NWERC 2019 presentation of solutions

NWERC 2019 Jury

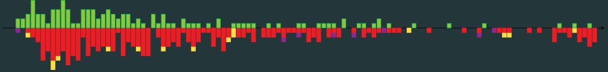
- **Per Austrin**
  KTH Royal Institute of Technology
- **Jeroen Bransen**
  Chordify
- **Alexander Dietsch**
  FAU Erlangen-Nürnberg
- **Ragnar Groot Koerkamp**
  Google
- **Bjarki Ágúst Guðmundsson**
  Reykjavík University
- **Nils Gustafsson**
  KTH Royal Institute of Technology
- **Irina Kostitsyna**
  Eindhoven University of Technology
- **Robin Lee**
  Google
- **Lukáš Poláček**
  Innovatrics
- **Paul Wild**
  FAU Erlangen-Nürnberg

Big thanks to our test solvers

- **Alexander Raß**
  FAU Erlangen-Nürnberg
- **Tobias Werth**
  Google

I: Inverted Deck

Problem Author: Robin Lee



Problem

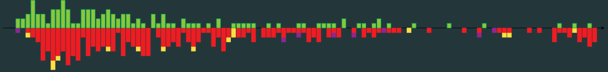
Sort a list of integers by inverting a sublist.

Faster solution

- Find the first element `list[i]` whose successor is smaller.
- Find the last element `list[j]` whose predecessor is larger.
- Find smallest `i'` $\leq$ `i` s.t. `list[i']==list[i]`, and largest `j'` $\geq$ `j` s.t. `list[j']==list[j]`.
- Invert the sublist between the indices `i` and `j'`.
- Check if the list is now sorted!
- $O(n)$ running time.

I: Inverted Deck

Problem Author: Robin Lee

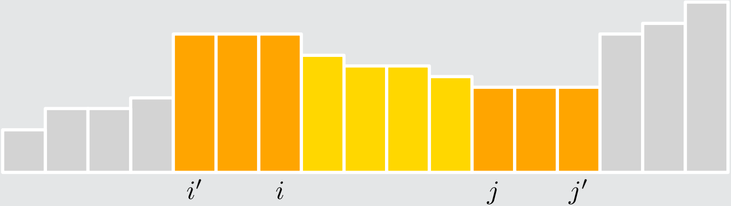


Problem

Sort a list of integers by inverting a sublist.

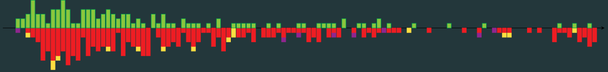
Pitfalls

- Index out of range when the input is already sorted.
- Forget to check if the list is sorted after inverting the sublist.
- Non-strict growth of values.



I: Inverted Deck

Problem Author: Robin Lee



Problem

Sort a list of integers by inverting a sublist.

Easier solution

- Sort the list.
- Find the first and the last indices where the elements in the sorted list and the original list differ.
- Invert the sublist between the two indices in the original list.
- Check if the list is now sorted!
- $O(n \log n)$ running time.

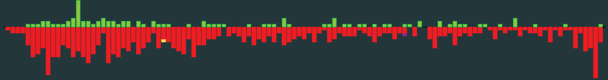
Pitfalls

- Index out of range when the input is already sorted.
- Forget to check if the list is sorted after inverting the sublist.

Statistics: 374 submissions, 120 accepted

E: Expeditious Cubing

Problem Author: Paul Wild



Problem

Out of 5 numbers, the largest and smallest are removed and the remaining 3 averaged. Given 4 of the numbers, you have to choose the 5th one such that the average is at most t.

How large can this number be? If it is impossible, or any number at all is OK, output this information instead.

Direct solution

- Let $\min$, $\max$ and sum be the min, max and sum of the 4 initial numbers, respectively.
- Let x be the 5th number. Then:
  - If $x > \max$, the average is $(\mathrm{sum} - \min)/3$
  - If $x < \min$, the average is $(\mathrm{sum} - \max)/3$
  - If $x \in [\min, \max]$, the average is $(\mathrm{sum} - \min - \max + x)/3$
- This is maximized when $x > \max$: Output "`infinite`" if $(\mathrm{sum} - \min)/3 \leq t$
- This is minimized when $x < \min$: Output "`impossible`" if $(\mathrm{sum} - \max)/3 > t$
- Otherwise solve $(\mathrm{sum} - \min - \max + x)/3 \leq t$ for x, giving $3t - \mathrm{sum} + \min + \max$ as the answer.

