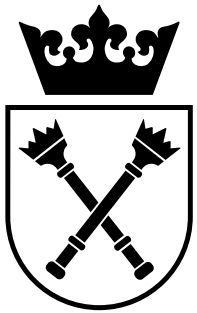
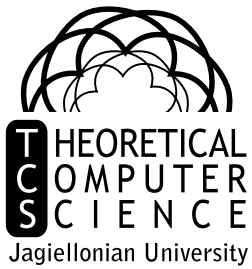
# UZI: Solutions

TCS UJ

24.10.2020





# Problem H
# Squareball

Submitted: 60
Accepted: 34

First solution:
Andrei Mishchanka
0:07

Author: Lech Duraj

Given a quadrilateral ABCD, determine if it is a square.

- **Solution 1**: Check if $|AB|^2 = |BC|^2 = |CD|^2 = |DA|^2$ and if $AB \perp BC$.
- **Solution 2**: Move the midpoint of $AC$ to $(0, 0)$. Now if $A = (x, y)$, then it should be $B = (-y, x), C = (-x, -y), D = (y, -x)$.
- **Solution 3**: Compute all the squares of distances between points $(|AB|^2, |AC|^2, \ldots, |CD|^2)$, sort the sequence, and check if it is $(a, a, a, a, 2a, 2a)$ for some $a$.

# Problem F
# Anniversary commission

Submitted: 61
Accepted: 26

First solution:
Andrei Mishchanka
0:13

Author: Lech Duraj

For a given word $w$, how much letters need to be changed for $w$ to be $v^r$ for some word $v$ and $n \geq 2$?

For a given word $w$, how much letters need to be changed for $w$ to be $v^r$ for some word $v$ and $n \geq 2$?

Suppose we know $r$ and let $s = n/r$. For any $i = 1, 2, \ldots, s$, all the symbols $w[i], w[i+s], w[i+2s], \ldots$ must be identical – to make them so, find the most popular symbol and change all other ones to the winner.

For a given word $w$, how much letters need to be changed for $w$ to be $v^r$ for some word $v$ and $n \geq 2$?

Suppose we know $r$ and let $s = n/r$. For any $i = 1, 2, \ldots, s$, all the symbols $w[i], w[i+s], w[i+2s], \ldots$ must be identical – to make them so, find the most popular symbol and change all other ones to the winner.

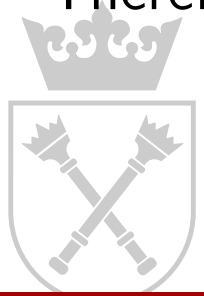
Of course, $r$ must be a divisor of $n$ – which leads to $\mathcal{O}(n \cdot d(n))$ algorithm (formally, $d(n) \sim n^{1/\log\log n}$). This *might* get accepted, but we can do better...

For a given word $w$, how much letters need to be changed for $w$ to be $v^r$ for some word $v$ and $n \geq 2$?

Suppose we know $r$ and let $s = n/r$. For any $i = 1, 2, \ldots, s$, all the symbols $w[i], w[i+s], w[i+2s], \ldots$ must be identical – to make them so, find the most popular symbol and change all other ones to the winner.

Of course, $r$ must be a divisor of $n$ – which leads to $\mathcal{O}(n \cdot d(n))$ algorithm (formally, $d(n) \sim n^{1/\log\log n}$). This *might* get accepted, but we can do better...

Observe that if $r' | r$, then the answer for $r'$ is better (at least not worse). Therefore it is enough to check all **prime** divisors of $n$. This is $\mathcal{O}(n \log n)$.



# Problem D
# Almost-composite

Submitted: 52
Accepted: 25

First solution:
Hubert Zięba
1:22



Author: Krzysztof Maziarz

## Statement

A positive integer is *almost-composite* if one can remove zero or more digits from its decimal representation to obtain a composite number. Given a positive integer $k$, find the $k$-th smallest almost-composite number.

## Statement

A positive integer is *almost-composite* if one can remove zero or more digits from its decimal representation to obtain a composite number. Given a positive integer $k$, find the $k$-th smallest almost-composite number.

## Solution

The only non-almost-composite positive integers are:

$$2, 3, 5, 7, 11, 13, 17, 23, 31, 37, 53, 71, 73, 113,$$
$$131, 137, 173, 311, 313, 317, 373, 1373, 3137,$$

and they can be generated with a simple program. The task is hence to find $k$-th number which is not listed above. This can be achieved in $O(1)$.



THEORETICAL COMPUTER SCIENCE
Jagiellonian University

How to be sure we have found all non-almost-composite numbers?

- Option 1: Prove that any number with 6 or more digits must be almost-composite (Hint: if there are any zeros then great, otherwise divide into three 2-digit blocks and analyse modulo 3). Then run the program to check all the numbers between 1 and 100'000.
- Option 2: Run the program, observe that there are no 5-digit non-almost-composite numbers. Therefore, there also cannot exist any such numbers with more than 5 digits.

