Problem Set 6

Questionable k-way heaps and
minimizing some trees
Due Thursday, November 6th at 11:59pm

A warmup: k-way heaps

In a binary heap, each node can have a left and right child. In a k-way heap, a node can have up to k children. The heap property is the same: the children of a node all have values greater than the node itself. In this part of the assignment you will implement a k-way heap. You may assume that only integers will be inserted into your heap. Feel free to modify the code in the book. This part of the assignment is a lot easier if you do.

Requirements

Use an array

Your heap must be array based. There should be no node class.

__init__(self, k)

Your constructor should take k as an argument. This will instantiate your heap as a k-way heap.

is_empty(self)

Returns True if the heap is empty. False otherwise,

min(self)

Returns the minimum element of in the heap.

remove_min(self)

Removes the minimum element from the heap and returns that element.

add(self, x)

Adds x to the heap. No return value.

index_of_children(self, j)

An iterator for the index of the children of the jth node.

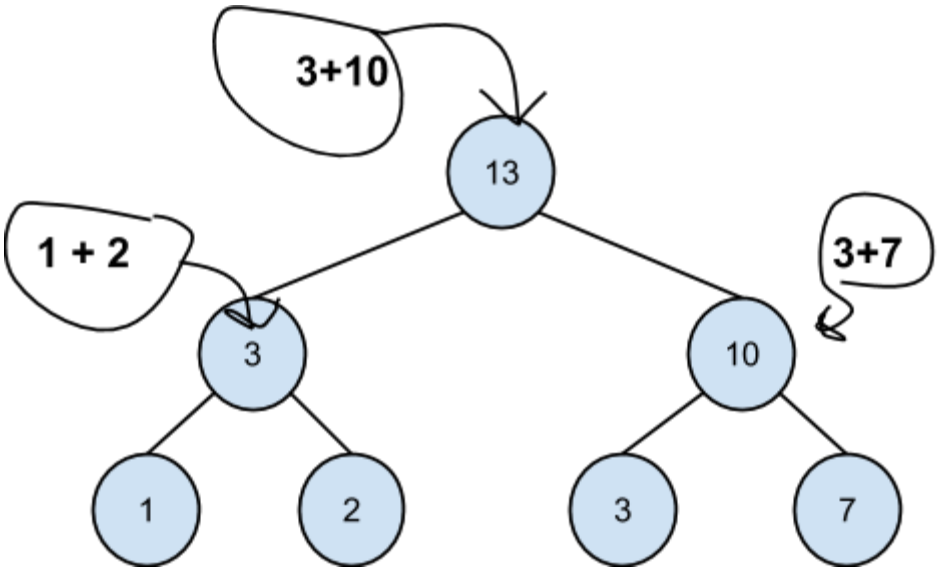
Consider whether or not a k-way heap is in any way better than a binary heap
On one hand, implementing a k-way heap may be nothing more than an exercise. On the other hand, Wikipedia says it's actually good for something.

Do the next part of the assignment
This was just a warmup.

Sum Trees

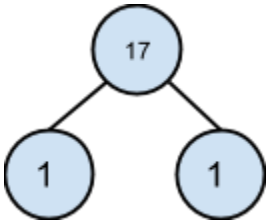
Consider a binary tree T whose nodes hold integers. T is a *sum tree* if every interior node's value is the sum of its left and right childrens' values.

Below is an example of a sum tree.

