Student Name: Kapil Gautam Student NetID: KXG180032

**University of Texas at Dallas**
**Department of Computer Science**
**CS 6364.001 – Artificial Intelligence**
**Fall 2020**
**Instructor: Dr. Sanda Harabagiu**

**Instructions:** Do not communicate with anyone in any shape or form. This is an independent exam. Do not delete any problem formulation, just attach your answer in the space provided. If the problem is deleted and you send only the answer, you shall receive ZERO points.

Copy and paste the Mid-Term Exam into a Word document, enter your answers (either by typing in Word, or by inserting a VERY CLEAR picture of your hand-written solution) and transform the file of the exam into a PDF format. If we cannot clearly read the picture, you will get ZERO for that answer! Make sure that you insert EACH answer immediately after EACH question. Failure to do so will result in ZERO points for the entire exam! Submit the PDF file with the name **MidTerM_Exam_netID.pdf**, where netID is your unique netid provided by UTD. If you submit your exam in any other format your will receive ZERO points. The Midterm shall be submitted in eLearning before the deadline. No late submissions shall be graded! Any cheating attempt will determine the ENTIRE grade of the mid-term to become ZERO.

---

**Problem 1 (50 points)**

*Proteins have an amino acid "alphabet" of 11 elements:* AM1, AM2, ..., AM11*. Amino acids are chemically linked together to form protein chains. Between amino acids there are chemical links of different strengths. Suppose you examine under microscope a sample of a protein that belongs to an alien species, having only 11 amino acids. You want to generate an optimal path between* AM1 *and* AM2 *using the A* search algorithm. You are given the strengths of the chemical links in the sample as a graph representation:*

```
 Oracle distance to AM2          |            The Graph
_____________________________________________________________________________
AM1                 160          |        AM11  ----- AM4    :::: 50
AM3                 100          |        AM11  ----- AM10   :::: 150
AM4                 200          |        AM11  ----- AM9    :::: 15
AM5                 120          |        AM4   ----- AM7    :::: 40
AM6                  80          |        AM7   ----- AM8    :::: 180
AM7                 250          |        AM7   ----- AM6    :::: 110
AM8                  40          |        AM9   ----- AM8    :::: 70
AM9                  60          |        AM10  ----- AM2    :::: 30
AM10                 25          |        AM8   ----- AM2    :::: 45
AM11                100          |        AM10  ----- AM3    :::: 80
******************************   |        AM3   ----- AM5    :::: 50
******************************   |        AM5   ----- AM1    :::: 40
******************************   |        AM1   ----- AM6    :::: 70
******************************   |        AM6   ----- AM8    :::: 20
******************************   |        AM1   ----- AM4    :::: 350
******************************   |_______________________________________
```

*An oracle also gives you the heuristic distance values to* AM2 *from each other amino acid in the sample. This heuristic is consistent. Specify if you will use TREE-SEARCH or GRAPH-SEARCH. Motivate your decision.* **(5 points)**
*Provide the path of amino acids from* AM1 *to* AM2 *as well as the cost of obtaining it.*

*it. Describe <u>at each step of the search</u> (1) what amino acids you have on the search frontier; (2) the current list of explored amino acids; (3) the current path from AM1 to the current amino acid and the cost of that path. Show the successors of each current node, show how you compute all the evaluation functions and which node you select for the next step.* **(45 points)**

**<u>Solution:</u>**

The heuristic is consistent, so we will use GRAPH-SEARCH.

<u>Motivation</u>: As mentioned in the problem statement, the heuristic is consistent. A heuristic is consistent if for every node *n* and for every successor *n'* generated by some action *a*, the estimated cost of reaching the goal from n is no greater than the estimated cost of reaching the goal from n', i.e., they satisfy the triangle inequality:
$h(n) <= c(n,a,n') + h(n')$.  -- 1

Now, n' is a successor of n:
$g(n') = g(n) + c(n,a,n')$  --- 2
$f(n') = g(n') + h(n') = g(n) + c(n,a,n') + h(n')$  --- Using 2
$f(n') = g(n) + c(n,a,n') + h(n')$  --- 3

Since $h(n) <= c(n,a,n') + h(n')$  --- Using 1
Adding g(n) to both sides:
$g(n) + h(n) <= g(n) + c(n,a,n') + h(n')$

Since $f(n) = g(n) + h(n)$ from A* search,
$f(n) <= f(n')$  --- Using 3

From above, we see that each successor evaluation function value will be no less than the parent evaluation function value. A* search always selects the node in order of increasing f value. Since the sequence of nodes expanded by GRAPH-SEARCH is in non-decreasing order of evaluation function, so the Graph-search is an optimal search if the heuristic is consistent.

