CERC 2012: Presentation of solutions

Jagiellonian University

November 28, 2012

Some numbers

Total submits: 1006

Accepted submits: 310

Some numbers

Total submits: 1006
Accepted submits: 310

First accept: 0:06:29, University of Wrocław
Last accept: 4:59:39, University of Zagreb

Some numbers

Total submits: 1006
Accepted submits: 310

First accept: 0:06:29, University of Wrocław
Last accept: 4:59:39, University of Zagreb

Most determined team: CTU Prague, 23rd attempt on task C succesful.

Problem H
Darts

Submits: 92
Accepted: 77

First solved by:
University of Wroclaw
(Bartłomiej Dudek, Maciej Dulęba, Mateusz Gołębiewski)
0:06:29

Author: Prof. Paweł Idziak

Problem C
Chemist's Vows

Submits: 197
Accepted: 64

First solved by:
Charles University in Prague
(Jakub Zíka, Filip Hlásek, Lukáš Folwarczný)
0:13:33

Author: Lech Duraj

Very simple dynamic programming.

Very simple dynamic programming.

Can say $word[1..k]$ if:

Very simple dynamic programming.

Can say $word[1..k]$ if:

- can say $word[1..k-2]$ and last two letters are an element symbol,

Very simple dynamic programming.

Can say $word[1..k]$ if:

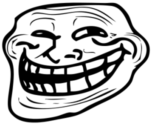
- can say $word[1..k-2]$ and last two letters are an element symbol,
- or can say $word[1..k-1]$ and last letter is an element symbol.

Very simple dynamic programming.

Can say $word[1..k]$ if:

- can say $word[1..k-2]$ and last two letters are an element symbol,
- or can say $word[1..k-1]$ and last letter is an element symbol.

For every k iterate through all elements.



PROBLEM?

Problem A
Kingdoms

Submits: 115
Accepted: 42

First solved by:
University of Warsaw
(Tomasz Kociumaka, Marcin Andrychowicz, Maciej Klimek)
0:16:21

Author: Leszek Horwath

- Make a graph:

- Make a graph:
  - $2^n - 1$ vertices—every nonempty subset of kingdoms;

- Make a graph:
  - $2^n - 1$ vertices—every nonempty subset of kingdoms;
  - for every vertex compute which kingdoms can bankrupt and add corresponding edges.

- Make a graph:
  - $2^n - 1$ vertices—every nonempty subset of kingdoms;
  - for every vertex compute which kingdoms can bankrupt and add corresponding edges.
- Use DFS to find vertices reachable from the vertex $\{1, 2, \ldots, n\}$.

- Make a graph:
  - $2^n - 1$ vertices—every nonempty subset of kingdoms;
  - for every vertex compute which kingdoms can bankrupt and add corresponding edges.
- Use DFS to find vertices reachable from the vertex $\{1, 2, \ldots, n\}$.
- For each i check whether the vertex $\{i\}$ is reachable.

- Make a graph:
  - $2^n - 1$ vertices—every nonempty subset of kingdoms;
  - for every vertex compute which kingdoms can bankrupt and add corresponding edges.
- Use DFS to find vertices reachable from the vertex $\{1, 2, \ldots, n\}$.
- For each i check whether the vertex $\{i\}$ is reachable.

- Make a graph:
  - $2^n - 1$ vertices—every nonempty subset of kingdoms;
  - for every vertex compute which kingdoms can bankrupt and add corresponding edges.
- Use DFS to find vertices reachable from the vertex $\{1, 2, \ldots, n\}$.
- For each i check whether the vertex $\{i\}$ is reachable.

Running time: $O(2^n n^2)$

Problem J
Conservation

Submits: 132
Accepted: 40

First solved by:
Jagiellonian University in Kraków
(Piotr Bejda, Michał Sapalski, Igor Adamski)
0:28:25

Author: Adam Polak

Sort the stages topologically:

Sort the stages topologically:

- hold a queue Q of stages with indegree 0
- take stages from Q and remove them from the graph

Sort the stages topologically:

- hold a queue Q of stages with indegree 0
- take stages from Q and remove them from the graph

Optimize greedily:

Sort the stages topologically:

- hold a queue Q of stages with indegree 0
- take stages from Q and remove them from the graph

Optimize greedily:

- completing a stage cannot harm the others
- we can lose nothing by performing it immediately

Sort the stages topologically:

- hold a queue Q of stages with indegree 0
- take stages from Q and remove them from the graph

Optimize greedily:

