Q5: One Wish Pacman

Q5: One Wish Pacman

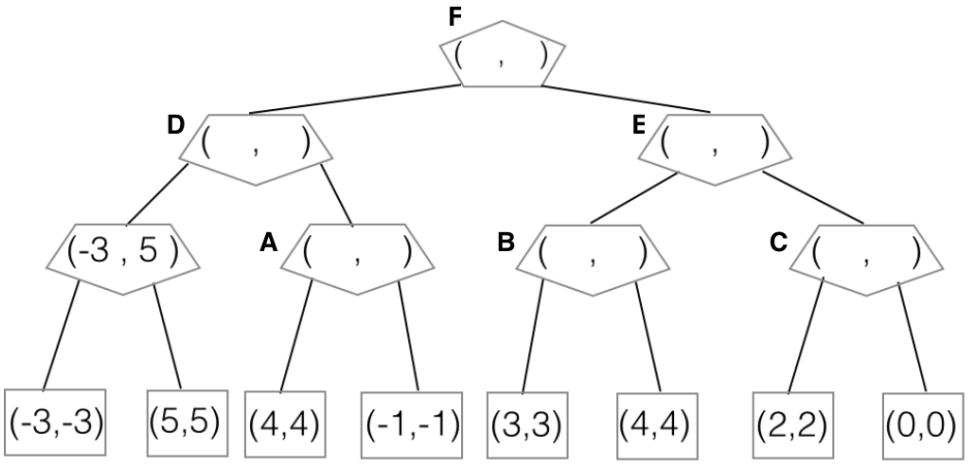
Power Search

Pacman has a special power: *once* in the entire game when a ghost is selecting an action, Pacman can make the ghost choose any desired action instead of the min-action which the ghost would normally take. **The ghosts know about this special power and act accordingly.**

Similar to the minimax algorithm, where the value of each node is determined by the game subtree hanging from that node, we define a value pair (u, v) for each node: u is the value of the subtree if the power is not used in that subtree; v is the value of the subtree if the power is used once in that subtree. For example, in the below subtree with values (-3, 5), if Pacman does not use the power, the ghost acting as a minimizer would choose -3; however, with the special power, Pacman can make the ghost choose the value more desirable to Pacman, in this case 5.

Reminder: Being allowed to use the power once during the game is different from being allowed to use the power in only one node in the game tree below. For example, if Pacman's strategy was to always use the special power on the second ghost then that would only use the power once during execution of the game, but the power would be used in four possible different nodes in the game tree.

For the terminal states we set $u = v = Utility\,(State)$.



Part 1

0.0/2.0 points (graded)

Fill in the (u, v) values corresponding to each labeled node in the modified minimax tree above. Pacman is the root and there are two ghosts.

Node	u		v	
A	-1	**Answer:** -1	4	**Answer:** 4
B	3	**Answer:** 3	4	**Answer:** 4
C	0	**Answer:** 0	2	**Answer:** 2
D	-3	**Answer:** -3	4	**Answer:** 4
E	0	**Answer:** 0	3	**Answer:** 3

F | 0 | **Answer:** 0 | 4 | **Answer:** 4

Explanation
Please see the solution of the general algorithm in the next part to see how the u, v values get propagated up the game tree.

Submit | You have used 0 of 2 attempts

Complete the algorithm below, which is a modification of the minimax algorithm, to work in the general case: Pacman can use the power at most once in the game but Pacman and ghosts can have multiple turns in the game.

```
function VALUE(state)
    if state is leaf then
        u ← UTILITY(state)
        v ← UTILITY(state)
        return (u, v)
    end if
    if state is Max-Node then
        return MAX-VALUE(state)
    else
        return MIN-VALUE(state)
    end if
end function
```

```
function MAX-VALUE(state)
    uList ← [ ], vList ← [ ]
    for successor in SUCCESSORS(state) do
        (u′, v′) ← VALUE(successor)
        uList.append(u′)
        vList.append(v′)
    end for
    u ← max(uList)
    v ← max(vList)
    return (u, v)
end function
```

```
function MIN-VALUE(state)
    uList ← [ ], vList ← [ ]
    for successor in SUCCESSORS(state) do
        (u′, v′) ← VALUE(successor)
        uList.append(u′)
        vList.append(v′)
    end for

    u ← _____

    v ← _____

    return (u, v)
end function
```

Part 2

0.0/2.0 points (graded)
Which one of the following correctly calculates the value of u

- ◉ $\min(uList)$ ✔
- ○ $\min(vList)$
- ○ $\max(uList)$
- ○ $\max(vList)$
- ○ $\min(\min(uList), \min(vList))$

Explanation
See the explanation in Part 3

Submit You have used 0 of 1 attempt

Answers are displayed within the problem

Part 3

0.0/2.0 points (graded)
Which one of the following correctly calculates the value of v

- ○ $\max(uList)$
- ○ $\max(vList)$
- ◉ $\max(\max(uList), \min(vList))$ ✔
- ○ $\min(\max(uList), \min(vList))$
- ○ $\min(\max(uList), \max(vList))$

- $\max(\max(uList), \max(vList))$

Explanation
The u value of a min-node corresponds to the case if Pacman does not use his power in the game subtree hanging from the current min-node. Therefore, it is equal to the minimum of the u values of the children of the node.
The v value of the min-node corresponds to the case when pacman uses his power once in the subtree. Pacman has two choices here - a) To use the power on the current node, or b) To use the power further down in the subtree.
In case a), the value of the node corresponds to choosing the best among the children's u values = $\max(uList)$ (we consider u values of children as Pacman is using his power on this node and therefore, cannot use it in the subtrees of the node's children).
In case b), Pacman uses his power in one of the child subtrees so we consider the v values of the children, Since Pacman is not using his power on this node, the current node acts as a minimizer, making the value in case b) = $\min(vList)$
The v value at the current node is the best of the above two cases.

