# CERC 2013: Presentation of solutions

Jagiellonian University

November 17, 2013

## Some numbers

Total submits: 835
Accepted submits: 347

## Some numbers

Total submits: 835
Accepted submits: 347

First accept: 0:03:03,
University of Warsaw
(Jan Kanty Milczek, Leonid Logvinov, Adam Karczmarz)

Last accept: 4:56:53,
Czech Technical University in Prague
(Petr Šefčík, Martin Švorc, Michal Peroutka)

## Some numbers

Most determined teams:

Charles University in Prague
(Karel Tesař, Lukáš Folwarczný, Vlastimil Dort)
9 attempts at problem C

VŠB - Technical University of Ostrava
(Lukáš Tomaszek, Jiri Cága, Marek Záškodný)
9 attempts at problem B

# Problem L
Bus

Submits: 81
Accepted: 72

First solved by:
University of Warsaw
(Jan Kanty Milczek, Leonid Logvinov, Adam Karczmarz)
0:03:03

Author: a popular riddle

Simply output $2^k - 1$.

# Problem B
What does the fox say?

Submits: 130
Accepted: 71

First solved by:
University of Zagreb
(Marin Tomić, Gustav Matula, Ivica Kicic)
0:10:38

Author: Ylvis

No real algoritm, just filter out given words from the string.

(image by sir-boo.deviantart.com)

# Problem F
# Draughts

Submits: 151
Accepted: 66

First solved by:
Charles University in Prague
(Jakub Zíka, Štěpán Šimsa, Filip Hlásek)
0:25:55

Author: Lech Duraj

Problem: you are given a $10 \times 10$ draughts board. How many captures can the white player perform in his next move?

Problem: you are given a $10 \times 10$ draughts board. How many captures can the white player perform in his next move?

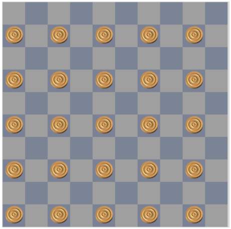
Solution: simply use backtracking to check all possible moves and hope it runs fast enough...

Problem: you are given a $10 \times 10$ draughts board. How many captures can the white player perform in his next move?

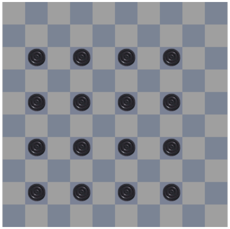
Solution: simply use backtracking to check all possible moves and hope it runs fast enough...

...or prove it.

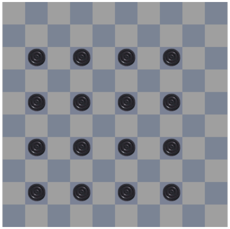
When you start from one of these fields you can never visit any other field.



So you can only capture these pieces – there are 16 of them.



So you can only capture these pieces – there are 16 of them.



In every step (apart from the first one) you can go in one of three directions (you cannot go back) so the number of moves to check is at most

$$3^{16} \approx 40\,000\,000.$$

# Problem I
Crane

Submits: 71
Accepted: 48

First solved by:
University of Warsaw
(Marek Sommer, Błażej Magnowski, Kamil Dębowski)
0:27:37

Author: Lech Duraj

Problem: you are given a permutation of numbers $\{1, 2, \ldots, n\}$. You have to sort it using at most $50n$ *moves*. A single *move* consists of

1. selecting an even length interval;
2. swapping its two halves.

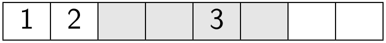
Problem: you are given a permutation of numbers $\{1, 2, \dots, n\}$. You have to sort it using at most $50n$ *moves*. A single *move* consists of

1. selecting an even length interval;
2. swapping its two halves.

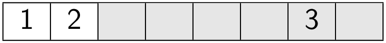
There are several quite different solutions of this problem producing sequences of $O(n)$ or $O(n \lg n)$ moves. We present presumably the simplest one, which always uses at most $2n$ moves.

Assume that numbers $\{1, 2, \ldots, k-1\}$ are already in their place. We'll move $k$ to its place in at most two moves.

It is either in the first half of the yet unsorted tail and we need only one move...



Or it is in the second half and with one swap we can move it to the first half.



The only problem left is finding $k$ in the unsorted tail. You can do it in $O(n \lg n)$ time with balanced BST but for $n \leq 10\,000$ simple quadratic simulation was fast enough.

# Problem C
Magical GCD

Submits: 126
Accepted: 29

First solved by:
University of Warsaw
(Jakub Oćwieja, Tomasz Kociumaka, Jarosław Błasiok)
0:34:35

Author: Adam Polak

## Magical GCD

$MGCD((a_1, a_2, ..., a_k)) := k \cdot GCD(a_1, a_2, ..., a_k)$

You are given a sequence $(a_1, a_2, ..., a_n)$.
Compute the largest MGCD of its connected subsequence.

$n \leq 10^5$, $1 \leq a_i \leq 10^{12}$.