Search steps from AM1 to AM2:

**Step 1:**
Current Node: AM1
Explored Nodes: {AM1}

Children of AM1: {AM4, AM5, AM6}
f(AM4) = g(AM1, AM4) + h(AM4) = 350 + 200 = 550
f(AM5) = g(AM1, AM5) + h(AM5) = 40 + 120 = 160
f(AM6) = g(AM1, AM6) + h(AM6) = 70 + 80 = 150

Frontier Nodes: {(150, AM6), (160, AM5), (550, AM4)}
Next Node: AM6

**Step 2:**
Current Node: AM6
Explored Nodes: {AM1, AM6}
Current Path: AM1 -> AM6
Total Path cost until now: 70

Children of AM6: {AM7, AM8, AM1}
f(AM1) = Already in explored list, not evaluating
f(AM7) = g(AM1, AM6) + g(AM6, AM7) + h(AM7) = 70 + 110 + 250 = 430
f(AM8) = g(AM1, AM6) + g(AM6, AM8) + h(AM8) = 70 + 20 + 40 = 130

Frontier Nodes: {(130, AM8), (160, AM5), (430, AM7), (550, AM4)}
Next Node: AM8

**Step 3:**
Current Node: AM8
Explored Nodes: {AM1, AM6, AM8}
Current Path: AM1 -> AM6 -> AM8
Total Path cost until now: 70+20 = 90

Children of AM8: {AM2, AM6, AM7, AM9}
f(AM2) = g(AM1, AM6) + g(AM6, AM8) + g(AM8, AM2) + h(AM2)
= 70 + 20 + 45 + 0 = 135
f(AM6) = Already in explored list, not evaluating
f(AM7) = g(AM1, AM6) + g(AM6, AM8) + g(AM8, AM7) + h(AM7)
= 70 + 20 + 180 + 250 = 520
AM7 is in frontier list with value 430 < 520, don’t update

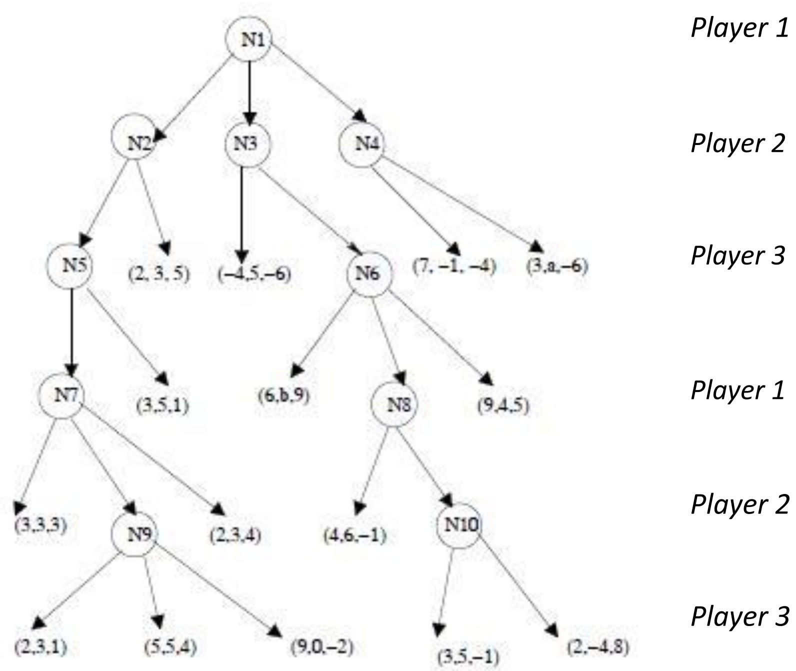
f(AM9) = g(AM1, AM6)  +  g(AM6, AM8) +  g(AM8, AM9) + h(AM9)
= 70 + 20 + 70 + 60 = 220

