Day 1 Editorial

April 26, 2016

ETH Zurich ACM ICPC Training Camp. April 2016

A. Bandits

Problem statement

- There were m bandits
- They wanted to divide n diamonds between each other
- Each bandit can make a proposal:
  - Proposal is an array $a_1, a_2 \dots a_m$, a_i — how many diamonds does the i-th bandit gets
    - $\sum_{i=1}^{m} a_i = n$
  - Each bandit votes **for** or **against** the proposal
  - Bandit i votes **for** the proposal if otherwise he survives and gets at least a_i diamonds
  - If the number of votes **for** didn't exceed $\frac{m}{2}$, then the bandit which proposed is killed
    - Then next bandit makes the proposal and so on
- Find the maximum number of diamonds the bandit that proposes first can get

A. Bandits

Solution

- Let's start from the end
  - If only one bandit left, he gets all the diamonds

A. Bandits

Solution

- Let's start from the end
  - If only one bandit left, he gets all the diamonds
  - Suppose $a_1, a_2, \ldots, a_k$ is the number of diamonds each of the k alive bandits get (and -1, if bandit dies)

A. Bandits

Solution

- Let's start from the end
  - If only one bandit left, he gets all the diamonds
  - Suppose $a_1, a_2, \ldots, a_k$ is the number of diamonds each of the k alive bandits get (and -1, if bandit dies)
  - When $k+1$ bandits left, the bandit that makes the proposal has to attract at least $\lfloor \frac{k}{2} \rfloor$ bandits on his side

A. Bandits

Solution

- Let's start from the end
  - If only one bandit left, he gets all the diamonds
  - Suppose $a_1, a_2, \ldots, a_k$ is the number of diamonds each of the k alive bandits get (and -1, if bandit dies)
  - When $k + 1$ bandits left, the bandit that makes the proposal has to attract at least $\lfloor \frac{k}{2} \rfloor$ bandits on his side
  - So he must propose them more than a_i

A. Bandits

Solution

- Let's start from the end
  - If only one bandit left, he gets all the diamonds
  - Suppose $a_1, a_2, \ldots, a_k$ is the number of diamonds each of the k alive bandits get (and -1, if bandit dies)
  - When $k + 1$ bandits left, the bandit that makes the proposal has to attract at least $\lfloor \frac{k}{2} \rfloor$ bandits on his side
  - So he must propose them more than a_i
- Which $\lfloor \frac{k}{2} \rfloor$ bandits to choose?

A. Bandits

Solution

- Let's start from the end
  - If only one bandit left, he gets all the diamonds
  - Suppose $a_1, a_2, \ldots, a_k$ is the number of diamonds each of the k alive bandits get (and -1, if bandit dies)
  - When $k+1$ bandits left, the bandit that makes the proposal has to attract at least $\lfloor \frac{k}{2} \rfloor$ bandits on his side
  - So he must propose them more than a_i
- Which $\lfloor \frac{k}{2} \rfloor$ bandits to choose?
  - With minimal a_i
    - Give them $a_i + 1$ diamonds, give nothing to others

A. Bandits

Solution

- Let's start from the end
  - If only one bandit left, he gets all the diamonds
  - Suppose $a_1, a_2, \ldots, a_k$ is the number of diamonds each of the k alive bandits get (and -1, if bandit dies)
  - When $k+1$ bandits left, the bandit that makes the proposal has to attract at least $\lfloor \frac{k}{2} \rfloor$ bandits on his side
  - So he must propose them more than a_i
- Which $\lfloor \frac{k}{2} \rfloor$ bandits to choose?
  - With minimal a_i
    - Give them $a_i + 1$ diamonds, give nothing to others
- If you don't have enough diamonds, then you are dead

A. Bandits

Solution

- Let's start from the end
  - If only one bandit left, he gets all the diamonds
  - Suppose $a_1, a_2, \ldots, a_k$ is the number of diamonds each of the k alive bandits get (and -1, if bandit dies)
  - When $k + 1$ bandits left, the bandit that makes the proposal has to attract at least $\lfloor \frac{k}{2} \rfloor$ bandits on his side
  - So he must propose them more than a_i
- Which $\lfloor \frac{k}{2} \rfloor$ bandits to choose?
  - With minimal a_i
    - Give them $a_i + 1$ diamonds, give nothing to others
- If you don't have enough diamonds, then you are dead
- Otherwise you get all the rest