## Magical GCD

|  |  |  |  |  | (j) |  |  |
|---|---|---|---|---|---|---|---|
| $a_i$ | 5 | 8 | 6 | 2 | 6 | 8 | 8 |
| $i$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

For a fixed $j$ and every $i$, consider $GCD(a_i, a_{i+1}, ..., a_j)$.

$$\begin{aligned}
1 &= GCD(5, 8, 6, 2, 6) \\
2 &= GCD(8, 6, 2, 6) \\
2 &= GCD(6, 2, 6) \\
2 &= GCD(2, 6) \\
6 &= GCD(6)
\end{aligned}$$

$GCD(a_i, a_{i+1}, ..., a_j) \mid GCD(a_{i+1}, a_{i+2}, ..., a_j)$.

$|\{GCD(a_i, a_{i+1}, ..., a_j) \textbf{ for } i = 1, 2, ..., j\}| = O(\log a_j)$.

## Magical GCD

|  |  |  |  |  |  (j) |  |  |
|---|---|---|---|---|---|---|---|
| $a_i$ | 5 | 8 | 6 | 2 | 6 | 8 | 8 |
| $i$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

Store the leftmost $i$ for each distinct value of $GCD(a_i, a_{i+1}, ..., a_j)$.

| GCD | left end |
|---:|---:|
| 1 | 1 |
| 2 | 2 |
| 6 | 5 |

## Magical GCD

|  |  |  |  |  |  |  | $(j+1)$ |
|---|---|---|---|---|---|---|---|
| $a_i$ | 5 | 8 | 6 | 2 | 6 | 8 | 8 |
| $i$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

Store the leftmost $i$ for each distinct value of $GCD(a_i, a_{i+1}, ..., a_j)$.

| GCD | left end |
|---|---|
| 1 | 1 |
| 2 | 2 |
| 2 | 5 |
| 8 | 6 |

## Magical GCD

| | | | | | | | $(j+1)$ | |
|---|---|---|---|---|---|---|---|---|
| $a_i$ | 5 | 8 | 6 | 2 | 6 | 8 | | 8 |
| $i$ | 1 | 2 | 3 | 4 | 5 | 6 | | 7 |

Store the leftmost $i$ for each distinct value of $GCD(a_i, a_{i+1}, ..., a_j)$.

| GCD | left end |
|---|---|
| 1 | 1 |
| 2 | 2 |
| 8 | 6 |

# Problem K
# Digraphs

Submits: 135
Accepted: 32

First solved by:
University of Warsaw
(Jakub Oćwieja, Tomasz Kociumaka, Jarosław Błasiok)
0:58:30

Author: Adam Polak

Problem: given a list of forbidden two-letters words find the biggest square which does not contain any of them in any row or column. If it is possible to get a square bigger than $20 \times 20$ print only $20 \times 20$ square.

Assume there exists a $2n-1$-letters word which does not contain any forbidden word:

$$a_1, a_2, \dots, a_{2n-1}.$$

Then you can construct an $n \times n$ square:

$$\begin{matrix} a_1 & a_2 & \dots & a_n \\ a_2 & a_3 & \dots & a_{n+1} \\ \vdots & \vdots & \ddots & \dots \\ a_n & a_{n+1} & \dots & a_{2n-1} \end{matrix}$$

The opposite is also true – if there exists an $n \times n$ square without forbidden words, you can also construct a $2n-1$-letters word by taking the topmost row and the rightmost column of the square.

How to find the longest possible word which does not contain any forbidden word?
Construct a directed graph with nodes representing letters. There is an edge between vertices $A$ and $B$ if $AB$ is not a forbidden word. Now you need to find the longest path in this graph. It is either acyclic and you can use DP approach or it has a cycle and you can loop through that cycle to generate a 39-letters word which is enough to create a $20 \times 20$ square.

# Problem D
Subway

Submits: 48
Accepted: 12

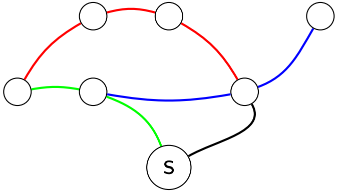
First solved by:
University of Warsaw
(Marcin Smulewicz, Grzegorz Prusak, Wojciech Nadara)
2:03:27

Author: Grzegorz Herman

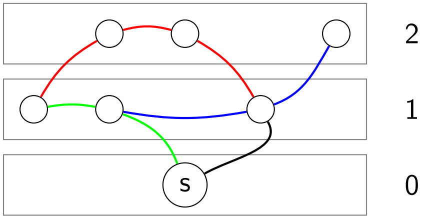
## Problem

Given a graph of subway stops and lines connecting multiple stops,
find a route from $s$ to $t$ which
minimizes the number of lines used,
and maximizes the number of stops travelled.

## Minimizing the number of line changes

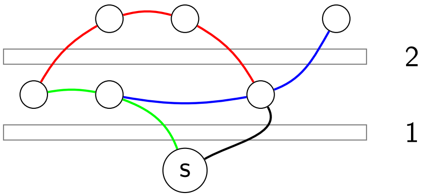


## Minimizing the number of line changes



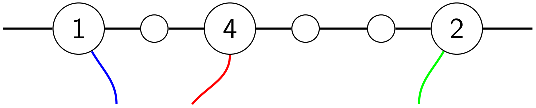
- regular BFS. . .

