2020-2021 "Orz Panda" Cup Programming Contest

Closing Ceremony

Programming Contest Training Base

Xidian University

Nov. 27, 2020

Part I

Editorial

- Teams passed/tried: $16/17$ (32 tries)
- We want:

$$\max_{1 \le i \le j \le n} \left(\sum_{i \le k \le j} w_k \right) \left(\min_{i \le k \le j} h_k \right)$$

- Teams passed/tried: 16/17 (32 tries)
- We want:

$$\max_{1\le i\le j\le n}\left(\sum_{i\le k\le j} w_k\right)\left(\min_{i\le k\le j} h_k\right)$$

- Brute force approach is $\mathcal{O}(n^2)$, which would cause TLE
- We can modify the above equation, to be

$$\max_{1\le i\le k\le j\le n,\, h_k=\min\limits_{i\le p\le j} h_p} h_k\left(\sum_{i\le p\le j} w_p\right)$$

- Note that $w_i > 0\ \cdots$

Limiting i and j

- If $i' \leq i$, and $j \leq j'$,

$$\sum_{i' \leq p \leq j'} w_p \geq \sum_{i \leq p \leq j} w_p$$

- So for some k, we should choose i as small as possible, and j as large as possible

$$i = \arg \min_{i \leq k,\ \min\limits_{i \leq p \leq k} h_p = h_k}$$

$$j = \arg \min_{j \geq k,\ \min\limits_{k \leq p \leq j} h_p = h_k}$$

- We can use monotonic stack to find (i, j) for all values of k, in $\mathcal{O}(n)$
- Then with a prefix sum we can calculate the answer for each instance of k, in $\mathcal{O}(1)$

- Teams passed/tried: $14/15$ (44 tries)

- Teams passed/tried: 14/15 (44 tries)
- If the vertices of a polygon is $A_1, \cdots, A_n$, its area is

$$\left|\sum_{i=0}^{n-1} \overrightarrow{OA_i} \times \overrightarrow{OA_{i+1}}\right|$$

Where $A_0 = A_n$

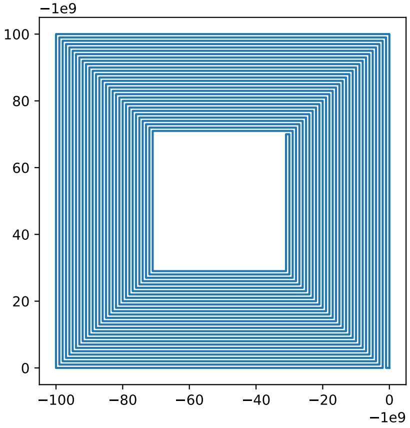
- Teams passed/tried: 14/15 (44 tries)
- If the vertices of a polygon is $A_1, \cdots, A_n$, its area is

$$\left| \sum_{i=0}^{n-1} \overrightarrow{OA_i} \times \overrightarrow{OA_{i+1}} \right|$$

  Where $A_0 = A_n$
- Pitfall 1: the precision of `double` is not enough (sample # 2)
- Pitfall 2: though the answer won't exceed 4×10^{18}, the intermediate values may be very large (sample #3 & #4)

A "Snake"

The sum will soon exceed 2^{64} with inputs like this



Consequences

- The precision of `long double` is not enough for intermediate values, so using it will results inaccurate answer
- The intermediate values won't fit in `long long`, using it will results **undefined behavior**, but it happens to give correct answer in the case
- One of the correct ways is to use `unsigned long long` for modulo 2^{64} arithmatics, then cast it to a `long long`
- Output: `"%lld.%s", x / 2, x & 1 ? ".50" : ".00"`

- Teams passed / tried: $10/13$ (26 tries)