- completing a stage cannot harm the others
- we can lose nothing by performing it immediately
- use two queues Q_1 and Q_2, switching only when you have to

Running time: $O(n + m)$

Problem E
Word equations

Submits: 154
Accepted: 29

First solved by:
Comenius University
(Tomáš Belan, Vladimír Boža, Peter Fulla)
0:36:20

Author: Lech Duraj

Without equations—simple greedy algorithm:

Without equations—simple greedy algorithm:

- keep a count C of matched pattern symbols
- for each symbol in the text, increase C if the symbols match

Without equations—simple greedy algorithm:

- keep a count C of matched pattern symbols
- for each symbol in the text, increase C if the symbols match

Running time: $O(|T||P|) = O(2^k|P|)$

With equations—dynamic programming:

With equations—dynamic programming:

- define $m(S, i) =$ the value of C after checking S, assuming the check started with $C = i$

With equations—dynamic programming:

- define $m(S, i)$ = the value of C after checking S, assuming the check started with $C = i$
- for equations "S = word", calculate $m(S, i)$ greedily
- for equations "$S = S_1 + S_2$", we have $m(S, i) = m(S_2, m(S_1, i))$

With equations—dynamic programming:

- define $m(S, i)$ = the value of C after checking S, assuming the check started with $C = i$
- for equations "$S =$ word", calculate $m(S, i)$ greedily
- for equations "$S = S_1 + S_2$", we have $m(S, i) = m(S_2, m(S_1, i))$
- calculate bottom-up or use memoization

Running time: $O(k|P|)$

With equations—dynamic programming:

- define $m(S, i)$ = the value of C after checking S, assuming the check started with $C = i$
- for equations "S = word", calculate $m(S, i)$ greedily
- for equations "$S = S_1 + S_2$", we have $m(S, i) = m(S_2, m(S_1, i))$
- calculate bottom-up or use memoization

Running time: $O(k|P|)$

Simplification: C never decreases, therefore it is enough to memoize the last query for each S.

Problem D
Non-boring sequences

Submits: 139
Accepted: 20

First solved by:
University of Zagreb
(Ivan Katanic, Stjepan Glavina, Goran Žužić)
1:18:26

Author: Adam Polak

The idea is pretty simple:

The idea is pretty simple:

- Find any unique element in the whole sequence . . .

The idea is pretty simple:

- Find any unique element in the whole sequence ...
- ...and recurse on both halves.

$$\underbrace{1\,2\,1}_{\text{recurse}}\ \color{red}{4}\ \underbrace{1\,2\,1\,3\,1\,2}_{\text{recurse}}$$

The idea is pretty simple:

- Find any unique element in the whole sequence ...
- ...and recurse on both halves.

$$\underbrace{1\,2\,1}_{\text{recurse}}\ \color{red}{4}\ \underbrace{1\,2\,1\,3\,1\,2}_{\text{recurse}}$$

How to find a unique element effectively?

How to find a unique element effectively?

How to find a unique element effectively?

- For every element x, compute the positions of the closest ones (in both directions) identical to x.

How to find a unique element effectively?

- For every element x, compute the positions of the closest ones (in both directions) identical to x.
- Now you can find whether a given element is unique in a given interval in $O(1)$ time.

How to find a unique element effectively?

- For every element x, compute the positions of the closest ones (in both directions) identical to x.
- Now you can find whether a given element is unique in a given interval in $O(1)$ time.
- In every recursive step, simply iterate over all elements . . .

How to find a unique element effectively?

- For every element x, compute the positions of the closest ones (in both directions) identical to x.
- Now you can find whether a given element is unique in a given interval in $O(1)$ time.
- In every recursive step, simply iterate over all elements . . .
- . . . in parallel, starting from both sides.

How to find a unique element effectively?

- For every element x, compute the positions of the closest ones (in both directions) identical to x.
- Now you can find whether a given element is unique in a given interval in $O(1)$ time.
- In every recursive step, simply iterate over all elements ...
- ... in parallel, starting from both sides.

What is the running time?

How to find a unique element effectively?

- For every element x, compute the positions of the closest ones (in both directions) identical to x.
- Now you can find whether a given element is unique in a given interval in $O(1)$ time.
- In every recursive step, simply iterate over all elements . . .
- . . . in parallel, starting from both sides.

What is the running time?

$$T(n) = \max_{0<k<n} T(k) + T(n-k) + \min(k, n-k)$$

How to find a unique element effectively?

- For every element x, compute the positions of the closest ones (in both directions) identical to x.
- Now you can find whether a given element is unique in a given interval in $O(1)$ time.
- In every recursive step, simply iterate over all elements ...
- ...in parallel, starting from both sides.

