Day 5 Editorial

May 1, 2016

ETH Zurich ACM ICPC Training Camp. April 2016

A. Complexity

Problem statement

- Given N and X
- You can add products of N, $\lceil \log N \rceil$, $\lceil \log \lceil \log N \rceil \rceil$ and so on
- Use the least number of $\mathtt{N}$'s to produce X

A. Complexity

Solution

- If taking logarithms: $10^5 \to 16 \to 4 \to 2$
- Generate all the products possible: there would be not much of them
- Compute dynamic programming: `f[n]` — the number of `N`'s to produce n
- Try every product to calculate `f[n]`

B. Divisibility Tree

Problem statement

- You are given a tree with $n \leqslant 1000$ vertices
- There are some numbers in leaves
- You need to put some positive integers in other vertices such that:
- each vertex has bigger value than its child and also divisible by child's value

B. Divisibility Tree

Solution

- If you want to choose a number for some vertex and you know numbers for children than best option is their GCD
- But it can be equal to some child so let's save that you need to divide it by some prime
- Now each vertex can be repsresented as a pair (value, k), where k is a number of times it should be divided by some prime
- If you have a vertex with children $(v_1, k_1), (v_2, k_2), \ldots, (v_m, k_m)$ than vertex should be $(v, max(0, k_1 + 1 - divs(\frac{v_1}{v}), k_2 + 1 - divs(\frac{v_2}{v}), \ldots))$, where $v = gcd(v_1, v_2, \ldots, v_m)$ and div(x) is number of times x can be divided by primes
- Than go from root recursively and simply divide numbers by any primes in a way which follows the rules

C. Progressing Fraction

Problem statement

- Given n, b and q
- Consider numbers $b \cdot q^i$, for $i \geqslant 0$
- c_k is the number of $b \cdot q^i$ starting with n in decimal notation for $i \leqslant k$
- Find $\lim\limits_{i \to +\infty} \frac{c_i}{i}$

C. Progressing Fraction

Solution

- $b \cdot q^i$ starts with n iff:
  - there exists k, such that $10^k \cdot n \leqslant b \cdot q^i \leqslant 10^k \cdot (n+1)$
- Let's take logarithm based 10:
  $k + \log n \leqslant \log b + i \cdot \log q \leqslant k + \log(n+1)$
- We can subtract integer from all the sides, and there is no integer x, so that $\log n < x < \log(n+1)$
  - We can leave the fractional part of all numbers
- $\text{frac}(\log n) \leqslant \text{frac}(\log b + i \cdot \log q) \leqslant \text{frac}(\log(n+1))$
- When increasing i, $\log q$ added to $\text{frac}(\log b + i \cdot \log q)$
- If $\log q$ is irrational, then all $\text{frac}(i \cdot \log q)$ are different, and the probability is $\log(n+1) - \log n$
- $\log q$ is rational iff $q = 10^k$

D. 4-Character Percentage

Problem Statement

- You are given a string s of length at most 10000
- Consider all 4-character words and number of times each contains as subsequence of s
- You need to output all frequent 4-character words (that form at least 1% of all such strings)

D. 4-Character Percentage

Solution

- Total number of occurences 4-character word as subsequnce is $\binom{|s|}{4}$
- We can find number of occurences of fixed word in a string in $O(|s|)$ using dynamic programming
- We can pick random 4 positions of string and check such string (try this T times)
- What is a probability that we forgot about some word? It is less than $(\frac{99}{100})^T$
- We have at most 100 words to output, so probability of finding all is at least $1 - 100(\frac{99}{100})^T$
- So if we check 5000 random words, probabily of getting WA is less than 10^{-19}. On real data it's ok to check 1000 strings.

E. Random strings

Problem statement

- There are two ways of generating strings
- Fisrt: each character if chosen uniformly and independently at random
- Second: firstly all characters has equal probability. After using some character its probability is divided by 2 and than all probabilities are normalized
- You are given some strings which were generated in one of two possible ways. You need to find which one was used for each of them.