E: Expeditious Cubing

Problem Author: Paul Wild

Gotchas

- Output to exactly two decimal places, even when there are trailing zeros.
- Use epsilons when comparing floating point numbers.
- When using integers, also print two digits. 1708 should *not* be printed as 17.8.

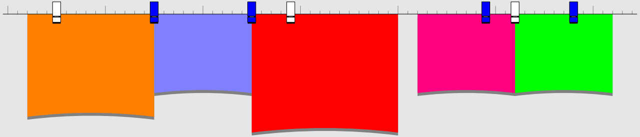
Alternative solutions

- Binary search x.
- Simply try all possible x.

Statistics: 634 submissions, 98 accepted

Problem

Hold each of a set of (non-overlapping) canvases with exactly two pegs. Use as few new pegs as possible, given that some have already been added.

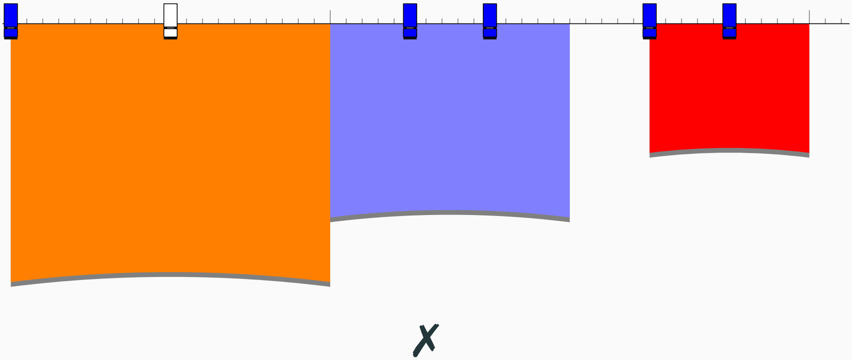


One peg can hold:

- either no canvas at all;
- or one canvas from somewhere in the middle;
- or two canvases, if they share a corner.

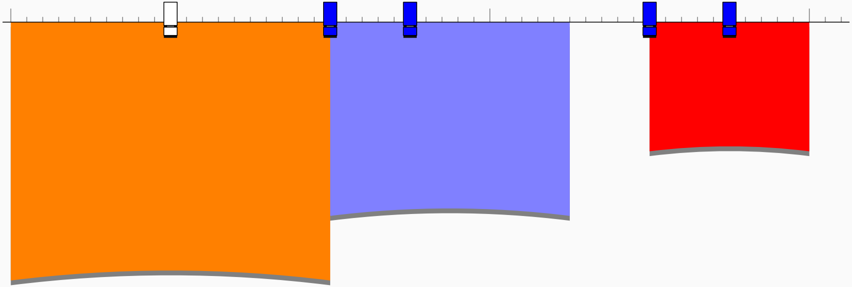
Greedy algorithm

- One option is to try and insert pegs left-to-right so long as they are allowed.
- This will not always give the best answer:



Greedy algorithm

- One option is to try and insert pegs left-to-right so long as they are allowed.
- This will not always give the best answer:



✓

C: Canvas Line

Problem Author: Irina Kostitsyna

Greedy algorithm, improved

- Adding pegs at shared corners is always better than adding them inside a canvas.
- Thus, give a higher priority to those ones by inserting them first.
- First, for each shared corner, see if there is still space to add a peg.
- Second, fill in the gaps in peg count with extra pegs in the middle.
- Finally, check that every canvas now has exactly two pegs.
- Keep track of number of pegs on a canvas at all times.

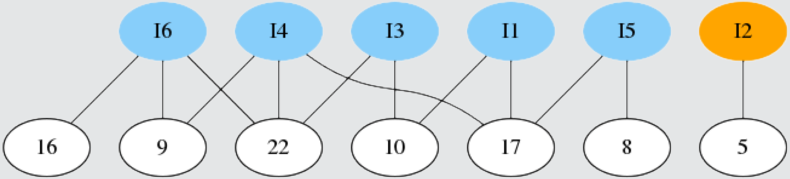
Statistics: 273 submissions, 94 accepted

F: Firetrucks Are Red

Problem Author: Bjarki Ágúst Guðmundsson

Problem

Each person is associated with some numbers. Two people being associated with the same number means they are connected.