# Problem A
# Robbery

Submitted: 43
Accepted: 15

First solution:
Andrei Mishchanka
0:46



Author: Krzysztof Maziarz

## Statement

Given sequence $v_1, \ldots, v_n$ and number $k$ ($n \leq 500\,000, k \leq 5$), compute

$$\sum_{\substack{1 \leq i,j \leq n \\ j-i+1 \geq k}} b_k(i,j),$$

where $b_k$ denotes the $k$-th largest value among $v_i, \ldots, v_j$.

## Statement

Given sequence $v_1, \ldots, v_n$ and number $k$ ($n \leq 500\,000, k \leq 5$), compute

$$\sum_{\substack{1 \leq i,j \leq n \\ j-i+1 \geq k}} b_k(i,j),$$

where $b_k$ denotes the $k$-th largest value among $v_i, \ldots, v_j$.

## Solution

For each $v_i$, compute how many intervals have $v_i$ as $k$-th largest value.

## Statement

Given sequence $v_1, \ldots, v_n$ and number $k$ ($n \leq 500\,000, k \leq 5$), compute

$$\sum_{\substack{1 \leq i,j \leq n \\ j-i+1 \geq k}} b_k(i,j),$$

where $b_k$ denotes the $k$-th largest value among $v_i, \ldots, v_j$.

## Solution

For each $v_i$, compute how many intervals have $v_i$ as $k$-th largest value.

## Example

$(k = 3)$

## Statement

Given sequence $v_1, \ldots, v_n$ and number $k$ ($n \leq 500\,000, k \leq 5$), compute

$$\sum_{\substack{1 \leq i,j \leq n \\ j-i+1 \geq k}} b_k(i,j),$$

where $b_k$ denotes the $k$-th largest value among $v_i, \ldots, v_j$.

## Solution

For each $v_i$, compute how many intervals have $v_i$ as $k$-th largest value.

## Example

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | 9 | | $(k = 3)$ |

## Statement

Given sequence $v_1, \ldots, v_n$ and number $k$ ($n \leq 500\,000, k \leq 5$), compute

$$\sum_{\substack{1 \leq i,j \leq n \\ j-i+1 \geq k}} b_k(i,j),$$

where $b_k$ denotes the $k$-th largest value among $v_i, \ldots, v_j$.

## Solution

For each $v_i$, compute how many intervals have $v_i$ as $k$-th largest value.

## Example

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | 8 | 9 | | $(k = 3)$ |

## Statement

Given sequence $v_1, \ldots, v_n$ and number $k$ ($n \leq 500\,000, k \leq 5$), compute

$$\sum_{\substack{1 \leq i,j \leq n \\ j-i+1 \geq k}} b_k(i,j),$$

where $b_k$ denotes the $k$-th largest value among $v_i, \ldots, v_j$.

## Solution

For each $v_i$, compute how many intervals have $v_i$ as $k$-th largest value.

## Example

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | 6 | | | | | | 8 | 9 | | $(k = 3)$ |

## Statement

Given sequence $v_1, \ldots, v_n$ and number $k$ ($n \leq 500\,000, k \leq 5$), compute

$$\sum_{\substack{1 \leq i,j \leq n \\ j-i+1 \geq k}} b_k(i,j),$$

where $b_k$ denotes the $k$-th largest value among $v_i, \ldots, v_j$.

## Solution

For each $v_i$, compute how many intervals have $v_i$ as $k$-th largest value.

## Example

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 5 | | | 6 | | | | | | 8 | 9 | | $(k = 3)$ |

## Statement

Given sequence $v_1, \ldots, v_n$ and number $k$ ($n \leq 500\,000, k \leq 5$), compute

$$\sum_{\substack{1 \leq i,j \leq n \\ j-i+1 \geq k}} b_k(i,j),$$

where $b_k$ denotes the $k$-th largest value among $v_i, \ldots, v_j$.

## Solution

For each $v_i$, compute how many intervals have $v_i$ as $k$-th largest value.

## Example

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 5 | 3 | | 6 | | | | | | 8 | 9 | | $(k = 3)$ |

## Statement

Given sequence $v_1, \ldots, v_n$ and number $k$ ($n \leq 500\,000, k \leq 5$), compute

$$\sum_{\substack{1 \leq i,j \leq n \\ j-i+1 \geq k}} b_k(i,j),$$

where $b_k$ denotes the $k$-th largest value among $v_i, \ldots, v_j$.

## Solution

For each $v_i$, compute how many intervals have $v_i$ as $k$-th largest value.