- Teams passed / tried: $10/13$ (26 tries)
- There are at most 2 bits flipped
- The value of $d(0)$ should ensure the xor-sum of $d(i)$ for each i is 0
- If it's true, we know that 0 or 2 bits are flipped
- Then check if the data block is correct, output "`good`" or "`broken`"
- Now consider if the xor-sum of all bits is 1 $\cdots$

One-bit Error

- The value of $d(2^k)$ ensures

$$\bigoplus_{(x \cap 2^k) \neq 0} d(x) = 0$$

- Assume that the position of filpped bit is x
- If it's true, we know that all bits with positions with its k-th binary bit as 1 are correct, so the position of the flipped bit must have its k-th binary bit as 0
- Likewisely, if it's false, we know the position of the flipped bit must have its k-th binary bit as 1
- We now have all binary bits of x, output it!

- Teams passed / tried: $3/9$ $(28$ tries)

- Teams passed / tried: $3/9$ (28 tries)
- Use a `std::set` with a special comparator, to store the segments of unoccupied closestools and find the best one
- Use another `std::set` with a "normal" comparator, to store the positions of Orz Pandas in the toliet so we can find the segment(s) should be merged in type 2 operations

- Teams passed / tried: $3/5$ (9 tries)

- Teams passed / tried: $3/5$ (9 tries)
- The time of reassignments in the t-th operation, $g(t)$, is a *stochastic process*
- We "want" the *time average* of $g(t)$

$$\lim_{T\to\infty} \frac{f(T)}{T} = \lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{m} g(t)$$

- If a stochastic process is *ergodic*, the time average will converge *almost everywhere* to its *ensemble average* or *expectation*

$$P\left\{ \lim_{T\to\infty} \frac{f(T)}{T} = \lim_{T\to\infty} E(g(T)) \right\} = 1$$

- Or

$$\frac{f(T)}{T} \to E(g(T)) \quad (\text{a. e., } T \to \infty)$$

- *Th.* If a Markov process has a unique invariant distribution π, then it is an ergodic process
- *Th.* A Markov process has a unique invariant distribution *iff.* there is exactly one irreducible closed subset of non-null recurrent states
- In the problem:
  - Those states where some internal node has no preferred child are *transient*
    Proof. Once an internal node gets an preferred child, it can not "orphan" it without getting a new one
  - Those states where each internal node has one preferred child are *non-null recurrent*, and those states belong to one irreducible closed subset
    Proof. To get state B from state A, just query each preferred child in B, in reversed order of the *depth* of the child
- Now we can see the ergodicity, so we just need $E(g(\infty))$

Calculate $E(g(\infty))$

$$E(g(\infty)) = \frac{1}{n}\sum_{u} \sum_{v \text{ is on the path from } u \text{ to the root}} E([v_{i+1} \text{ is not the preferred child of } fa(v)])$$

- $E([\text{some event}]) = P\{\text{some event}\}$
- $P\{v \text{ is not the preferred child of } fa(v)\} = 1 - \frac{s(v)}{s(fa(v))-1}$
- $fa(u)$ is the parent of u, and $s(u)$ is the size of the subtree with u as the root
- Now $E(g(\infty))$ can be calculated in $\mathcal{O}(n)$ with an interchange of the summations

$$E(g(\infty)) = \sum_{v}\left(1 - \frac{s(v)}{s(fa(v))}\right)\sum_{u}[v \text{ is an ancester of } u] = \sum_{v}\left(1 - \frac{s(v)}{s(fa(v))}\right)s(v)$$

- Teams passed / tried: $2/9$ $(31$ tries)

- Teams passed / tried: $2/9$ (31 tries)
- Consider the problem bitwisely
- For the i-th bit, *exclusive or* is equivalent to *modulo-*2 *add*
- So we can consider k times of *prefix sum* insteadly

$$(1 + z^1 + z^2 + \cdots)^k = (1 - z)^{-k}$$

$$\left(\frac{\mathrm{d}}{\mathrm{d}z}\right)^t (1-z)^{-k} = k^{\overline{t}}(1-z)^{-k-t}$$

$$(1-z)^{-k} = \sum_{t=0}^{\infty} \frac{k^{\overline{t}} z^t}{t!} = \sum_{t=0}^{\infty} \binom{k+t}{t} z^t$$

