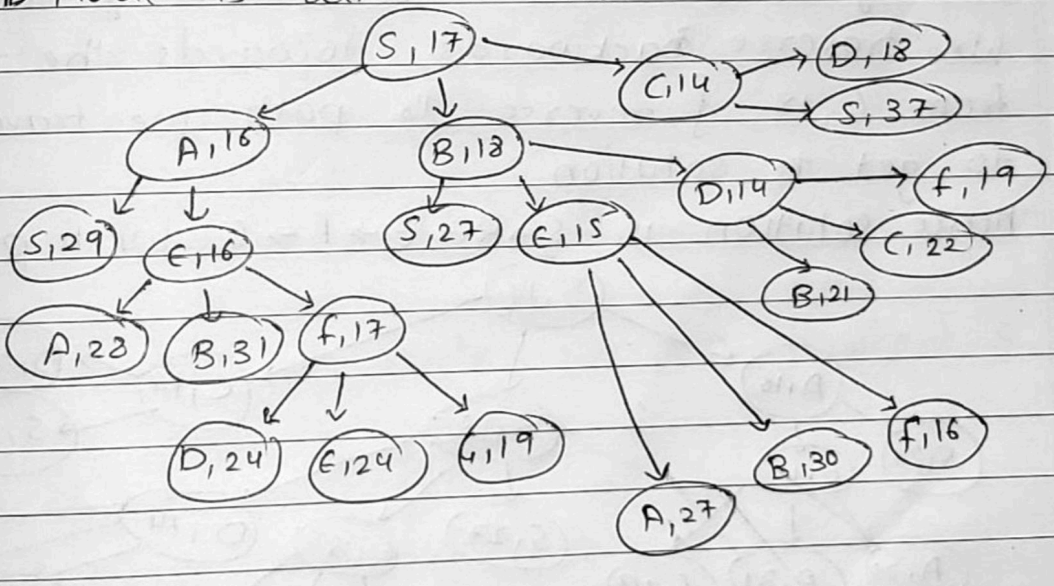
$f(D) = (7+7)+2 = 14$

k) D is not goal node. Generates its successor B, C & f & 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.