## Example

| 5 | 3 | | 6 | | | 2 | | | 8 | 9 | |
|---|---|---|---|---|---|---|---|---|---|---|---|

$(k = 3)$

## Statement

Given sequence $v_1, \ldots, v_n$ and number $k$ ($n \leq 500\,000, k \leq 5$), compute

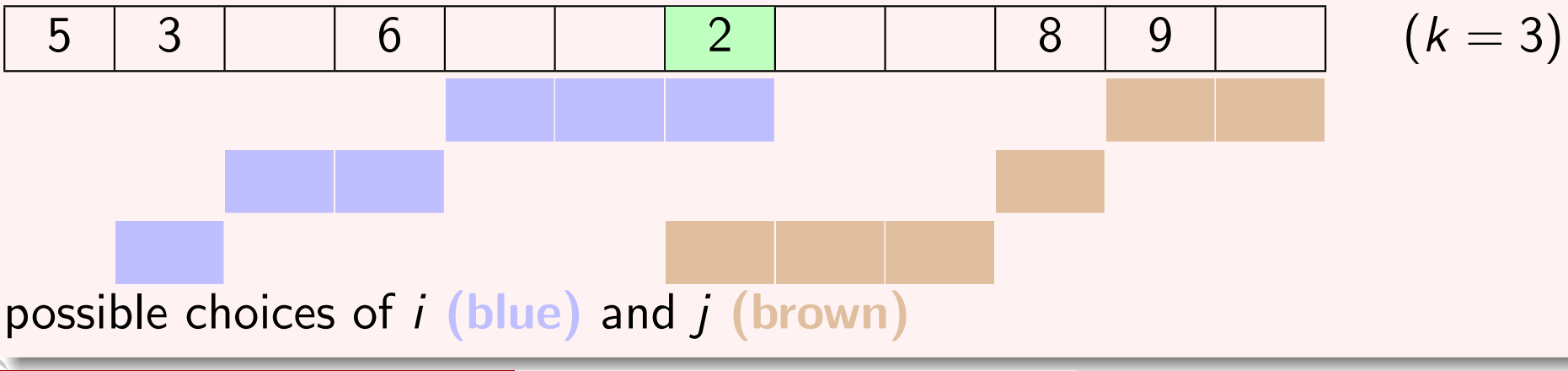
$$\sum_{\substack{1 \leq i,j \leq n \\ j-i+1 \geq k}} b_k(i,j),$$

where $b_k$ denotes the $k$-th largest value among $v_i, \ldots, v_j$.

## Solution

For each $v_i$, compute how many intervals have $v_i$ as $k$-th largest value.

## Example




possible choices of $i$ (blue) and $j$ (brown)

# Problem G
# No turn

Submitted: 49
Accepted: 12

First solution:
Jacek Salata
0:49



Author: Krzysztof Kleiner

## Statement

Given a graph with $n$ nodes, $m$ edges and $k$ no turn signs (i.e. pairs of roads $(e, f)$ such that you cannot use $f$ immediately after $e$), determine the shortest routes from the starting vertex to all other vertices.

## Statement

Given a graph with $n$ nodes, $m$ edges and $k$ no turn signs (i.e. pairs of roads $(e, f)$ such that you cannot use $f$ immediately after $e$), determine the shortest routes from the starting vertex to all other vertices.

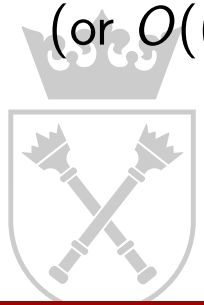
## Solution

Run Dijkstra. While resolving a vertex, use only those outgoing edges which are legal for your incoming edge. Don't mark the vertex as *visited* - instead, replace the list of its (outgoing) edges with the list of those edges which you had to omit due to a no-turn sign.

## Solution

Run Dijkstra. While resolving a vertex, use only those outgoing edges which are legal for your incoming edge. Don't mark the vertex as *visited* - instead, replace the list of its (outgoing) edges with the list of those edges which you had to omit due to a no-turn sign.

*Complexity analysis*:
A vertex can be visited multiple times, but every (outgoing) edge $f$ can be checked at most as many times as the number of $(*, f)$ no-turn signs appearing on the input, before (at most) one final check which will result in the Dijkstra algorithm actually going through it (i.e. adding its end vertex to the queue). Therefore, the whole Dijkstra runs in $O(k + m\log m)$ (or $O((k + m)\log(k + m))$ in a less careful implementation, which is also OK).



## Solution 2 (not recommended)

For every vertex $v$, build a segment tree on $v$'s outgoing edges. For an incoming edge, compute the set of the tree's intervals covering all the 'legal' outgoing edges and mark those intervals as *visited* after performing the Dijkstra step. On a next visit to this vertex, go down the tree starting from the set of intervals covering all the 'legal' edges of your new incoming edge, but do not descend further down the tree whenever you meet an interval which is already fully visited.

## Solution 2 (not recommended)

For every vertex $v$, build a segment tree on $v$'s outgoing edges. For an incoming edge, compute the set of the tree's intervals covering all the 'legal' outgoing edges and mark those intervals as *visited* after performing the Dijkstra step. On a next visit to this vertex, go down the tree starting from the set of intervals covering all the 'legal' edges of your new incoming edge, but do not descend further down the tree whenever you meet an interval which is already fully visited.

