3A Contest

July 9, 2019

Ilya Zban

Discover Vladivostok 2019

A. Maximal Difference

In this problem we need to find the maximal possible difference between two n-digit numbers in base-b with the same sum of digits

A. Maximal Difference

The maximal possible difference is achieved between the maximal and the minimal number with the same sum of digits

A. Maximal Difference

The maximal possible difference is achieved between the maximal and the minimal number with the same sum of digits

We can iterate over the sum of digits s from 1 to $bn - 1$ and build the maximal and minimal numbers max_s and min_s with the sum of digits s. Base is $s = 1$, and $min_1 = max_1 = b^{n-1}$ — there is only one n-digit number with sum of digits 1.

A. Maximal Difference

The maximal possible difference is achieved between the maximal and the minimal number with the same sum of digits

We can iterate over the sum of digits s from 1 to $bn - 1$ and build the maximal and minimal numbers max_s and min_s with the sum of digits s. Base is $s = 1$, and $min_1 = max_1 = b^{n-1}$ — there is only one n-digit number with sum of digits 1.

$max_{s+1} = max_s + b^{x_s}$, where x_s is the largest index such that max_s doesn't contain $b - 1$ at x_s-th digit in b-ary representation.

$min_{s+1} = min_s + b^{y_s}$, where y_s is the smallest index such that min_s doesn't contain $b - 1$ at y_s-th digit in b-ary representation.

A. Maximal Difference

We can see that $max_{s+1} - min_{s+1} = max_s - min_s + b^{x_s} - b^{y_s}$. So, we should increase s while $b^{x_s} - b^{y_s}$ is positive.

Constructing numbers max_s and min_s can be done in $\mathcal{O}(nb)$ or $\mathcal{O}(n)$ time, and the difference $max_s - min_s$ can be found in $\mathcal{O}(n)$ time.

B. Polygon Construction

In this problem we are given a sequence of integers $a_1, a_2, \ldots, a_n$, and we need to construct a convex polygon with the sides' length $a_1, a_2, \ldots, a_n$.

B. Polygon Construction

We can construct the polygon if $\forall i \ a_i > 0$ and $2a_{max} < \sum_{i=1}^{n} a_i$.

Let's try to construct the polygon, inscribed in circle. Diameter d of such circle should be not less than a_{max}.

B. Polygon Construction

First case, let the center of the circumcircle be inside of polygon. Then, sum of all angles to the edges of polygon should be equal to 2π:

$$f(d) = 2\sum_{i=1}^{n} \arcsin \frac{a_i}{d} = 2\pi$$

B. Polygon Construction

First case, let the center of the circumcircle be inside of polygon. Then, sum of all angles to the edges of polygon should be equal to 2π:

$$f(d) = 2\sum_{i=1}^{n} \arcsin\frac{a_i}{d} = 2\pi$$

Function $f(d)$ is decreasing, and if $f(a_{max}) \geqslant 2\pi$, there will be $d \geqslant a_{max}$ such that $f(d) = 2\pi$, and we can find such d by binary search.

B. Polygon Construction

First case, let the center of the circumcircle be inside of polygon. Then, sum of all angles to the edges of polygon should be equal to 2π:

$$f(d) = 2\sum_{i=1}^{n} \arcsin\frac{a_i}{d} = 2\pi$$

Function $f(d)$ is decreasing, and if $f(a_{max}) \geqslant 2\pi$, there will be $d \geqslant a_{max}$ such that $f(d) = 2\pi$, and we can find such d by binary search.

Given d, we can make a polygon with

$$x_i = d/2 \cdot cos\varphi_i, y_i = d/2 \cdot sin\varphi_i$$

, where $\varphi_0 = 0, \varphi_i = \varphi_{i-1} + \arcsin\frac{a_{i-1}}{d}$.

B. Polygon Construction

Second case, $f(a_{max}) < 2\pi$. Then the center of the circumcircle is outside of polygon. In this case the angle to the maximal side and sum of angles to all other sides should be equal.

B. Polygon Construction

Second case, $f(a_{max}) < 2\pi$. Then the center of the circumcircle is outside of polygon. In this case the angle to the maximal side and sum of angles to all other sides should be equal. Let's consider function g:

$$g(d) = \sum_{i=1}^{n} 2 \arcsin \frac{a_i}{d} - 2 \arcsin \frac{a_{max}}{d}$$

B. Polygon Construction

Second case, $f(a_{max}) < 2\pi$. Then the center of the circumcircle is outside of polygon. In this case the angle to the maximal side and sum of angles to all other sides should be equal. Let's consider function g:

$$g(d) = \sum_{i=1}^{n} 2 \arcsin \frac{a_i}{d} - 2 \arcsin \frac{a_{max}}{d}$$

$g(a_{max}) < 0$ (because $f(a_{max}) < 2\pi$), and $\lim_{d \to +\infty} g(d) > 0$, $g(d)$ is continuous, so $\exists d \; g(d) = 0$, which can be found by binary search. We can inscribe the polygon with required side lengths in the circle with diameter d, and if $a_{max} = a_n$, then formulas from previous case work fine.