Convolution

With the coefficients of $(1-z)^{-k}$

$$b_t = \binom{k+t}{t}$$

We can calculate the answer of the i-th bit as a modulo-2 convolution

$$c_t = \left(\sum_{\tau=0}^{t} b_\tau a_{t-\tau}\right) \bmod 2$$

Use FFT to make the convolution $\mathcal{O}(n\log n)$, the time complexity overall would be $\mathcal{O}(n\log n\log\max a_i)$

$b_t \bmod 2$

b_t is a binomial coefficient which may be very large. But we only care $b_t \bmod 2$. It's either 0 (if b_t is even) or 1 (if b_t is odd). We can see

$$b_t = \frac{(k+t)!}{k!\,t!}$$

The number of the prime divisor 2 in the factorial $m!$ is

$$f(m) = \sum_{j=1}^{\infty} \left\lfloor \frac{m}{2^j} \right\rfloor$$

It's easy to see

$$\left\lfloor \frac{a}{2^j} \right\rfloor + \left\lfloor \frac{b}{2^j} \right\rfloor \le \left\lfloor \frac{a+b}{2^j} \right\rfloor$$

And the left side equals to the right side *iff.* for all $j > 0$

$$(a \bmod 2^j) + (b \bmod 2^j) < 2^j$$

We can now see if there is some $j > 0$ with

$$(k \bmod 2^j) + (t \bmod 2^j) \geq 2^j$$

Then the numerator $(k+t)!$ has more two's in its prime decomposition than the denominator $k!t!$. It's easy to see this criteria holds *iff.*

$$(k \cap t) = 0$$

So

$$\binom{k+t}{t} \bmod 2 = 1$$

iff. $(k \cap t) = 0$
Alternatively you can skip the paper work and calculate $f(a)$, $f(b)$, $f(a+b)$ with brute force, in $\mathcal{O}(\log(a+b))$.

Problem G

Gery's Problem and Orz Pandas

- Teams passed / tried: $1/1$ (2 tries)

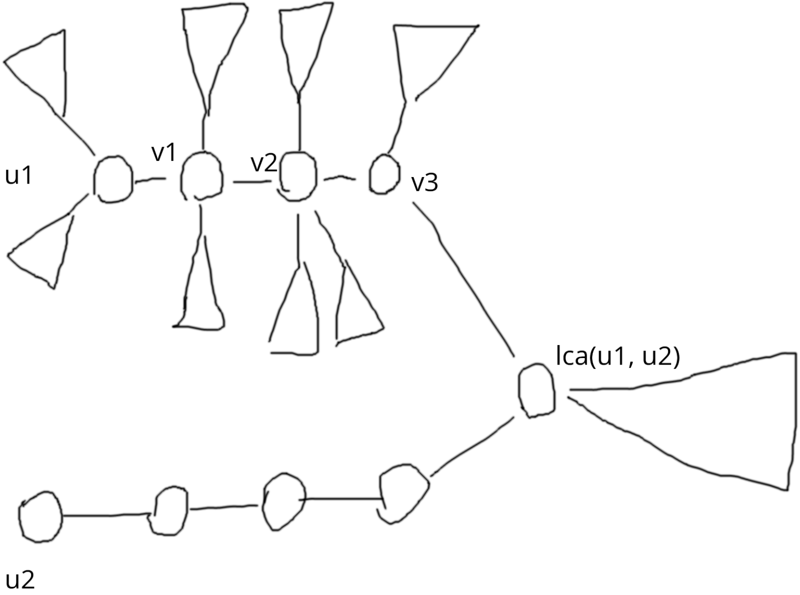
- Teams passed / tried: $1/1$ (2 tries)
- Consider the node which is on the path and closet to the root
- Consider the path from $u_1 = v_0$ to $lca_1(u_1, u_2)$, $v_0, v_1, \cdots, v_k, lca_1(u_1, u_2)$, there are

