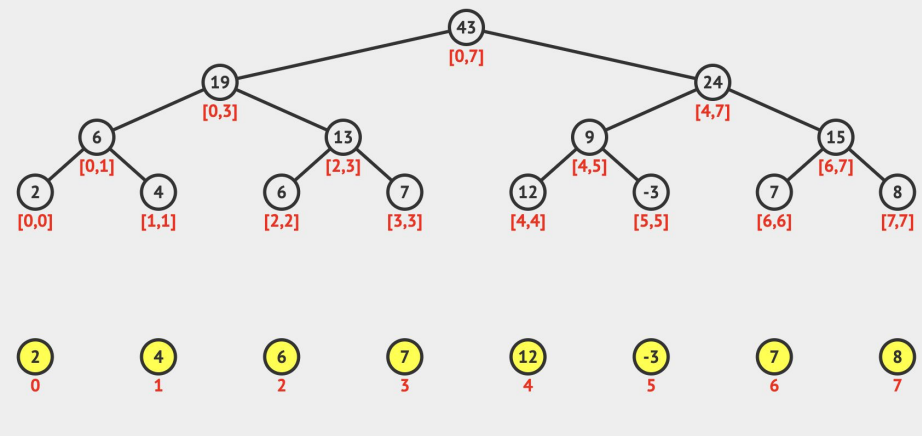
Probabilities of Blackjack Using Segment Trees

Brandon Charette, Andrew Dionizio, Kaidan Campbell, Kyle DaSilva

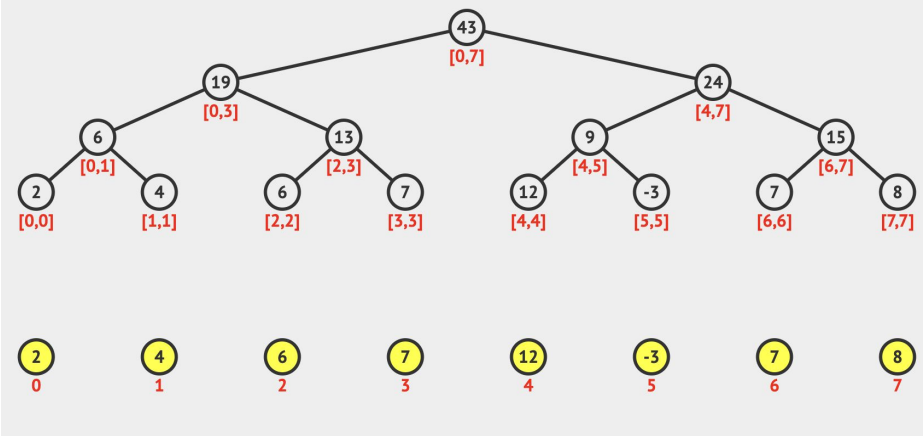
Segment Tree

- a data structure that takes values from an array in intervals and stores them into a binary tree, which means it can be represented as a 1D array in c++
- used to find queries of different ranges like the sum of values, minimum value and maximum value



Segment Tree: Sum

- Root equals sum of entire array
- continuously split into halves until the intervals are equal
- Each parent node is then filled with sum of two children
- Ex) -> the range [2,3] sum is the range [2,2] and [3,3] added together



Blackjack

Objective: To be dealt a higher count than the dealer without exceeding 21

Count For Each Card:

1. Aces- 1 or 11
2. Face Cards- 10
3. Numbered Cards- Number shown on card

Setup:

- 8 decks of cards used (total 416 cards)
- Bets placed first
- Each player and dealer dealt 2 cards

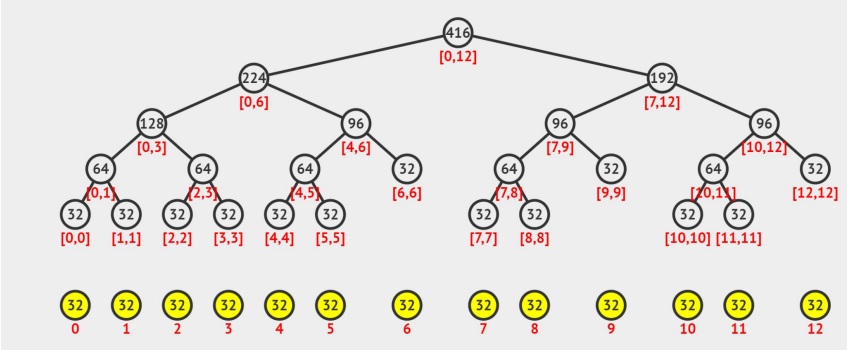
How To Play:

- Choose to hit or stand
  - Hit → Dealt another card
  - Stand → No more cards dealt
- Dealer will deal themself cards until they get a count of at least 17 or bust

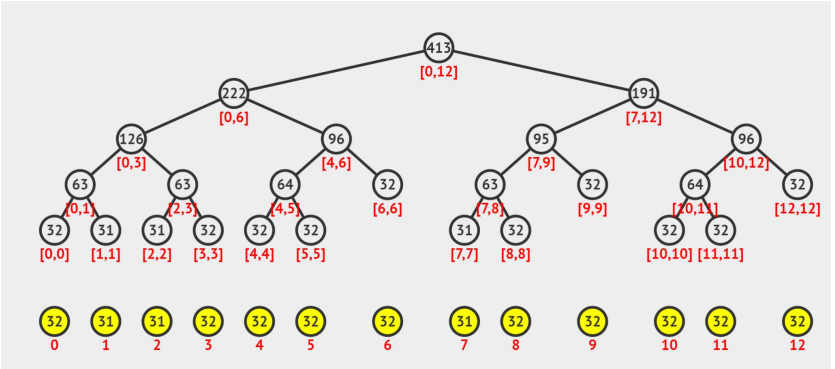
Implementation

- value array is has the size of 13 (amount of different cards) with each value being 32 (8 decks).
- everytime a card is pulled the value array is updated and in turn updates the segment tree.
- After the segment tree is updated, our probability functions run

Note: The nodes that do not have children that are not on the last level of the tree have zeros as placeholders in the segment tree 1D array.



After first 3 cards are dealt



Probability Functions

Dealer having a better card on next turn:

- Your value - dealers value = How many cards the dealer has that will surpass your value
- Use segment tree to find the sum of cards in that range the dealer has better than you
- Divide that sum by the total number of cards left and multiply by 100 to get the probability

Next card not busting:

- 21 - Your Value = Highest Card That Doesn't Bust
- Use Segment tree to find the sum of the range that doesn't bust
- Divide that sum by the total number of cards left and multiply by 100 to get the probability

Having a Favorable Hand:

- Loops through and finds card that will make score 17,18,19,20, or 21 when added
- Uses segment to find total number of possible cards
- Prints the Probability