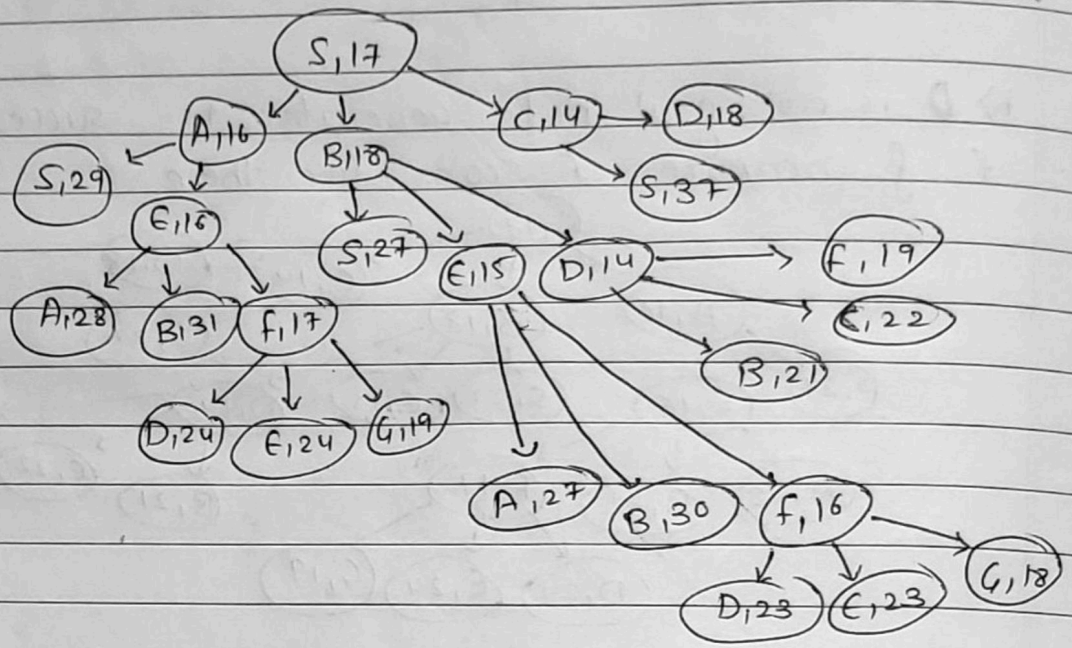


f) $f(A) = (5+6+6)+10 = 27$

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

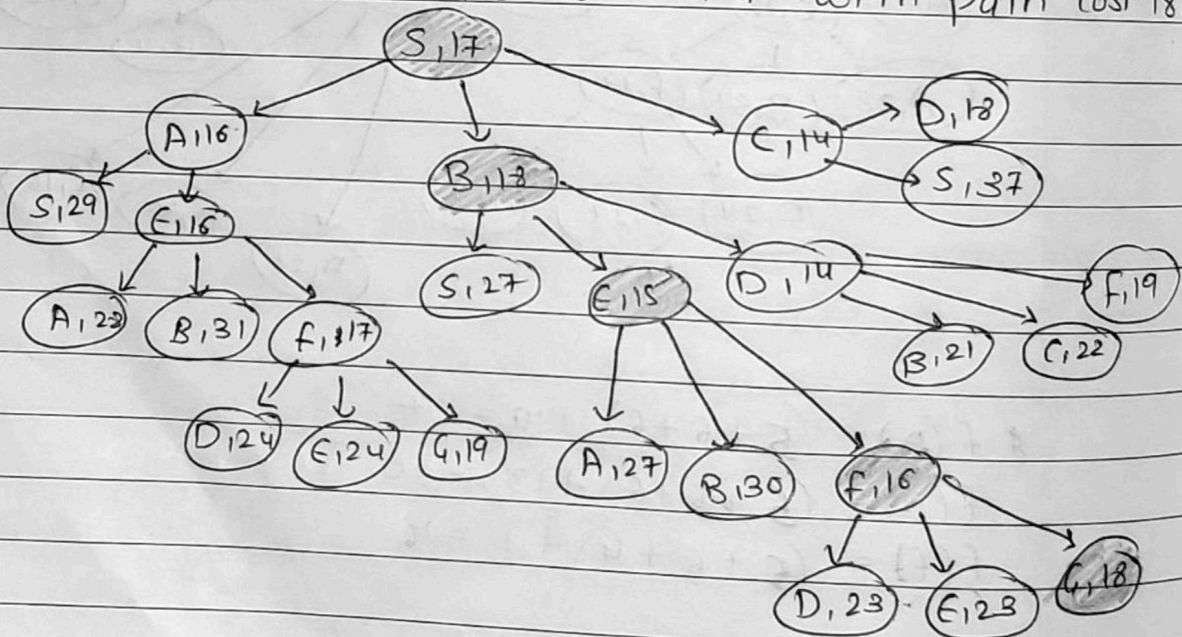
$f(f) = (5+6+4)+1 = 16$

My 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 & return it. We traverse backwards towards the root from G,18 & reverse the path we traverse to get the solution.

Hence solution is S → B → E → F → G with path cost 18.



Q2 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 Heuristic functions defined below
h1 : misplaced tiles count except space.
h2 : correctly placed tiles count except space.
h3 : sum of manhalton distance between current and correct position of all times except space

Answer the following questions :

a) In the 8 puzzle problem we are connected with getting to goal configuration within least number of steps. All moves are thus equally costly. Define g(n) in your 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 arbitory 8 puzzle instance which gives solution in 6 steps