$$s(v_i) - s(v_{i-1})$$

  nodes which can make $v_i (i > 0)$ closet to it on the path
- The contribution to the answer on the path is

$$\sum_i (s(v_i) - s(v_{i-1})) \times i(q - i)$$

  Where q is the number of edges on the path on u_1 to u_2, $s(v)$ is the size of the subtree with v as the root, and the root of the entire tree is 1


u1
v1
v2
v3
lca(u1, u2)
u2

Split the Sum

For v_k, we have

$$k = d(u) - d(v_k)$$

Where $d(v)$ is the depth of node v. So the equation above can be rewrited

$$\sum (s(v_i) - s(v_{i-1}))(d(u) - d(v_i))(q - d(u) + d(v_i))$$

We can split it into 8 sums, with only v_i as the parameter. For example

$$-(q - d(u)) \sum s(v_i) d(v_i)$$

And

$$-\sum s(v_i) d(fa(v_i))^2$$

Where $fa(v)$ is the parent of node v, with node 1 as the root.

Get the Sum

- Now all the sums splitted have only v_i as the parameter, so we can use heavy-light decomposition to calculate it in $\mathcal{O}(n \log n)$.
- We can improve it to $\mathcal{O}(n)$, with prefix sum on the tree.
- The contribution of $lca(u_1, u_2)$ is calculated specially.

Problem J

Junction of Orz Pandas

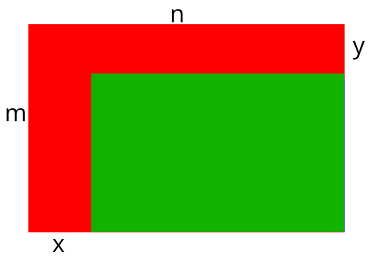
- Teams passed / tried: $0/1$ (3 tries)

Problem J
Junction of Orz Pandas

- Teams passed / tried: $0/1$ $(3$ tries)
- At first note that we can swap a_i and a_j without changing the answer
- Likewise for b_i and b_j
- So we can sort a and b

Reduce the Problem

- After the sorting, generally those lines with minimum a or b will form a "L"-shape area



- We can solve the problem for the remaining area *independently*
- Then just multiply the answer of two areas

Solve the subproblem

- Using the inclusion-exclusion rule:

$$\sum_{ij}\binom{x}{i}\binom{y}{j}(-1)^{ij}q^{(mx+ny-xy)-(mi+nj-ij)}(q-1)^{mi+nj-ij}$$

- It's $\mathcal{O}(xy)$, we can put all terms depending on j together

$$\sum_{i}\binom{x}{i}(-1)^{i}q^{mx+ny-xy-i(m-y)}\sum_{j}\binom{x}{j}(-1)^{j}q^{-j(n-i)}(q-1)^{jn-i}$$

- Note there is an expansion of a binomial

$$\sum_{i}\binom{x}{i}(-1)^{i}q^{mx+ny-xy-i(m-y)}\left(1-\left(\frac{q-1}{q}\right)^{n-i}\right)^{y}$$

- Now it's $\mathcal{O}(x\log x)$

Problem B

Bracelets of Orz Pandas

- Teams passed/tried: $0/2$ (4 tries)

