实验报告

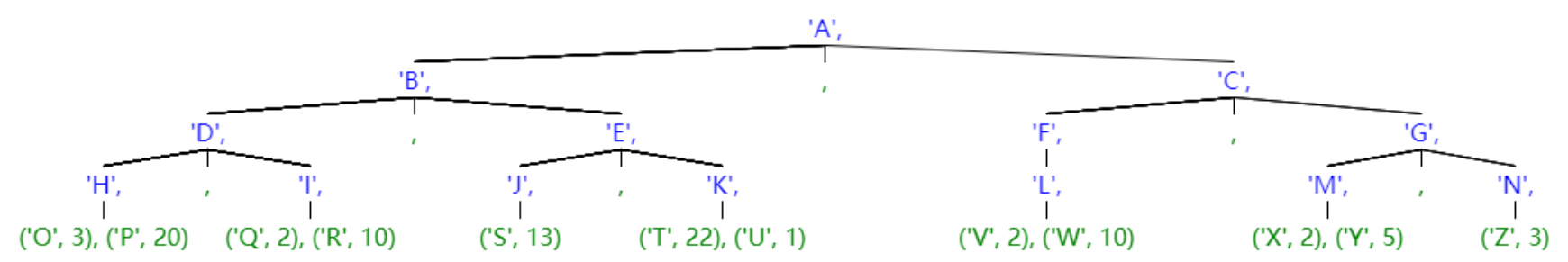
实验二：alpha-beta剪枝问题

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 姓名 | 封兆欣 | 学号 | 202011010083 | 专业 | 教育技术学 |

# Alpha-beta问题

1. 知识回顾

请给出alpha-beta算法剪枝的原理，过程：



1. 从A开始深度优先搜索，比较O和P的值，得到结点H值为3
2. D的值更新为大于等于3
3. 继续搜索I和Q，得到I的值为小于等于2，此时发生剪枝，不再搜索R（因为D已经确定走H）
4. B的值更新为小于等于3
5. 再次搜索E->J->S，得到E的值大于等于13，此时发生剪枝，不再搜索K（因为B已经确定走D）
6. 搜索右子树，C->F->L，得到C的值小于等于2，此时发生剪枝，不再搜索G（因为A已经确定走B）

最后得到搜索过的路径是：B D H O H P D I Q B E J S C F L V

最优动作是选择B，此时结点B的值为3

1. 树的生成：

给定list形式的树的数据，需要按一定方式生成一颗树，生成树之后，为了检验生成的树是否正确，写一个树打印函数，能够实现给出任意节点，从生成的树中读取数据，再以输入list的形式输出。请给出打印函数的代码：

# 以列表形式打印已经创建好的树  
def printTree(self, node):  
 # 如果该结点存在子结点  
 if node.children:  
 print('[', end="")  
 print('\'', node.name, '\'', end="")  
 for i in range(0, len(node.children)):  
 self.printTree(node.children[i])  
 print(']', end="")  
 # 若该节点为叶节点  
 else:  
 print('(', '\'', node.name, '\'', ',', node.val, ')', end=" ")

1. Alpha-beta剪枝结果

以MAX作为初始方，根节点作为函数的初始状态。

利用生成的树，完成alpha-beta剪枝算法，首先输出剪枝时经过节点的名字：

·搜索路径：B D H O H P D I Q B E J S C F L V

输出下一步抉择的节点名，得分：B，得分为3

1. 关键程序注释

请给出𝐴𝑙𝑝ℎ𝑎𝐵𝑒𝑡𝑎𝐷𝑒𝑐𝑖𝑠𝑖𝑜𝑛()，MinValue()，MaxValue()三个函数的程序并给以详细注释（注明在哪里剪枝，如何完成剪枝）：

# 使用AlphaBeta剪枝进行搜索，返回下一步应该决策的结点v  
def minmax\_with\_alphabeta(self, node):  
 v = self.max\_value(node, -10000, 10000)  
 for child in node.children:  
 if child.val == v:  
 return child  
  
# 计算最大值,alpha是剪枝区间下限，beta是剪枝区间上限，v是子节点中的最大值  
def max\_value(self, node, alpha, beta):  
 # 如果是叶子节点直接返回  
 if self.isTerminal(node):  
 # 输出搜索过的结点名称  
 print(node.name, end=" ")  
 return self.get\_value(node)  
 v = -10000  
 # 搜索子节点，如果某个子节点大于beta，则返回，否则一直寻找子结点中的最大值，并且试图更新alpha值  
 for child in node.children:  
 print(node.name, end=" ")  
 v = max(v, self.min\_value(child, alpha, beta))  
 if v >= beta:  
 return v  
 alpha = max(alpha, v)  
 node.val = v  
 return v  
  
# 计算最小值,alpha是剪枝区间下限，beta是剪枝区间上限，v是子节点中的最小值  
def min\_value(self, node, alpha, beta):  
 if self.isTerminal(node):  
 # 输出搜索过的结点名称  
 print(node.name, end=" ")  
 return self.get\_value(node)  
 v = 10000  
 # 搜索子节点，如果某个子节点小于等于alpha，则返回 ;否则一直寻找子节点中的最小值，并试图更新beta  
 for child in node.children:  
 print(node.name, end=" ")  
 v = min(v, self.max\_value(child, alpha, beta))  
 if v <= alpha:  
 return v  
 beta = min(v, beta)  
 node.val = v  
 return v

# 实验总结

1. 使用递归的方法生成树，本实验的树是指任意树，并不是指二叉树，同时也可以使用栈的方式实现
2. Alpha-Beta剪枝的难点在于要想到需要不断更新alpha值和beta值（如果是叶子结点则直接返回，否则一直寻找子节点中的最小值或者最大值，并试图更新alpha值和beta值），同时不断使用函数之间的相互调用完成结点值的估计。