## Minimizing the number of line changes

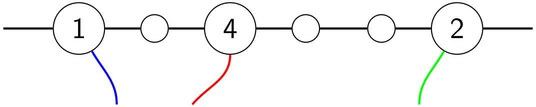


- regular BFS. . .
- on *lines*

## Maximizing travel time

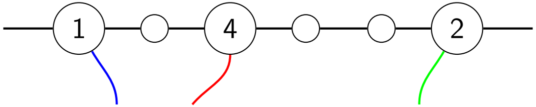


## Maximizing travel time



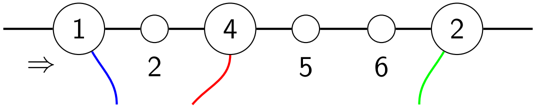
- for every non-visited stop. . .

## Maximizing travel time



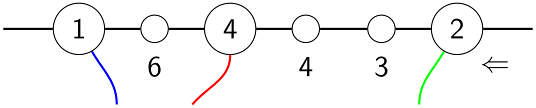
- for every non-visited stop. . .
- need to consider every visited as a potential source

## Maximizing travel time



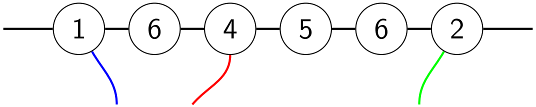
- for every non-visited stop...
- need to consider every visited as a potential source
- solution: sweep

## Maximizing travel time



- for every non-visited stop. . .
- need to consider every visited as a potential source
- solution: sweep in both directions

## Maximizing travel time



- for every non-visited stop. . .
- need to consider every visited as a potential source
- solution: sweep in both directions

# Problem H
Chain & Co.

Submits: 25
Accepted: 11

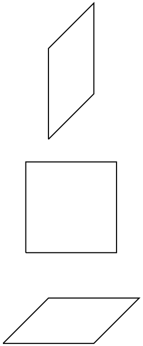
First solved by:
University of Warsaw
(Jakub Oćwieja, Tomasz Kociumaka, Jarosław Błasiok)
2:17:46

Author: Arkadiusz Pawlik

## Problem

Given a set of *links* – equal size squares (without interior), axis aligned in three dimensions, determine if it can be divided into two proper subsets, with each link in one set inseparable from each link of the other.

## Divide links into three groups



## Structure of a solution

Separability of links:

- same orientation: always separable (cannot touch!)
- different orientation: may be inseparable

## Structure of a solution

Separability of links:

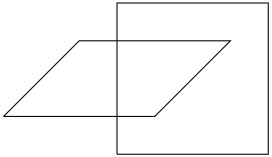
- same orientation: always separable (cannot touch!)
- different orientation: may be inseparable

Possible solutions:

- $A = X, B = Y \cup Z$
- $A = Y, B = Z \cup X$
- $A = Z, B = X \cup Y$

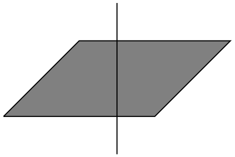
## Checking a possible solution

A link $y \in Y$ is inseparable from a link $z \in Z$



## Checking a possible solution

A link $y \in Y$ is inseparable from a link $z \in Z$
iff
one of $z$-aligned segments of $y$ "goes through" $z$



## Checking a possible solution

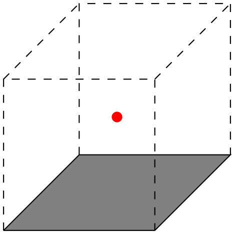
A link $y \in Y$ is inseparable from a link $z \in Z$

iff

one of $z$-aligned segments of $y$ “goes through” $z$

iff

the upper end of this segment lies in the interior of the “upper box” of $z$

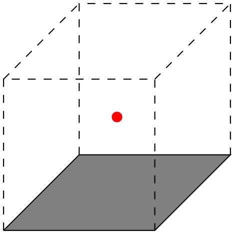


## Checking a possible solution

A link $y \in Y$ is inseparable from *every* link $z \in Z$

iff

the upper end . . . lies in the *intersection* of the “upper boxes” of $Z$



# Problem A
Rubik's Rectangle

Submits: 27
Accepted: 5

First solved by:
University of Warsaw
(Jakub Oćwieja, Tomasz Kociumaka, Jarosław Błasiok)
2:41:27

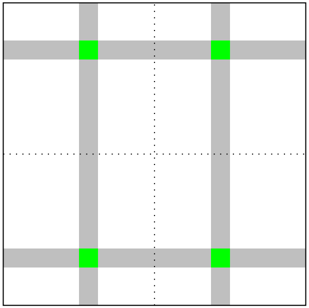
Author: Lech Duraj & Jakub Pachocki

## Problem

Sort a rectangular board
(holding a permutation of numbers from 1 to $W \cdot H$)
by a sequence of flips (reversals)
of individual rows or columns.

