

F. T-shirts

(<https://codeforces.com/contest/702/problem/F>)

Input:

```
8
30 3
22 3
22 16
30 16
15 9
10 12
30 7
15 10
9
10 40 50 10 70 20 25 10 35
```

Algorithm:

T-shirts:

cost →	22	30	10	15	15	30	22	30
quality →	16	16	12	10	9	7	3	3

sorted by descending order of quality and ascending order of price

Customers:

money →	10	40	50	10	70	20	25	10	35
id →	0	1	2	3	4	5	6	7	8

Legend:



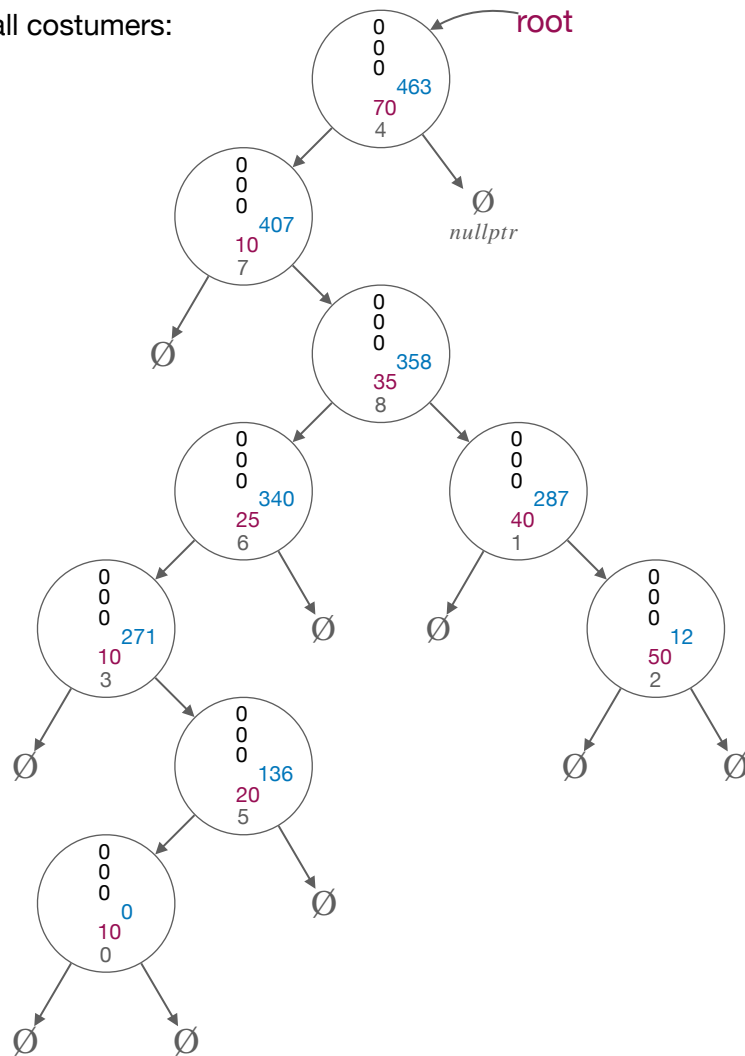
shirts = number of T-shirts purchased from a customer with identification number id;
lazy1 = money to be spent on T-shirts;
lazy2 = number of T-shirts to be purchased;
prior = a random number that keeps the tree balanced;
key = customer's money;
id = customer's identification number.

To generate random numbers for the nodes of the Treap we use `srand(489)` when the Treap is initialized and `rand()%489` at the initialization of each Treap's node.

We will create a Treap from all the customers using their money as key. After inserting all costumers, the Treap will look the following way shown below:

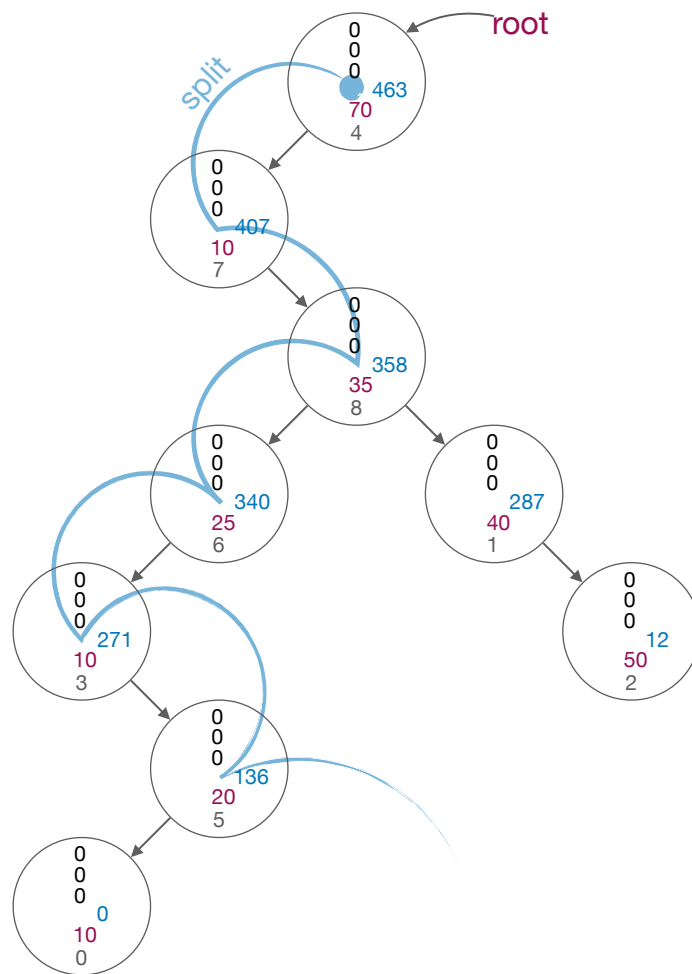
Note: we build the Treap by inserting each client sequentially, which takes us $O(n \cdot \log(n))$ time complexity. This construction can also take linear time if we build the Treap as a balanced binary search tree and then heapify.

