Problemset Analysis

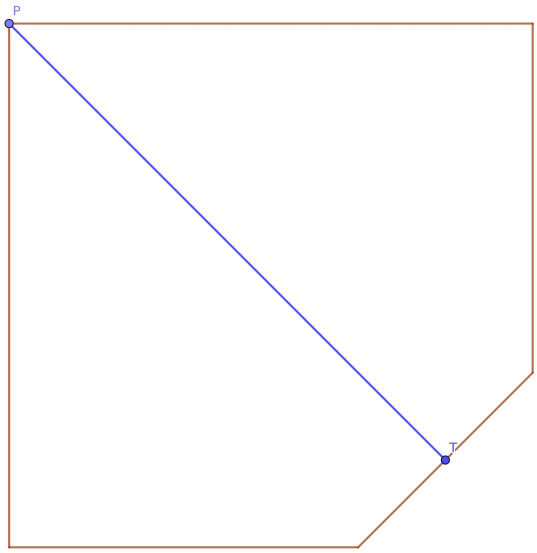
BUET IUPC 2022

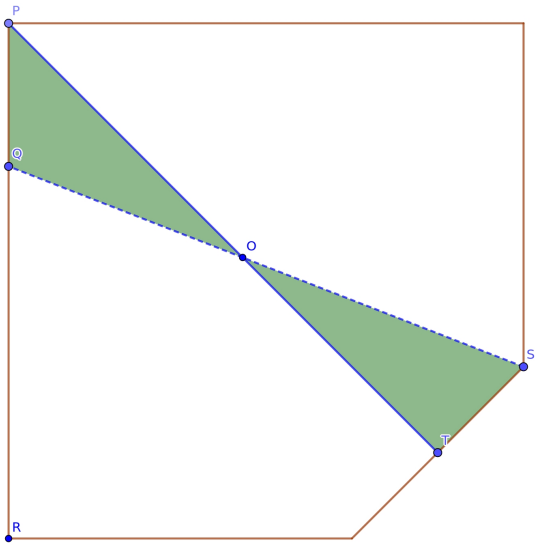
July 2022

A. Chop Chop

Setter: Rafid Bin Mostofa
Techniques: Geometry, Ternary Search

It must be a be straight line segment which divides the polygon into two, otherwise the two new polygons will not be convex. Notice that the sum of perimeters will be the perimeter of the input polygon itself and twice the length of the splitting line segment. Thus, we only need to find a line segment which divides the polygon into half and the length of the segment is minimum possible.





We first choose any random vertex P and find out the segment from this vertex PT which divides the polygon equally. Our plan is to keep rotating this line along the sides of the polygon and find out which one of these states produce the minimal length. We keep rotating this line until one of its endpoints go through a vertex. We also need to make sure the two triangles formed by the previous line and the rotated one have equal areas, which ensures equal division of the polygon.

In the figure above, we kept rotating PT and stopped at S. It's trivial to find Q such that $\Delta OTS = \Delta OPQ$. We can now ternary search on PQ to find an endpoint of the shortest line on PQ. If we fix one endpoint on PQ, it is trivial to find the other endpoint on TS. Similarly, we can also ternary search on TS if we want to, instead of searching on PQ.

We will now rotate the new rotated line even more. Notice that, the line goes through a new vertex. In the figure above, the new rotated line SQ goes through vertex S. We follow the same procedure of rotating line, finding two equal triangles and ternary searching between them. We stop rotating the line once it has rotated more than 180 degree.

Complexity: $O(n \log_{1.5} M)$ where n is the number of vertices and M is the maximum length of a side.

B. Up You Go

Setter: Anik Sarkar

Note that we can divide the queries into 3 cases: Arya's ball hasn't started the journey, Arya's ball is floating on the air, Arya's ball has finished the journey.

In case 1, we only need to count all balls that are on the air at that particular time. We can count this in several ways, we can maintain the intervals during which a ball is on the air or we can binary search on start and end times to figure out which balls have started but not ended their journey during the query time. Note that For a ball with initial velocity u and start time t it's journey is during the interval $[t, t + \frac{2u}{g}]$. Case 3 is similar to case 1.

For Case 2, we can forget about the other cases, and consider only the equation $h(t) = u(t - t_0) - \frac{1}{2}g(t - t_0)^2$ where t_0 is the starting time. $h_i(t) - h_1(t)$ is a linear function $at + b$. So we can consider all the roots of these linear functions to keep a record of which ball is crossing Arya's ball at which direction. This way, for a particular query time, we can use binary search on those crossing points to figure out how many balls are above Arya's ball.

