

Experiment 4

Objective

To be able to apply Min-Max Algorithm for game playing.

Problem Statement

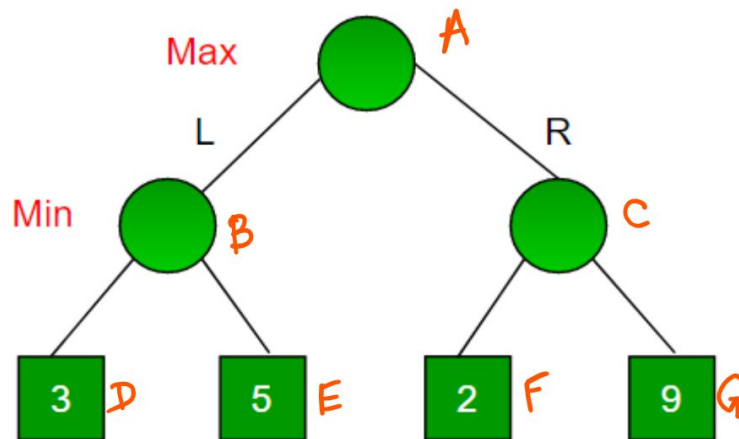
To apply an adversarial search technique (Min-Max Algorithm) for finding out the minimum cost path from source node to goal node in graph.

Lab Exercise

Write the program for finding out the path from source node to a goal node in a graph using the Min-Max algorithm in a game tree. Ask the user to provide the details of the tree, from the user i.e. number of nodes, number of edges, utility values at the edge nodes, etc.

Program Code

Example:



```

1 class node:
2     def __init__(self,name,utility,children=None):
3         self.name = name
4         self.utility = utility
5         self.children = children
6
7     def terminal(self):
8         return not self.children
9
10 def minmax(node,maxi,value={}):
11     if node.terminal():
12         return node.utility,value
13
14     if maxi:
15         maxeva = float('-inf')
16         for item in node.children:
17             eva,v = minmax(item,False,value)
18             maxeva = max(maxeva,eva)
19             value.update({str(node.name):int(maxeva)})
20             #print(str(node.name),int(maxeva))
21         return maxeva,value
22     else:
23         mineva = float('inf')
24         for item in node.children:
25             eva,v = minmax(item,True,value)
26             mineva = min(mineva,eva)
27             value.update({str(node.name):int(mineva)})
28             #print(str(node.name),int(mineva))
29         return mineva,value

