Q1. Search problems

It is training day for Pacbabies, also known as Hungry Running Maze Games day. Each of k Pacbabies starts in its own assigned start location s_i in a large maze of size MxN and must return to its own Pacdad who is waiting patiently but proudly at g_i; along the way, the Pacbabies must, between them, eat all the dots in the maze.

At each step, all k Pacbabies move one unit to any open adjacent square. The only legal actions are Up, Down, Left, or Right. It is illegal for a Pacbaby to wait in a square, attempt to move into a wall, or attempt to occupy the same square as another Pacbaby. To set a record, the Pacbabies must find an optimal collective solution.

(a) Define a minimal state space representation for this problem.

The state space is defined by the current locations of k Pacbabies and, for each square, a Boolean variable indicating the presence of food.

(b) How large is the state space?

$(MN)^k \cdot 2^{MN}$

(c) What is the maximum branching factor for this problem?

- ● 4^k
- ○ 8^k
- ○ $4^k 2^{MN}$
- ○ $4^k 2^4$

Each of k Pacbabies has a choice of 4 actions.

(d) Let $MH(p, q)$ be the Manhattan distance between positions p and q and F be the set of all positions of remaining food pellets and p_i be the current position of Pacbaby i. Which of the following are admissible heuristics?

- ■ h_A: $\frac{\sum_{i=1}^{k} MH(p_i, g_i)}{k}$
- ■ h_B: $\max_{1 \le i \le k} MH(p_i, g_i)$
- □ h_C: $\max_{1 \le i \le k}[\max_{f \in F} MH(p_i, f)]$
- □ h_D: $\max_{1 \le i \le k}[\min_{f \in F} MH(p_i, f)]$
- ■ h_E: $\min_{1 \le i \le k}[\min_{f \in F} MH(p_i, f)]$
- □ h_F: $\min_{f \in F}[\max_{1 \le i \le k} MH(p_i, f)]$

h_A is admissible because the total Pacbaby–Pacdad distance can be reduced by at most k at each time step.
h_B is admissible because it will take at least this many teps for the furthest Pacbaby to reach its Pacdad.
h_C is inadmissible because it looks at the distance from each Pacbaby to its most distant food square; but of course the optimal solution might another Pacbaby going to that square; same problem for h_D.
h_E is admissible because some Pacbaby will have to travel at least this far to eat one piece of food (but it's not very accurate).
h_F is inadmissible because it connects each food square to the most distant Pacbaby, which may not be the one who eats it.
A different heuristic, $h_G = \max_{f \in F}[\min_{1 \le i \le k} MH(p_i, f)]$, *would* be admisible: it connects each food square to its closest Pacbaby and then considers the most difficult square for any Pacbaby to reach.

Q2. Pacman's Life

Suppose a maze has height M and width N and there are F food pellets at the beginning. Pacman can move North, South, East or West in the maze.

(a) In this subquestion, the position of Pacman is known, and he wants to pick up all F food pellets in the maze. However, Pacman can move North at most two times overall.

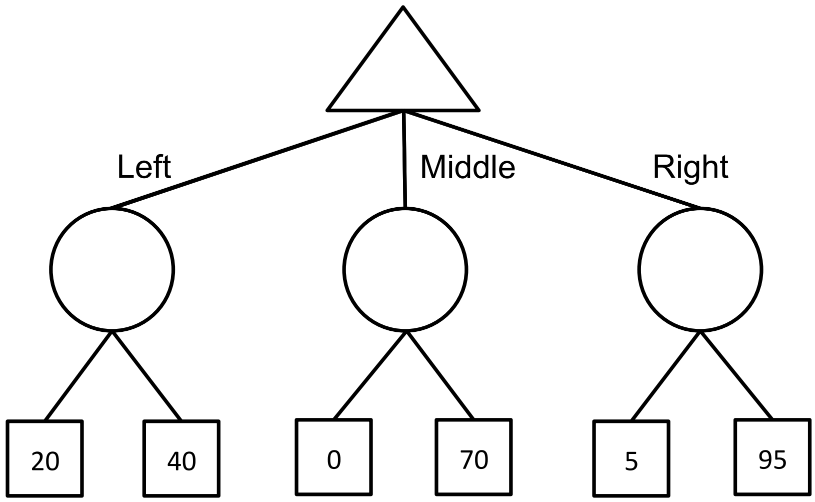
What is the size of a minimal state space for this problem? Give your answer as a product of terms that reference problem quantities such as (but not limited to) M, N, F, etc. Below each term, state the information it encodes. For example, you might write $4 \times MN$ and write number of directions underneath the first term and Pacman's position under the second.
$MN \times 2^F \times 3$. Pacman's position, a boolean vector representing whether a certain food pellet has been eaten, and the number of times Pacman has moved North (which could be 0, 1 or 2).

(b) In this subquestion, Pacman is lost in the maze, and does not know his location. However, Pacman still wants to visit every single square (he does not care about collecting the food pellets any more). Pacman's task is to find a sequence of actions which guarantees that he will visit every single square.