C. Peculiar Primes

Setter: Pritom Kundu

Observation 1. *Let S be the sum of the array and n be the number of elements in the array. Then for any sub-sequence X, $gcd(sum(X), sum(\overline{X})) = gcd(sum(X), S)$*

Observation 2. *If the sum S is prime, there is no valid partition.*

This is because $sum(X) < S$, and any positive number less than a prime is co-prime with it. Now we can assume, S is composite.

Observation 3. *If p is smallest prime factor of S, and p is smaller than n, then the answer is always YES.*

Because there will at least two prefix, both having same prefix sum modulo p (Pigeonhole Principle). Constructing the subset is quite easy here.

Observation 4. *Only case left is, where S is composite and $p \geq n$.*
So, $S \geq p^2$, hence, $S \geq n^2$.
As $A[i]$ can be at most 1000, maximum sum S can be $n \times 1000$, hence, $n \times 1000 \geq n^2$. That is, for this case, $n \leq 1000$.

As $n \leq 1000$, generate all possible subset sum by dynamic programming with bitset (sum can be upto 10^6). Lastly, check gcd with each possible subset sum and S. If $gcd > 1$ is found, the answer is YES.
For constructing the answer, store intermediate states of bitset of dynamic programming.

D. Yet Another Frog Jumping Problem

Setter: Tariq Sajal

There are 40 distinct values of D and it always non-decreasing. Assume color D continues for **consecutive** x days, the frog **starts** at stone idx_1 on the first of these days, **starts** at stone idx_2 on the second of these days and so on. Then it must hold that $idx_i \neq idx_j$(modulo D) if $i \neq j$.
Let us denote first of these days as day T.
So, idx_1 must be first uncolored stone among all stones on day T.
idx_2 must be second uncolored stone among all with condition that $idx_2 \neq idx_1$ (modulo D).
idx_3 must be third uncolored stone among all with condition that $idx_3 \neq idx_1$ (modulo D) and $idx_3 \neq idx_2$ (modulo D).

...
So, binary searching over how many **first** distinct stones with different modulo value were selected as some idx, we can find the value of x.

We need to do the same for every color-change-event. Total 40 binary searches will be needed (hence maximum query count is 720) and rest of the part is brute force.

E. Make More Money

Setter: Md. Shariful Islam

Let $prefix(i)$ means the sum of first i numbers (1-indexed) and $prefix(0)$ is 0. Here we are asked to find $max(prefix(i) + prefix(i-1) - prefix(j))$ where, $0 \leq j < i \leq n$.
Here, $prefix(i) + prefix(i-1) * 2 - prefix(j)$ can be written as $a_i + prefix(i-1) * 2 - prefix(j)$. So, minimizing $prefix(j)$ will be effective. We will iterate over all possible i and use minimum prefix sum so far to find our answer. Therefore, we will maintain minimum prefix sum so far $mprefix$ (initialized with 0), current prefix sum $cprefix$ (initialized with 0) and ans (initialized with $-\infty$). Then, iterate over $1 \leq i \leq n$ to do the following in the given order:
1. Take a_i
2. Update $ans = max(ans, a_i + cprefix * 2 - mprefix)$
3. Update $cprefix = cprefix + a_i$
4. Update $mprefix = min(mprefix, cprefix)$

When the iterations are complete, ans will be our answer.

F. Choco Fun

Setter: Ashiqul Islam

Simpler problem: Let us solve this problem for $A + K = C$. In this case, we need to choose every chocolate. For each chocolate either of P_i or F_i will be added to total happiness value. Total happiness value can be written as: (Sum of A chosen F_i) + (Sum of K chosen P_i). Let $D_i = F_i - P_i$. So total happiness = (Sum of A chosen F_i) + (Sum of K chosen $F_i - D_i$) = (Sum of $A + K$ chosen F_i) - (Sum of K chosen D_i). Here, Sum of $A + K$ chosen F_i is simply sum of every F_i. To maximize total happiness value, we need to choose K chocolate such that sum of chosen D_i is minimum. This can be done by choosing K chocolate with minimum D_i.

Given problem: Now in this problem, $K + A \leq C$. There can be more chocolate than we choose. Lets first choose $K + A$ chocolate and then we can greedily pick K chocolate with minimum D_i and calculate maximum happiness value for the chosen chocolates. To do this efficiently, Lets sort the given input by non-decreasing order of D_i. In this order, for any sub-sequence of chosen $K + A$ chocolate, maximum happiness value is sum of first K P_i and sum of last A F_i. In optimal answer, let last index of chosen P_i is x. So optimal answer would be maximum sum of K P_i from first x chocolate + maximum sum of A F_i from last $C - x$ chocolate. Finally we need to check answer of every x in range $[K, C - A]$ and print their maximum.

