CROATIAN OPEN COMPETITION IN INFORMATICS

6th ROUND

SOLUTIONS

COCI 2009/2010 **Task: KAJAK**

6th round, 6. March 2010. **Autor:** Marko Ivanković

We start from the finish and work our way to the start. As we encounter kayaks along the way, we mark them in increasing order. Just be careful to mark two kayaks in the same column with the same number.

Necesarry skills:

strings

Tags:

ad-hoc

COCI 2009/2010 **Task NATJECANJE**

6th round, 6. March 2010. **Autor:** Marko Ivanković

This task can be solved using a simple greedy algorithm. First we take care of teams that have lost their kayaks and brought reserves. They are the same as teams that have not lost their kayaks and did not bring reserves, so they can be counted as such. Now, starting from the first team, we work our way up. For each kayakless team we first check their lower-numbered neighbor first, if they can borrow that kayak, then they do it. If not, they check their upper-numbered neighbor. If they have a spare kayak, they borrow that one. If not, they are kayakless and increment the solution by one.

Necesarry skills:

greedy algorithm

Tags:

greedy algorithm

COCI 2009/2010 **Task DOSADAN**

6th round, 6. March 2010. **Autor:** Marko Ivanković

This task was inspired by a puzzle from **+Ma's Reversing** site.

OTP is theoretically perfect algorithm, **if implemented correctly.** In most cases if it fails, it fails due to human error. In this task, the key bit of information is the structure of the plaintext and the key. Let us take a look at the binary coding of plaintext letters. We se that letters 'a' - 'z' all have the 7^{th} bit equal to 1. Numbers '0' - '9' present in the key all have the 7^{th} bit equal to 0. Space and full stop also have the 7^{th} bit equal to 0. This is enough information to conclude that in the ciphertext, all letters will have the 7^{th} bit equal to 1, and space and full stop 0. This is enough information to solve this task.

For extra points, you might have realized that this now provides enough information for you to try and decrypt the test data completely. Using some guesswork, of course.

Necesarry skills:

understanding ASCII coding

Tags:

cryptography

COCI 2009/2010 **Task XOR**

6th round, 6. March 2010. **Autor:** Goran Žužić

Tags:

???

COCI 2009/2010 **Task HOLMES**

6th round, 6. March 2010. **Autor:** Marko Ivanković

This task cannot be solved by simply finding dominators for the graph. But it's not far away. The following example illustrates the problem with dominators:

```
4 4 1
1 3
2 3
1 4
2 4
3
```

Event 3 is proven by the Yard. Event 3 can be caused by either event 1, event 2 or both. Even though we do not know which one, note that one of them **had** to be. Since event 4 can be caused by either event 1, event 2 or both, this means that event 4 is also part of the solution. Even though we do not know the exact cause.

Let us see how we can approach this problem. First, we add all events proven by Scotland Yard to the solution. This is a no brainer. Then, we flood fill from them, adding all "consequences" of proven events to the solution. This leaves "causes" and "consequences of causes". For each not yet proven event, we try to prove it by contradiction. We start by supposing that the event X could not have happened. Then we optimistically presume all other events that do not cause event X to happen to have happened. This now leaves a set of events leading to X and some leading from X marked as "did not happen". If one of the proven events ends up "not happened" this is a contradiction and we can surmise that event X had to have happened.

How can we check this quickly? For each event X, we can iterate through all other events Y. If the set of "causes" of X contains all "causes" of Y then Y must not have happened. "causes" is the minimal set of events that can cause X. Note that all "causes" do not contain entry edges. Causes for each event can be precomputed ahead of time.

On a side not, talking in past perfect conditional tense in English is extremely confusing.

Necesarry skills:
graph theory, logic

Tags:
graph theory

COCI 2009/2010 **Task GREMLINI**

6th round, 6. March 2010. **Autor:** Luka Kalinovčić

Let $x_D[i]$ be the number of years until the birth of the first type i gremlin with D ancestors. Since there is one of each type of gremlins born in the accident we can set $x_0[i] = 0$, for each i.

Examine one type i gremlin with D ancestors. After Y_i years he spawns one type j egg, that needs $Z_{i,j}$ years to hatch. That egg has D+1 ancestors. So we now know that $x_{D+1}[j]$ is less than or equal to $x_D[i] + Y_i + Z$. Obviously, finding the minimum of all such values for all gremlins that hatch type j eggs we obtain $x_{D+1}[j]$. We can observe all types at once by using matrix algebra. We have:

$$A \otimes X_D = X_{D+1}$$

Where A is a rectangular matrix with i-th row and j-th column equal to $Y_j + Z_{j,i}$ if type j gremlin hatches type i egg, or ∞ if not. Matrices X_D and X_{D+1} are self explanatory, and $\otimes$ marks min-plus matrix multiplication.

Matrix A can be raised to arbitrary power fast using binary exponentiation. After multiplying A to the power of D and performing min-plus with X_0 we scan the obtained X_D matrix. If each number is larger than T, there are no gremlins with D ancestors. Otherwise, there is at least one. Using binary search on D we find the largest possible D.

Kategorija:

min-plus algebra, matrix multiplication and powers, binary search