Show via $n-1$ connections that show everyone is connected, or (as pictured) report back that not everyone is connected.

F: Firetrucks Are Red

Problem Author: Bjarki Ágúst Guðmundsson

Solution

- Too slow:
  - For every number, connect every pair of people associated with that number.
  - Find a spanning tree of the resulting graph.
  - Complexity: $O(n^2)$
- Instead:
  - For every number, choose a person as a representative and connect it to every other person associated with that number.
  - Find a spanning tree of the resulting graph.
  - Complexity: $O(n)$
- Alternatively:
  - For every number, create another vertex and connect it to every person associated with that number.
  - Find a spanning tree of the resulting graph.
  - Complexity: $O(n)$
- To find a spanning tree, you can for example use DFS, BFS, or union-find.

Statistics: 218 submissions, 103 accepted

G: Gnoll Hypothesis

Problem Author: Alexander Dietsch

Problem

We get numbers $p_1, \ldots, p_n$. Exactly k of the n numbers will be chosen. Non-chosen numbers are added to the next chosen one (wrapping modulo n if necessary) and then set to 0.

What are the average resulting numbers $q_1, \ldots, q_n$, over all possible choices of which k numbers to keep?

G: Gnoll Hypothesis

Problem Author: Alexander Dietsch

Solution

- Number $i-d$ gets added to number i if and only if number i is kept, and none of the numbers $i-1, i-2, \ldots, i-d$ are chosen (with $i-d$ wrapping modulo n).
- This happens for $\binom{n-d-1}{k-1}$ out of the $\binom{n}{k}$ ways of choosing the numbers.
- By the power of linearity of expectation, resulting averaged numbers are

$$q_i = \sum_{d=0}^{n-1} p_{i-d} \cdot \frac{\binom{n-d-1}{k-1}}{\binom{n}{k}} = \sum_{d=0}^{n-1} p_{i-d} \cdot r_d.$$

- Leads to immediate $O(n^2)$ time solution.
- Potential pitfall: binomial coefficients are very large and do not fit in 64-bit integers, use doubles.
- Can also be solved in $O(n \log n)$:
  - $(q_1, \ldots, q_n)$ is the (circular) *convolution* of the vectors $(p_1, \ldots, p_n)$ and $(r_1, \ldots, r_n)$.
  - The Fast Fourier Transform can be used to evaluate convolutions in $O(n \log n)$.

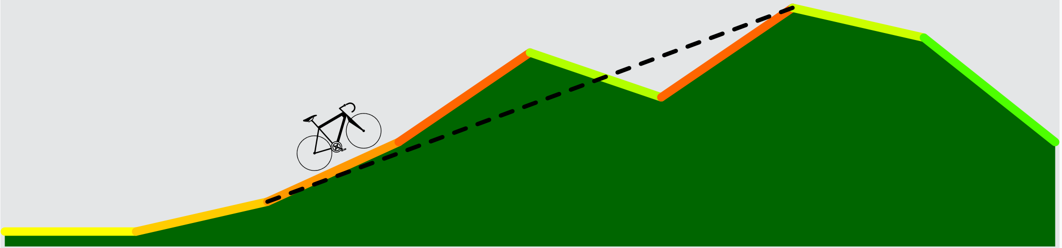
Statistics: 237 submissions, 76 accepted

H: Height Profile

Problem Author: Ragnar Groot Koerkamp

Problem

A bicyclist is travelling through a rustic two-dimensional landscape, represented as a series of y coordinates along an x axis. They are looking for a challenging section of road where the average incline is at least g.



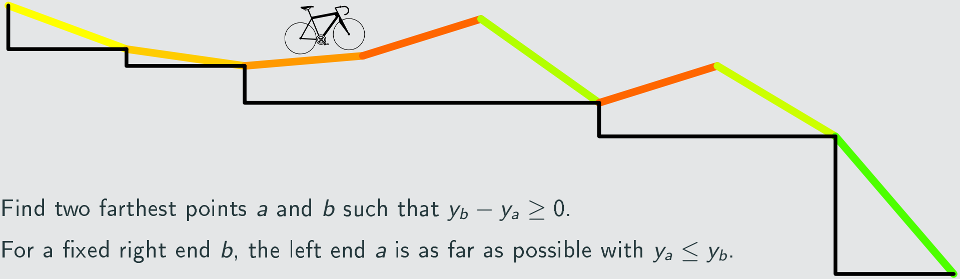
What is the longest such section of road?