## Quads

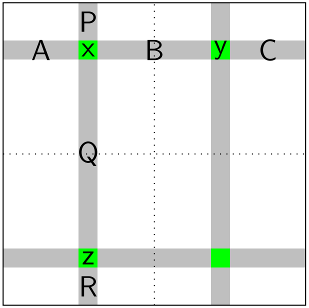
Tiles symmetric w.r.t. the symmetry axes of the board form a *quad*:



No flips can move a tile away from its quad.

## Even permutations

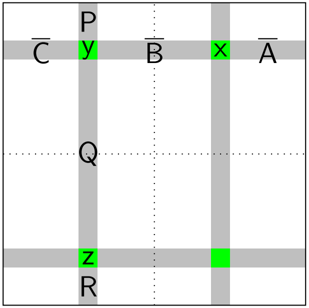
You can "rotate" three tiles of a quad by:



## Even permutations

You can "rotate" three tiles of a quad by:

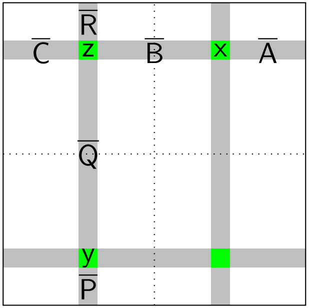
- flipping the top row



## Even permutations

You can "rotate" three tiles of a quad by:

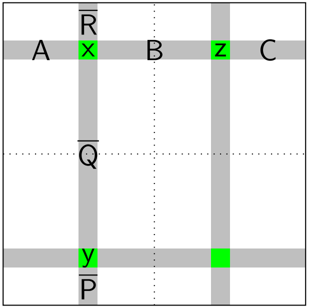
- flipping the top row
- flipping the left column



## Even permutations

You can "rotate" three tiles of a quad by:

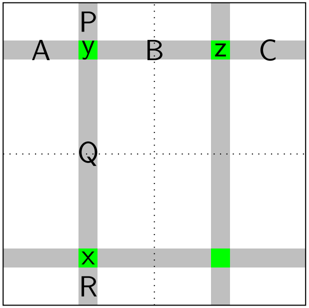
- flipping the top row
- flipping the left column
- flipping the top row again



## Even permutations

You can “rotate” three tiles of a quad by:

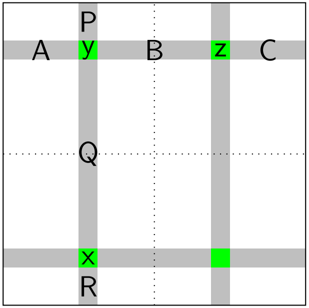
- flipping the top row
- flipping the left column
- flipping the top row again
- flipping the left column again



## Even permutations

You can "rotate" three tiles of a quad by:

- flipping the top row
- flipping the left column
- flipping the top row again
- flipping the left column again

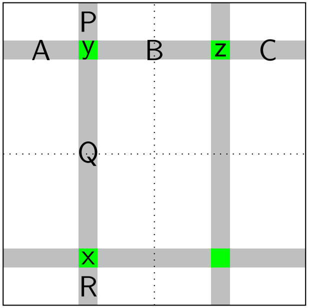


Such move leaves the rest of the board unchanged.

## Even permutations

You can "rotate" three tiles of a quad by:

- flipping the top row
- flipping the left column
- flipping the top row again
- flipping the left column again



Such move leaves the rest of the board unchanged.

Similarily, every *even* permutation of a quad can be performed.

## Odd permutations

Assume we flip a subset $R$ of rows and a subset $C$ of columns – the parity of each quad's permutation will change accordingly.

## Odd permutations

Assume we flip a subset $R$ of rows and a subset $C$ of columns – the parity of each quad's permutation will change accordingly.

If all resulting permutations are even, we can fix them.

## Odd permutations

Assume we flip a subset $R$ of rows and a subset $C$ of columns – the parity of each quad's permutation will change accordingly.

If all resulting permutations are even, we can fix them.

Thus, a solution exists (and can be easily generated)
iff
there exist boolean variables $r_1, \ldots, r_{H/2}, c_1, \ldots, c_{W/2}$,
such that the parity of every quad $(i, j)$ equals $r_i \oplus c_j$.

## Solving the equations

Instead of solving by Gauss elimination or SAT
(which was accepted if well-implemented),
we can use an equivalent condition:

$$par(i,j) = par(1,1) \oplus par(1,j) \oplus par(i,1)$$
for every $1 < i \leq H/2$, $1 < j \leq W/2$

Now, we can first solve the quads in the first row and column (for odd permutations, flipping their column or row, respectively) – the others either become even, or there is no solution at all.

# Problem J
# Captain Obvious and the Rabbit-Man

Submits: 16
Accepted: 1

First solved by:
Jagiellonian University
(Piotr Bejda, Igor Adamski, Jakub Adamek)
4:28:55

Author: Lech Duraj

## a.k.a. The Insane Problem

A sequence $p_i$ is defined as follows:

$$p_i = c_1 \cdot F_1^i + c_2 \cdot F_2^i + \ldots + c_k \cdot F_k^i$$

with $F_1, F_2, \ldots, F_k$ being the Fibonacci numbers.