A. Bandits

Example

- 5 bandits, 1000 diamonds

A. Bandits

Example

- 5 bandits, 1000 diamonds
  - 1 bandit: he gets 1000

A. Bandits

Example

- 5 bandits, 1000 diamonds
  - 1 bandit: he gets 1000
  - 2 bandits: must give 1001 to first bandit, cannot do so, bandits get (1000, -1)

A. Bandits

Example

- 5 bandits, 1000 diamonds
  - 1 bandit: he gets 1000
  - 2 bandits: must give 1001 to first bandit, cannot do so, bandits get (1000, -1)
  - 3 bandits: must give 0 to second bandit can give 0 to first bandit, bandits get (0, 0, 1000)

A. Bandits

Example

- 5 bandits, 1000 diamonds
  - 1 bandit: he gets 1000
  - 2 bandits: must give 1001 to first bandit, cannot do so, bandits get (1000, -1)
  - 3 bandits: must give 0 to second bandit can give 0 to first bandit, bandits get (0, 0, 1000)
  - 4 bandits: (1, 1, 0, 999)

A. Bandits

Example

- 5 bandits, 1000 diamonds
  - 1 bandit: he gets 1000
  - 2 bandits: must give 1001 to first bandit, cannot do so, bandits get (1000, -1)
  - 3 bandits: must give 0 to second bandit can give 0 to first bandit, bandits get (0, 0, 1000)
  - 4 bandits: (1, 1, 0, 999)
  - 5 bandits: (0, 2, 1, 997) or (2, 0, 1, 997)

B. Fitness Club

Problem statement

- n training sessions are to be in fitness club
- $a_i + b_i$ men visit i-th of them
- a_i of these close lockers they use, and b_i don't
- Find the minimum number of opened lockers after the last training session

Solution

- Minimize the number of open lockers after each session
- Greedily give open lockers to the visitors first, if it's not enough give some of the closed, too
- Let x be the number of open lockers before the session
- After the session the number of open lockers is $\max(x - a_i, b_i)$

C. Graduated Lexicographical Ordering

Problem statement

- Consider integer number from 1 to n
- *grlex ordering* is: a is before b if $(w(a) < w(b))$ or $(w(a) = w(b)$ and $a_{10} < b_{10})$
  - where $w(x)$ is the sum of digits in decimal representation of x
  - and x_{10} is the decimal representation itself
- Find the position of k in this ordering

C. Graduated Lexicographical Ordering

Solution

- You have to find the number of numbers that are before k
- First find the number of those, that $w(x) < w(k)$
- You have to calculate $f(s)$ — the number of x from 1 to n, such that sum of digits of x equals to s
- To calculate that find $c(L, S)$ — number of x consisting of L digits and $w(x) = S$
    - Get all numbers, which has length less than $|n_{10}|$
    - Then for each p find the number of x having longest common prefix with n equal to p and of the same length as n

C. Graduated Lexicographical Ordering

Find those having the same $w(x)$

- For every p let's consider those p, that $\mathrm{LCP}(x_{10}, k_{10}) = p$
  - Try every next digit that is less, than the next digit in k
  - Count the number of ways to append something to get number not greater than n
  - To do that iterate over p_2: $\mathrm{LCP}(n_{10}, x_{10})$
    - Check if p and p_2 are not contradictive
    - Count the way to append $n - \max(p, p_2) - 1$ digits

D. Network Wars

Problem Statement

- You are given a graph
- Find the cut with minimum mean cost

D. Network Wars

Solution

- Common approach for such problems: binary search for the answer
  - Need to check: given z, whether there exists cut with mean cost at most z

Given z, is there a cut with mean cost at most z?

