Homework 1, CSE 573

Alexander Van Roijen

February 12, 2020

I 1

I.I a

DFS returns the optimal path $A \to B \to y \to G$ with cost 10
The search tree is a, b, y, c, d, d, G d is explored twice, as a non intelligent DFS would still explore d in a different fashion to see if it is faster that way to get to our goal G rather than going in the roundabout fashion determined before hand

I.II b

UCS returns the optimal path as well $A \to B \to y \to G$ with cost 10
The search tree is a, c, d, b, y, G

I.III c

H_1 is an admissible heuristic
H_2 is not because of node C, to be admissible it should be $H(C) \leq 14$ as that is the true cost of the nearest goal. when in fact $H_2(C) = 16$

I.IV d

optimal $A \to B \to y \to G$ with cost 10
Expanded order: a, b, y, g

I.V e

optimal $A \to B \to y \to G$ with cost 10
a, b, y, G Neither c or d are expanded as their combined cost with the heuristic is larger than the actual cost to the goal which will be unqueued before they are.

II 2

II.I a

Number of parcels in place. It is admissible assuming that a package in the same square without being dropped is still considered in place / delivered, as we know that if they are not in place, they require at least cost one away from their destination (pickup and drop dont count). Thus if 3 packets are out of place, I need at least 3 moves to resolve it, if not more. For it to be consistent, we would need that the difference in consecutive heuristics is $\leq$ actual cost between the two states. So technically, this heuristic is consistent assuming that once the robot is in the destination square, the package is considered delivered, even if it isnt yet dropped. If you did not consider the package delievered, that would mean we have two different states, where the robot is in the same square and hasnt yet dropped the box and where it has then dropped the box, my heuristic would return $h(in_square) - h(dropbox) = 1 - 0 = 1 \not\leq 0$ as 0 would be the actual cost to drop the box.

II.II b

Under the assumption that being in the appropriate square with the parcel in hand is considered delivered, this heuristic is still admissible and consistent, as even if I carry multiple packages, only one package gets assigned one tile, meaning that the cost of delivering all my packages, even if I had them all on my person, would still be the minimum distance to travel between all nodes, which is always at least the number of parcels not yet delivered. Consistency is guaranteed as we are still in the same situation as listed above in part a, so the proof still holds.

II.III c

- We can use the previous heuristic as it is both admissible and consistent, as explained in the previous section.
- Sum of manhattan distances between packages and their destination. So this would basically calculate for each package, the manhattan distance between itself and its destination while ignoring walls. If there were no walls, we know that the sum of all

these distances wherever these parcels are is the minimum possible distance needed to be traversed by the robot to deliver all packages. This will be consistent as if the robot carries multiple packages, each step will calculate the new distances in the new state, thus if multiple packages move closer to their destination in one move, the heuristic captures exactly that much change as well.

III 3

III.I a & b

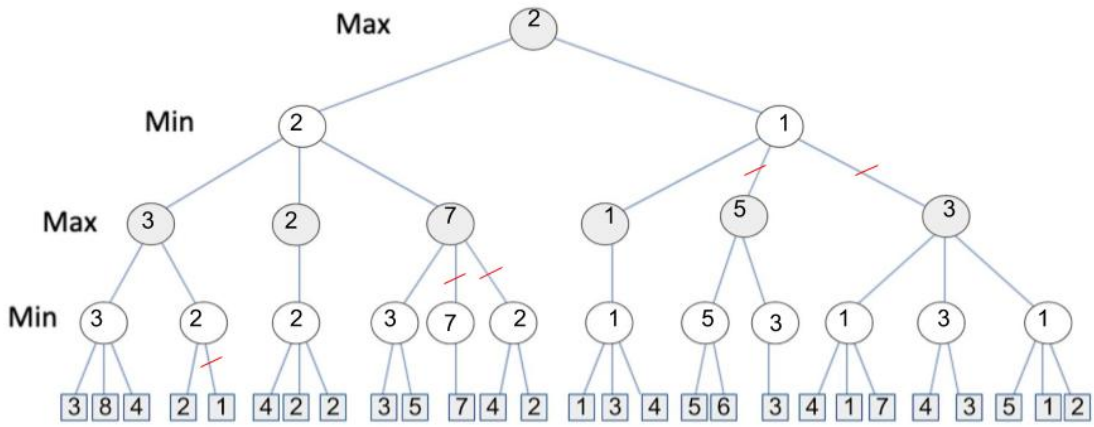

Figure 1: Value of nodes and red slashes indicating pruning

III.II c

change the leftmost leaf of the right subtree which holds value one into a leaf that holds value 3 instead. this will ensure we somewhat explore both of the other subtrees on the right side rather than cutting them out.