Implementation: Now the problems is to find sum of maximum P value for every suffix/prefix of a array. To do this for every prefix, we iterate from left to right, and maintain a min priority queue of our chosen value and their sum. After adding an element to the priority queue, if size of priority queue becomes greater than P, we remove minimum value from the priority queue and update current sum of element inside the priority queue correspondingly. Insert and removal from priority queue runs in $O(\log n)$. We can apply same procedure of suffix also. Final solution of this problems runs in $O(n \log n)$ for each tests.

G. Red Blue Graph

Setter: Arghya Pal

Impossible case: Let all nodes have even degree and total number of edges be odd. Even degree nodes must have equal number of red and blue edges. As all nodes have even degree, total number of Red edges must be equal to total number of Blue edges. Thus, total number of edges is even. A contradiction!

Possible case: All other cases are possible. We prove by construction.
If the graph is an Euler circuit, just color along the circuit alternately. Otherwise there exists some number of odd degree nodes in the graph. Let the number of odd degree nodes be $2k$ (obviously number of odd degree nodes is even). Therefore we can add k edges to make the graph an Euler circuit. Now color the edges alternately similar to the previous case and then remove the extra edges. Now, vertices with even degree would have $degree_R = degree_B$. And vertices with odd degree would have exactly 1 extra edge removed. Therefore after the removal, $\mid degree_R - degree_B \mid = 1$ holds for odd degree vertices.

H. Tomar Mobile E Games Ache?

Setter: Kazi Md Irshad

The horizontal and vertical movements of the snake are independent. Therefore the number of vertical moves is $min(|i_1 - i_2|, 16 - |i_1 - i_2|)$. And similarly, number of horizontal moves is $min(|j_1 - j_2|, 20 - |j_1 - j_2|)$.

I. Colded LCM

Setter: Iftekhar Hakim Kaowsar
Techniques: Number Theory, Math

Abridged statement: Given L and R. Find a and b, such that $L \leq a < b \leq R$ and $LCM(a, b)$ is minimum possible.
Idea: Let us fix $GCD(a, b)$ as g. Then, a would be first multiple of g greater than or equals to L. Let $a = g * x$. So, $b = a + g = g * (x + 1)$. And $LCM(a, b)$ will be $\frac{a*b}{g} = g * x * (x + 1)$. Now, $a <= R$, so either $g \leq sqrt(R)$ or $x \leq sqrt(R)$.
So, we have to loop over all possible g and x.

- When we loop over g upto $sqrt(R)$, we have to find smallest possible x such that $L \leq g * x$ and check whether $g * (x + 1) \leq R$.
- When we loop over x upto $sqrt(R)$, we have to find smallest possible g such that $L \leq g * x$ and check whether $g * (x + 1) \leq R$.

Among all of those pairs g and x, find minimum value of $g * x * (x + 1)$. That is our answer. Time complexity $O(T * sqrt(R))$

Another idea: If we fix $GCD(a, b)$ as g, then $a = \lceil \frac{L}{g} \rceil * g$ and $b = (\lceil \frac{L}{g} \rceil + 1) * g$. So, $LCM(a, b) = \frac{a*b}{g} = g * \lceil \frac{L}{g} \rceil * (\lceil \frac{L}{g} \rceil + 1)$. We know if we check $g = 1, 2, \ldots, L$, value of $\lceil \frac{L}{g} \rceil$ changes at most $O(sqrt(L))$ times. Find those changing g values, check whether both a and b are in given range and find minimum answer. Time complexity $O(T * sqrt(L))$.

J. Shine on You Crazy Diamond

Setter: Pritom Kundu

Observation: It is always beneficial to select prime numbers as x and y.

Solution: Let us denote $cnt[x]$ as the number of multiples of x in the array. Instead of calculating the minimum size of the remaining array, we will calculate the maximum number of elements removed. After we have calculated it, we will subtract it from n to get the correct answer.

- First calculate for each prime $p(1 \leq p \leq 10^6)$, the number of multiples of p in the array.
- If we select x and y, then our answer will be $cnt[x] + cnt[y] - cnt[x \times y]$.
- When $x \times y > 10^6$, $cnt[x \times y]$ will be 0.
- So, iterate over all x and y such that $x \times y \leq 10^6$, calculate $cnt[x] + cnt[y] - cnt[x \times y]$ and update the answer. There aren't many of them.
- For $x \times y > 10^6$, first iterate over all possible x. Now we need the y whose $cnt[y]$ is maximum and $x \times y > 10^6$. This can be done in many ways for example calculate suffix maximum in the cnt array of primes.