H: Height Profile

Problem Author: Ragnar Groot Koerkamp

Solution

Shear the input by subtracting the incline equation $y = g \cdot x$ from the (x, y) coordinates given.



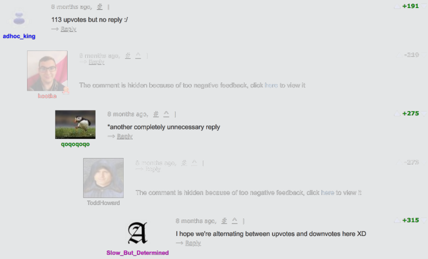
- Find two farthest points a and b such that $y_b - y_a \geq 0$.
- For a fixed right end b, the left end a is as far as possible with $y_a \leq y_b$.
- Iterate b over $0, 1, 2, \ldots$ horizontal kilometers and keep a "staircase" of decreasing values of y. Binary search the staircase to find the farthest y to the left that is smaller than y_b.
- Extend the interval at most 1 kilometer either to the right or left until $y_a = y_b$.

Statistics: 109 submissions, 14 accepted

Problem

Make a comment chain alternating positive and negative vote scores. You have two innovative methods available:

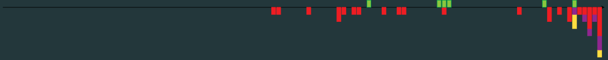
- Create a fake account and upvote/downvote comments.
- Report a comment so that it will be removed.



How much time do you need to spend on this?

J: Jackdaws and Crows

Problem Author: Nils Gustafsson



Insights

- With f fake accounts all scores s with $|s| < f$ become wildcards (can be voted to positive or negative score).
  $\Rightarrow$ Only $n + 1$ interesting numbers of fake accounts.
- It is possible to calculate whether one comments needs to be removed between any two adjacent non-wildcards in $\mathcal{O}(1)$. E.g. the chain + ? ? ? - needs one comment removed.

Solution

- Calculate in $\mathcal{O}(n)$ the number of removed comments in the original chain.
- For every interesting number of fake accounts change the corresponding scores to wildcards and check how that affects the adjacent non-wildcards. Update the number of removed comments in $\mathcal{O}(1)$.
- Special case 0 *fake accounts*: All comments with score 0 need to be removed.

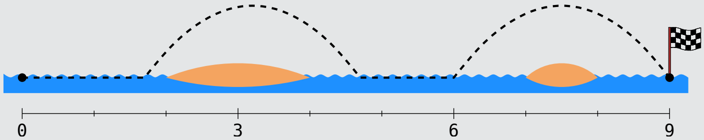
Statistics: 43 submissions, 6 accepted

K: Kitesurfing

Problem Author: Nils Gustafsson

Problem

Find the fastest way to jump over n islands along a straight line.



There are two ways to move:

- Surf some distance at a speed of 1.
- Jump up to D units ahead. This takes some fixed time, regardless of distance.

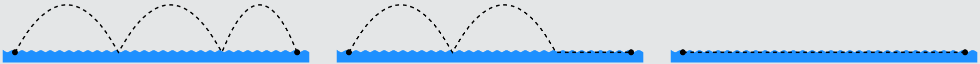
Insight 1

For any jump that follows a surf section, move the take-off position to the left until the landing position hits an island:

Now every surf section ends either exactly D units before the end of an island, or at the finish line.

Insight 2

When there are no obstacles between two points, there are three options:

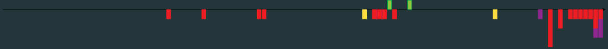


(a) only jump (b) jump, then surf (c) only surf

Any of these may be optimal, depending on jump length and jump time.

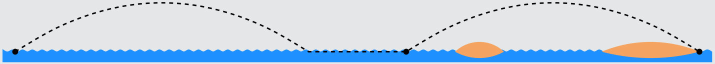
K: Kitesurfing

Problem Author: Nils Gustafsson

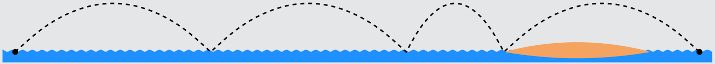


Solution

- Dynamic programming over positions, only visiting those we need to.
- From every position, try two options:
    - Advance to one of the take-off positions, then jump.