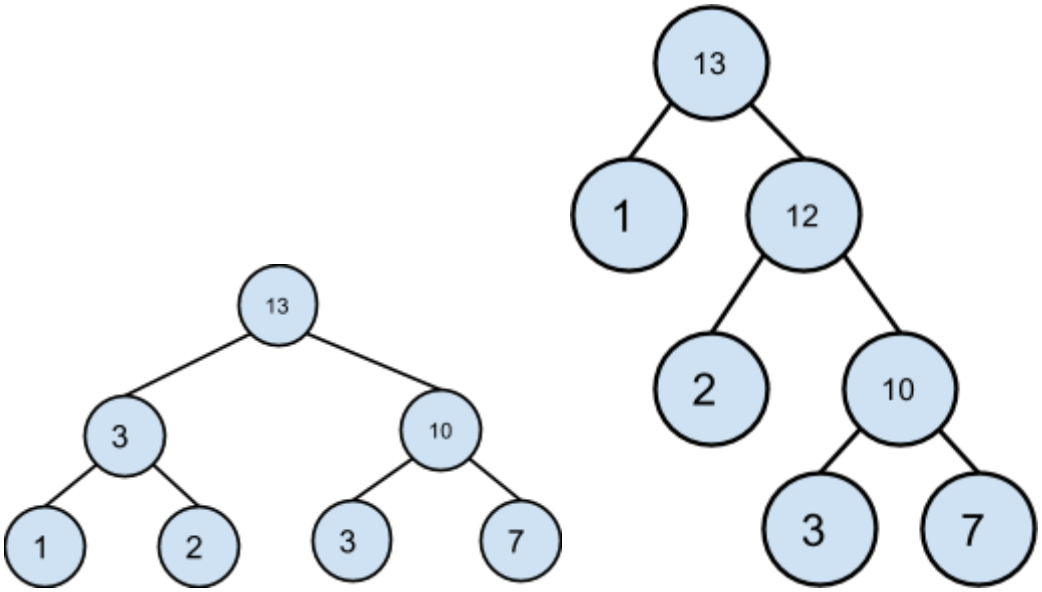


Note that a tree with a single node as well as an empty tree are sum trees as well.

Here is an example of something that IS NOT a sum tree:



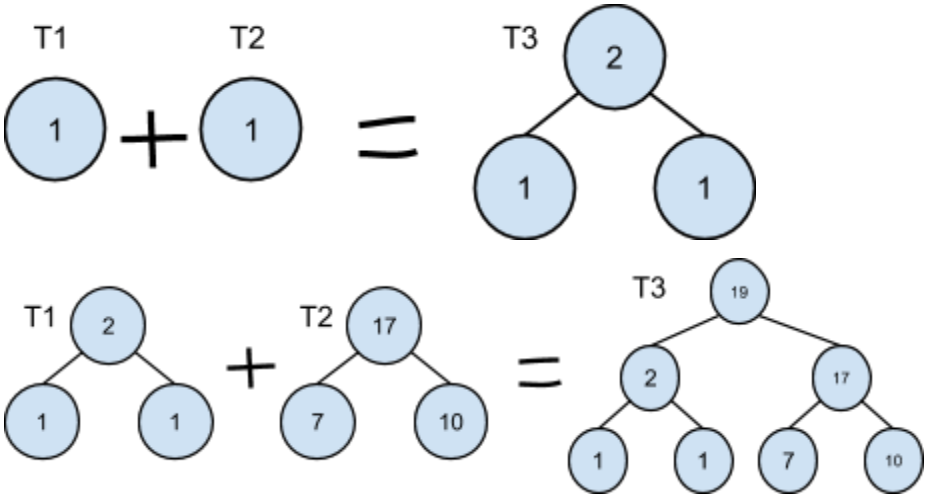
because, as we all know from math, $1 + 1 \neq 17$.



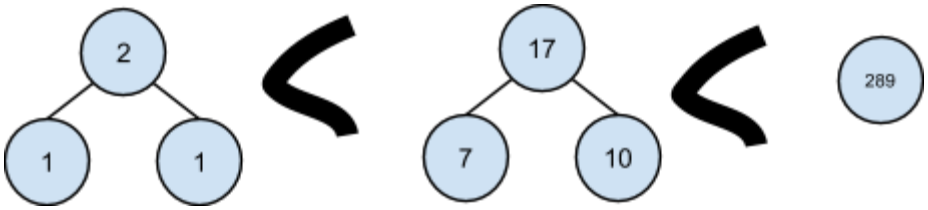
Keeping the leaf nodes the same, the value at the root node does not depend on the structure of the tree (try it for yourself). However, the structure of the tree does affect the sum of the values in the tree. Despite having the same values at their leaves, in the tree on the left, its values sum to 39. Meanwhile, in the tree on the right, its values sum to 49. In this assignment, given a set of leaf node values, we will construct their minimum sum tree, that is, we will construct the sum tree whose nodes' values have the least sum. (By the way, neither of these two trees are the minimum sum tree for 1, 2, 3, and 7.)

