

**CROATIAN OPEN COMPETITION IN
INFORMATICS**

Round 3

SOLUTIONS

COCI 2011/2012	Task DIGITALNA
Round 3, December 17th, 2011	Authors: Goran Gašić, Marko Ivanković

This problem can be solved by following this algorithm:

1. Move the arrow down until it's pointing to the BLJTV1 channel.
2. Move the arrow and selected channel up until they reach the first position.
3. Move the arrow down until it's pointing to the BLJTV2 channel.
4. Move the arrow and selected channel up until they reach the second position.

First two steps are very similar to other two, and can be implemented using the same function, as it's done in official solutions.

Necessary skills: strings, *for* or *while* loop

Category: ad hoc

COCI 2011/2012	Task D'HONDT
Round 3, December 17th, 2011	Author: Nikola Dmitrović

As we read input data, we immediately check if this party won more than 5% of the total number of votes. If it did, we add fourteen entries to some sequence. Each entry should contain this party's identifier letter, and a number of votes divided by an integer in range 1 to 14.

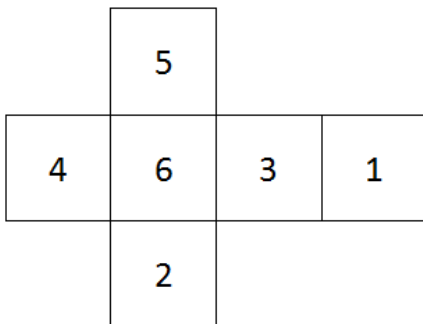
We now sort this sequence in descending order of calculated quotients, and take a look at the first 14 entries. Among these, we must count how many times each party appears. We must be careful to output all the parties that had more than 5% of the votes, and to output them in sorted order.

Necessary skills: sequence, structures, sorting, loops

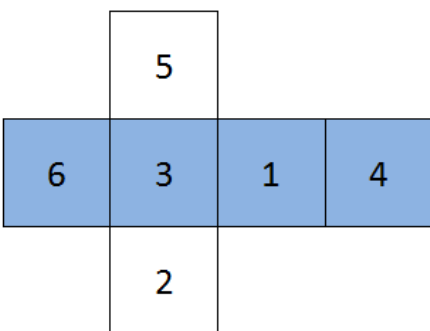
Category: ad hoc

COCI 2011/2012	Task POGODAK
Round 3, December 17th, 2011	Author: Frane Kurtović

First thing to do is decide how to store the cube. One possible way is to lay it's faces to the plane, like in the figure below. This figure shows the cube in its starting position, i.e. in the upper left corner of the matrix.



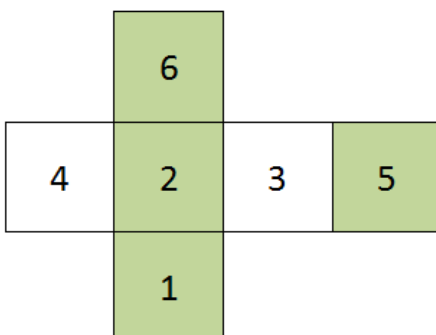
Let's roll the cube one field to the right, and see what it looks like now.



We can see that this move only changed the blue row in the figure, and that this row **shifted circullary to the left**.

Also note that one move to the left is the same as three moves to the right. This will be helpful later.

Let's see what happens if we roll the cube one field down. Figure shows what the cube looks like after moving it one field down from the starting position.



Only green faces marked 2,6,5,1 changed, and we can observe that they **shifted circullary to the right**, i.e 2651 became 1265.

With observations above, we can simulate all the moves, which would lead to an $O(\mathbf{RS})$ solution and would be too slow.

We can improve our solution by finding a faster way to calculate the sum of each row and the final state of the cube at the end of that row.

After four horizontal moves we end up in exactly the same state, so $S-1$ horizontal moves gives the same state of the cube as if we made only $(S-1) \bmod 4$ moves.

To find the sum of whole row, we must also add the rest of the moves. These moves can be divided into groups of 4, and it's not hard to see that each of these groups has the sum of 14, since one group contains two pairs of opposing faces.

Final complexity is now $O(R)$.

Necessary skills: periodic patterns, wrapping an object into structure

Category: ad hoc

COCI 2011/2012	Task ROBOT
Round 3, December 17th, 2011	Author: Adrian Satja Kurdija

We will find a way to quickly keep track of the current sum after each move of the robot.

Assume that robot moved to the right (east) from (a, b) to $(a+1, b)$. Distance to any control point changed by 1 or -1 after this move. To be more precise, distance increased by 1 for every control point with x-coordinate no greater than a , and decreased by 1 for rest of them. If we denote the number of points in the first group $f(a)$, then the sum of these distances changed by $1 * f(a) + (-1) * (N - f(a))$. We can do the same for y-coordinate.

Next we must figure out how to calculate $f(a)$ efficiently. One of the ways to do this is to keep the control points in two arrays, one sorted by x-coordinate and other sorted by y-coordinate. Now we can easily binary search $f(a)$ for any value a .

This solution has the complexity of $O(N \log N + M \log N)$. Better complexity can be achieved by precomputing all the $f(a)$ values using dynamic programming.

Necessary skills: mathematical problem analysis

Category: ad hoc

COCI 2011/2012	Task PLAĆE
Round 3, December 17th, 2011	Author: Ivan Katanić

First, notice that we can represent the company hierarchy using a tree. Each node in a tree represents an employee, and corresponding subtree his subordinates.

