# Q1. MedianMiniMax

You're living in utopia! Despite living in utopia, you still believe that you need to maximize your utility in life, other people want to minimize your utility, and the world is a 0 sum game. But because you live in utopia, a benevolent social planner occasionally steps in and chooses an option that is a compromise. Essentially, the social planner (represented as the pentagon) is a median node that chooses the successor with median utility. Your struggle with your fellow citizens can be modelled as follows:
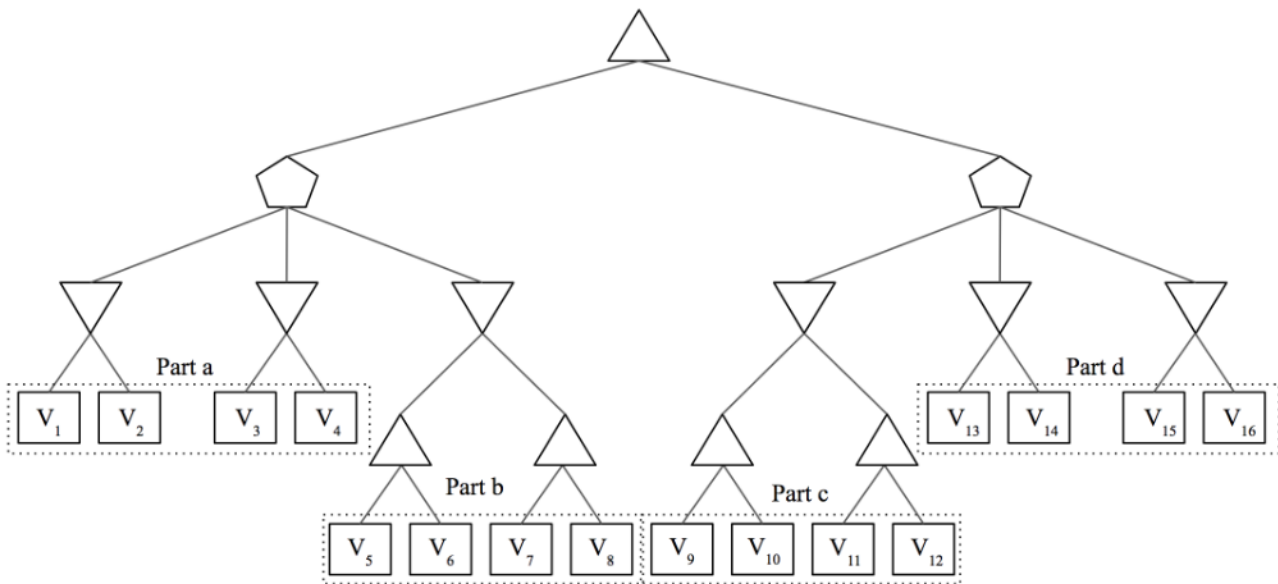
There are some nodes that we are sometimes able to prune. In each part, mark all of the terminal nodes such that **there exists a possible situation** for which the node **can be pruned**. In other words, you must consider **all** possible pruning situations. Assume that evaluation order is **left to right** and all V_i's are **distinct.**

Note that as long as there exists ANY pruning situation (does not have to be the same situation for every node), you should mark the node as prunable. Also, alpha-beta pruning does not apply here, simply prune a sub-tree when you can reason that its value will not affect your final utility.

(a)
- [ ] V_1
- [ ] V_2
- [ ] V_3
- [ ] V_4
- [ ] None

(b)
- [ ] V_5
- [ ] V_6
- [ ] V_7
- [ ] V_8
- [ ] None

(c)
- [ ] V_9
- [ ] V_{10}
- [ ] V_{11}
- [ ] V_{12}
- [ ] None

(d)
- [ ] V_{13}
- [ ] V_{14}
- [ ] V_{15}
- [ ] V_{16}
- [ ] None

Q2. Games

Alice is playing a two-player game with Bob, in which they move alternately. Alice is a maximizer. Although Bob is also a maximizer, Alice believes Bob is a minimizer with probability 0.5, and a maximizer with probability 0.5. Bob is aware of Alice's assumption.

In the game tree below, square nodes are the outcomes, triangular nodes are Alice's moves, and round nodes are Bob's moves. Each node for Alice/Bob contains a tuple, the left value being Alice's expectation of the outcome, and the right value being Bob's expectation of the outcome.