C. Beautiful Patterns

In this problem we are given a rectangle with *NxM* square tiles, some of which are colored in black or white color.

The black/white tile coloring is a beautiful if in every 2x2 square there are 3 black and 1 white or 1 black and 3 white tiles. We need to find a number of beautiful colorings of tiles.

C. Beautiful Patterns

Let $A_{i,j}$ be a color of tile (i, j). The condition from statement can be written as $A_{i,j} + A_{i,j+1} + A_{i+1,j} + A_{i+1,j+1} = 1 \mod 2$.

C. Beautiful Patterns

Let $A_{i,j}$ be a color of tile (i, j). The condition from statement can be written as $A_{i,j} + A_{i,j+1} + A_{i+1,j} + A_{i+1,j+1} = 1 \mod 2$.

We have $(N-1)(M-1)$ equations and NM variables. We can see that if there are no colored tiles, this linear system has a solution (e.g. $A_{i,j} = ij \mod 2$), there are $N + M - 1$ free variables, so this system has 2^{N+M-1} solutions.

C. Beautiful Patterns

Let $A_{i,j}$ be a color of tile (i, j). The condition from statement can be written as $A_{i,j} + A_{i,j+1} + A_{i+1,j} + A_{i+1,j+1} = 1 \mod 2$.

We have $(N-1)(M-1)$ equations and NM variables. We can see that if there are no colored tiles, this linear system has a solution (e.g. $A_{i,j} = ij \mod 2$), there are $N + M - 1$ free variables, so this system has 2^{N+M-1} solutions.

Let's say that these $N + M - 1$ basis variables are variables $A_{i,0}$ and $A_{0,j}$. We know that if all these variables are zero, then $A_{i,j} = ij \mod 2$, and we can check that in general

$$A_{i,j} = A_{0,0} \oplus A_{0,j} \oplus A_{i,0} \oplus ij \mod 2.$$

C. Beautiful Patterns

Now we can imagine a bipartite graph with $n + m$ vertices, where for each colored tile $A_{i,j}$ there is a corresponding edge between vertex i in left part and vertex j in right part of a graph. We can see that for any cycle in this graph we can deduce the value of each edge (which correspond to tile in a grid) by xor-ing all other edges in this cycle. Also, we can show that acyclic subset of edges contains a set of independent tiles of grid.

So, we can make a DFS in this graph, ensure that coloring is non-contradictory and calculate answer as 2^{cc-1}, where cc is a number of connected components in this graph.

D. String without repetitions

In this problem we need to construct the string of length n on the smallest possible alphabet without tandem repeats.

D. String without repetitions

Thue-Morse string s is defined as $s_i = a$ if i has an even number of ones in binary representation, and $s_i = b$ otherwise.
$s = abbabaabbaababba\ldots$.

D. String without repetitions

Thue-Morse string s is defined as $s_i = a$ if i has an even number of ones in binary representation, and $s_i = b$ otherwise.
$s = abbabaabbaababba\ldots$.

Let's see at some simple facts about Thue-Morse string:

D. String without repetitions

Thue-Morse string s is defined as $s_i = a$ if i has an even number of ones in binary representation, and $s_i = b$ otherwise.
$s = abbabaabbaababba\ldots$.

Let's see at some simple facts about Thue-Morse string:

1. $s_{2i} = s_i$.

D. String without repetitions

Thue-Morse string s is defined as $s_i = a$ if i has an even number of ones in binary representation, and $s_i = b$ otherwise.
$s = abbabaabbaababba\ldots$.

Let's see at some simple facts about Thue-Morse string:

1. $s_{2i} = s_i$.
2. $s_{2i+1} = -s_i$ (let $-a = b, -b = a$).

D. String without repetitions

Thue-Morse string s is defined as $s_i = a$ if i has an even number of ones in binary representation, and $s_i = b$ otherwise.
$s = abbabaabbaababba\ldots$.

Let's see at some simple facts about Thue-Morse string:

1. $s_{2i} = s_i$.
2. $s_{2i+1} = -s_i$ (let $-a = b, -b = a$).
3. $s_j = s_{j+1} \to j$ is odd.

D. String without repetitions

Thue-Morse string s is defined as $s_i = a$ if i has an even number of ones in binary representation, and $s_i = b$ otherwise.
$s = abbabaabbaababba\ldots$.

Let's see at some simple facts about Thue-Morse string:

1. $s_{2i} = s_i$.
2. $s_{2i+1} = -s_i$ (let $-a = b, -b = a$).
3. $s_j = s_{j+1} \to j$ is odd.
4. $\nexists j : s_j = s_{j+1} = s_{j+2}$.