*Complexity*: $O((k+m)\log(k+m))$

## Solution 2 (not recommended)

For every vertex $v$, build a segment tree on $v$'s outgoing edges. For an incoming edge, compute the set of the tree's intervals covering all the 'legal' outgoing edges and mark those intervals as *visited* after performing the Dijkstra step. On a next visit to this vertex, go down the tree starting from the set of intervals covering all the 'legal' edges of your new incoming edge, but do not descend further down the tree whenever you meet an interval which is already fully visited.

*Complexity*: $O((k+m)\log(k+m))$

## Solution 2a (not recommended)

Do not create separate segment trees, but instead modify the graph by injecting a segment tree (of the outgoing edges) in place of each vertex. Add relevant edges with wieghts 0. Running Dijkstra on this graph simulates the behaviour of Solution 2.

*Complexity*: $O((k+m)\log(k+m))$ as well (not $\log^2$, actually!)

# Problem C
# Skyscraper

Submitted: 69
Accepted: 7

First solution:
Jan Klimczak
1:24



Author: Michał Seweryn

## Statement

In a building with $10^9$ floors there is an elevator which starts at floor 0 and moves up one floor every $s$ second. There are $n$ people, and the $i$-th of them wants to get to the floor $f_i$. Each person can either use the elevator and walk using a staircase. It takes the $i$-th person $u_i$ seconds to walk one floor upstairs, and $d_i$ seconds to walk one floor downstairs. Schedule at most $k$ floors where the people can leave the elevator, to minimize the time in which everyone can reach their floor

## Solution

Binary search the answer.

How to check if everyone can reach their floor in $T$ seconds?

## Observation 1

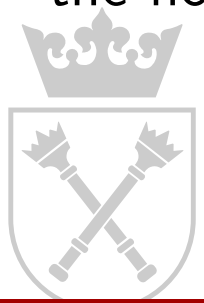
if $u_i \cdot f_i \leq T$, then the $i$-th person does not need to use the elevator – we can ignore such people

## Observation 2

if $u_i \cdot f_i > T$ and $s \cdot f_i > T$ for some $i$, then the answer is NO.

It remains to consider indices $i$ such that $s \cdot f_i \leq T < u_i \cdot f_i$. For each such $i$, the $i$-th person can reach their floor in at most $T$ seconds if and only if the floor stops at a floor from the interval

$$[a_i, b_i] = \left[ \frac{T - f_i \cdot u_i}{s - u_i}, \frac{T + f_i \cdot d_i}{s + d_i} \right].$$



The problem of determining, whether there exists a set $S$ of size at most $k$ integers which intersects each interval $[a_i, b_i]$ can be solved with a greedy algorithm in $O(n \log n)$.

**No element of $S$ should exceed $10^9$! (the levitating elevator problem)** The total complexity of the algorithm is $O(\log M \cdot n \log n)$ where $M = 10^9$ is the height of the skyscraper.

# Problem B
# Chocolate

Submitted: 29
Accepted: 2

First solution:
Nazarii Denha
2:18



Author: Krzysztof Maziarz

## Problem

Given an $a$ x $b$ rectangle, cut out as many 2 x 3 and 3 x 2 blocks as possible.

First, we can assume $a, b \geq 2$, otherwise we cannot cut out anything.

First, we can assume $a, b \geq 2$, otherwise we cannot cut out anything.

The amount of leftovers is at least $(a \cdot b) \mod 6$, we will show a solution that matches this lower bound.

## Observation

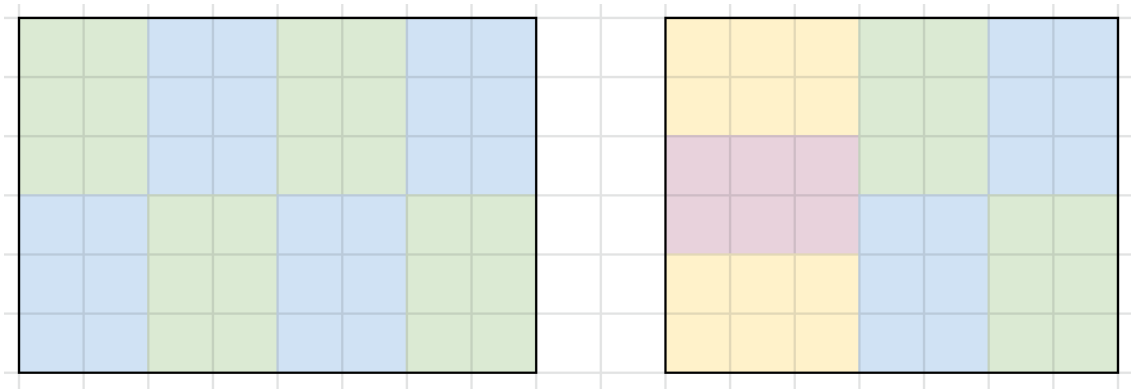
For any $a \geq 2$, we can fully decompose a rectangle 6 x $a$.