Tie-breaking: choose the left branch.
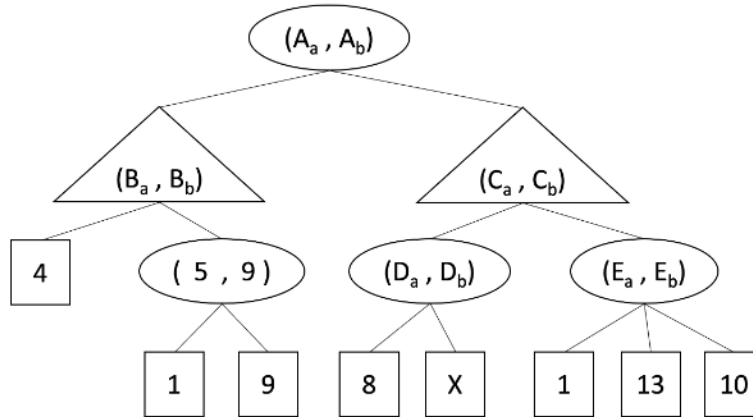
(a) In the blanks below, fill in the tuple values for tuples (B_a, B_b) and (E_a, E_b) from the above game tree.

$(B_a, B_b) = ($ _____ , _____ $)$
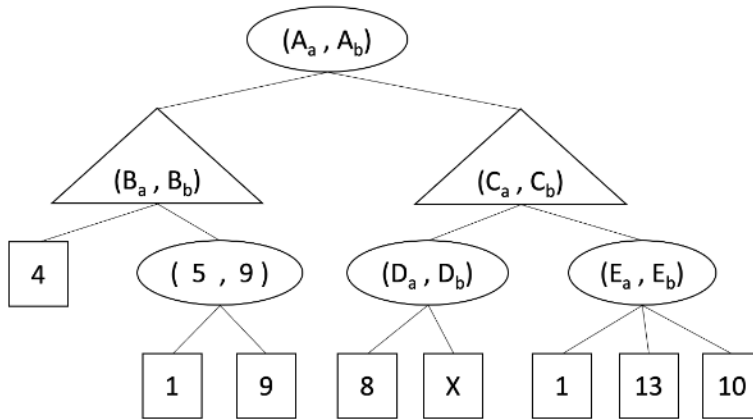
$(E_a, E_b) = ($ _____ , _____ $)$

(b) In this part, we will determine the values for tuple (D_a, D_b).

   (i) $D_a =$ ◯ 8 ◯ X ◯ 8+X ◯ 4+0.5X ◯ min(8,X) ◯ max(8,X)

   (ii) $D_b =$ ◯ 8 ◯ X ◯ 8+X ◯ 4+0.5X ◯ min(8,X) ◯ max(8,X)

(A_a , A_b)

(B_a , B_b)   (C_a , C_b)

4   (5 , 9)   (D_a , D_b)   (E_a , E_b)

1   9   8   X   1   13   10

(The graph of the tree is copied for your convenience. You may do problem e on this graph.)

(c) Fill in the values for tuple (C_a, C_b) below. For the bounds of X, you may write scalars, ∞ or $-\infty$.
If your answer contains a fraction, please write down the corresponding **simplified decimal value** in its place. (i.e., 4 instead of $\frac{8}{2}$, and 0.5 instead of $\frac{1}{2}$).

1. If $-\infty < X <$ [ ] , $(C_a, C_b) = ($ [ ] , [ ] $)$

2. Else, $(C_a, C_b) = ($ [ ] , max([ ] , [ ] $))$

(d) Fill in the values for tuple (A_a, A_b) below. For the bounds of X, you may write scalars, ∞ or $-\infty$.
If your answer contains a fraction, please write down the corresponding **simplified decimal value** in its place. (i.e., 4 instead of $\frac{8}{2}$, and 0.5 instead of $\frac{1}{2}$).

1. If $-\infty < X <$ [ ] , $(A_a, A_b) = ($ [ ] , [ ] $)$

2. Else, $(A_a, A_b) = ($ [ ] , max([ ] , [ ] $))$

(e) When Alice computes the left values in the tree, some branches can be pruned and do not need to be explored. In the game tree graph on this page, put an 'X' on these branches. If no branches can be pruned, mark the "Not possible" choice below.
Assume that the children of a node are visited in left-to-right order and that you should not prune on equality.

○ Not possible

3

Q3. Local Search

In this problem, we will look at the classic Traveling Salesman Problem (TSP) and how it relates to local search algorithms. The goal of the Traveling Salesman Problem is "Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city exactly once and returns to the origin city?" In other words, you're trying to find the shortest path that starts and end at a city and goes through every other city.

(a) Why might we choose to use local search (i.e.: hill-climbing) over an algorithm like A* search for solving the Traveling Salesman Problem? In general, when would using local search be better than using A* search?

(b) Suppose that you connect all the cities into a single path (not a cycle). Devise a hill-climbing algorithm to solve TSPs. (Hint: Think about breaking the path)

(c) Is it possible to reach a local optimum (but not the global optimum) using your algorithm?