Now we'll talk about how to add two sum trees, what it means when one sum tree is less than another, and, finally, how to build the minimum sum tree.

T1+T2 = T3 means T3's root's value is the sum of T1's root's value and T2's root's value and T3's left and right subtrees are T1 and T2:



Given two sum trees T1 and T2, we say T1 is less than T2 if T1's root's value is less than T2's root's value.



Making a Minimum Sum Tree

Now we discuss how to make a minimum sum tree. Given a list L of the values of the leaves, instantiate a sum tree for each of the values (so each of these trees will have one node). Then, take the two smallest sum trees and add them together. Keep doing this until you have only one tree left. If you think you should use a heap for this, you are correct. For the heap, you may use python's heapq library, the book's code, or the k-way heap that you will code for the other part of this assignment.

Requirements

There are five requirements:

__lt__(self, other)

A SumTree method. Returns True if self is less than other, as defined above. False otherwise.

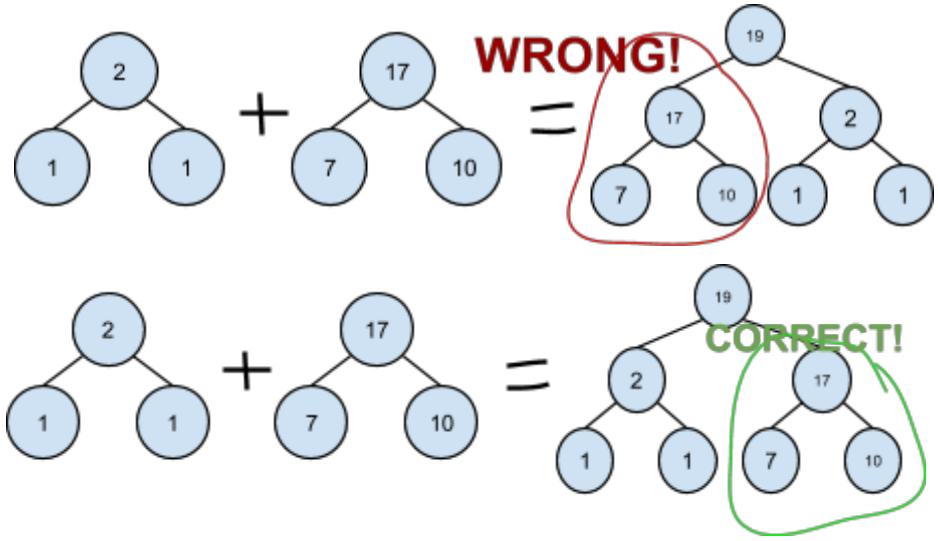
sum(self)

A SumTree method. Sums the values of the nodes and returns it. Write it recursively.

createMinSumTree(L)

Takes a list L of integers representing leaf node values and returns the minimum sum tree.

Draw your minimum sum tree in pygame.
There's always a pygame part! For uniformity amongst your assignments, **a node's left child's value should be less than or equal to the right child's value**. See the following example.



How to get better than a B:

To get an A-, do the following. In a new file called A.py, implement a k-way Heap whose constructor, in addition to k, also takes a function f as an argument. f(x,y) will be a function that basically defines x < y. For example, let t1 = ('a', 'z'), t2 = ('c', 'a'). Normally, t1 < t2 but if f(t1, t2) returns t1[1] < t2[1], then according to f, t2 is "less than" t1.

To get a A, tell me during your presentation why our algorithm works for making the minimum sum tree. I will be expecting a rigorous argument and you should expect me to question your logic. Feel free to tell each other the proof. As long as you can clearly present the proof and show that you actually understand it, the B+ is yours. There is no need to submit anything and you cannot bring any notes to the presentation. You may use a whiteboard for your presentation.

You don't have to do both parts to get an A but you may if you want. In that case, failing to get an A does not preclude you from getting an A-.

Submission

Put the k-way heap code in kwayHeap.py, the sum tree code in sumTree.py, and the the extra credit code in A.py. Put those two files in a folder called lastnameFirstnamePset6 and upload that to Google Drive.