What is the size of a minimal state space for this problem? As in part(a), give your answer as a product of terms along with the information encoded by each term. You will receive partial credit for a complete but non-minimal state space.
$2^{((MN)^2)}$. For every starting location, we need a boolean for every position (MN) to keep track of all the visited locations. In other words, we need MN sets of MN booleans for a total of $(MN)^2$ booleans. Hence, the state space is $2^{((MN)^2)}$.

Q3. Bounded Expectimax

(a) **Expectimax.** Consider the game tree below, where the terminal values are the *payoffs* of the game. Fill in the expectimax values, assuming that player 1 is maximizing expected payoff and player 2 plays uniformly at random (i.e., each action available has equal probability).



(b) Again, assume that Player 1 follows an expectimax strategy (i.e., maximizes expected payoff) and Player 2 plays uniformly at random (i.e., each action available has equal probability).

(i) What is Player 1's expected payoff if she takes the expectimax optimal action?

50

(ii) Multiple outcomes are possible from Player 1's expectimax play. What is the worst possible payoff she could see from that action?

5

(c) Even if the average outcome is good, Player 1 doesn't like that very bad outcomes are possible. Therefore, rather than purely maximizing expected payoff using expectimax, Player 1 chooses to perform a modified search. In particular, she only considers actions whose worst-case outcome is 10 or better.

(i) Which action does Player 1 choose for this tree?

Left

(ii) What is the expected payoff for that action?

30

(iii) What is the worst payoff possible for that action?

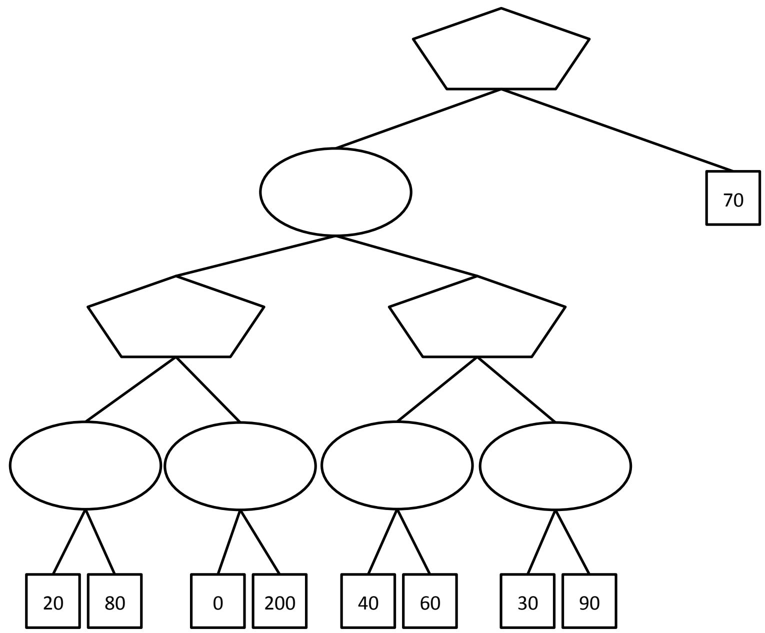
20

(d) Now let's consider a more general case. Player 1 has the following preferences:

- Player 1 prefers any lottery with worst-case outcome of 10 or higher over any lottery with worst-case outcome lower than 10.
- Among two lotteries with worst-case outcome of 10 or higher, Player 1 chooses the one with the highest expected payoff.
- Among two lotteries with worst-case outcome lower than 10, Player 1 chooses the one with the highest worst-case outcome (breaking ties by highest expected payoff).

Player 2 still always plays uniformly at random.

To compute the appropriate values of tree nodes, Player 1 must consider both expectations and worst-case values at each node. For each node in the game tree below, fill in a pair of numbers (e, w). Here e is the expected value under Player 1's preferences and w is the value of the worst-case outcome under those preferences, assuming that Player 1 and Player 2 play according to the criteria described above.



Last expect-layer, (50, 20), (100, 0), (50, 40), (60, 30)
Funny max layer on top of lowest expect layer, (50, 20), (60, 30)
Expect layer, (55, 20)
Funny max at top, (70, 70)

(e) Now let's consider the general case, where the lower bound used by Player 1 is a number L not necessarily equal to 10, and not referring to the particular tree above. Player 2 still plays uniformly at random.

(i) Suppose a Player 1 node has two children: the first child passes up values (e_1, w_1), and the second child passes up values (e_2, w_2). What values (e, w) will be passed up by a Player 1 node if

1. $w_1 < w_2 < L$ (e_2, w_2)

2. $w_1 < L < w_2$ (e_2, w_2)

3. $L < w_1 < w_2$ $(max(e_1, e_2), w_{argmax(e_1,e_2)})$

(ii) Now consider a Player 2 node with two children: the first child passes up values (e_1, w_1) and the second child passes up values (e_2, w_2). What values (e, w) will be passed up by a Player 2 node if

1. $w_1 < w_2 < L$ $(mean(e_1, e_2), min(w1, w2))$

2. $w_1 < L < w_2$ $(mean(e_1, e_2), min(w_1, w_2))$

3. $L < w_1 < w_2$ $(mean(e_1, e_2), min(w_1, w_2))$