COCI 2010/11	Task UPIT
7. kolo, 9. travnja 2011.	**Author:** Goran Žužić

We can solve this task by building red-black tree or splay tree using the input data. Each node will keep track of the sum of all the nodes beneath it. We can easily answer the queries of type 3 and 4 by using standard binary tree manipulation techniques, and adjusting some sums along the way. Supporting query types 1 and 2 is a bit trickier. We will use the algorithm called lazy propagation.

Assume that we have to set all elements within given range to the same value. We begin by finding the set of O(log N) nodes that exactly cover that range. For each node in that set, we mark that all of the node's children are set to wanted value. Later on, when we encounter the node that has this mark set and would like to descend into that node's children, we simply transfer on this mark to all the children. We can now support queries of type 1.

Solving the queries of type 2 can be achieved in a similar way. We store two integers A and B for each node. Those integers suggest that k-th child of that node should be increased by A*k+B. We "propagate" A and B in the same way as described above. We must be careful in order to keep the correct sum stored in nodes at all times.

We suggest the following sources for further reading:

- http://en.wikipedia.org/wiki/Red_black_tree,
- http://en.wikipedia.org/wiki/Splay_tree,
- Robert Sedgewick: Algorithms.

Necessary skills:

Tournament trees, balansed trees, tree augmentation

Tags:

Data structures