What is the running time?

$$T(n) = \max_{0<k<n} T(k) + T(n-k) + \min(k, n-k)$$

$$T(n) = O(n \lg n)$$

Problem I
The Dragon and the knights

Submits: 50
Accepted: 14

First solved by:
Jagiellonian University in Kraków
(Jakub Adamek, Grzegorz Guśpiel, Jonasz Pamuła)
1:10:45

Author: Bartosz Walczak

An $O((n^2 + m) \log n)$ solution: plane partition and point location

An $O((n^2 + m) \log n)$ solution: plane partition and point location

Another $O((n^2 + m) \log n)$ solution: plane sweep

An $O((n^2 + m)\log n)$ solution: plane partition and point location

Another $O((n^2 + m)\log n)$ solution: plane sweep

Easy $O(n \cdot m)$ solution:

1. Count the number of all districts.
2. Count the number of occupied districts.
3. Answer `PROTECTED` if the two numbers are equal.

An $O((n^2 + m) \log n)$ solution: plane partition and point location

Another $O((n^2 + m) \log n)$ solution: plane sweep

Easy $O(n \cdot m)$ solution:

1. Count the number of all districts.
2. Count the number of occupied districts.
3. Answer `PROTECTED` if the two numbers are equal.

Counting all districts:

- $n =$ the number of rivers
- $p =$ the number of pairs of non-parallel rivers
- the number of districts $= p + n + 1$ (induction, Euler's formula, etc.)

An $O((n^2 + m) \log n)$ solution: plane partition and point location

Another $O((n^2 + m) \log n)$ solution: plane sweep

Easy $O(n \cdot m)$ solution:

1. Count the number of all districts.
2. Count the number of occupied districts.
3. Answer `PROTECTED` if the two numbers are equal.

Counting all districts:

- $n =$ the number of rivers
- $p =$ the number of pairs of non-parallel rivers
- the number of districts $= p + n + 1$ (induction, Euler's formula, etc.)

Counting protected districts:

- start with a single class containing all the knights
- add rivers one by one; one river may split each partition class into two
- count the final number of partition classes

Problem K
Graphic Madness

Submits: 28
Accepted: 11

First solved by:
University of Wrocław
(Bartłomiej Dudek, Maciej Dulęba, Mateusz Gołębiewski)
2:34:41

Author: Jakub Pachocki

How to find a Hamiltonian cycle in a graph composed of two trees T_1, T_2 joined by leaves?

How to find a Hamiltonian cycle in a graph composed of two trees T_1, T_2 joined by leaves?

Note that the intersection of any such cycle and one of the trees is a matching between leaves with disjoint paths.

How to find a Hamiltonian cycle in a graph composed of two trees T_1, T_2 joined by leaves?

Note that the intersection of any such cycle and one of the trees is a matching between leaves with disjoint paths.

Root T_i. For every vertex v other than the root, discard the edge leading up from v if the subtree rooted in v contains an even number of leaves.

How to find a Hamiltonian cycle in a graph composed of two trees T_1, T_2 joined by leaves?

Note that the intersection of any such cycle and one of the trees is a matching between leaves with disjoint paths.

Root T_i. For every vertex v other than the root, discard the edge leading up from v if the subtree rooted in v contains an even number of leaves.

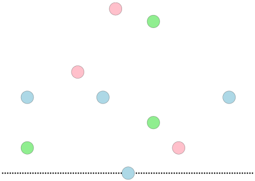
Check if the remaining edges form a Hamiltonian cycle.

Problem G
Jewel heist

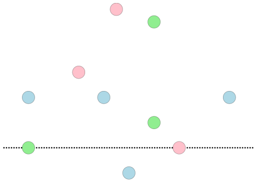
Submits: 38
Accepted: 10

First solved by:
Jagiellonian University in Kraków
(Piotr Bejda, Michał Sapalski, Igor Adamski)
1:33:57

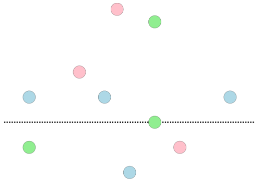
Author: Piotr Micek & Lech Duraj



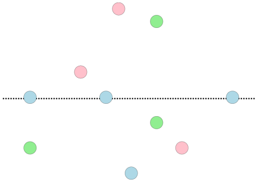
Imagine that we "sweep" the plane bottom-up with a horizontal line.



