# Cutting Sticks Problem

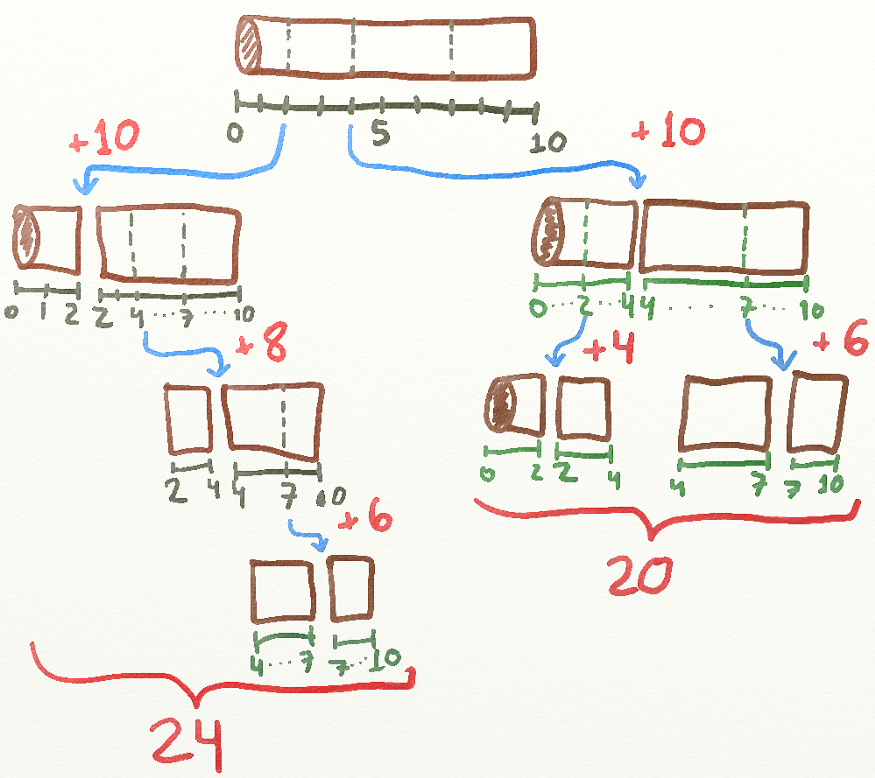
# Problem Statement

## **You have to cut a wood stick into pieces. The most affordable company, charges money according to the length of the stick being cut. Their procedure of work requires that they only make one cut at a time. It is easy to notice that different selections in the order of cutting can led to different prices.**

## **For example: Consider a stick of length 10 meters that has to be cut at 2, 4 and 7 meters from one end. There are several choices. One can be cutting first at 2, then at 4, then at 7. This leads to a price of 10 + 8 + 6 = 24 because the first stick was of 10 meters, the resulting of 8 and the last one of 6. Another choice could be cutting at 4, then at 2, then at 7. This would lead to a price of 10 + 4 + 6 = 20, which is a better price. Your boss trusts your computer abilities to find out the minimum cost for cutting a given stick.**

# Explanation & Algorithm

Image



## Explanation

* You must cut the stick in **all** marked places. Eg. consider a stick of size 10 and 3 places marked in it (positions 2, 4 and 7) where the cuts must be made.
* The cost to make a cut is the same as the size of the stick. Eg. if you have a stick of size 10, cut at ay point will also cost you 10.
* Depending on the order of cuts, your total cost will be different.
* Whenever we make a cut, we divide the stick into 2 other sticks. Those 2 new sticks will then need to be cut or will remain intact (if there is no further cutting to be made).
* The 2 new sticks that need to be cut represent the same problem again, except that now we have a new stick with a new size and new places (i.e, new indexes) to cut it.
* So, we just need to find the best way to cut those 2 new sub-sticks and sum their cost with the cost of the original cut you made.

## Algorithm

### If start index is less than last index return 0 else go to next step (Wood cannot be further separated)

### Calculate `currCost` based on current length( Diff between end and start length)

### Initialize `minCost` with any large number

### Call algorithm to calculate min price for starting and ending part of wood for each element of array (Cutting point) `int currentMinCost = getMinCost(arr, startLength, arr[i], startIdx, i - 1)`

### Compare `currentMinCost` and Assign minimun price to `minCost`

### Calculate total cost as `currCost` (Step 3) + newly added `minCost`(Step 5)

# Time Complexities

O (3 ^ (n-1))

* For loop with 2 recursive calls inside it

# Test Cases

## Test Case 1

## Input: { 2, 4, 7 }, 10

## Output: 20

## Description: 10 + 4 + 6

## Test Case 2

## Input: { 2, 4, 7, 12 }, 20

## Output: 43

## Description: 20 (12,8) + 12(7,5) + 7(4,3) + 4(2,2)

## Test Case 3

## Input: { 25, 50, 75 }, 100

## Output: 200

## Test Case 4

## Input: { 4, 5, 7, 8 }, 10

## Output: 22

## Test Case 5

Input: { 4, 5, 7, 8 }, 2

Output: Exception