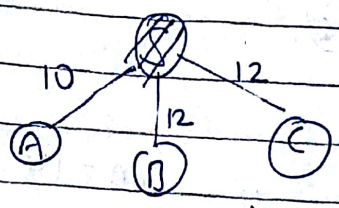
LAKSHYA BANSAL
2016240
BTech CSAM 2016

Dt. _____ B+
Pg. _____

Artificial Intelligence
Written Assignment - 1

(2) Initial bound = 0 ∵ min = 0



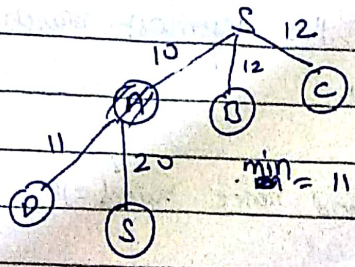
min = 10

S	0
A	0
B	4
C	3
D	0
G	0

⇒ bound = 10

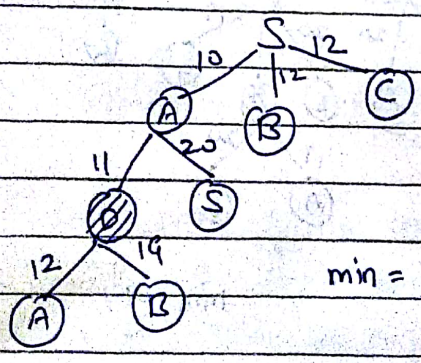
★ At each step, edge weight is the f = g + h of that node.