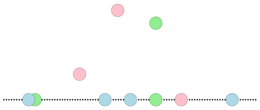
Imagine that we “sweep” the plane bottom-up with a horizontal line.



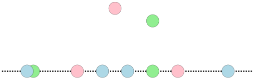
Imagine that we “sweep” the plane bottom-up with a horizontal line.



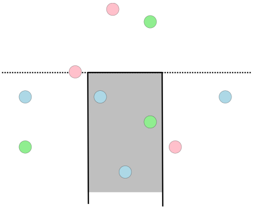
Imagine that we "sweep" the plane bottom-up with a horizontal line.
Every point that has been swept stays on the line.



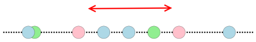
Imagine that we “sweep” the plane bottom-up with a horizontal line.
Every point that has been swept stays on the line.



Imagine that we “sweep” the plane bottom-up with a horizontal line.
Every point that has been swept stays on the line.



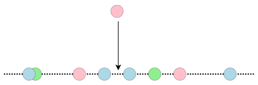
Imagine that we "sweep" the plane bottom-up with a horizontal line.
Every point that has been swept stays on the line.
A bottomless rectangle now corresponds to a *gap* between consecutive points of the same colour.



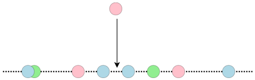
Imagine that we "sweep" the plane bottom-up with a horizontal line.
Every point that has been swept stays on the line.
A bottomless rectangle now corresponds to a *gap* between consecutive points of the same colour.

Regionals 2012
acm International Collegiate Programming Contest
IBM event sponsor

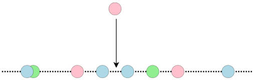
THEORETICAL COMPUTER SCIENCE Jagiellonian University



We detect a gap at the moment it is destroyed.



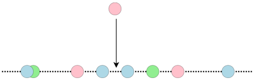
We detect a gap at the moment it is destroyed.
When a new point is inserted, we look at its same-colour neighbours and count the points between them.



We detect a gap at the moment it is destroyed.

When a new point is inserted, we look at its same-colour neighbours and count the points between them.

It is enough to know how many points are to the left of a given one.



We detect a gap at the moment it is destroyed.

When a new point is inserted, we look at its same-colour neighbours and count the points between them.

It is enough to know how many points are to the left of a given one.

Fenwick tree, $O(n \log n)$ total time.

We detect a gap at the moment it is destroyed.

When a new point is inserted, we look at its same-colour neighbours and count the points between them.

It is enough to know how many points are to the left of a given one.

Fenwick tree, $O(n \log n)$ total time.

Some gaps survived the whole sweeping. We count them with one additional loop at the end.

Problem B
Who wants to live forever?

Submits: 58

Accepted: 3

First solved by:

University of Warsaw

(Jakub Oćwieja, Mirosław Michalski, Jarosław Błasiok)

2:12:22

Author: Arkadiusz Pawlik

Assume our sequence is $x_1 x_2 x_3 \ldots x_n$.

Assume our sequence is $x_1 x_2 x_3 \ldots x_n$.

- If it is all zeroes, the universe definitely dies.

Assume our sequence is $x_1 x_2 x_3 \ldots x_n$.

- If it is all zeroes, the universe definitely dies.
- Otherwise, if n is even, the universe lives forever.

Assume our sequence is $x_1 x_2 x_3 \ldots x_n$.

- If it is all zeroes, the universe definitely dies.
- Otherwise, if n is even, the universe lives forever.
- Otherwise, let x' be x after one discrete step.

Assume our sequence is $x_1 x_2 x_3 \ldots x_n$.

- If it is all zeroes, the universe definitely dies.
- Otherwise, if n is even, the universe lives forever.
- Otherwise, let x' be x after one discrete step.
- Then the universe dies if and only if both $x_2 x_4 x_6 \ldots x_{n-1}$ and $x'_2 x'_4 x'_6 \ldots x'_{n-1}$ die.

This is because the even steps of the evolution of $x_2, x_4, \ldots, x_{n-1}$ are independent of the rest of the sequence.

Assume our sequence is $x_1 x_2 x_3 \ldots x_n$.

- If it is all zeroes, the universe definitely dies.
- Otherwise, if n is even, the universe lives forever.
- Otherwise, let x' be x after one discrete step.
- Then the universe dies if and only if both $x_2 x_4 x_6 \ldots x_{n-1}$ and $x'_2 x'_4 x'_6 \ldots x'_{n-1}$ die.