Treap representing all costumers:

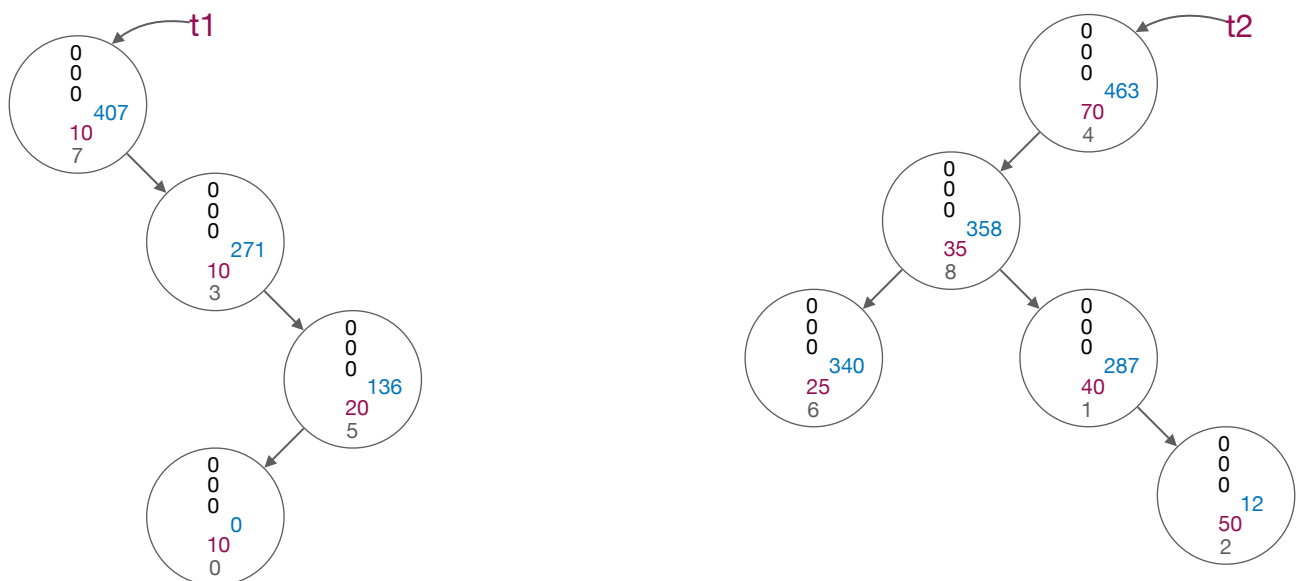


To simplify the visualization of the Treap, we will omit its empty nodes and we will know that if a given node has only a pointer to one existing child, then the other pointer will point to \emptyset (nullptr in context of C++ programming language) and if it does not point (has an arrow) to any child, then it is a leaf node.

We start traversing the Treap with every T-shirt starting with the one with largest quality and smallest price.



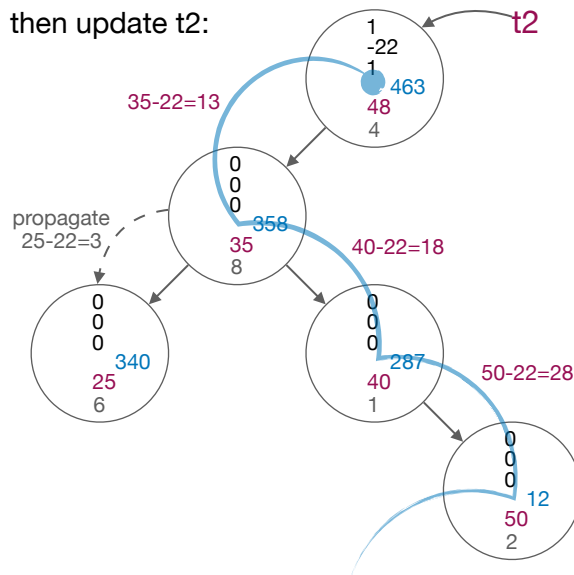
We split the Treap by key = T-shirt[0].cost:



After the split we know that all the nodes in the left Treap (t1) have keys $<$ T-shirt[0].costTreap and all the nodes in the right Treap (t2) have keys \geq T-shirt[0].cost.

if t2 is not equal to nullptr, then update t2:

key += -22
 shirts += 1
 lazy1 += -22
 lazy2 += 1



After the split we know that all the nodes in the left Treap (t2) have keys $< \text{T-shirt}[0].\text{costTreap}$ and all the nodes in the right Treap (t3) have keys $\geq \text{T-shirt}[0].\text{cost}$, i.e. we took out these customers, which purchase amount has decreased so much that it has become less than the price of the current T-shirt and they have to join the left Treap representing buyers who could not afford the current T-shirt, but without the ability to propagate to them.