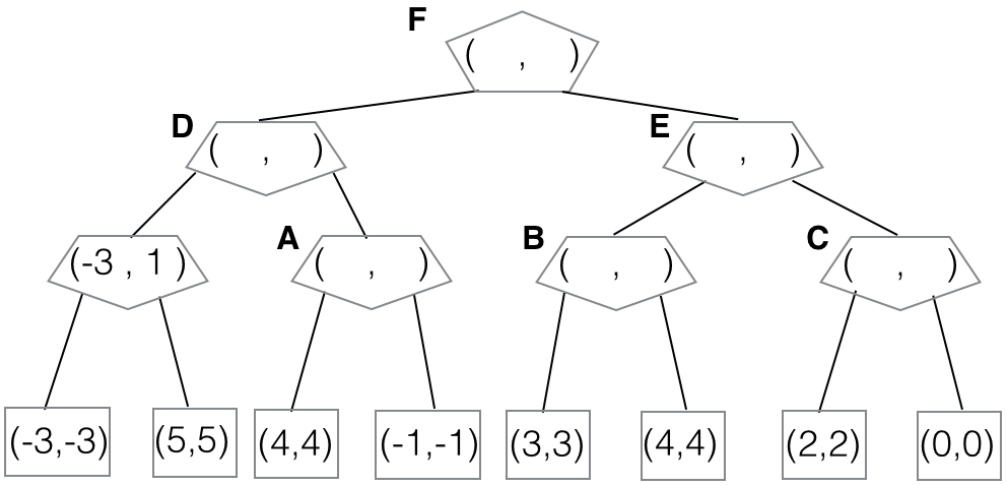
Submit You have used 0 of 1 attempt

Answers are displayed within the problem

Weak-Power Search

Now, rather than giving Pacman control over a ghost move once in the game, the special power allows Pacman to once make a ghost act randomly. The ghosts know about Pacman's power and act accordingly.

The propagated values (u, v) are defined similarly as in the preceding question: u is the value of the subtree if the power is not used in that subtree; v is the value of the subtree if the power is used once in that subtree.



Part 4

0.0/2.0 points (graded)

Fill in the (u, v) values in the modified minimax tree below, where there are two ghosts.

Node	u	v	
A	-1	1.5	
	Answer: -1	**Answer:** 1.5	
B	3	3.5	
	Answer: 3	**Answer:** 3.5	
C	0	1	**Answer:** 1
	Answer: 0		
D	-3	1	**Answer:** 1
	Answer: -3		
E	0	1.5	
	Answer: 0	**Answer:** 1.5	

F

0

Answer: 0

1.5

Answer: 1.5

Explanation
Please see the solution of the general algorithm in the next part to see how the u, v values get propagated up the game tree.

Submit

You have used 0 of 2 attempts

Complete the algorithm below, which is a modification of the minimax algorithm, to work in the general case: Pacman can use the weak power at most once in the game but Pacman and ghosts can have multiple turns in the game.
Hint: you can make use of a min, max, and average function

```
function VALUE(state)
    if state is leaf then
        u ← UTILITY(state)
        v ← UTILITY(state)
        return (u, v)
    end if
    if state is Max-Node then
        return MAX-VALUE(state)
    else
        return MIN-VALUE(state)
    end if
end function
```

```
function MAX-VALUE(state)
    uList ← [ ], vList ← [ ]
    for successor in SUCCESSORS(state) do
        (u′, v′) ← VALUE(successor)
        uList.append(u′)
        vList.append(v′)
    end for
    u ← max(uList)
    v ← max(vList)
    return (u, v)
end function
```

```
function MIN-VALUE(state)
    uList ← [ ], vList ← [ ]
    for successor in SUCCESSORS(state) do
        (u′, v′) ← VALUE(successor)
        uList.append(u′)
        vList.append(v′)
    end for

    u ← _____

    v ← _____

    return (u, v)
end function
```

Part 5

0.0/2.0 points (graded)
Which one of the following correctly calculates the value of u

- $\min(uList)$ ✔
- $\min(vList)$
- $avg(uList)$
- $avg(vList)$
- $\min(\min(uList), \min(vList))$

Explanation
See the explanation in Part 6

Submit You have used 0 of 1 attempt

Answers are displayed within the problem

Part 6

0.0/2.0 points (graded)
Which one of the following correctly calculates the value of v

- $avg(uList)$
- $avg(vList)$
- $\max(avg(uList), avg(vList))$
- $\max(avg(uList), \min(vList))$ ✔

- $avg\left(\max\left(uList\right), \min\left(vList\right)\right)$
- $avg\left(\min\left(uList\right), \max\left(vList\right)\right)$
- $\max\left(\max\left(uList\right), avg\left(vList\right)\right)$

Explanation

The solution to this scenario is same as before, except that when considering case a) for the v value of a min-node, the value of the node corresponds to choosing the average of the children's u values = $avg\left(uList\right)$

Submit You have used 0 of 1 attempt

Answers are displayed within the problem