- Subtract z from all costs
- Mean cost of every cut decreased by z
- Is there cut with mean cost at most 0?
- Let's find minimal cost cut
  - Get all negative edges
  - For every positive edge add the edge with such capacity in network
  - Find maximal flow

E. N-gons

Problem statement

- Find numbers $a_1, a_2, \ldots, a_m$ such that
  - $1 \leqslant a_i \leqslant k$
  - For any subset of size n, it was impossible to construct polygon with these side lengths
- Maximize m

E. N-gons

Solution

- Can make a polygon from $b_1 \leqslant b_2 \leqslant \ldots \leqslant b_k$ if $b_1 + b_2 + \ldots + b_{k-1} > b_k$
- Order a_i by increasing of their values
- It's enough to check $a_i, a_{i+1}, \ldots, a_{i+n-1}$
- Let $a_1 = a_2 = \ldots = a_{n-1} = 1$
- Let $a_i = a_{i-1} + a_{i-2} + \ldots + a_{i-n+1}$, for $i \geqslant n$
- Continue, while $a_i \leqslant k$

F. Numbers to Numbers

Problem statement

- Given text containing numbers in English
- Convert some of them to digits
- Leave the minimum number of words unconverted
- Maximize the first converted number, then the second and so on

F. Numbers to Numbers

Solution

- Dynamic programming approach: $f(i)$ — the least number of words left unconverted when converting text starting from word i
- You either don't convert i-th word, then $f(i) = f(i+1) + 1$
- Or you convert, then try all j, so that words from i to j form a number and choose minimal $f(j+1)$, so $f(i) = f(j+1)$
- Restoring the answer:
  - if $f(i) = f(i+1) + 1$, then don't convert i-th word
  - Otherwise choose such j, so that $f(i) = f(j+1)$ and the number formed is maximized
- There is not more than 18 words in a number

G. Beautiful Permutation

Problem statement

- Find permutation, such that maximal monotonic subsequence is minimized
- Find lexicographically smallest such permutation

G. Beautiful Permutation

Solution

- Maximal monotonic subsequence is at least $\lceil \sqrt{n} \rceil$
  - Consider `up[i]` — longest increasing subsequence ending at i-th element
  - Consider `down[i]` — longest increasing subsequence starting at i-th element
  - Pairs (`up[i]`, `down[i]`) are distinct
    - Suppose `up[i]` $=$ `up[j]` and `down[i]` $=$ `down[j]`, for $i < j$
    - If $a_i < a_j$ then `up[i]` $\leqslant$ `up[j]`
    - If $a_i > a_j$ then `down[i]` $\geqslant$ `down[j]`

Getting such permutation

- $n = u^2$
  - $u, u-1, \ldots, 1, 2u, 2u-1, \ldots, 2u-u+1, \ldots, u^2, \ldots (u-1)u+1$
- $n = u^2 - t$
  - Use pattern above and remove all element greater than n

H. Beautiful Numbers

Problem statement

- Given a phone number
- There are some patterns that give you bonus
- Find the number partition into substrings to get maximum bonus

Solution

- Iterate over all different partitions
- Find the bonus
- Choose the best

I. Flipping Bits

Problem statement

- You have a string of '0'-s and '1'-s of length n
- You also have an integer m
- In one move you can either flip one bit, or flip first $k \cdot m$ bits for some positive integer k
- What is the minimal number of moves needed to get a string, prefix and suffix of length $n - m$ of which are equal to each other

I. Flipping Bits

Solution

- If prefix of length $n - m$ equals to suffix of length $n - m$, then the string has period m ($s_i = s_{i+m}$)
- If $m \leqslant \sqrt{n}$:
  - Try all prefixes of length m, there are 2^m of them, let's call it p
  - For every $s[im..(i+1)m)$ you can calculate the number of moves needed to make it equal to p and to p^r
    - p^r is flipped p
  - Do the dynamic programming $f(i, j)$ — the number of flips to make, if you put all s_j for $j \geqslant i$ correctly and the parity of number of big flips made for $k > i$ is j ($j = 0$ or $j = 1$)
  - Transition is: You either get p or p^r and depending on j you have to do the flip or not