## a.k.a. The Insane Problem

A sequence $p_i$ is defined as follows:

$$p_i = c_1 \cdot F_1^i + c_2 \cdot F_2^i + \ldots + c_k \cdot F_k^i$$

with $F_1, F_2, \ldots, F_k$ being the Fibonacci numbers.

We know $p_1, \ldots, p_k$ and have to guess $p_{k+1}$.

## a.k.a. The Insane Problem

A sequence $p_i$ is defined as follows:

$$p_i = c_1 \cdot F_1^i + c_2 \cdot F_2^i + \ldots + c_k \cdot F_k^i$$

with $F_1, F_2, \ldots, F_k$ being the Fibonacci numbers.

We know $p_1, \ldots, p_k$ and have to guess $p_{k+1}$.

This is the sum of geometrical sequences, and this has something to do with linear recursion.

## Magic formula

Compute the polynomial:

$$(X-1)(X-2)(X-3)(X-5)\ldots(X-F_k) = X^k + a_1X^{k-1} + a_2X^{k-2} + \ldots + a_k.$$

Then $p_i + a_1p_{i-1} + a_2p_{i-2} + \ldots + a_kp_{i-k} = 0.$

## Magic formula

Compute the polynomial:

$$(X-1)(X-2)(X-3)(X-5)\dots(X-F_k) = X^k + a_1X^{k-1} + a_2X^{k-2} + \dots + a_k.$$

Then $p_i + a_1 p_{i-1} + a_2 p_{i-2} + \dots + a_k p_{i-k} = 0$.

…which is enough to compute $p_{k+1}$ from the previous elements.

## Magic formula

Compute the polynomial:

$$(X-1)(X-2)(X-3)(X-5)\dots(X-F_k) = X^k + a_1X^{k-1} + a_2X^{k-2} + \dots + a_k.$$

Then $p_i + a_1 p_{i-1} + a_2 p_{i-2} + \dots + a_k p_{i-k} = 0$.

…which is enough to compute $p_{k+1}$ from the previous elements.

**WHAT?**

## Magic formula

We claim that the coefficients $a_i$ of the polynomial:

$$A(X) = (X-1)(X-2)(X-3)(X-5)\dots(X-F_k) =$$
$$X^k + a_1X^{k-1} + a_2X^{k-2} + \dots + a_k$$

satisfy the magic formula $p_i + a_1p_{i-1} + a_2p_{i-2} + \dots + a_kp_{i-k} = 0$ with our sequence $p_i$.

## Magic formula

We claim that the coefficients $a_i$ of the polynomial:

$$A(X) = (X-1)(X-2)(X-3)(X-5)\dots(X-F_k) = X^k + a_1X^{k-1} + a_2X^{k-2} + \dots + a_k$$

satisfy the magic formula $p_i + a_1p_{i-1} + a_2p_{i-2} + \dots + a_kp_{i-k} = 0$ with our sequence $p_i$.

Observe that if we insert $p_i = 2^i$, then the formula boils down to $A(2) = 0$, which is...well, obvious.

## Magic formula

We claim that the coefficients $a_i$ of the polynomial:

$$A(X) = (X-1)(X-2)(X-3)(X-5)\dots(X-F_k) = X^k + a_1X^{k-1} + a_2X^{k-2} + \dots + a_k$$

satisfy the magic formula $p_i + a_1p_{i-1} + a_2p_{i-2} + \dots + a_kp_{i-k} = 0$ with our sequence $p_i$.

Observe that if we insert $p_i = 2^i$, then the formula boils down to $A(2) = 0$, which is...well, obvious.

Similarly for $p_i = 3^i$, or $5^i$, or $\dots$, or $F_k^i$.

All these sequences satisfy the magic formula.

## Magic formula

We claim that the coefficients $a_i$ of the polynomial:

$$A(X) = (X-1)(X-2)(X-3)(X-5)\dots(X-F_k) = \\ X^k + a_1X^{k-1} + a_2X^{k-2} + \ldots + a_k$$

satisfy the magic formula $p_i + a_1p_{i-1} + a_2p_{i-2} + \ldots + a_kp_{i-k} = 0$ with our sequence $p_i$.

Observe that if we insert $p_i = 2^i$, then the formula boils down to $A(2) = 0$, which is...well, obvious.

Similarly for $p_i = 3^i$, or $5^i$, or $\ldots$, or $F_k^i$.

All these sequences satisfy the magic formula.

But if they all do, their linear combination fits as well.

So, it is enough to compute the coefficients of $(X-1)(X-2)(X-3)(X-5)\dots(X-F_k)$ modulo $M$.
This is easily done in $O(k^2)$ time.

So, it is enough to compute the coefficients of
$(X-1)(X-2)(X-3)(X-5)\dots(X-F_k)$ modulo $M$.
This is easily done in $O(k^2)$ time.
It could be, in fact, done in $O(k\log^2 k)$ with Fast Fourier Transform...

# Problem E
Escape

Submits: 21
Accepted: 0

First solved by:

—

—

Author: Łukasz Matylla & Lech Duraj

There is a tree with integer labels on nodes.

We gain or lose HP when we first enter a node.

