# Day 3 Editorial

April 28, 2016

ETH Zurich ACM ICPC Training Camp. April 2016

# A. Arithmetic Rectangle

## Problem statement

- You are given an $n \times m$ matrix $(n, m \leqslant 3000)$
- Find biggest "arithmetic" submatrix
- Matrix is called "arithmetic" iff each row and each column is arithmetic sequence

# A. Arithmetic Rectangle

## Solution

- For each cell find end of longest arithmetic sequence goes right and goes down
- It can be done in $O(n \cdot m)$ using dynamic programming
- Let's answer is rectange $[x_1..x_2] \times [y_1..y_2]$
- Let's fix $x_1$ (first column). Than $x_2 = \min_{y=y_1}^{y_2} right[x_1][y]$, but we can only use this rectangle if $down[x_1][y_2] \leqslant y_1$ and $down[x_1 + 1][y_2] \leqslant y_1$
- We can solve this in a similar way as a biggest zero submatrix problem using stack or DSU
- Total complexity is $O(n \cdot m)$

# B. Bytean Road Race

## Solution

- For every vertex define $A(v)$:
  - Downward arc if it exists
  - Otherwise, rightward arc
- Define $B(v)$:
  - Rightward arc if it exists
  - Otherwise, downward arc
- Let's call downmost path from $v$ is $v, A(v), A(A(v)), \ldots$
- And rightmost path from $v$ is $v, B(v), B(B(v)), \ldots$
- To check, whether you can reach vertex $u$, check if $u$ is between rightmost and downmost paths from $v$

# C. Will It Stop?

## Problem statement

Given a program:

$$
\begin{array}{l}
\textbf{while } n > 1 \textbf{ do} \\
\quad \textbf{if } n \bmod 2 = 0 \textbf{ then} \\
\quad\quad n := n/2 \\
\quad \textbf{else} \\
\quad\quad n := 3 \cdot n + 3
\end{array}
$$

- Given $n$
- Find whether the program stops or not

## Solution

- The program stops iff $n = 2^k$ for some integer $k$
  - When $n := 3 \cdot n + 3$ is done, $n$ becomes multiple of 3
  - Both operations keep $n$ divisible to 3

# E. Gophers

## Problem statement

- We have some CD players on the x-axis
- Each of them don't let sleep on segment $[x - l, x + l]$
- Some of them are moved
- You are to answer the queries how many points are there, that someone can't sleep in them

# E. Gophers

## Solution

- We need a data structure that can add and subtract 1 on a segment and count number of zeros in array
- Just use interval tree that maintains the minimum and the number of values that equal to minimum
  - When making a move, we can count the number of zeros created and destroyed
    - Just take minimum, if it equals to zero, count number of minimums

# F. Laundry

## Problem statement

- There are $n \leqslant 10^6$ friends who want to do laundry
- $i$-th of them need $2 \cdot a_i$ clothespin of same color to pin socks and $3 \cdot a_i$ clothespin of same color to pin shirts
- $i$-th person can use clothespins of same color for socks and shirts if he wants
- But different persons can't use clothespins of same color
- You are also given how much clothespins of each color you have
- You need to give closespins to persons in a way which minimizes total number of used colors

# F. Laundry

## Solution

- We can use greedy
- Let's handle all $a_i$ in decreasing order
- If there are exist some color that there are $\geqslant 5 \cdot a_i$ clothespins of that color, than we will use it for person $a_i$
- Otherwise try to find clothespins which amount is smallest but greater than $2(3) \cdot a_i$ for current persons socks (shirts)
- Total complexity is $O(n \log n)$

# G. Bits Generator

## Problem statement

- You have a random bit generator
- Find initial seed that could be to generate given string

## Solution

- Build graph of seeds, there are at most $m$ seeds
- Each vertex has one outcoming edge
- Let's calculate hash function of string generated starting with each seed
- To do that use binary climbing: $q(s, i)$ — seed after $2^i$ bits generated, $h(s, i)$ — hash of generated string of length $2^i$
- Can get hash function in $O(\log m)$
- One can also come up with solution similar to building suffix array using class equivalences for $2^i$ length substrings

# H. Afternoon Tea

## Problem statement

- Byteasar has a teacup and he does following $n \leqslant 10^5$ times:
- Drink half of cup and add tea or milk
- You are to find if he drank more tea or milk

## Solution

- For each adding of milk or tea we can find how much of it Byteasar drank in the end
- If he made *len* actions then he drank $1 - 0.5^{len-i}$ part of liquid, which he add during $i$-th action
- So we can store balance as a 2-based value and maintain it. Total complexity is $O(n)$
- There is also a case-analysis solution

# I. Intelligence Quotient

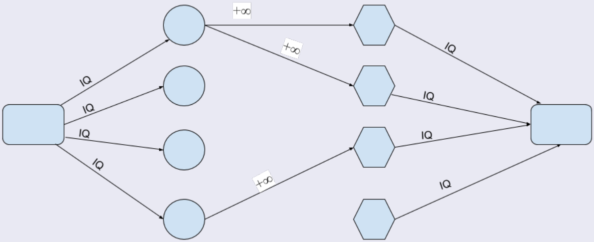
## Problem statement

- You are given two cliques
- Some of the pairs of vertices from different cliques are connected with an edge
- Find the clique with largest sum of vertex weights

# I. Intelligence Quotient

## Solution

- If you get the complement of the graph, then you have to choose an independent set of maximum weight in a bipartite graph
  - It's the same as minimum weight vertex cover
- Any finite cut in network below corresponds to a vertex cover
- Just find maximum flow



# J. Cave

## Problem statement

- You are given a tree consisting of $n \leqslant 3 \cdot 10^6$ vertices
- You want to split it on $k$ disjoint sets of vertices, such that each set is connected
- What $k$ can be used?

# J. Cave

## Solution

- We only need to try $k$ such that $n \bmod k = 0$ (at most 336 variants)
- We must put two adjecent vertices to the same set if size of subtree is not divisible by $\frac{n}{k}$
- If we want to split vertices on $k$ sets, there should exist exactly $k - 1$ edges such that size of subtree is divisible by $\frac{n}{k}$
- Let's count number of vertices such that its subtree has size $i$ for each $i$ upto $n$
- Now we can found number of "good" vertices in $O(k)$ just as $\sum_{i=1}^{k} cnt[\frac{n \cdot i}{k}]$

# K. Cross Spider

## Problem statement

- You are given $n \leqslant 10^5$ points in a 3D space
- You need to check if there is a plane which contains all points

## Solution

- Find a point which doesn't lie on a line built on first and second point
- If there is no such point than we can use any plane contains this line
- Otherwise build a plane on this 3 points and check if all other points are on it
- Total complexity is $O(n)$