## Observation

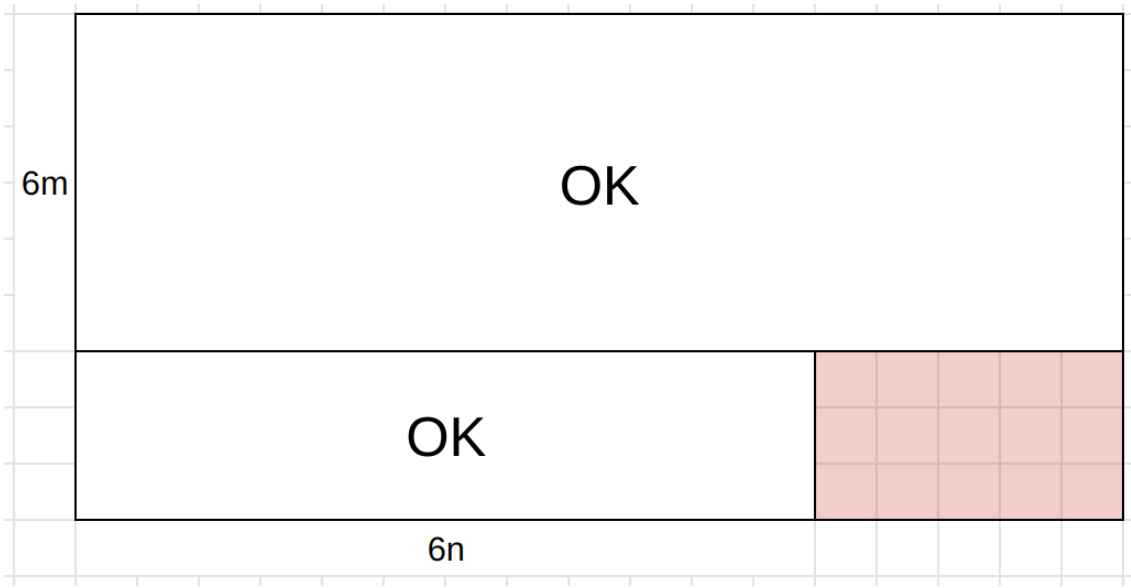
For any $a \geq 2$, we can fully decompose a rectangle 6 x $a$.

## Proof

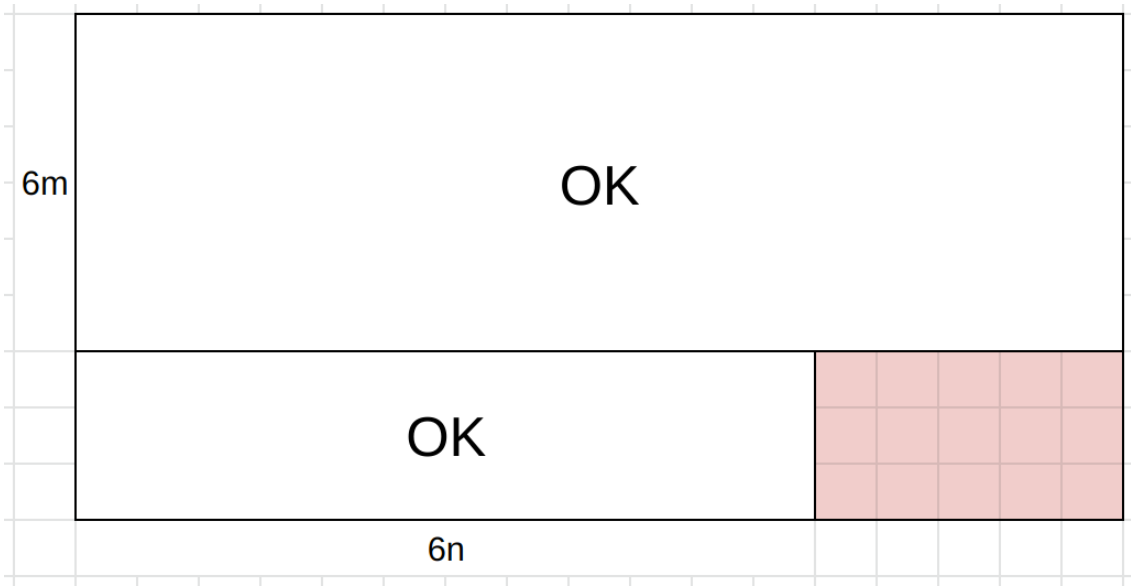
If $a$ is even, we can tile everything vertically. If it's odd, we can do one horizontal stripe, and then the rest vertically.



Given our rectangle, we can cut any multiple of 6 from both sides.