    - Keep jumping until you pass over an island. The next position can be found in constant time.

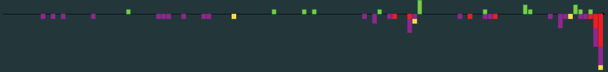


- The number of reached positions is $\mathcal{O}(n^2)$, for a total time of $\mathcal{O}(n^3)$.

Statistics: 30 submissions, 2 accepted

A: Average Rank

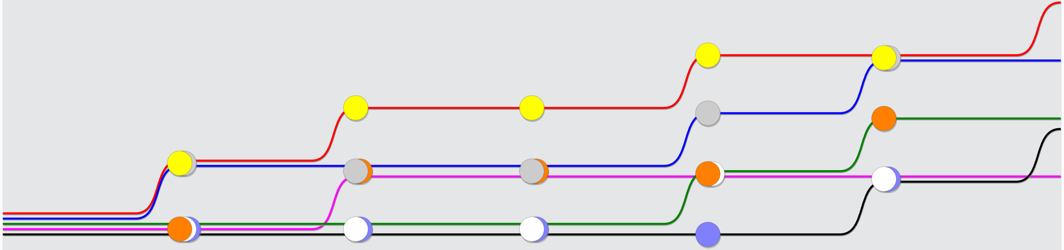
Problem Author: Lukáš Poláček



Problem

Competitors in a contest are ranked by their total score each of w weeks, sharing the same rank in case of a tie. For example, two teams with 4 points could share third place, the next team taking fifth place.

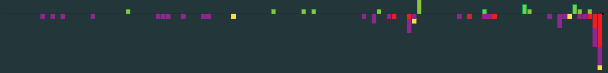
Every week a competitor can either keep the same score, or increase their score by 1 point.



Question: What is the average rank of each competitor, over all of the w weeks?

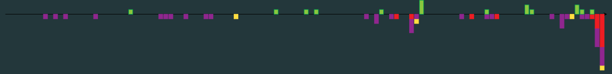
A: Average Rank

Problem Author: Lukáš Poláček



Solution

- Q: When does the rank of person x change?
  - A: When either x gains a point or somebody else with the same score gains a point.
  - There can be $\Theta(nw)$ changes of rank!
- Q: When does the rank of everybody with score s change?
  - A: When somebody with s points gets a point, this rank increases by 1.
  - There can only be $O(p)$ such changes in total over all s, with p the total number of points given!
- For each score, keep track of when its rank changes and to which value.
- When person x with s points gets another point, add their total rank while they had s points to their overall total.



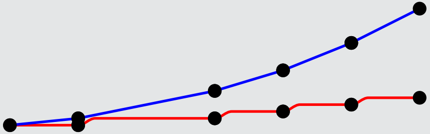
A: Average Rank

Problem Author: Lukáš Poláček

Solution

The prefix sums of the rank at score s can be computed efficiently:

- The rank is piecewise constant.
- The prefix sums are piecewise linear.
- Store the last slope and offset and update whenever the rank changes.
- When a person goes from score s to $s+1$:
  - Add the accumulated rank at score s.
  - Subtract the current accumulated rank at score $s+1$.



Statistics: 71 submissions, 16 accepted

D: Disposable Switches

Problem Author: Nils Gustafsson

Problem

You are given an undirected graph where the ith edge has weight $\ell_i/v + c$, where $v > 0$ and $c \geq 0$ are unknown constants.

Determine which vertices cannot be on a shortest path from vertex 1 to vertex n, no matter what the actual values of v and c are.