Frontier Nodes: {(135, AM2), (160, AM5), (220, AM9) (430, AM7), (550, AM4)}
Next Node: AM2

**Step 4:**
Current Node: AM2
Explored Nodes: {AM1, AM6, AM8, AM2}
AM2 is the goal node.
Current Path: AM1 -> AM6 -> AM8 -> AM2
Total Path cost: g(AM1, AM6)  +  g(AM6, AM8) +  g(AM8, AM2) = 70+20+45 = 135

**Problem 2 (50 points)**

(a) (**15 points**) Given the following game tree, find the possible values of variables *a* and *b* such that the minimax values in node N1 are (7, −1, −4). Also compute the minimax values at nodes: N2, N3, N4, N5, N6, N7, N8, N9 and N10.




**Solution:**

Minimax(N10) = Player2Max ( (3,5,-1), (2,-4,8) ) = (3,5,-1)
Minimax(N9) = Player2Max ( (2,3,1), (5,5,4), (9,0,-2) ) = (5,5,4)

Minimax(N8) = Player1Max ( (4,6,-1), Minimax(N10) ) = Player1Max ( (4,6,-1), (3,5,-1) ) = (4,6,-1)

Minimax(N7) = Player1Max ( (3,3,3), Minimax(N9), (2,3,4) ) = Player1Max ( (3,3,3), (5,5,4), (2,3,4)) = (5,5,4)

Minimax(N6) = Player3Max ( (6,b,9), Minimax(N8), (9,4,5) ) = Player3Max ( (6,b,9), (4,6,-1), (9,4,5) ) = (6,b,9)
Minimax(N5) = Player3Max ( Minimax(N7), (3,5,1) ) = Player3Max ( (5,5,4), (3,5,1) ) = (5,5,4)

Minimax(N4) = Player2Max ( (7,-1,-4), (3,a,-6) )
Minimax(N3) = Player2Max ( (-4,5,-6), Minimax(N6) ) = Player2Max ( (-4,5,-6), (6,b,9) )
Minimax(N2) = Player2Max ( Minimax(N5), (2,3,5)) = Player2Max ( (5,5,4), (2,3,5) ) = (5,5,4)

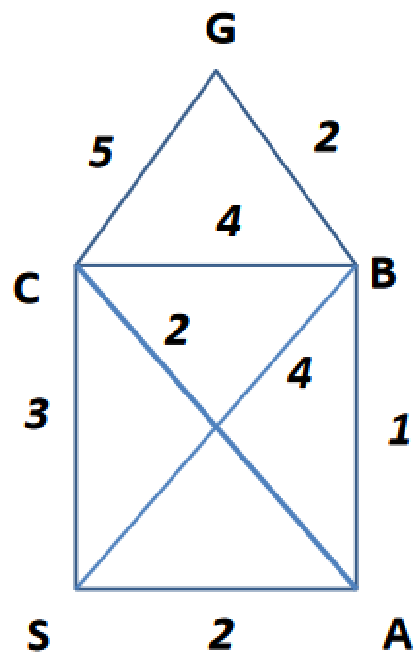
Minimax(N1) = Player1Max ( Minimax(N2), Minimax(N3), Minimax(N4) )
= Player1Max ( (5,5,4),
Player2Max ( (-4,5,-6), (6,b,9) ),
Player2Max ( (7,-1,-4), (3,a,-6) ))
Given Minimax(N1) = (7,-1,-4)

This value is only possible when Minimax(N4) = (7,-1,-4) is chosen for Player1 in Minimax(N1).
This implies for Minimax(N4) = (7,-1,-4) = Player2Max ( (7,-1,-4), (3,a,-6) )
=> -1 is greater than a => a < -1

For Minimax(N1) to be (7,-1,-4) , we can either choose (-4,5,-6) or (6,b,9) for Minimax(N3).
So b value will have no effect.

