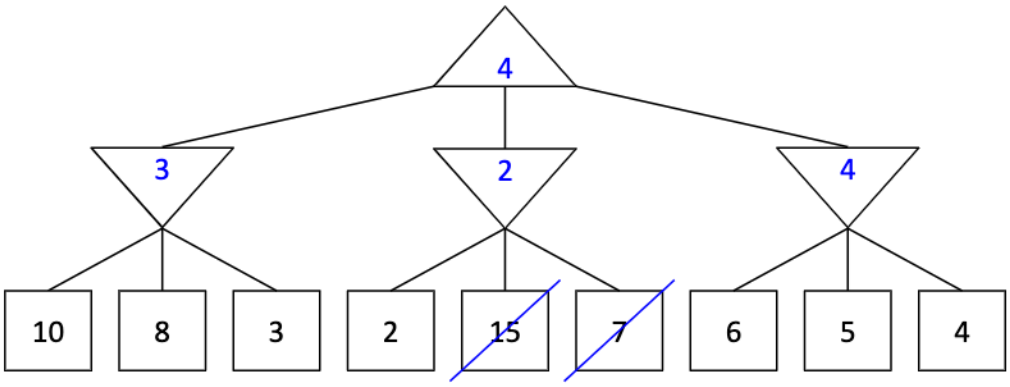
CS 188
Spring 2021

Regular Discussion 3 Solutions

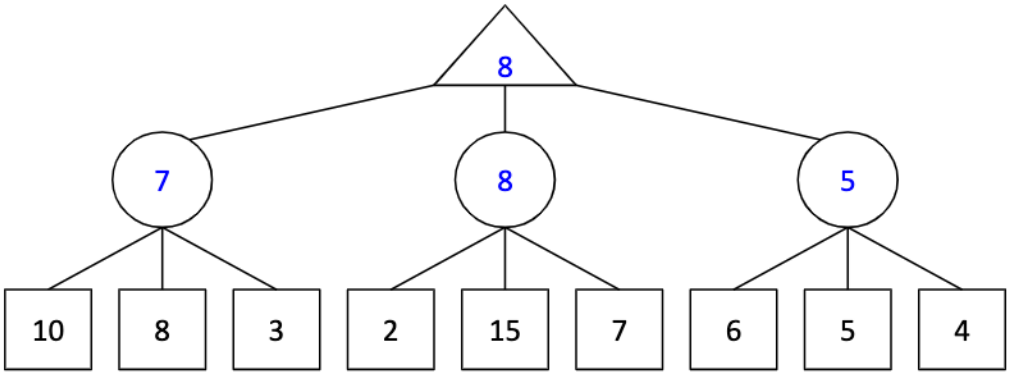
1 Games

(a) Consider the zero-sum game tree shown below. Triangles that point up, such as at the top node (root), represent choices for the maximizing player; triangles that point down represent choices for the minimizing player. Assuming both players act optimally, fill in the minimax value of each node.



(b) Which nodes can be pruned from the game tree above through alpha-beta pruning? If no nodes can be pruned, explain why not. Assume the search goes from left to right; when choosing which child to visit first, choose the left-most unvisited child.

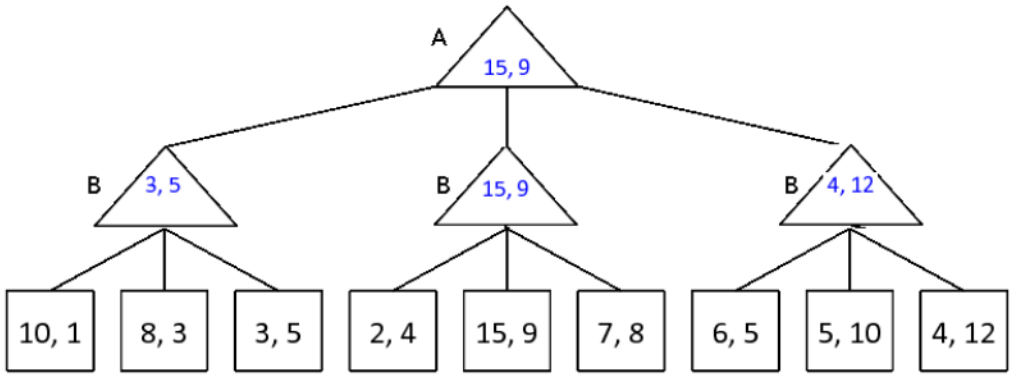
(c) Again, consider the same zero-sum game tree, except that now, instead of a minimizing player, we have a chance node that will select one of the three values uniformly at random. Fill in the expectimax value of each node. The game tree is redrawn below for your convenience.