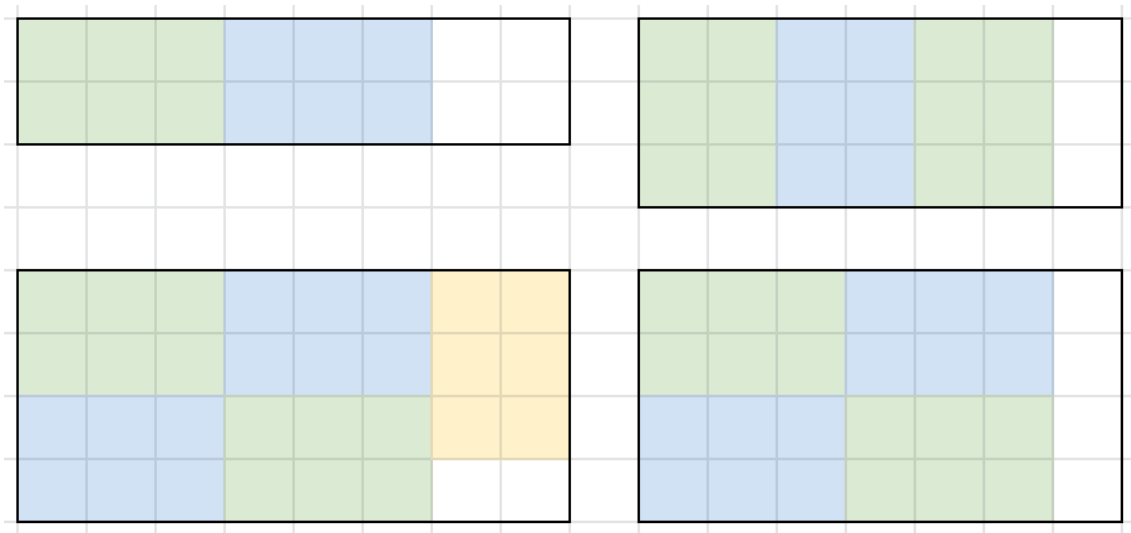


Given our rectangle, we can cut any multiple of 6 from both sides.

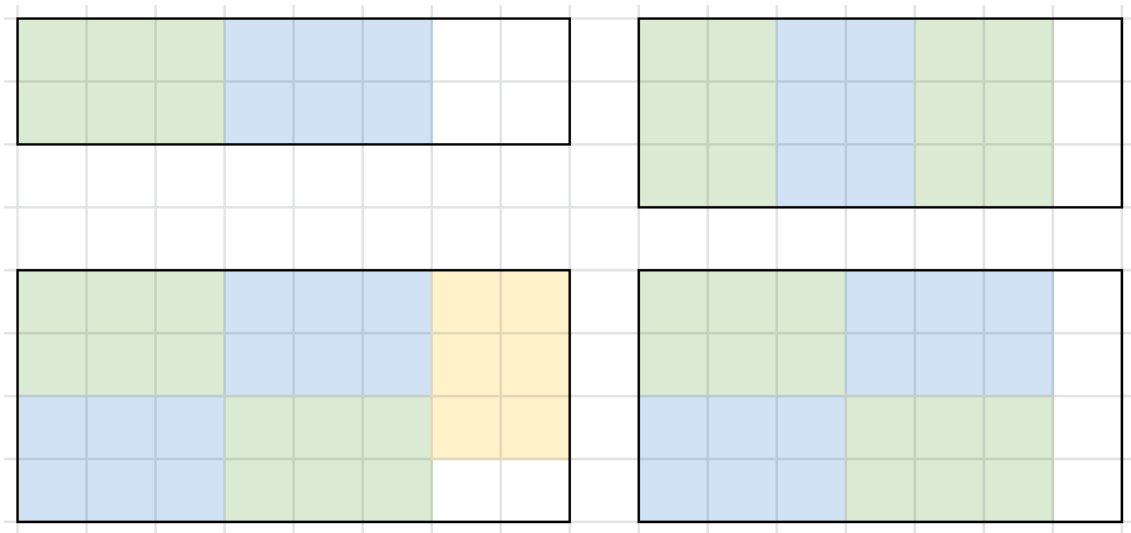



It remains to handle the corner, which is a rectangle of shape $r_a \times r_b$ where $r_a, r_b \in \{2, 3, 4, 5, \underline{7}\}$.

Cases of $r_a \in \{2, 3, 4\}$ are easy.