Therefore
Range of a: $-\infty < a < -1$
Range of b: $-\infty < b < +\infty$

(b) (**15 points**) An agent stating in state *S* should reach the goal state *G*. If the possible states the agent can reach are *A, B, C* or *G*, as depicted bellow:



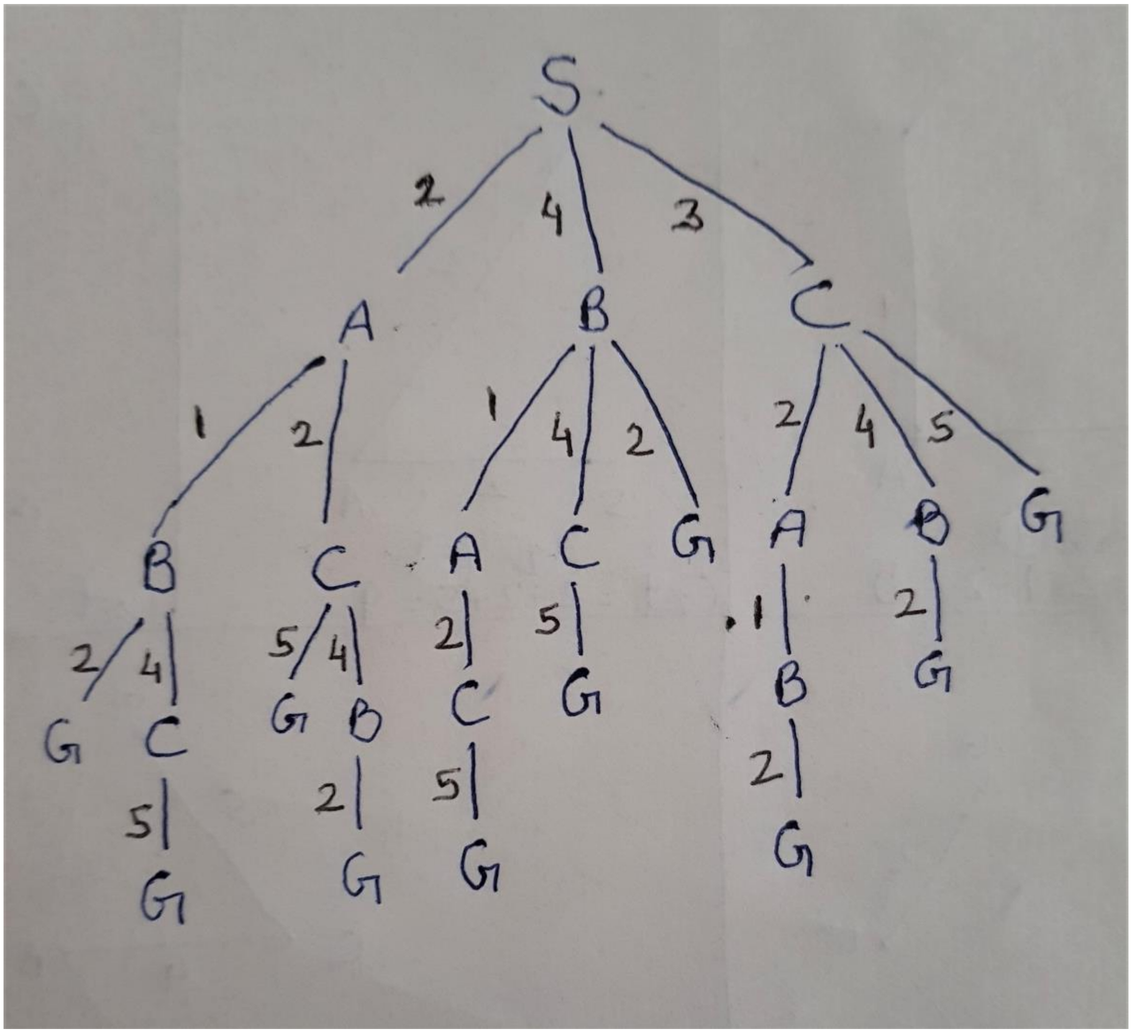

And as shown in the figure: cost(S⟶A) = 2; cost(S⟶B) = 4; cost(S⟶C) = 3; cost(A⟶B) = 1; cost(A⟶C) = 2; cost(B⟶C) = 4; cost(B⟶G) = 2; cost(C⟶G) = 5; you are asked to:

(a) draw the search tree that allows the agent to travel from *S* to *G*, knowing that the agent *cannot ever visit S again*, and cannot visit any state more than once. Show in the search tree all the ways in which the agent can get from the state *S* to the goal state *G*; (**5 points**) How many ways of getting to the goal state *G* from *S* are there? (**2 points**) *HINT:* Any solution path starts in *S* and ends in *G* but does not have to visit all other nodes! However, it cannot visit more than one any node!