Is it possible to go from 1 to target node $t$?

Simple observation: We attach a new target $t'$ to $t$ and give a large weight $W$ to $t'$.

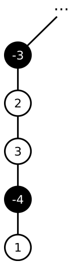
Instead of reaching $t$, we can now ask about gaining at least $W$ hitpoints.

## Path subtree

Suppose that there is a part of the tree that is a path.

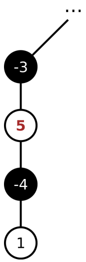
## Path subtree

Suppose that there is a part of the tree that is a path.
We can merge every two consecutive positive vertices.



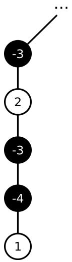
## Path subtree

Suppose that there is a part of the tree that is a path.
We can merge every two consecutive positive vertices.



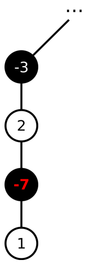
## Path subtree

We can merge every two consecutive negative vertices.



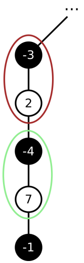
## Path subtree

We can merge every two consecutive negative vertices.



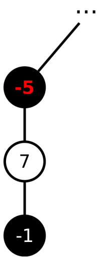
## Path subtree

We can merge every two consecutive pairs if the first positive vertex is not stronger than first negative one.



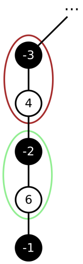
## Path subtree

We can merge every two consecutive pairs if the first positive vertex is not stronger than first negative one.



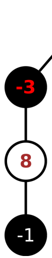
## Path subtree

Now, we can assume the negative vertices are in decreasing order – if not, we can merge them as well.

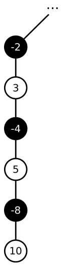


## Path subtree

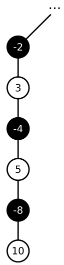
Now, we can assume the negative vertices are in decreasing order – if not, we can merge them as well.



We call the path now a *proper* path.



We call the path now a *proper* path.



Every path can be substituted with a proper path without changing the solution.

## Any subtree

There's more:

**Every subtree can be substituted with a proper path!**

## Any subtree

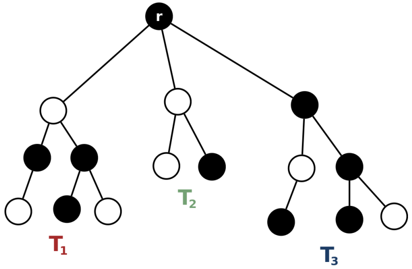
There's more:

**Every subtree can be substituted with a proper path!**

Here's the inductive proof:
(which is also a recursive algorithm for conversion)
Take a tree $T$ with root $r$ and subtrees $T_1, T_2, \dots, T_k$.

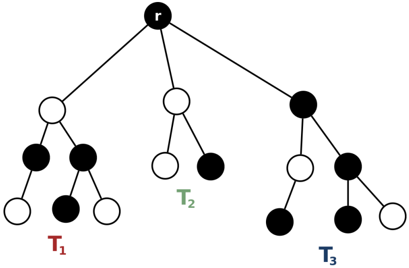
## Any subtree

Take a tree $T$ with root $r$ and subtrees $T_1, T_2, \ldots, T_k$.



## Any subtree

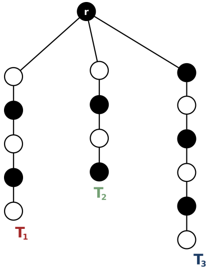
Take a tree $T$ with root $r$ and subtrees $T_1, T_2, \ldots, T_k$.



We first recursively substitute $T_1, T_2, \ldots, T_k$ with proper paths.

## Any subtree

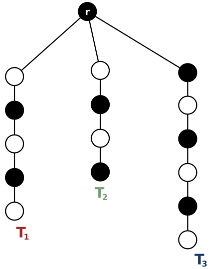
Take a tree $T$ with root $r$ and subtrees $T_1, T_2, \ldots, T_k$.



We first recursively substitute $T_1, T_2, \ldots, T_k$ with proper paths.

## Any subtree

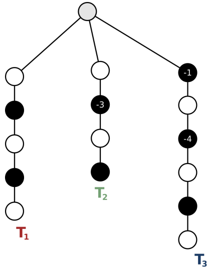
Take a tree $T$ with root $r$ and subtrees $T_1, T_2, \ldots, T_k$.



Now, assume the root has been eaten.

## Any subtree

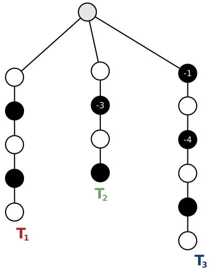
Take a tree $T$ with root $r$ and subtrees $T_1, T_2, \ldots, T_k$.



Now, assume the root has been eaten.

## Any subtree

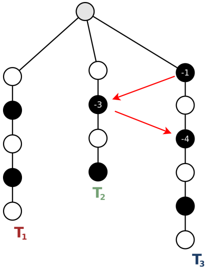
Take a tree $T$ with root $r$ and subtrees $T_1, T_2, \ldots, T_k$.