- Teams passed/tried: $0/2$ (4 tries)
- For length n, the number of different bracelets is

$$f(n) = n^{-1} \sum_{d|n} \phi(d) g(n/d)$$

Where $g(x) = fib(x) + fib(x-2) + 2[x \text{ is even}]$

Total Prize

- The answer is

$$\sum_{n=1}^{m} \sum_{d|n} \phi(d) g(n/d)$$

- Interchange the order of summation

$$\sum_{d=1}^{m} \phi(d) \sum_{i=1}^{\lfloor m/d \rfloor} g(i)$$

- $\lfloor m/d \rfloor$ has only $\mathcal{O}(\sqrt{m})$ different values

Total Prize

- The answer is

$$\sum_{n=1}^{m} \sum_{d|n} \phi(d) g(n/d)$$

- Interchange the order of summation

$$\sum_{d=1}^{m} \phi(d) \sum_{i=1}^{\lfloor m/d \rfloor} g(i)$$

- $\lfloor m/d \rfloor$ has only $\mathcal{O}(\sqrt{m})$ different values
- If we can caluclate the prefix sum of ϕ and g quickly (in time T), we can solve the problem in $\mathcal{O}(\sqrt{m}) \times T$
- For ϕ, we have `apiadu`'s sieve
- For g, we have $fib(x+2) - 1 = \sum_{x=1}^{n} fib(x)$

Time complexity

- $\mathcal{O}(m^{7/6})$?

Time complexity

- $\mathcal{O}(m^{7/6})$?
- Note that in the implementation of `apiadu`'s sieve, once we calculated $s(n) = \sum_{i=1}^{n} \phi(i)$, the values of $s(\lfloor n/d \rfloor)$ are already stored into the hashtable
- So the time complexity is only $\mathcal{O}(m^{2/3} + m^{1/2} \log m)$

- Teams passed / tried: $0/1$ (1 tries)

- Teams passed / tried: $0/1$ (1 tries)
- It's easy to find one possible solution (if there is any)
- For any loop containing edges $e_1, e_2, \cdots, e_k$, assume we add a very small loop current δ on the loop, the cost will change

$$\sum_i \left(\frac{\mathrm{d}}{\mathrm{d}f_i} (c_i f_i^2) \right) \delta$$

- To make the cost won't increase for any positive of negative δ, we must have

$$2 \sum_i f_i c_i = 0$$

- This is a linear equation of f_i, if we can find enough equations we can solve f_i

Find Equations

- For each *independent* loops, we can find one equation
- For each *non-tree* edge, we can find one independant loop. There are $|E| - |V| + 1$ loops
- For each vertex v (except the super source vertex), we can find one *conservation* equation

$$w_v = \sum_{e \in \text{edge connecting } v} f_e$$

  There are $|V| - 1$ conservation equations
- Now we have $|E|$ independent linear equations, so we can solve f_i with Gauss elimination
- Note that our equations are exactly Kirchhoff voltage and current equation
- You may have to handle the special case where $c_i = 0$

Part II

Awards

Fastest to Solve

- Fastest to Solve A: `rand`
- Fastest to Solve I: `the path to ac`
- Fastest to Solve C: `Symplectic Geometric Rhythm`
- Fastest to Solve E: `Symplectic Geometric Rhythm`
- Fastest to Solve D: `the path to ac`
- Fastest to Solve H: `Nothing Gold Can Stay`
- Fastest to Solve G: `Symplectic Geometric Rhythm`

Bronze Medals

- HTT: 2/202
- Triy: 3/504
- EasyMath: 3/496
- TakeYourTime: 3/481

Silver Medals

- TriWater: 3/433
- Meow: 3/374
- Three binary trees: 3/364
- Nothing Gold Can Stay: 3/249

Gold Medal
4th Place

`the path to ac`: 4/506

- Fan Zhang
- Zhibin Mai
- Dongchen Chai

Gold Medal

3rd Place

Celeste: 4/466

- Quyang Pan
- Linqi Zhu
- Haozhao Liu

Gold Medal
2nd Place

`rand`: 5/658

- Mengxi Wang
- Shidong Li
- Pengfei Shi

Symplectic Geometric Rhythm: 7/752

- Chang Feng
- Zhongsheng Zhan
- Can Zhou

GL & HF!