This is because the even steps of the evolution of $x_2, x_4, \ldots, x_{n-1}$ are independent of the rest of the sequence. This leads to an $O(n \log n)$ solution.

We can also characterize the 'finite' universes directly.

We can also characterize the 'finite' universes directly.

- Let k be maximum such that $2^k | n + 1$.

We can also characterize the 'finite' universes directly.

- Let k be maximum such that $2^k | n+1$.
- Then, the universe dies if and only if it is of the form:

$$w0\hat{w}0w\ldots0\hat{w}0w$$

where w is some binary string of length $2^k - 1$ and $\hat{w}$ is its reverse.

Problem F
Farm and factory

Submits: 0
Accepted: 0

First solved by:
nobody :(

Author: Jakub Pachocki

Let G be the original graph and G' be the graph with the capital c added. Let $d(u,v)$ be the distance between u and v in G and $d'(u,v)$ be the distance between u and v in G'.

Let G be the original graph and G' be the graph with the capital c added. Let $d(u, v)$ be the distance between u and v in G and $d'(u, v)$ be the distance between u and v in G'.

Note that $d(1, u) = d'(1, u)$ and $d(2, u) = d'(2, u)$ for all u, v in G. Let us denote $x_u = d(1, u)$ and $y_u = d(2, u)$.

Let G be the original graph and G' be the graph with the capital c added. Let $d(u, v)$ be the distance between u and v in G and $d'(u, v)$ be the distance between u and v in G'.

Note that $d(1, u) = d'(1, u)$ and $d(2, u) = d'(2, u)$ for all u, v in G. Let us denote $x_u = d(1, u)$ and $y_u = d(2, u)$.

Then it must hold for all u, v that $d'(u, v) \geq \max(|x_u - x_v|, |y_u - y_v|)$.

Let G be the original graph and G' be the graph with the capital c added. Let $d(u, v)$ be the distance between u and v in G and $d'(u, v)$ be the distance between u and v in G'.

Note that $d(1, u) = d'(1, u)$ and $d(2, u) = d'(2, u)$ for all u, v in G. Let us denote $x_u = d(1, u)$ and $y_u = d(2, u)$.

Then it must hold for all u, v that $d'(u, v) \geq \max(|x_u - x_v|, |y_u - y_v|)$.

If we fix some nonnegative x_c and y_c where $x_c + y_c \geq d(1, 2)$, then for all u we can add the edge (c, u) with cost $\max(|x_c - x_u|, |y_c - y_u|)$.

Let G be the original graph and G' be the graph with the capital c added. Let $d(u, v)$ be the distance between u and v in G and $d'(u, v)$ be the distance between u and v in G'.

Note that $d(1, u) = d'(1, u)$ and $d(2, u) = d'(2, u)$ for all u, v in G. Let us denote $x_u = d(1, u)$ and $y_u = d(2, u)$.

Then it must hold for all u, v that $d'(u, v) \geq \max(|x_u - x_v|, |y_u - y_v|)$.

If we fix some nonnegative x_c and y_c where $x_c + y_c \geq d(1, 2)$, then for all u we can add the edge (c, u) with cost $\max(|x_c - x_u|, |y_c - y_u|)$.

The cost will therefore be equal to:

$$\sum_u \max(|x_c - x_u|, |y_c - y_u|)$$

How to select the best x_c, y_c?

How to select the best x_c, y_c?

- First, draw all points (x_u, y_u) on the plane. We want to find a 'median' of the points in the maximum metric: $\max(|x_1 - x_2|, |y_1 - y_2|)$.

How to select the best x_c, y_c?

- First, draw all points (x_u, y_u) on the plane. We want to find a 'median' of the points in the maximum metric: $\max(|x_1 - x_2|, |y_1 - y_2|)$.
- Rotate the plane by 45 degrees.

How to select the best x_c, y_c?

- First, draw all points (x_u, y_u) on the plane. We want to find a 'median' of the points in the maximum metric: $\max(|x_1 - x_2|, |y_1 - y_2|)$.
- Rotate the plane by 45 degrees.
- Now we want to find a 'median' in the Manhattan distance metric: $|x_1 - x_2| + |y_1 - y_2|$.

How to select the best x_c, y_c?

- First, draw all points (x_u, y_u) on the plane. We want to find a 'median' of the points in the maximum metric: $\max(|x_1 - x_2|, |y_1 - y_2|)$.
- Rotate the plane by 45 degrees.
- Now we want to find a 'median' in the Manhattan distance metric: $|x_1 - x_2| + |y_1 - y_2|$.
- It is easy: just find the median of the new x and y coordinates!