★ highlighted node is the current node



min = 11

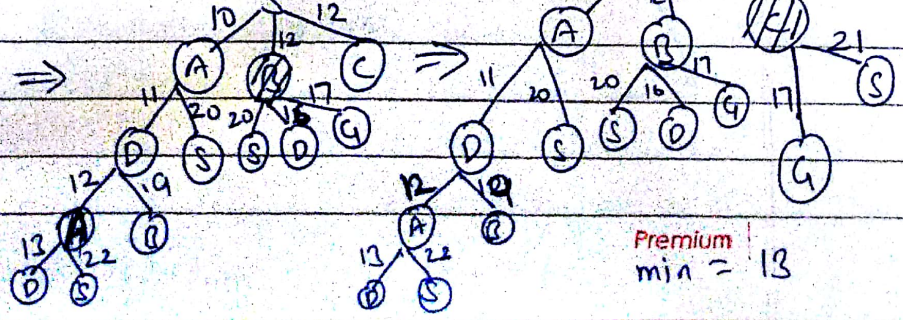
bound = 11



min = 12

bound = 12

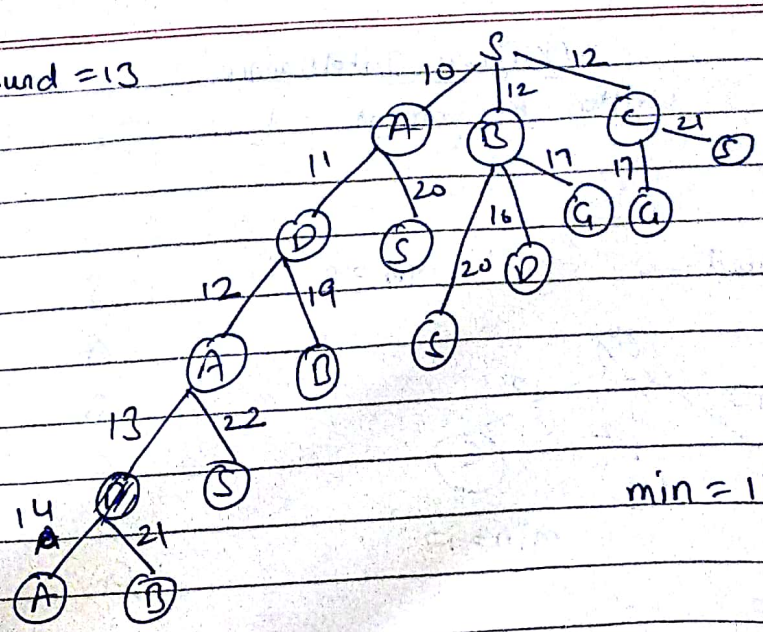
prev. tree ⇒



Premium
min = 13

Dt. _____ B+
Pg. _____

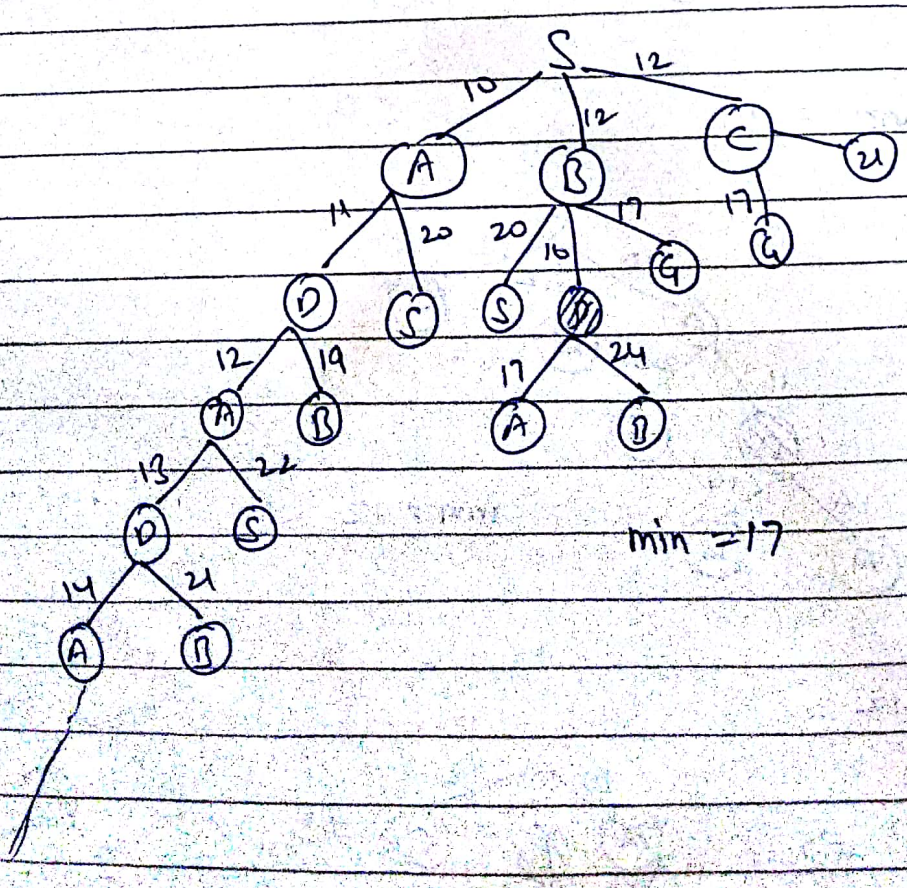
bound = 13



min = 14

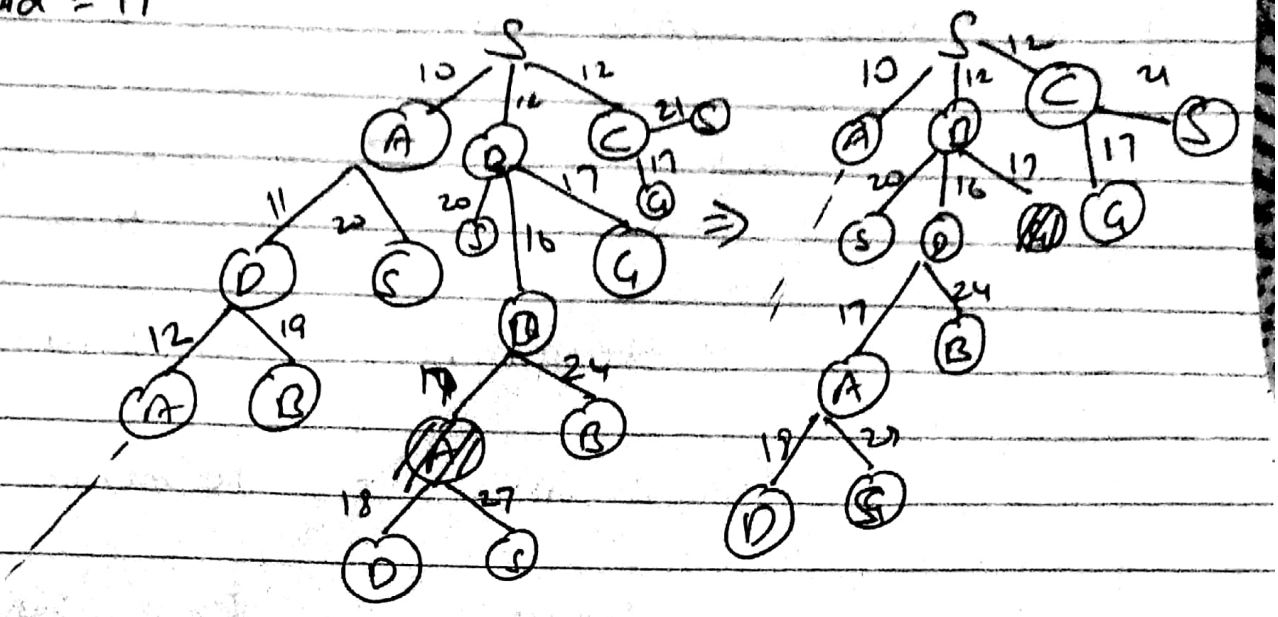
for bound = 14, and 15, the leftmost branch in the above DFS tree will be evaluated.

for when min = 16 and hence bound = 16.



min = 17

bound = 17



Hence path is S → B → G with cost 17.

(3)

	Binary
p1 = 17	10001
p2 = 21	10101
p3 - 4	00100
p4 - 28	11100

optimal solution = 10 01010

Since none of the chromosomes have 1 in their second bit from the right ~~...~~, which is required for the optimal solution. Hence, ~~no~~ any number of one point crossovers can't obtain the optimal solution of 10, Therefore, the probability is 0.

Atleast 1 mutation is required.

Dt. B+
Pg.

(4) a) Medical Diagnosis System.

Performance Measure = Accuracy diagnostic, cost, treatment measure.

Environment = hospital, staff (assuming the hospital to be the diagnostic area)

Actuators = The machine displaying result.

Sensors. = Reports, initial patients replies to questions about issues,

b) Refinery Controller

Performance Measure = purity of oil, output

Environment = factory, workers.

Actuators = refining machines, valve, heaters, displays.

Sensors = quantity sensor, pressure sensor, quality sensor, temp

c) Amazon Go

P = accuracy of the products picked, accurate pricing, correct name of person on bill

E = shop, staff, products

A = billing machine, system to transfer bill to correct person.

S = RFID, Cameras,

(6) Construct two graphs, one for each key. In each graph, connect each node(city) to a city 2 hops ahead and to a city 5 hops ahead. and terminate the branch at the destination city. Apply BFS to both graphs, terminating the BFS whenever the destination city node is encountered. If the height of both branches is same, each branch is the sequence of transfers and since BFS will find the destination node in min. height steps, the height is the min. no. of transfers required.
If not, delete the branch with lesser height and start BFS again from scratch. If all nodes are deleted, in any of the trees, no solution exists.

