hw2 4710

jh4ctf

September 2021

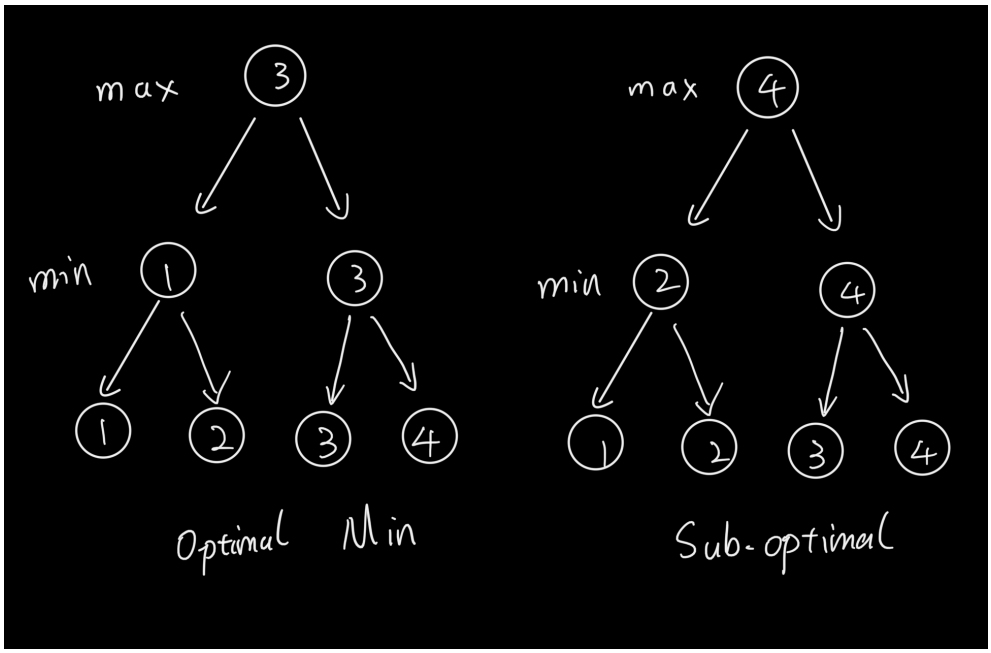
1

In addition to MinMax function and other algorithm, we can also create a evaluation function which decides how some of the Pacman's feature can affect its moving direction. In the end, we are putting different weights on different features for directions.
One of weights is cumulative numbers of reward dot which Pacman can eat contribute its moving because it is one of the goals to eat up as many dots as possible. Thus, direction with more access to more dots (its sub-direction has more dots)weights higher than other directions. (In scale of -5.0 to +10.0 for each direction, etc.) Also, the ghost which is "deadly" to the Pacman also weighted different on the direction. It should be weighted in a higher priority than the dots because it causes immediate end to the game. Thus, it should be in a relative scale of (-20.0 to +30.0). Similarly, features like quickest route to the last dot are also weighted but in a lower frequency.// With this evaluation method, each direction should generated weighted decision respectively. node

2

We can use induction to prove this statement. In this proof, we are basing the utility of "sub-optimal" as one sub-node that is not smaller than the optimal node :
Consider there nodes and their children nodes being the terminal nodes.
If the MIN turns sub-optimally, then the value of node is equal or bigger (not less) than that of nodes if they weren't sub-optimal: $MINx_{sub} \geq MINx_{opt}$. Thus, the value of MAX node which is the MIN node's parent will increase. $MAXx_{sub} \geq MAXx_{opt}$.

This case can then be applied all the way to the root as a game tree can be made of as many base case tree above as possible.



3

In the function for Arc Consistency Checking, when X loses a value, we recheck the consistency and detects the failure prior to the forward checking with the CSP function. While in the function of rechecking, removing and en-queuing utilize a while loop and removing values utilize a for loop. Above steps in the CSP function runs in the efficiency of n^2. Furthermore, in the removing function, iterating through the domain of X_i utilizes a for loop, and checking if y is in domain of X_j also takes up efficiency of n, in this case runs in d^2. Thus, updating the numbers of ar consistency is in total time of O(n^2d^2).