```

```

30
31
32 if __name__ == '__main__':
33     h=[]
34     t=int(input("Enter total no of node: "))
35     l=int(input("Enter total no of terminal node: "))
36     p=t-1
37
38     for i in range(l):
39         tem1 = input('Enter Terminal Node Name: ')
40         tem2 = int(input('Enter Value: '))
41         h.insert(i,node(tem1,tem2))
42
43
44     print(" %%% Terminal nodes finished %%% ")
45
46     for i in range(1,t):
47         tem1 = input('Enter New Node Name : ')
48         tem2 = int(input('Enter Value : '))
49         ncn=int(input("Enter no of child nodes : "))
50
51         for w in range(0,i):
52             print(w,h[w].name)
53         listt=[]
54
55         for tob in range(ncn):
56             t = int(input("Enter "+str(tob)+ " th child number from list: "))
57             listt.append(h[t])
58
59         h.insert(i,node(tem1,tem2,listt))
60
61
62     for w in range(0,(t+1)):
63         print(w,h[w].name)
64     k = int(input("Enter index of node you want to search: "))
65     mm=bool(int(input("Do yo want Max search: ")))
66     result,val = minmax(h[k],mm,value={})
67     print(val)
68     print(result)

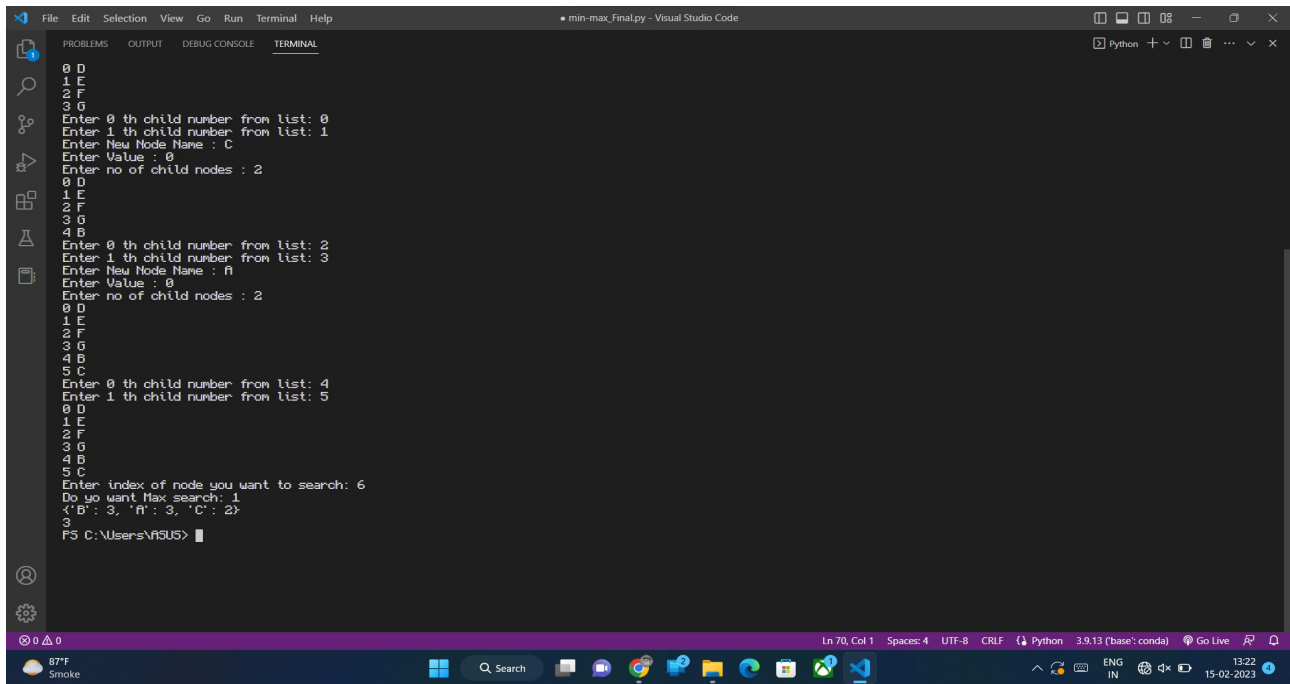
```

Output

```

Enter total no of node: 7
Enter total no of terminal node: 4
Enter Terminal Node Name: D
Enter Value: 3
Enter Terminal Node Name: E
Enter Value: 5
Enter Terminal Node Name: F
Enter Value: 2
Enter Terminal Node Name: G
Enter Value: 9
%%% Terminal nodes finished %%%
Enter New Node Name : B
0 D
1 E
2 F
3 G
Enter 0 th child number from list: 0
Enter 1 th child number from list: 1
Enter New Node Name : C
Enter Value : 0
Enter no of child nodes : 2
0 D
1 E
2 F
3 G
4 B
Enter 0 th child number from list: 2
Enter 1 th child number from list: 3
Enter New Node Name : A
Enter Value : 0
Enter no of child nodes : 2
0 D
1 E
2 F
3 G
4 B
5 C
Enter 0 th child number from list: 4
Enter 1 th child number from list: 5
0 D
1 E
2 F
3 G
4 B
5 C

```



```
0 0
1 1
2 1
3 0
Enter 0 th child number from list: 0
Enter 1 th child number from list: 1
Enter New Node Name : C
Enter Value : 0
Enter no of child nodes : 2
0 0
1 1
2 1
3 0
4 0
Enter 0 th child number from list: 2
Enter 1 th child number from list: 3
Enter New Node Name : A
Enter Value : 0
Enter no of child nodes : 2
0 0
1 1
2 1
3 0
4 0
5 0
Enter 0 th child number from list: 4
Enter 1 th child number from list: 5
0 0
1 1
2 1
3 0
4 0
5 0
Enter index of node you want to search: 6
Do you want Max search: 1
{'B': 3, 'A': 3, 'C': 2}
3
PS C:\Users\ASUS>
```

Conclusion

In this experiment, we looked at how well the Min-Max Algorithm handled the game-playing issue. Our findings show that the Min-Max Algorithm is a strong tool that may be utilised to successfully and efficiently address this problem. We saw that the system could efficiently search across vast search areas and identify ideal solutions. The Min-Max Algorithm is a useful tool for resolving game-playing issues, and it has important real-world implications in areas like artificial intelligence, game theory, and decision-making, according to our results. The Min-Max Algorithm, in particular, is a helpful tool for addressing issues in a wide range of fields, which underscores the significance of knowing and utilising algorithms in computer science.