D: Disposable Switches

Problem Author: Nils Gustafsson

Solution

- Since $v > 0$ we can scale all edge weights by v without affecting the answer.
- The length of a path $P = \{e_1, \ldots, e_k\}$ from vertex 1 to vertex n is then

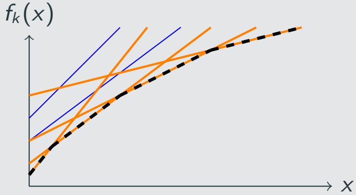
$$\sum_{i=1}^{k} v \cdot (\ell_i / v + c) = \sum_{i=1}^{k} (\ell_i + v \cdot c) = L(p) + k \cdot x$$

  where $L(P) = \sum_{e_i \in P} \ell_i$ and $x = v \cdot c$
- Among all paths with exactly k edges, let $\mathrm{best}_k := L(P^\star)$ on a path $P^\star$ that minimises this quantity.
  - Can be computed with dynamic programming for $k \in \{0, \ldots, n-1\}$ in $O(n(n+m))$ time.
  - Paths with n or more edges will have unnecessary cycles, and thus cannot be shortest paths.
  - Paths with k edges that have $L(P) > \mathrm{best}_k$ cannot be shortest paths either.
- The optimal number of edges to use depends on x.

Solution (continued)

- For each k, the weight of the shortest path containing exactly k edges can now be represented as the line

$$f_k(x) = k \cdot x + \mathrm{best}_k$$



- Lines $f_k(x)$ that occur on the lower hull correspond to number of edges k that are optimal for some value of x. Find these lines in $O(n^2)$ or $O(n)$ with some basic math.
- For each of these optimal k, use the DP table to mark all vertices that occur on a path with k edges of total weight best_k in $O(n(n+m))$.
- Output the unmarked vertices. Overall time complexity: $O(n(n+m))$.

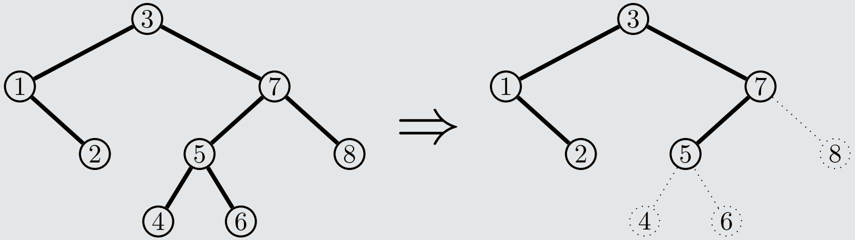
Statistics: 12 submissions, 2 accepted

B: Balanced Cut

Problem Author: Alexander Dietsch

Problem

Keep k nodes from an n-node balanced binary tree, such that the remaining tree is connected, balanced and includes the root.



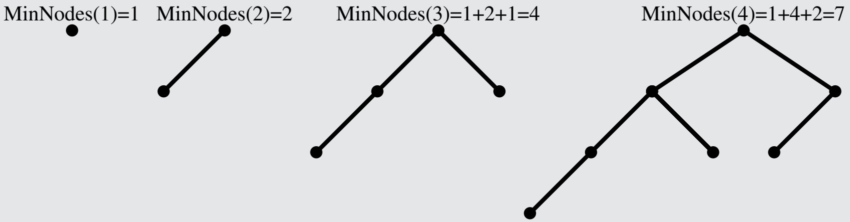
What is the lexicographically-largest tree we can keep, if we represent a tree as a string over 01 of which nodes are removed, and which nodes are kept?

B: Balanced Cut

Problem Author: Alexander Dietsch

Solution

Let's first compute MinNodes(h), the minimal number of nodes in a tree of height h.



$$\text{MinNodes}(h) = 1 + \text{MinNodes}(h-1) + \text{MinNodes}(h-2).$$

B: Balanced Cut

Problem Author: Alexander Dietsch

Solution

- Try to greedily add nodes in pre-order.
- We can only add u if the number of additional nodes a needed to preserve the balancing is small enough.
- For each ancestor p of u:
  - Compute the current height h_p and the new height h'_p after u is added.
  - If $p > u$: add $T(h'_p - 2) - T(h_p - 2)$ to a to reserve additional nodes for the right subtree of p.
  - If $p < u$: the left subtree of p was processed already.

  Add u if $1 + a \leq k$ $(+1$ for u itself).
- When descending to a right child, set its height to $h_u - 2$.
- If $h_u > 0$ when we enter u: add it for 'free' and propagate heights $h_u - 1$ and $h_u - 2$ to its children.

This is $O(n \log n)$ because a balanced tree has logarithmic depth.

B: Balanced Cut

Problem Author: Alexander Dietsch

Solution

- Other approaches:
  - Iterate over nodes from low to high, instead of pre-order.
  - Iterate over the nodes needed before adding u, instead of counting them.
- Linear time is possible!

Statistics: 47 submissions, 0 accepted

Language stats