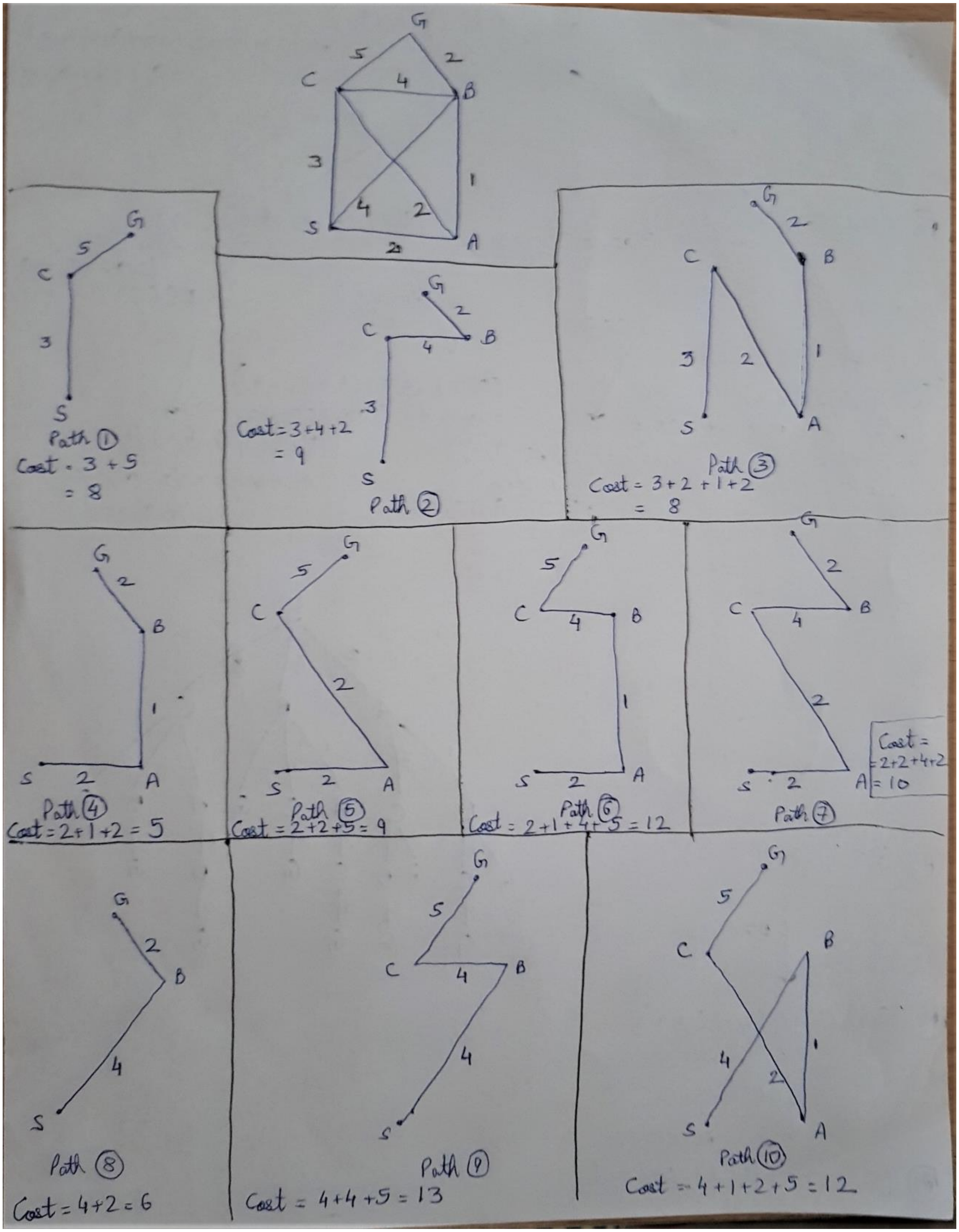
(b) What is the least costly and the costliest way for the agent to get from state *S* to state *G*? Show the least costly path (**2 points**) and specify how much it costs (**2 points**). Show the costliest path (**2 points**) and specify how much it costs (**2 points**) and show how you have computed the costs.