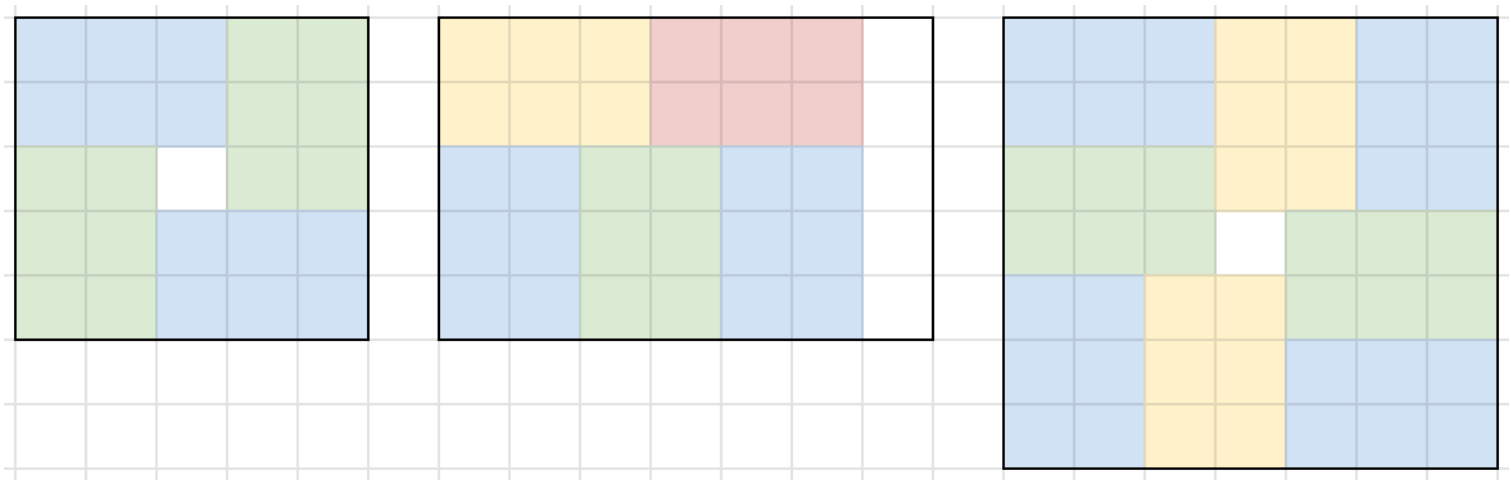


Cases of $r_a \in \{2, 3, 4\}$ are easy.



Similarly we can solve $r_b \in \{2, 3, 4\}$.

The only cases left are 5 x 5, 5 x 7 and 7 x 7. These have pretty solutions that we can find by hand:



For ease of implementation, we can first generate the blocks that are part of the solution, and then "color" them into $'Z' -' A' + 1$ colors.

For ease of implementation, we can first generate the blocks that are part of the solution, and then "color" them into $'Z' -' A' + 1$ colors.

We can go through the blocks and always pick the first colors that fits (i.e. "first fit").

For ease of implementation, we can first generate the blocks that are part of the solution, and then "color" them into $'Z' -' A' + 1$ colors.

We can go through the blocks and always pick the first colors that fits (i.e. "first fit").

Or, we can color a block that has its top-left corner in $(x, y)$ with color $(x + y) \mod C$; one can see that for $C \geq 5$ this will not produce two adjacent blocks of the same color.

# Problem E
# Civilizations

Submitted: 9
Accepted: 2

First solution:
Andrei Mishchanka
2:50

Author: Krzysztof Maziarz

## Problem

We're given an $n \times n$ field divided into $n^2$ unit squares; each square has an owner civilization and a value. For civilization $p$, we define its length of borders $l_p$ and sum-of-values $w_p$. There are $q$ events where a unit square changes owners, after each event determine the maximum value of $A \cdot w_p + B \cdot l_p + C \cdot w_p \cdot l_p$ ($A$, $B$, $C$ are different for each query).

We can think of civilizations as points $(l_p, w_p)$. Every change of owner for a single unit square impacts the values for the old owner, new owner, and owners of adjacent squares. In total, $\mathcal{O}(1)$ points change.

We can think of civilizations as points $(l_p, w_p)$. Every change of owner for a single unit square impacts the values for the old owner, new owner, and owners of adjacent squares. In total, $\mathcal{O}(1)$ points change.

If not for the $C \cdot w_p \cdot l_p$ part, we could think of the convex hull of our set of points. But handling the full scoring function for arbitrary points and updates seems tricky...

We can think of civilizations as points $(l_p, w_p)$. Every change of owner for a single unit square impacts the values for the old owner, new owner, and owners of adjacent squares. In total, $\mathcal{O}(1)$ points change.

If not for the $C \cdot w_p \cdot l_p$ part, we could think of the convex hull of our set of points. But handling the full scoring function for arbitrary points and updates seems tricky...

Is there anything special about the set of points $(l_p, w_p)$?

The range of values for $l_p$ and $w_p$ are large, that's not going to help. But...

The range of values for $l_p$ and $w_p$ are large, that's not going to help. But...

**Observation**

The sum of values of $l_p$ is $\mathcal{O}(n^2)$.

The range of values for $l_p$ and $w_p$ are large, that's not going to help. But...

**Observation**

The sum of values of $l_p$ is $\mathcal{O}(n^2)$.

**Fact**

If $x_1 + \cdots + x_k = m$, then there are only $\mathcal{O}(\sqrt{m})$ *distinct* values $x_i$.

So, we get a crucial observation: while there may be $\mathcal{O}(n^2)$ civilizations, there will only be $\mathcal{O}(n)$ different values of $l_p$ at any given time.

