CSE3013 – Artificial Intelligence (ETP)

(C2+TC2)

Digital Assignment – 1

Under the guidance of –

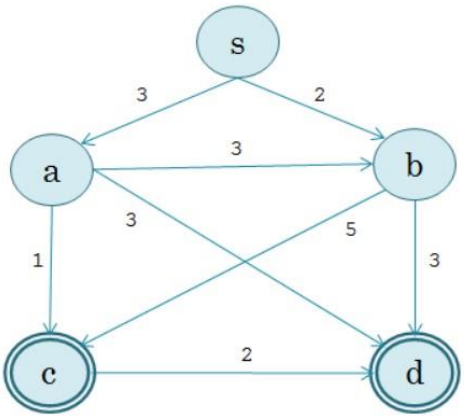
Prof. Anto S.

Presented By -

Kumar Sparsh

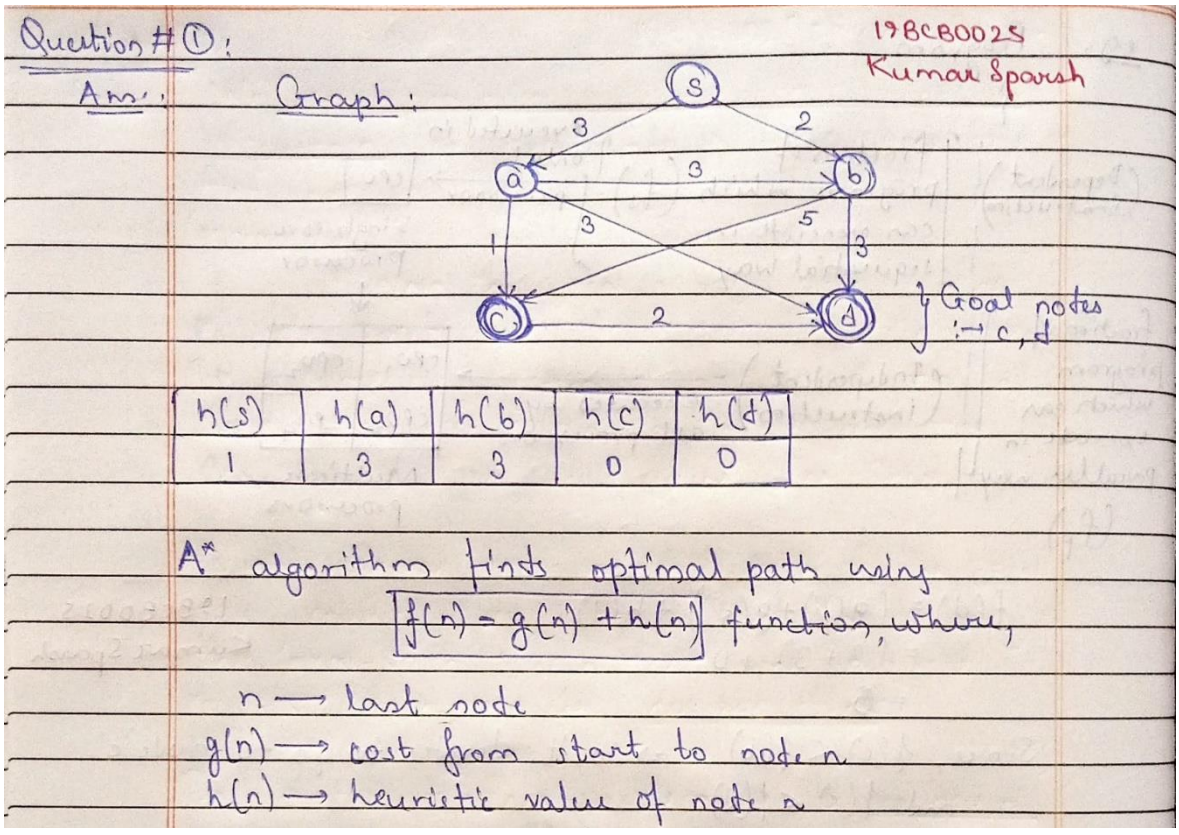
19BCB0025

1. **Using A* Algorithm, find the optimal path from the node S for the graph given in the diagram. Two goal nodes are given in the diagram . Explain how the algorithm can/cannot find the optimal solution for both goal nodes.**



h(s)	h(a)	h(b)	h(c)	h(d)
1	3	3	0	0

Ans.