**Solution:**

(a) Total ways to get from S to G without visiting more than once any node in a path: 10
(Assuming the cost for cost(B⟶C) = 4 === cost(C⟶B) = 4, similar for others)
Search Tree:




Search Paths visualized:




Search Paths:

| Path# | Path | Path Cost |
|---|---|---|
| 1. | S -> C -> G | 3+5 = 8 |
| 2. | S -> C -> B -> G | 3+4+2 = 9 |
| 3. | S -> C -> A -> B -> G | 3+2+1+2 = 8 |
| 4. | S -> A -> B -> G | 2+1+2 = 5 |
| 5. | S -> A -> C -> G | 2+2+5 = 9 |
| 6. | S -> A -> B -> C -> G | 2+1+4+5 = 12 |
| 7. | S -> A -> C -> B -> G | 2+2+4+2 = 10 |
| 8. | S -> B -> G | 4+2 = 6 |
| 9. | S -> B -> C -> G | 4+4+5 = 13 |
| 10. | S -> B -> A -> C-> G | 4+1+2+5 = 12 |

(b)

Least costly path:
Path#4 : S -> A -> B -> G
Cost: cost(S⟶A) + cost(A⟶B) + cost(B⟶G) = 2+1+2 = 5

Costliest path:
Path#9: S -> B -> C -> G
Cost: cost(S⟶B) + cost(B⟶C) + cost(C⟶G) = 4+4+5 = 13

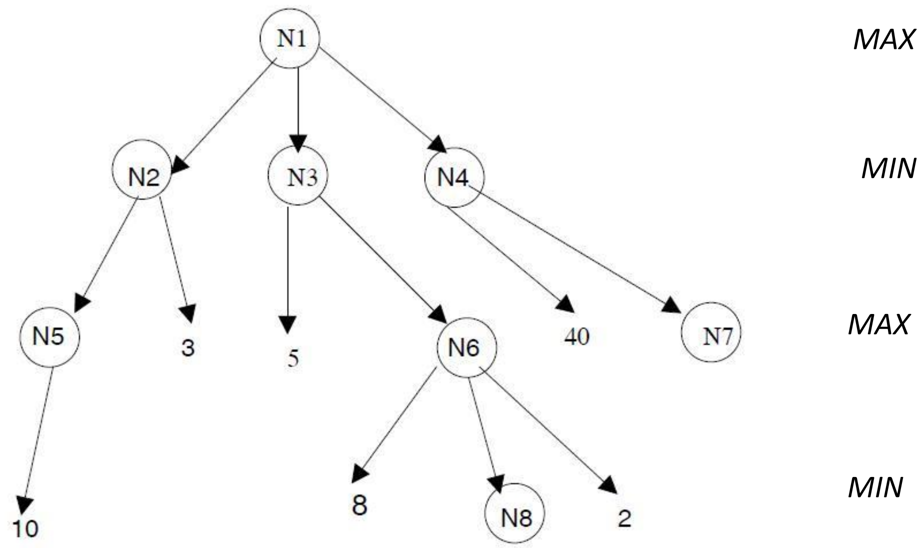
(c) (**20 points**) Given the game tree below, compute the value of alpha and beta at following nodes, if the order is the same as in depth-first search:

(1) alpha and beta at node N3 before and after visiting the terminal node with utility 5. Also show the values of alpha and beta in N3 after visiting N6 (Hint: Show also the values of alpha and beta at all nodes visited before you reached N3.); **(3 points)**

(2) alpha and beta at node N6; **(2 points)**

(3) alpha and beta at node N1 after visiting N7, if the node N7 has a child node with an utility value *x* (after you visited all nodes illustrated in the Figure) **(5 points)**

(4) should the game tree be pruned? If yes, how? **(10 points)**




**<u>Solution:</u>**

(c) Following depth-first search and the order of neighbors are assumed to be as given in graph for each node from left-to-right.