let g_1 & g_2 be the graphs.

```
process ( g1, g2 ) {
        if( g1.size ==0 || g2.size ==0 ): return -1;
{       h1 = BFS (g1)      // terminated when dest. city is found
        h2 = BFS (g2)      //  "  "  "  "  "  "  "
        if (h1 == h2):
                print ( seq obtained to dest. city is g1)
                print (  "   "   "   "   "   "   g2)
                print ( h1)
                return;
        else:
                if (h1 < h2):
                        delete( the path in g1 that returned h1)
                else:
                        delete (  "  "  "  g2  "  "  h2)
                process (g1, g2)
}
```

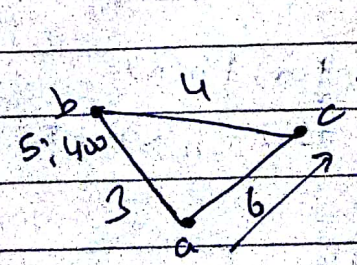
(1) The graph is constructed as follows: edge weight b/w any two cities is the distance + the time taken to defuse a bomb. We will assume that unit distance takes unit time to travel. All cities are connected to all cities, i.e., it is a complete graph.

In an A* search algo, we will take $f(n) = g(n) + h(n)$. In this question, $g(n)$ will take $g(n)$ to be the number of people saved if a city n is visited. So, if the time taken to reach city n + diffusion time exceeds time for that bomb, $g(n) = \infty$.

$h(n)$ is defined as follows:- It is the estimated max. no. of people saved for the rest of the cities. So, from n, it will visit a city y which has the maximum casualty count (it will visit y only if the bomb at y can be diffused, else it will visit the city with next highest casualty count.). From y, it will follow the same routine as above and move to a neighbouring city with highest casualty count. Hence $h(n)$ will calculate the estimated max. casualties that can be saved for the remaining unvisited cities.

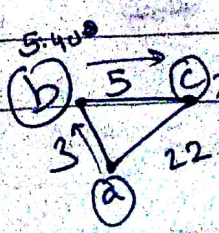
So the city n with max value of $g(n) + h(n)$ in the priority queue will be visited and the rest of the $f(n)$ values will be stored in priority queue.

So suppose we can visit cities 'c' and 'b' from 'a'.



$h(a) = 800$

but suppose this graph is



$h(a) = 1200$

DL. _____ B+
PE. _____

The given heuristic is admissible because the heuristic calculates at each step the estimated maximum more casualties that can be saved. The optimal solution that exists will either calculate greater than or equal to the number of casualties estimated by the heuristic. Hence the heuristic is never overestimating.

Since at every step we are taking the city with max value of $f(n) = g(n) + h(n)$, we are sure to find a path which will save optimal maximum no. of casualties.

Space complexity = O(no. of nodes)
Time complexity = $O(b^n * n * n)$. (∵ for each city we have to calculate heuristic).

(5) Given $h(n) = h^*(n) + \varepsilon$

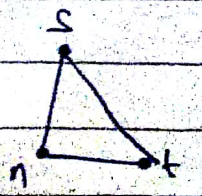
Since $h(n)$ overestimates the admissible heuristic, it will give a path atmost ε time more than the optimal path.

Now there ∃ a node n in the path obtained by using h^* as the heuristic, which is not in the path obtained by using h as the heuristic., such that.

$f(t) \leq f(n)$, else n would have been in the path obtained by using $h(\cdot)$ as the heuristic. We will take the 1st such node 'n'.

$\Rightarrow g(t) \leq g(n) + h(n)$ (∵ $h(t) = 0$)

$\Rightarrow g(t) \leq g(n) + h^*(n) + \varepsilon$ — (1)



From the △ on the left, it can be noted that

$h^*(s) \leq g(n) + h^*(n)$

$\Rightarrow h^*(s) + \varepsilon \leq g(n) + h^*(n) + \varepsilon$ — (2)

Assuming $\varepsilon \geq g(t) - h^*(s)$. — (3)

Combining (1), (2), (3), $\boxed{g(t) \leq h^*(s) + \varepsilon}$.