I. Flipping Bits

Solution

- If $m > \sqrt{n}$:
  - m is period, so the final string divides on equal blocks
    - The number of blocks is $\lceil \frac{n}{m} \rceil \leqslant \sqrt{n} + 1$
  - For every block decide, whether we do it equal to prefix, or equal to flipped prefix
    - There are $2^{\lceil \frac{n}{m} \rceil}$ ways to do that
    - Count the number of big flips to be made and make them
  - For every $0 \leqslant r < m$ count the number of s_{im+r} that need to change if $p_r = 0$ and if $p_r = 1$
  - For every r choose p_r, so that less changes is needed
  - Sum everything up
- Overall complexity is $O(2^{\sqrt{n}} n)$

J. Puzzle

Problem statement

- There are $k \leqslant 10$ maps
- Each map consist of $n \leqslant 50$ vertices and $m \leqslant 1500$ edges
- Start end finish vertices are selected on each map
- You must move a token on each map along some edge during a move
- Your goal is to move all tokens to finish vertices in minimum number of moves

J. Puzzle

Solution

- If there is a path of length X, there is a path of length $X + 2$ (because you can move back and forth alone an edge)
- So let's find shortest even and odd pathes on each map
- If there is no even path on some map you can't win with even number of moves
- Same for odd number of moves
- Answer is smaller of maximums for even and odd number of moves

K. Restore the tree

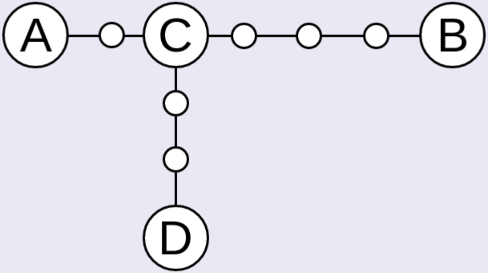
Problem statement

- There was a tree with n leafs
- $n \leqslant 200$
- You are given pairwise distances for all leafs
- You need to restore a tree with such distances or say that there is no such tree

K. Restore the tree

Solution

Let's look at some leafs A, B and D. We can find vertex C which is the only vertex belongs to all three pathes AB, AD and BD.



To do this use equation: $|AC| = \frac{|AB|+|AD|-|BD|}{2}$

K. Restore the tree

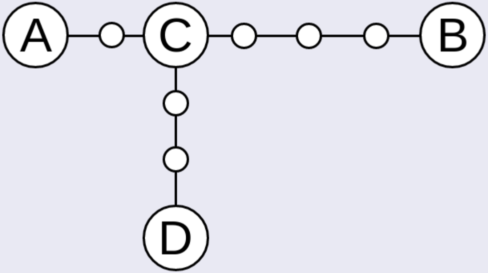
Solution

We can find a diametr of a tree. It is two leafs with biggest distance between them. Let's call them A and B.
Than iterate over all other leafs and hang them on a proper vertex of diametr.
Let's call a leaf we a currently looking at as D. And vertex on a path C. Now we need to consider some cases.

K. Restore the tree

Solution

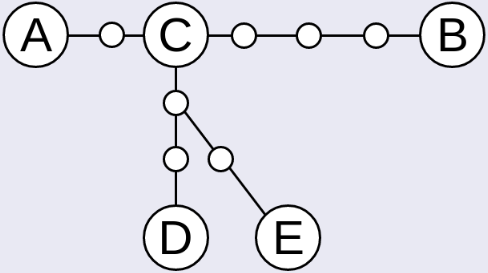
If there are no other vertices is connected to C just create a path of new vertices to hang vertex D.



K. Restore the tree

Solution

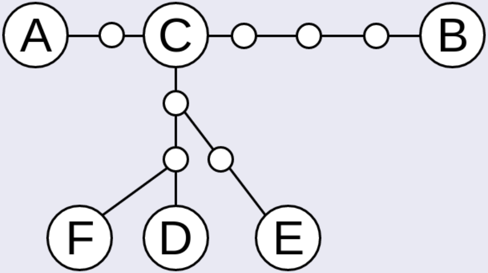
Otherwise iterate over all leafs in a subtree of vertex C. Choose a leaf which has longest common path from diametr. Create a new path starting from LCA.



K. Restore the tree

Solution

Otherwise iterate over all leafs in a subtree of vertex C. Choose a leaf which has longest common path from diametr. Create a new path starting from LCA.