(1) alpha and beta at node N3 before and after visiting the terminal node with utility 5. Also show the values of alpha and beta in N3 after visiting N6 (Hint: Show also the values of alpha and beta at all nodes visited before you reached N3.); **(3 points)**

| **Node** | **alpha value** | **beta value** | **Utility** |
|---|---|---|---|
| N1 | alpha = -∞ | beta = +∞ | max-value = -∞ |
| N2 | alpha = -∞ | beta = +∞ | min-value = +∞ |
| N5 | alpha = -∞ | beta = +∞ | max-value = -∞ |
| 10 | 10 | 10 | |
| N5 | alpha = 10 | beta = +∞ | max-value = 10 |
| N2 | alpha = -∞ | beta = 10 | min-value = 10 |
| 3 | 3 | 3 | |
| N2 | alpha = -∞ | beta = 3 | min-value = 3 |
| N1 | alpha = 3 | beta = +∞ | max-value = 3 |
| N3 | alpha = 3 | beta = +∞ | min-value = +∞ |
| 5 | 5 | 5 | |
| N3 | alpha = 3 | beta = 5 | min-value = 5 |
| N6 | alpha = 3 | beta = 5 | max-value = -∞ |
| 8 | 8 | 8 | |
| N6 | alpha = 3 | beta = 5 | max-value = 8<br>8 >= 5<br>beta pruning cut<br>1 |
| N3 | alpha = 3 | beta = 5 | min-value = 5 |

(2) alpha and beta at node N6; **(2 points)**

At Node N6:
alpha value = 3
beta value = 5

(3) alpha and beta at node N1 after visiting N7, if the node N7 has a child node with an utility value *x* (after you visited all nodes illustrated in the Figure) **(5 points)**

| Node | alpha value | beta value | Utility |
|---|---|---|---|
| N1 | alpha = -∞ | beta = +∞ | max-value = -∞ |
| N2 | alpha = -∞ | beta = +∞ | min-value = +∞ |
| N5 | alpha = -∞ | beta = +∞ | max-value = -∞ |
| 10 | 10 | 10 | |
| N5 | alpha = 10 | beta = +∞ | max-value = 10 |
| N2 | alpha = -∞ | beta = 10 | min-value = 10 |
| 3 | 3 | 3 | |
| N2 | alpha = -∞ | beta = 3 | min-value = 3 |
| N1 | alpha = 3 | beta = +∞ | max-value = 3 |
| N3 | alpha = 3 | beta = +∞ | min-value = +∞ |
| 5 | 5 | 5 | |
| N3 | alpha = 3 | beta = 5 | min-value = 5 |
| N6 | alpha = 3 | beta = 5 | max-value = -∞ |
| 8 | 8 | 8 | |
| N6 | alpha = 3 | beta = 5 | max-value = 8<br>8 >= 5<br>beta pruning cut<br>1 |
| N3 | alpha = 3 | beta = 5 | min-value = 5 |
| N1 | alpha = 5 | beta = +∞ | max-value = 5 |
| N4 | alpha = 5 | beta = +∞ | min-value = +∞ |
| 40 | 40 | 40 | |
| N4 | alpha = 5 | beta = 40 | min-value = 40 |
| N7 | alpha = 5 | beta = 40 | max-value = -∞ |
| N7child | alpha = 5 | alpha = 40 | min-value = x |

**Case 1: x <= 5**

| | | | |
|---|---|---|---|
| N7 | alpha = 5 | beta = 40 | max-value = x |
| N4 | alpha = 5 | beta = x | min-value = x<br>x <= alpha<br>No successors,<br>no cut |
| N1 | alpha = 5 | beta = +∞ | max-value = 5 |