8	7	6
2	1	5
-	3	4

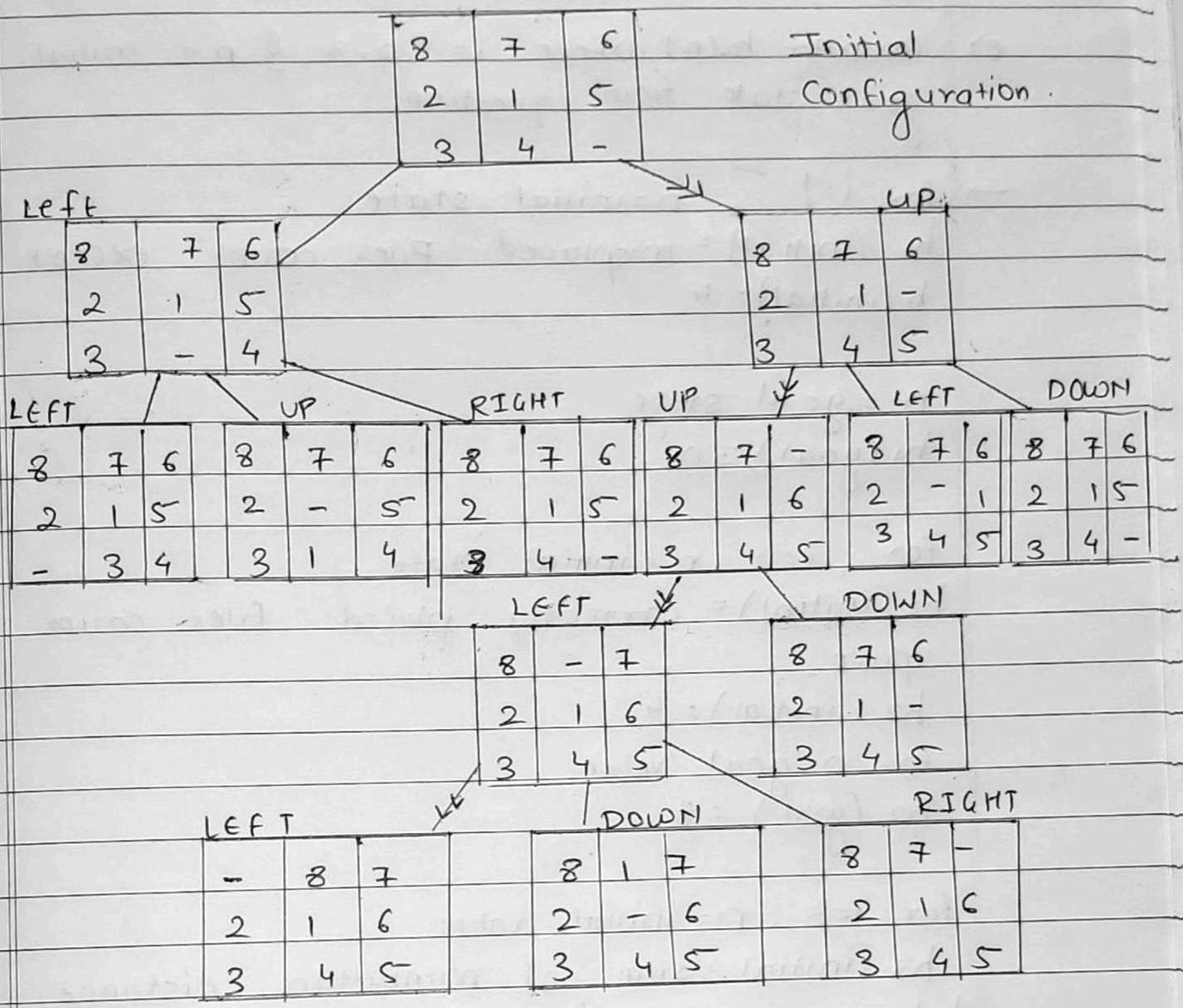
The solution can be represented as:
{{8,7,6}, {2,1,5}, {-,3,4}} → {{8,7,6} {4,5}, {3,-,4}} → {{8,7,6}, {2,15}, {3,4,-}} → {{8,7,6}, {2,1,-}, {3,4,5}} → {{8,7,-}, {2,16}, {3,4,5}} → {{8,-,7} {2,1,6}, {3,4,5}} → {{-,8,7}, {2,1,6}, {3,4,5}}

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

$$g(n) = 0$$

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

→



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 (initial) = 4

n = goal state

h_1 (goal) = 0

for $i=2$, n = initial state

h_2 (initial) = correctly placed files 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 & 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