IV 4

IV.I a

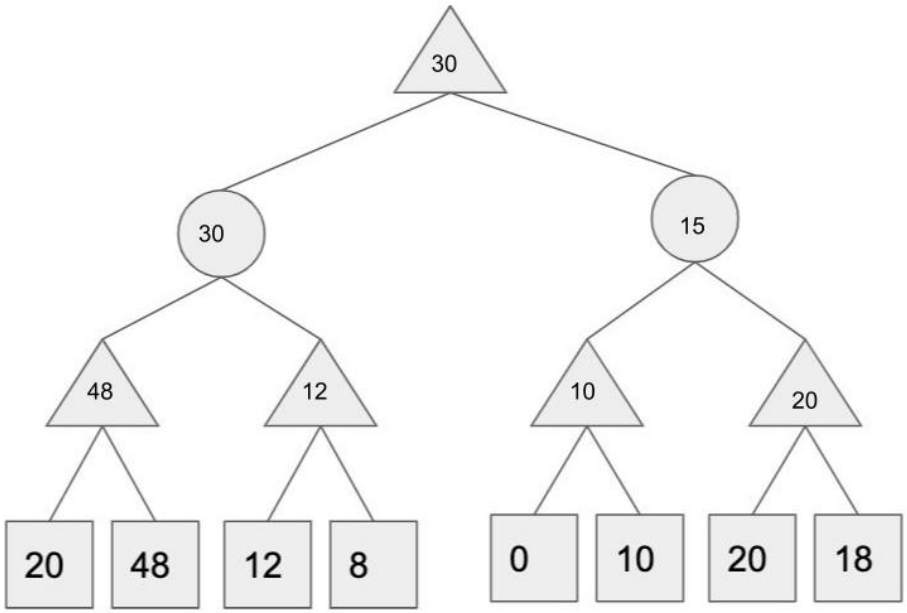

Figure 2: Value of nodes assuming uniform probability

IV.II b

our average on the left subtree expected node now becomes $p*48+(1-p)*12 = 36p+12$. Meanwhile the right has $10p + 20(1 - p) = -10p + 20$. To change the course of the root node, our right subtree must have a great expected return, which means $36p + 12 < -10p + 20 \rightarrow 46p < 8 \rightarrow p < 8/46 = 4/23$ which is roughly 0.174.