We can rewrite our scoring as

$$A \cdot w_p + B \cdot l_p + C \cdot w_p \cdot l_p = B \cdot l_p + (A + C \cdot l_p) \cdot w_p$$

If $l_p$ is fixed, it's optimal to choose either the largest or smallest $w_p$ (depending on the sign of $A + C \cdot l_p$).

We can rewrite our scoring as

$$A \cdot w_p + B \cdot l_p + C \cdot w_p \cdot l_p = B \cdot l_p + (A + C \cdot l_p) \cdot w_p$$

If $l_p$ is fixed, it's optimal to choose either the largest or smallest $w_p$ (depending on the sign of $A + C \cdot l_p$).

Lets store all civilizations grouped by $l_p$; within a single group sorted by $w_p$.

We can rewrite our scoring as

$$A \cdot w_p + B \cdot l_p + C \cdot w_p \cdot l_p = B \cdot l_p + (A + C \cdot l_p) \cdot w_p$$

If $l_p$ is fixed, it's optimal to choose either the largest or smallest $w_p$ (depending on the sign of $A + C \cdot l_p$).

Lets store all civilizations grouped by $l_p$; within a single group sorted by $w_p$.

To answer a query, we iterate through all groups of civilizations and check either the minimum or maximum element.

We can rewrite our scoring as

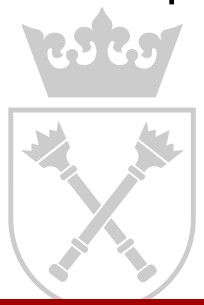
$$A \cdot w_p + B \cdot l_p + C \cdot w_p \cdot l_p = B \cdot l_p + (A + C \cdot l_p) \cdot w_p$$

If $l_p$ is fixed, it's optimal to choose either the largest or smallest $w_p$ (depending on the sign of $A + C \cdot l_p$).

Lets store all civilizations grouped by $l_p$; within a single group sorted by $w_p$.

To answer a query, we iterate through all groups of civilizations and check either the minimum or maximum element.

Complexity: $\mathcal{O}(n^2 + q \cdot n)$



# Jury

Lech Duraj
Grzegorz Guśpiel
Krzysztof Kleiner
Krzysztof Maziarz
Adam Polak
Michał Seweryn

# Results!

| Rank | Name | Score | Time | Tasks |
|---|---|---|---|---|
| **6** | Jacek Salata | 5 | 7:58:47 | H G* A F** D* |
| **7** | Justyna Jaworska | 5 | 12:52:44 | H F D A* C** |
| **8** | Krzysztof Pióro | 5 | 13:13:09 | H F A** D G* |
| **9** | Grzegorz Gawryał | 5 | 13:27:12 | H F* D C* A(6) |
| **10** | Rafał Pyzik | 5 | 13:56:48 | H* F D A G** |
| **11** | Katzper Michno | 5 | 14:13:27 | H** F A** D* C(5) |
| **12** | Pavel Sankin | 5 | 16:47:16 | H F D C(6)A(9) |

# Results!

| Rank | Name | Score | Time | Tasks |
|---|---|---|---|---|

# Results!

| Rank | Name | Score | Time | Tasks |
|---|---|---|---|---|
| **5** | Łukasz Janeczko | 6 | 17:44:20 | F* H B* D A G** |

# Results!

| Rank | Name | Score | Time | Tasks |
|---|---|---|---|---|
| **4** | Nazarii Denha | 6 | 16:28:03 | H* F B* G*** D A* |
| **5** | Łukasz Janeczko | 6 | 17:44:20 | F* H B* D A G** |

# Results!

| Rank | Name | Score | Time | Tasks |
|---|---|---|---|---|
| **3** | Jan Klimczak | 6 | 14:41:36 | H F C*** A* D* G |
| **4** | Nazarii Denha | 6 | 16:28:03 | H* F B* G*** D A* |
| **5** | Łukasz Janeczko | 6 | 17:44:20 | F* H B* D A G** |

# Results!

| Rank | Name | Score | Time | Tasks |
|---|---|---|---|---|
| **2** | Krzysztof Potępa | 6 | 12:20:57 | H F G D A* C**** |
| **3** | Jan Klimczak | 6 | 14:41:36 | H F C*** A* D* G |
| **4** | Nazarii Denha | 6 | 16:28:03 | H* F B* G*** D A* |
| **5** | Łukasz Janeczko | 6 | 17:44:20 | F* H B* D A G** |

# Results!

| Rank | Name | Score | Time | Tasks |
|---|---|---|---|---|
| **1** | Andrei Mishchanka | 7 | 13:11:20 | H F A* G D E*** C** |
| **2** | Krzysztof Potępa | 6 | 12:20:57 | H F G D A* C**** |
| **3** | Jan Klimczak | 6 | 14:41:36 | H F C*** A* D* G |
| **4** | Nazarii Denha | 6 | 16:28:03 | H* F B* G*** D A* |
| **5** | Łukasz Janeczko | 6 | 17:44:20 | F* H B* D A G** |