Step①: Starts with node 's', node 'a' and 'b' can be reached from node 's'.

Calculating f(a) and f(b),

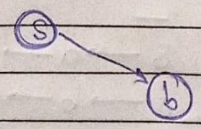
$f(a) = 3 + 3$

$= 6$

$f(b) = 2 + 3$

$= 5$

Since, $f(b) < f(a)$, so it decides to go to node 'b'.



Path: s → b

Step②: Node 'c' and node 'd' can be reached from node 'b'.

19BCB0025
Kumar Sparsh

Calculating f(c) and f(d),

$f(c) = (2+5) + 0$

$= (g(b) + g(c)) + h(c)$

$= 7$

$f(d) = (g(b) + g(d)) + h(d)$

$= (2+3) + 0$

$= 5$

Since, $f(d) < f(a)$ and, $f(d) < f(c)$ ⇒ so, it decides to go to node 'd'.

Path: s → b → d, here, cost is 5 that is less among all. Hence, it is optimal path.

So,

Optimal solution for node 'd' is:
s → b → d, with a cost of 5

Step③: There is no node which can be reached from node 'd'. So, from node 'a' and node 'c', $f(a) < f(c)$. So, it decides to go to node 'a'.

Path: s → a

Step④: Node 'b', 'c', and 'd' can be reached from node 'a'.

Calculating $f(b)$, $f(c)$, $f(d)$,

$f(b) = (g(a) + g(b)) + h(b)$
$= (3+3) + 3$
$= 9$

$f(c) = (g(a) + g(c)) + h(c)$
$= (3+1) + 0$
$= 4$

$f(d) = (g(a) + g(d)) + h(d)$
$= (3+3) + 0$
$= 6$

19BCB0025
Kumar Sparsh

Since, $f(c) < f(b)$ ⇒ so, it decides to go to node 'c'.
and, $f(c) < f(d)$

Path: $S \longrightarrow a \longrightarrow c$

Step⑤: As 4 is minimum cost among others, there is no any other optimal path possible.

i.e., Path: $S \rightarrow a \rightarrow c$, here, its cost is 4, that is least amongst all. So, it is optimal path.

So, **Optimal solution for node 'c' is: $S \longrightarrow a \longrightarrow c$, with a cost of 4**

⇒ **Optimal solutions :—**
for node C ⇒ $S \longrightarrow a \longrightarrow c$ (cost = 4)
for node D ⇒ $S \longrightarrow b \longrightarrow d$ (cost = 5)

Ans.

2. **If a program prunes away a move in chess that looks bad because it sacrifices valuable material that may just be the sacrificial move that would have checkmated the opponent in another few additional ply (or just gain a material and/or tactical advantage)? How can alpha-beta pruning be made safe for a chess program? Discuss whether this pruning algorithm can be used to achieve the best solution space for the given problem.**

Ans.

Question #②: 19BCB0025-Kumar Sparsh

Ans. We use alpha-beta pruning with a minimax algorithm for evaluating a large game tree in Artificial Intelligence.

Generally, when the game is very large, i.e., the possible positions or states are of huge numbers, for a computing system, it gets unrealistic and impractical to search the whole game tree for the required move in a game like Chess.

Here, it searches in a small portion of the game tree making fixed-depth or level. The tendency of the fixed-depth search gives some incorrect results, which is called "horizon effect".

One solution to this problem is to "see beyond the horizon" using the quiescence function, to be applied recursively on the terminal nodes and finding the best possible move.