IV.III c & d

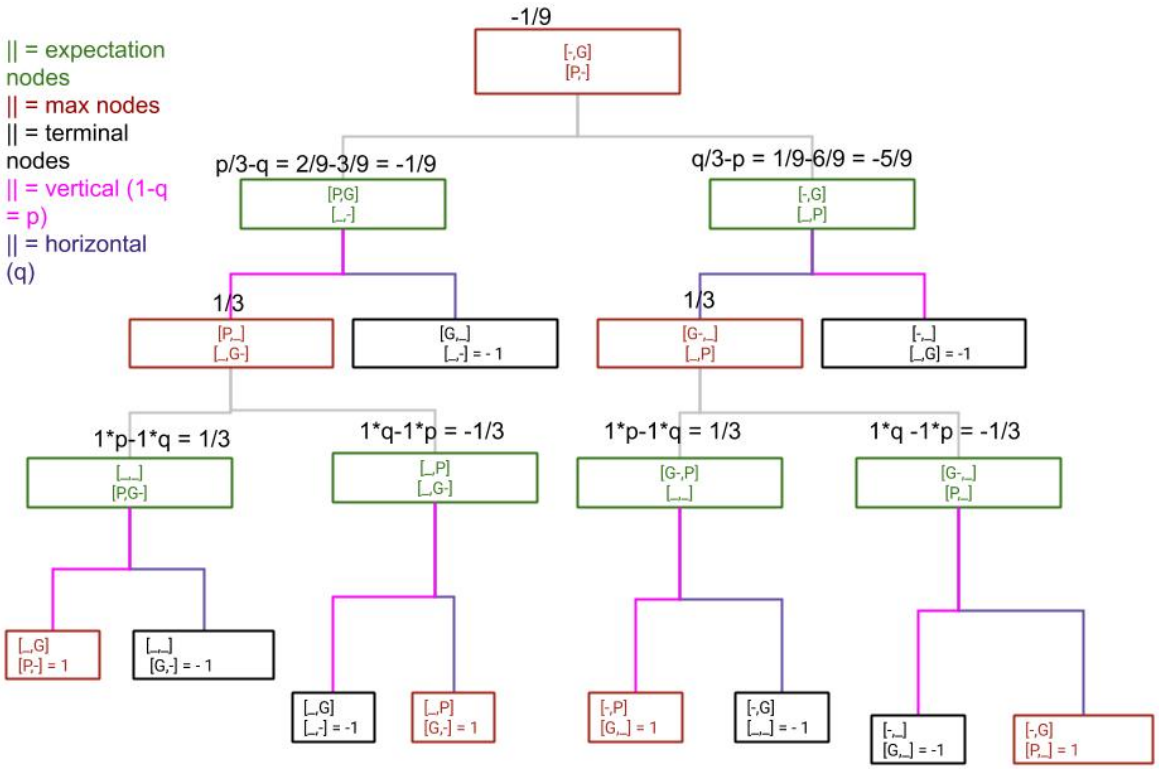

Figure 3: dashes represent food, and underscores represent empty. expected score from pacman is -1/9 by the end of it with depth 4

IV.IV e

It depends entirely on the probabilities of the ghosts actions. Since pacman cant help but being next to the ghost in this grid when he moves , pacman should always move in such a way that he is in the least probable action path possible from the ghost. That is, until he can access the last food pellet and end the game successfully. So in this example, pacman would move up assuming $q = 1/3$ and then move back down, hoping that each time the ghost moved vertically, and then finally moving into the vertical path of the ghost at the bottom right corner, but will have eaten the last food pellet before the ghost can reach him.

V 5

V.I a

It is easy to see that for an $n*n$ grid, the first player can choose one of $n*n$ choices, then for **each** of these $n*n$ choices, we have the entire space of $n*n-1$ spaces remaining for the second player to choose from. This space of possible states quickly becomes exorbitantly large with any n of decent size (imagine n = 10). So we should take this into consideration when under resource limits for our compute. Commonly, we limit the depth of our tree and add a evaluation function that approximates the end state value of a given subtree given the current state. For example, we could try to estimate how many tiles we need, considering the obstacles in front of us, to currently reach our end goal, perhaps with a similar cost added for how many the opponent needs.

V.II b

let $Eval(s) = \max NTTME\text{-}\min NTTOE$. Where NTTME = "Number of Tiles To My End" and NTTOE = "Number of Tiles To Opponents End". these are computed by computing the shortest distance from any tile to the opposite end considering the walls / barriers created by the opponent.

V.III c

$n^2 * (n^2) - 1 * (n^2) - 2 = \theta(n^6)$

VI 6

VI.I 1

States S = Lose, R1,R2,R3,R4,R5, Win

Actions A = pass,eliminated

Transition Function $T(s, a, s\prime)$
$T(R1, pass, R2) = \frac{4}{5}$, which means $T(R1, fail, Lose) = \frac{1}{5}$
$T(R2, pass, R3) = \frac{3}{4}$, which means $T(R2, fail, Lose) = \frac{1}{4}$
$T(R3, pass, R4) = \frac{1}{2}$, which means $T(R3, fail, Lose) = \frac{1}{2}$
$T(R4, pass, Win) = \frac{1}{5}$, which means $T(R4, fail, Win) = \frac{4}{5}$
The remaining transitions are all probability zero

Reward Function $R(s, a, s\prime)$
$R(R1, pass, R2) = 500$
$R(R2, pass, R3) = 0$
$R(R3, pass, R4) = 0$
$R(R4, pass, Win) = 100000$
The remaining rewards are all zero gain

Start State = R1

End state = Lose, or Win

VI.II 2

V(S1) = $\frac{4}{5}(500+\frac{3}{4}(0+\frac{1}{2}(0+\frac{1}{5}(100000)))) = \frac{4}{5}(500+\frac{3*1*1}{4*2*5}(100000)) = \frac{4}{5}(500+7500) = 6400$
V(S2) = $\frac{3}{4}(0+\frac{1}{2}(0+\frac{1}{5}(100000))) = \frac{3*1*1}{4*2*5}(100000) = 7500$
V(S3) = $\frac{1}{2}(0+\frac{1}{5}(100000)) = \frac{1*1}{2*5}(100000) = 10000$
V(S4) = $\frac{1}{5}(100000) = 20000$

VII Appendix

This code used to generate plots and more will be posted on github soon @ user: "bogyshi" https://github.com/bogyshi! stay posted!