If we traverse this tree using preorder, we will get a list of all the nodes. For every subtree, there is an interval in this list that contains exactly the nodes of that subtree.

$\mathbf{L} = \{ \}$

TRAVERSE(\mathbf{u}) :

append \mathbf{u} to sequence \mathbf{L}

for each unvisited \mathbf{v} adjacent to \mathbf{u} , do TRAVERSE(\mathbf{v})

We obtain the whole list by calling TRAVERSE(1).

We now need a structure that can increase some interval in \mathbf{L} by a given value, and return the value of an element at a given moment. This can be done by using interval (tournament) tree, or Fenwick tree. Overall complexity is $O((\mathbf{N}+\mathbf{M}) \log \mathbf{N})$.

Official solution uses Fenwick tree.

Necessary skills: dfs traversal, data structures

Category: graphs, data structure

COCI 2011/2012	Task TRAKA
Round 3, December 17th, 2011	Author: Gustav Matula

Let's formally define the condition from the task that says that no car can wait with none of the workers. Let's denote some car i and some worker j . By the time worker j is done with car $i+1$, worker $j+1$ must be done with car i . Note that worker $j+1$ can finish car i before this. If car $i+1$ is sent t_0 minutes after car i , following must hold:

$$t_0 + \sum_{k=0}^j F_{i+1} T_{j+1} \geq \sum_{k=0}^{j+1} F_i T_{j+1}$$

In other words, the time that it takes car $i+1$ to pass through all the workers up to and including j must be at least the time that takes car i to go through workers up to and including $j+1$.

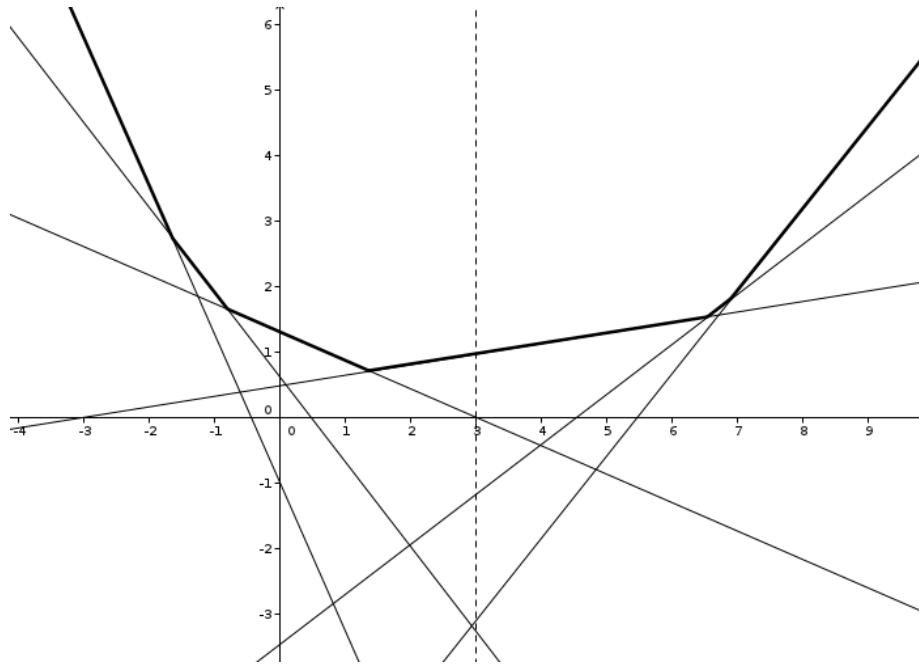
If we denote by S_j sum of T for workers through j , than minimum t_0 for cars i and $i+1$ can be written as:

$$t_0 = \max \{ F_i S_{j+1} - F_{i+1} S_j, 0 \leq j \leq N-2 \}$$

From this we can derive the brute-force solution. For each i , we go through all the workers to find the maximum value t_0 , and we sum up these maximum values with $F_{M-1} S_{N-1}$. This solution has the complexity of $O(NM)$.

We can come up with a faster solution by taking a look at the expression that we must maximize. Since i is fixed and all the factors are positive we can divide this expression by F_{i+1} . If we denote this quotient F_i/F_{i+1} with x , we get $f(x) = S_{j+1}x - S_j$, i.e. a line equation. For each j that satisfies mentioned limits we get a line in the plane. From these lines we must find the one that gives as the maximum value for some x .

This idea is often useful in algorithmic competitions, so we will describe it here in general case. Let's take a look at how this maximum of lines looks like.



It's a polygonal line enclosing a convex region of the plane. Lines that are part of this line are sorted in increasing order of their slopes. Also, no line is above previous two lines. These leads us to the following algorithm.

We sort the lines in increasing order of slopes. We push first two of them to the stack, and proceed traversing rest of the lines. When processing the next line in order, we pop the top line of the stack, as long as intersection of the top two lines on the stack is below the current line. When this is no longer true, we push the current line to the stack.

We denote the i th line that's pushed on the stack by $f_i(\mathbf{x})$.

We can find the maximum value for some \mathbf{x} by using binary search, since $f_i(\mathbf{x})$ is monotonic with respect to i .

Complexity of this approach is $O((N + M) \log N)$, or to be more precise, $O(N \log N + M \log H)$ where H is number of lines that are part of convex boundary.

There is also an alternative solution that represents the expression we are trying to maximize using dot product, and uses convex hulls.

Necessary skills: line set maximum

Category: geometry, optimization