E. Random strings

Solution

- There are a lot of different methods to solve this problem. One of them:
- Calculate a standart deviation of number of occurences of each character
- You can write a program which generated strings in a ways which were described and find that SD for strings from second methods is much smaller

F. Rotor Traversal

Problem Statement

- You are given a tree consisting of $n \leqslant 100$ vertices
- For each vertex let's fix an order of its neighbours
- Now let's define a traverse algorithm. When you first come to some vertex, you go to first neighbour in order. When you come second time, you go to second neighbour and so on. After last neighbour you go to the first one.
- You stop when you visit all vertices
- You need to find minimum and maximum possible time of walking if you can set neigbours orders

F. Rotor Traversal

Solution. Minimum case

- It is a best option to visit all children and than go back to parent
- If you end at vertex v than total length will be $2(n-1) - (\texttt{length from root to v})$
- So you just need to choose deepest vertex as last one

F. Rotor Traversal

Solution. Maximum case

- Let's look at the correct answer and consider all neighbours of root.
- There is a child which will be visited last. For subtrees of all other children we should use a "minimum strategy" because it fully visits a subtree each time and this is maximum possible length for subtree.
- For last child we firstly go to parent and after that we recursively create visiting strategy.
- If we unfold recursion, strategy can be described like this. There is a leaf which we are going to visit last. Whenever we are on path from root to this leaf we try to go up, than visit all other children, than go along path to leaf. If we are not on path, we just visit all children and than go up.

G. Possible Shifts

Problem statement

- There are two strings s and t of length upto 10^9
- You are given information in format i-th symbol of string s is (not) equal to j sybmbol of t
- We say i is a possible shift if for all j in range $[0..|t|)$ $s[i+j] = t[j]$ for some strings s and t
- There are at most $n \leqslant 100$ such facts
- You need to find number of possible shifts

G. Possible Shifts

Solution

- If there is a contradiction in input there are 0 possible variants
- There are at most n interesting positions in s and at most n in t
- Shifts where there is no pair of interesting positions are at the same position are always possible
- Number of other shifts is not more than n^2 and can be checked in $O(n)$ time each using DSU
- Total complexity is $O(n^3)$

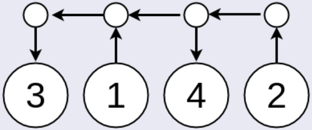
H. Small Graph

Problem statement

- You are given a permutation p of length $n \leqslant 1000$
- You need to create a directed graph of not more than $30n$ edges and $30n$ vertices such that:
- There is a path from i to j $(1 \leqslant i, j \leqslant n)$ iff $i < j$ and $p_i > p_j$

H. Small Graph

Solution

- You need to invent a construction. One of them looks like this:
- Use divide and conquer technique
- Divide permutation into two almost equal parts, solve recursively, solve main part
- For main part we can use such structure. Add k new vertices. Connnect them with edges from right to left. Add edges to new vertices from initial vertices with values smaller than $\frac{k}{2}$. Add edges from new vertices to initial vertices with values bigger than $\frac{k}{2}$.
- 

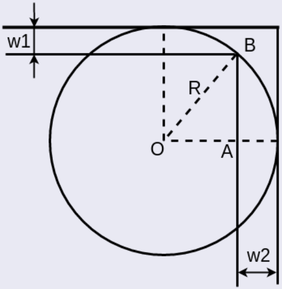
- Total $O(n \log n)$ vertices and edges will be used

I. High Speed

Problem Statement

- There is a 90-degree turn
- Car can move along a smooth sequence of circular arcs and segments
- What is the largest possible turning radius can be?

Solution



Let's look at triangle OAB:

$|OB| = R, |OA| = R - w_2, |AB| = R - w_1$

By Pythagorean theorem we have:

$R^2 = (R - w_1)^2 + (R - w_2)^2$

$R^2 - 2R(w_1 + w_2) + w_1^2 + w_2^2 = 0$

biggest possible R is $w_1 + w_2 + \sqrt{2 w_1 w_2}$