Now, assume the root has been eaten.
The paths vertices should be visited in decreasing order.

## Any subtree

Take a tree $T$ with root $r$ and subtrees $T_1, T_2, \ldots, T_k$.

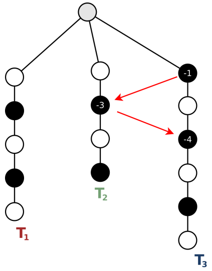


Now, assume the root has been eaten.

The paths vertices should be visited in decreasing order.

## Any subtree

Take a tree $T$ with root $r$ and subtrees $T_1, T_2, \ldots, T_k$.



We simply merge the sorted lists, add root on top, and restore proper path.

## The algorithm

If we change the whole tree to a proper path, it is easy to compute the largest possible HP we can gain.

The easiest implementation is to remember proper paths as sets and merge them by joining smaller one into larger one.

This leads to $O(n \log^2 n)$ algorithm. Using mergeable queues, we can do it in $O(n \log n)$. Not much faster in practice.

# Problem G
History course

Submits: 4

Accepted: 0

First solved by:

—

—

Author: Tomasz Krawczyk

Given a set of intervals on a line, construct an ordering that:

- Keeps non-intersecting intervals in the *natural* order.
- Minimizes the distance (in the order) between any two intersecting intervals.

## Observations

- We can do a binary search for the optimal value of a solution.

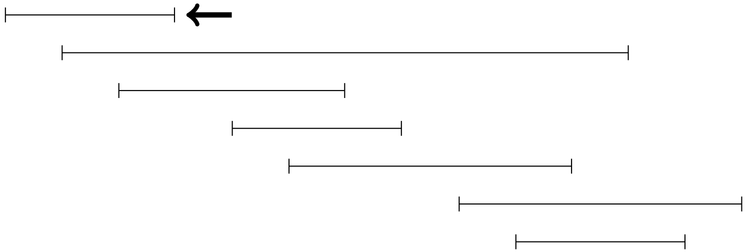
## Observations

- We can do a binary search for the optimal value of a solution.
- It's preferable to put first the intervals which end earlier.

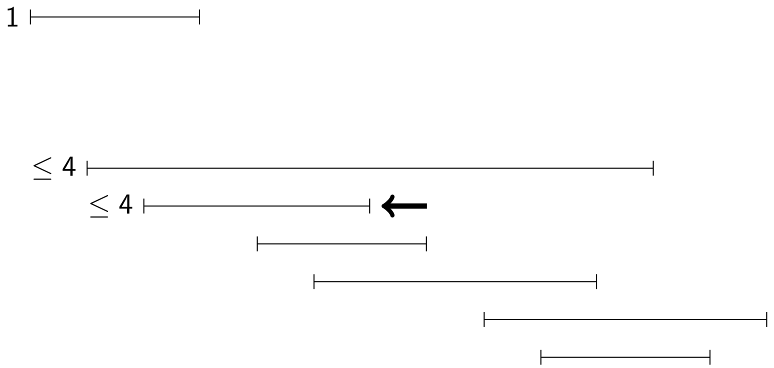
## Observations

- We can do a binary search for the optimal value of a solution.
- It's preferable to put first the intervals which end earlier.
- But it's not always possible.