(d) Which nodes can be pruned from the game tree above through alpha-beta pruning? If no nodes can be pruned, explain why not. No nodes can be pruned. There will always be the possibility that an as-yet-unvisited leaf of the current parent chance node will have a very high value, which increases the overall average value for that chance node. For example, when we see that leaf 4 has a value of 2, which is much less than the value of the left chance node, 7, at this point we cannot make any assumptions about how the value of the middle chance node will ultimately be more or less in value than the left chance node. As it turns out, the leaf 5 has a value of 15, which brings the expected value of the middle chance node to 8, which is greater than the value of the left chance node. In the case where there is an upper bound to the value of a leaf node, there is a possibility of pruning: suppose that an upper bound of +10 applies only to the children of the rightmost chance node. In this case, after seeing that leaf 7 has a value of 6 and leaf 8 has a value of 5, the best possible value that the rightmost chance node can take on is $\frac{6+5+10}{3} = 7$, which is less than 8, the value of the middle chance node. Therefore, it is possible to prune leaf 9 in this case.

2 Nonzero-sum Games

1. Let's look at a non-zero-sum version of a game. In this formulation, player A's utility will be represented as the first of the two leaf numbers, and player B's utility will be represented as the second of the two leaf numbers. Fill in this non-zero game tree assuming each player is acting optimally.



2. Which nodes can be pruned from the game tree above through alpha-beta pruning? If no nodes can be pruned, explain why not. No nodes can be pruned. Because this game is non-zero-sum, there can exist a leaf node anywhere in the tree that is good for both player A and player B.

3 Local Search

1. Give the name of the algorithm that results from each of the following special cases:

   (a) Local beam search with $k = 1$.

   Local beam search with $k = 1$ is hill-climbing search.

   (b) Local beam search with one initial state and no limit on the number of states retained.

   Local beam search with one initial state and no limit on the number of states retained, resembles breadth-first search in that it adds one complete layer of nodes before adding the next layer. Starting from one state, the algorithm would be essentially identical to breadth-first search except that each layer is generated all at once.

   (c) Simulated annealing with $T = 0$ at all times (and omitting the termination test).

   Simulated annealing with $T = 0$ at all times: ignoring the fact that the termination step would be triggered immediately, the search would be identical to first-choice hill climbing because every downward successor would be rejected with probability 1.

   (d) Simulated annealing with $T = \infty$ at all times.

   Simulated annealing with $T = \infty$ at all times is a random-walk search: it always accepts a new state.

   (e) Genetic algorithm with population size $N = 1$.

   Genetic algorithm with population size $N = 1$: if the population size is 1, then the two selected parents will be the same individual; crossover yields an exact copy of the individual; then there is a small chance of mutation. Thus, the algorithm executes a random walk in the space of individuals.

2. When might local search (i.e. hill climbing) be better than using A* search? When might it be worse? There are many possible answers.

   Many answers are possible here: local search helps to solve computationally difficult problems like TSP (shown below), local search can converge to a local optimum, but not global optimum.

   Another useful characteristic of local searches is that if something changes (e.g., a bridge is closed or a flight is cancelled), one can simply resume the search and get a good solution that is close to the current one. A* has to be restarted and may generate a quite different path.