**Case 2: x > 5, x < 40**

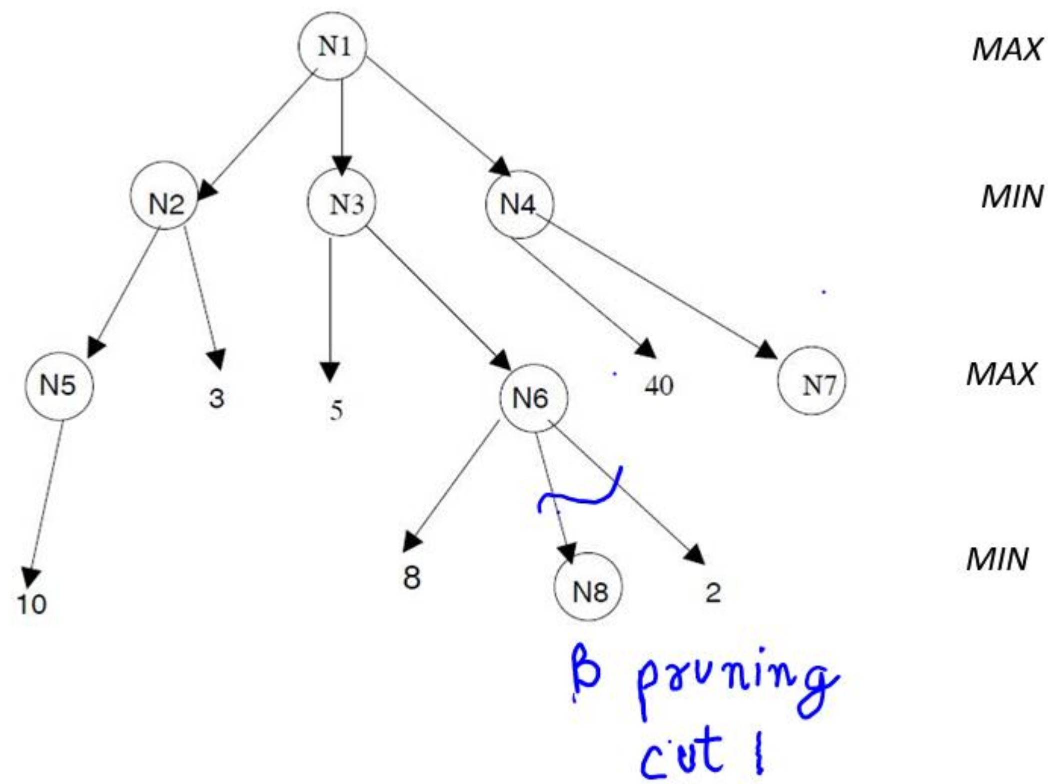
| | | | |
|---|---|---|---|
| N7 | alpha = x | beta = 40 | max-value = x |
| N4 | alpha = 5 | beta = x | min-value = x |
| N1 | alpha = x | beta = +∞ | max-value = x |

**Case 3: x >= 40**

| | | | |
|---|---|---|---|
| N7 | alpha = 5 | beta = 40 | max-value = x<br>x >= beta<br>But no successors, so no cut |
| N4 | alpha = 5 | beta = 40 | min-value = 40 |
| N1 | alpha = 40 | beta = +∞ | max-value = 40 |

(4) should the game tree be pruned? If yes, how? **(10 points)**
Yes, the game tree should be pruned as per the following table:

| **Node** | **alpha value** | **beta value** | **Utility** |
|---|---|---|---|
| N1 | alpha = -∞ | beta = +∞ | max-value = -∞ |
| N2 | alpha = -∞ | beta = +∞ | min-value = +∞ |
| N5 | alpha = -∞ | beta = +∞ | max-value = -∞ |
| 10 | 10 | 10 | |
| N5 | alpha = 10 | beta = +∞ | max-value = 10 |
| N2 | alpha = -∞ | beta = 10 | min-value = 10 |
| 3 | 3 | 3 | |
| N2 | alpha = -∞ | beta = 3 | min-value = 3 |
| N1 | alpha = 3 | beta = +∞ | max-value = 3 |
| N3 | alpha = 3 | beta = +∞ | min-value = +∞ |
| 5 | 5 | 5 | |
| N3 | alpha = 3 | beta = 5 | min-value = 5 |
| N6 | alpha = 3 | beta = 5 | max-value = -∞ |
| 8 | 8 | 8 | |
| N6 | alpha = 3 | beta = 5 | max-value = 8<br>8 >= 5<br>beta pruning cut 1 |
| N3 | alpha = 3 | beta = 5 | min-value = 5 |
| N1 | alpha = 5 | beta = +∞ | max-value = 5 |
| N4 | alpha = 5 | beta = +∞ | min-value = +∞ |
| 40 | 40 | 40 | |
| N4 | alpha = 5 | beta = 40 | min-value = 40 |
| N7 | alpha = 5 | beta = 40 | max-value = -∞ |



N1
N2
N3
N4
N5
3
5
N6
40
N7
10
8
N8
2
MAX
MIN
MAX
MIN
β pruning
cut 1