K. Restore the tree

Solution

Don't forget to check a tree in the end:

- All vertices from input are leafs
- All distances from input are correct

L. String

Problem statement

- You are given a recurent equation for strings:
  - s_0 is an empty string
  - $s_i = s_{i-1}$ if s_{i-1} contains decimal representation of i
  - $s_i = s_{i-1} + i$ otherwise
- Find s_n, $n \leqslant 500$

L. String

Solution

- $n \leqslant 500$
- So we can generate all s_i in a naive way
- It takes $O(length)$ to check if string contains a number
- Total complexity is $O(n^2)$

M. Shooting game

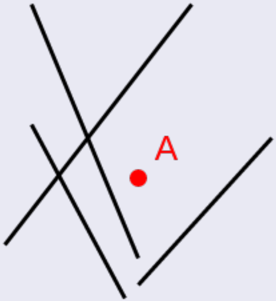
Problem statement

- There are $n \leqslant 8$ segments on a plane
- Let's call $dist(AB)$ for points A and B a number of segments from input which intersects segment $[AB]$
- Find $\min_A \max_B dist(AB)$ for all possible points A and B

M. Shooting game

Solution

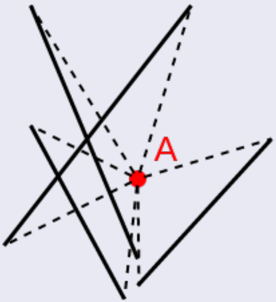
- Suppose we fixed point A
- How to find $\max_B dist(AB)$?



M. Shooting game

Solution

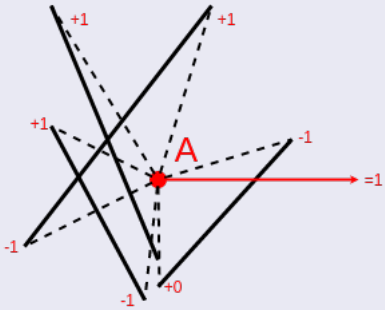
We can usa a sweeping line algorithm to find best position for B. Let's A has coordinates (0, 0). Count number of intersections if we place B in (+inf, 0).



M. Shooting game

Solution

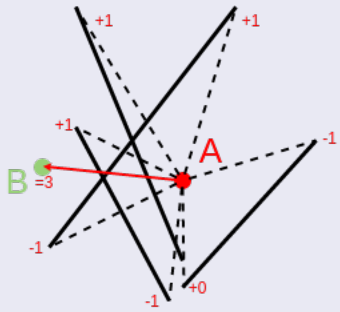
We can use a sweeping line algorithm to find best position for B. Let's A has coordinates $(0, 0)$. Count number of intersections if we place B in $(+\infty, 0)$. Move B counterclockwise and maintain number of intersections.



M. Shooting game

Solution

We can use a sweeping line algorithm to find best position for B. Let's A has coordinates $(0, 0)$. Count number of intersections if we place B in $(+\infty, 0)$. Move B counterclockwise and maintain number of intersections. Choose best position.



M. Shooting game

Solution

- Will the answer change if we move A “a little”?

M. Shooting game

Solution

- Will the answer change if we move A “a little”?
- It will not change if angles in which we see segments ends from A don’t change their order.

M. Shooting game

Solution

- Will the answer change if we move A "a little"?
- It will not change if angles in which we see segments ends from A don't change their order.
- For each pair of points there is a half plane where first point goes before second, and half plane where second goes before first.

M. Shooting game

Solution

- Will the answer change if we move A “a little”?
- It will not change if angles in which we see segments ends from A don’t change their order.
- For each pair of points there is a half plane where first point goes before second, and half plane where second goes before first.
- So divide whole plane into parts and solve separately for each part.

M. Shooting game

Solution

- Will the answer change if we move A “a little”?
- It will not change if angles in which we see segments ends from A don't change their order.
- For each pair of points there is a half plane where first point goes before second, and half plane where second goes before first.
- So divide whole plane into parts and solve separately for each part.
- There are $O(n^2)$ lines which divide plane on $O(n^4)$ parts. For each part we can solve problem in $O(n \log n)$ time, so total complexity is $O(n^5 \log n)$.