CROATIAN OPEN COMPETITION IN INFORMATICS

7th ROUND

SOLUTIONS

COCI 2010/11	Task ŠEĆER
7th round, April 9th, 2011	**Author:** Adrian Satja Kurdija

If N is a multiple of 5, it’s obviously best to carry only 5-kilogram packages.
If this is not the case, we will have to use at least one 3-kilogram package.

From this we derive the following algorithm: use 3-kilogram packages as long as remaining number is not a multiple of 5, and then use only 5-kilogram packages to use up the remaining sugar.

There are other approaches to solving this task. We could use two loops to try out all the possible combinations of 3 and 5-kilogram packages, and find out which one gives as the correct amount of sugar in least number of packages used.

Necessary skills:

Basic arithmetic operations

Tags:

Ad-hoc

COCI 2010/11	Task KOLO
7. kolo, 9. travnja 2011.	**Author:** Adrian Satja Kurdija

We will store the wheel as array of characters initially filled with question marks. We need to simulate every spin that Mirko makes and write the obtained letter into the corresponding field. If that field is already filled with letter different than the one we are trying to write, we output '!' and exit.

To simulate turning of the wheel in one direction, we will pretend to move the arrow around the wheel in the opposite direction. If at some point we reach the end of the array, we start from the beginning.

When we are done, we must check whether some letter appears twice in the array. If this is the case, we output '!'. Otherwise we output the array in requested order.

Necessary skills:

Array manipulation

Tags:

Ad-hoc

COCI 2010/11	Task GITARA
7. kolo, 9. travnja 2011.	**Author:** Adrian Satja Kurdija

We look at each string separately, since they are mutually independent. For each string, we will simulate the events on that string.

We will store the sequence of currently pressed frets, in ascending order.
When new fret is to be pressed, we must remove all the fingers from higher frets that are already pressed. We do this by removing the elements one by one from the back of the sequence until we have reached the lower fret, or the sequence becomes empty. Then we can safely append the new fret to the sequence.

Since each pressing is present exactly once in the sequence, this algorithm has linear complexity.
This kind of sequence, where elements are only added to and removed from the end, is called stack.

Necessary skills:

Stack manipulation

Tags:

Data structures

COCI 2010/11	Task POŠTAR
7. kolo, 9. travnja 2011.	Author: Goran Gašić

First idea is to check all routes that visit all the houses. This solution has exponential complexity and will yield 30% of the total number of points. In order to improve this, we observe that we don't need to know the exact route to solve the task.

Given matrix can be viewed as a graph, with nodes corresponding to cells of the matrix and edges representing adjacent cells. Nodes that we will pass through in our route will be a part of some connected component of this graph. Let's assume that we can only use nodes with attitudes from interval [l,r]. We can determine if it's possible to visit all the houses in the reduced graph by checking if some connected component contains all the houses and the starting point. This leads to an $O(N^6)$ solution as we can try out all the possible values for l and r. This solution is worth 60% of the total number of points.

We can further improve our solution by observing that if we fix the value of l, we can binary search for best r. Complexity now becomes $O(N^4 \log N^2)$ which is enough to obtain maximum number of points.

$O(N^4)$ solution also exists but we will not discuss it here.

Necessary skills:
Binary search, graph traversal (BFS or DFS)

Tags:
Graph theory

COCI 2010/11	**Task KUGLICE**
7. kolo, 9. travnja 2011.	**Author:** Filip Pavetić

We will solve the task using the off-line approach, i.e. we will load all the input data at the beginning and then analyse it in order to find the answers of given queries. It turns out that it's easier to solve the task if we execute queries in backwards order. Instead of removing the edges we will be adding them.
This allows us the solve the task using the small modification of the union-find algorithm.

Necessary skills:
Union-find algorithm

Tags:
Data structures

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