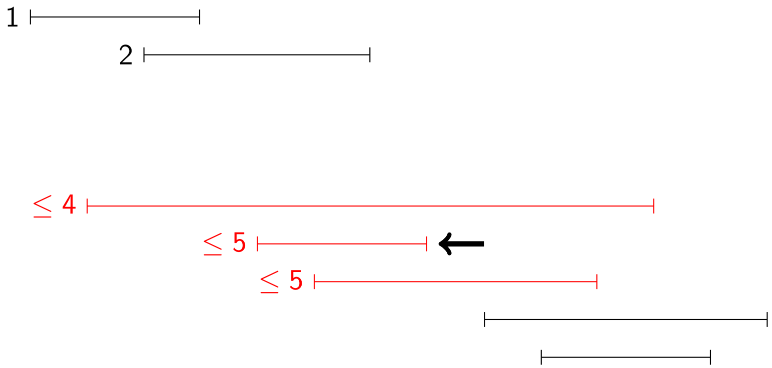
## Example



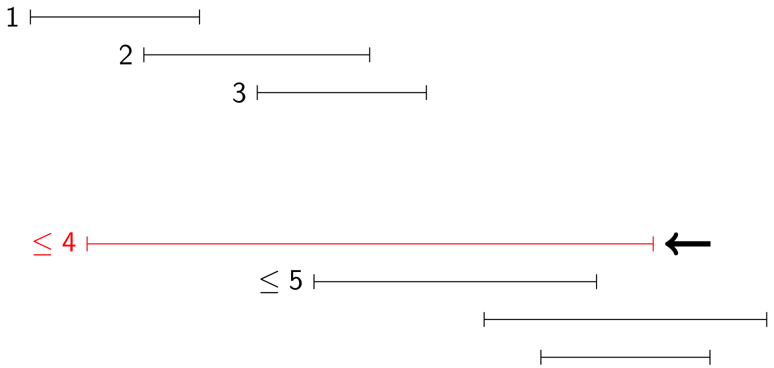
## Example



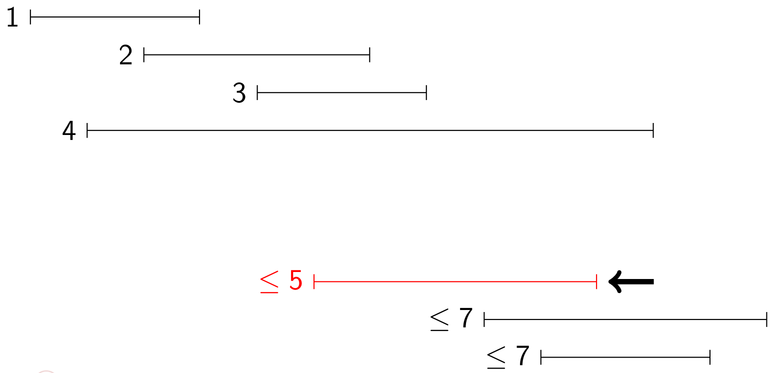
## Example



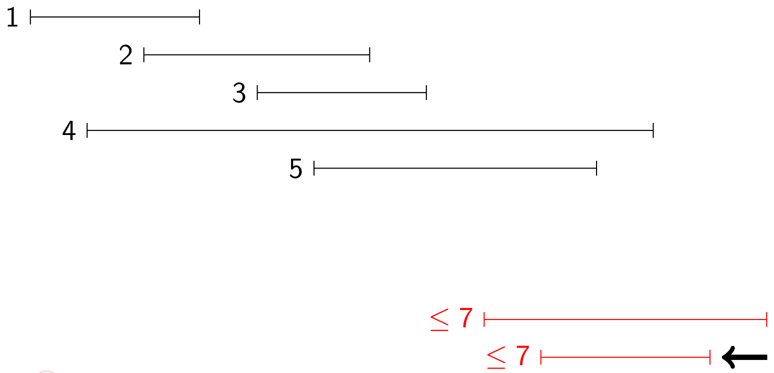
## Example



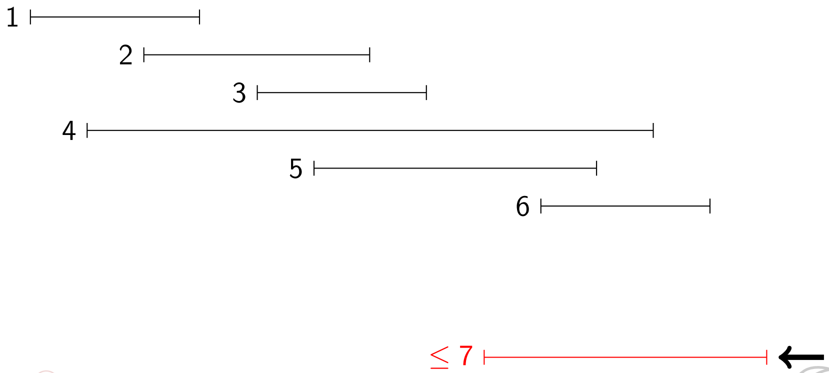
## Example



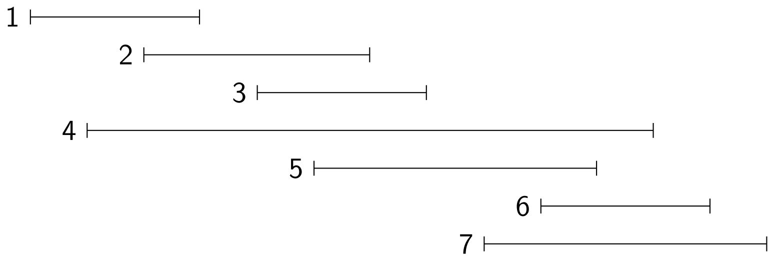
## Example



## Example



## Example



## Constructing optimal solution

- Construct the ordering from left to right.

## Constructing optimal solution

- Construct the ordering from left to right.
- Divide intervals into levels.

## Constructing optimal solution

- Construct the ordering from left to right.
- Divide intervals into levels.
- Find the first *dangerous* level $j$.

## Constructing optimal solution

- Construct the ordering from left to right.
- Divide intervals into levels.
- Find the first *dangerous* level $j$.
- Choose the earliest ending interval from levels up to $j$.

## Implementation

- Binary search.

## Implementation

- Binary search.
- Find the first *dangerous* level.

## Implementation

- Binary search.
- Find the first *dangerous* level.
- Find the earliest ending interval from the first $j$ levels.

## People involved

**Problem setters:**

Lech Duraj
Grzegorz Guśpiel
Grzegorz Gutowski
Grzegorz Herman
Arkadiusz Pawlik
Adam Polak
Bartosz Walczak

**Betatesters:**

Witold Jarnicki
Jonasz Pamuła
Maciej Wawro

## Animals involved

Unexpected night judge room guest: a striped field mouse.

(No animals were harmed in the making of this problemset.)

Thank you!