D. String without repetitions

Thue-Morse string s is defined as $s_i = a$ if i has an even number of ones in binary representation, and $s_i = b$ otherwise.
$s = abbabaabbaababba\ldots$.

Let's see at some simple facts about Thue-Morse string:

1. $s_{2i} = s_i$.
2. $s_{2i+1} = -s_i$ (let $-a = b, -b = a$).
3. $s_j = s_{j+1} \to j$ is odd.
4. $\nexists j : s_j = s_{j+1} = s_{j+2}$.
5. let $p(s) = s_0 s_2 s_4 \ldots$, $n(s) = s_1 s_3 s_5 \ldots$. $p(s) = s, n(s) = -s$.

D. String without repetitions

Thue-Morse string s is defined as $s_i = a$ if i has an even number of ones in binary representation, and $s_i = b$ otherwise.
$s = abbabaabbaababba\ldots$.

Let's see at some simple facts about Thue-Morse string:

1. $s_{2i} = s_i$.
2. $s_{2i+1} = -s_i$ (let $-a = b, -b = a$).
3. $s_j = s_{j+1} \to j$ is odd.
4. $\nexists j : s_j = s_{j+1} = s_{j+2}$.
5. let $p(s) = s_0 s_2 s_4 \ldots$, $n(s) = s_1 s_3 s_5 \ldots$. $p(s) = s, n(s) = -s$.
6. for every 5 consecutive symbols there are at least two consecutive equal symbols (only bad strings are ababa and babab, and they aren't substrings of s (either $p(s)$ or $n(s)$ would contradict with previous observation).

D. String without repetitions

Theorem

There are no such $i \geqslant 0, l > 0$ *such that* $s_i \dots s_{i+l} = s_{i+l} \dots s_{i+2l}$.

D. String without repetitions

Theorem

There are no such $i \geqslant 0, l > 0$ such that $s_i \dots s_{i+l} = s_{i+l} \dots s_{i+2l}$.

Proof.

Let l be an odd number. By obs.4 $l \neq 1$. $2l + 1 > 5$, so there is $j : i \leqslant j < 2l + 1$ such that $s_j = s_{j+1}$ (obs.6). $s_j s_{j+1}$ are contained either in $s_i \dots s_{i+l}$ or $s_{i+l} \dots s_{i+2l}$, and these string are equal, so either $s_{j+l} = s_{j+1+l}$, either $s_{j-l} = s_{j+1-l}$. But l is odd, so at least one of j and ($j - l$ or $j + l$) should be even, and that's contradiction with obs.3.

D. String without repetitions

Theorem

There are no such $i \geqslant 0, l > 0$ such that $s_i \dots s_{i+l} = s_{i+l} \dots s_{i+2l}$.

Proof.

Let l be an odd number. By obs.4 $l \neq 1$. $2l + 1 > 5$, so there is $j : i \leqslant j < 2l + 1$ such that $s_j = s_{j+1}$ (obs.6). $s_j s_{j+1}$ are contained either in $s_i \dots s_{i+l}$ or $s_{i+l} \dots s_{i+2l}$, and these string are equal, so either $s_{j+l} = s_{j+1+l}$, either $s_{j-l} = s_{j+1-l}$. But l is odd, so at least one of j and ($j - l$ or $j + l$) should be even, and that's contradiction with obs.3.

If l is even, we can reduce the repetition to $l/2$ by taking $s_i s_{i+2} \dots s_{i+l} = s_{i+l} s_{i+l+2} \dots s_{i+2l}$ until l would become odd, and that would lead us to contradiction. □

D. String without repetitions

Let's construct new string t, where $t_i = f(s_i s_{i+1})$, and $f(aa) = 0, f(ab) = 1, f(ba) = 2, f(bb) = 3$. $s = abbabaab\ldots$, so $t = 1321201\ldots$.

D. String without repetitions

Let's construct new string t, where $t_i = f(s_i s_{i+1})$, and $f(aa) = 0, f(ab) = 1, f(ba) = 2, f(bb) = 3$. $s = abbabaab\ldots$, so $t = 1321201\ldots$.

It is easy to see that there are no tandem repeats in t, because tandem repeat in t leads to a tandem repeat with an overlay in s, and that contradicts the theorem.

D. String without repetitions

Let's construct new string t, where $t_i = f(s_i s_{i+1})$, and $f(aa) = 0, f(ab) = 1, f(ba) = 2, f(bb) = 3$. $s = abbabaab\ldots$, so $t = 1321201\ldots$.

It is easy to see that there are no tandem repeats in t, because tandem repeat in t leads to a tandem repeat with an overlay in s, and that contradicts the theorem.

Also, every aa in s can occur only in $b\mathbf{aa}b$, and every bb can occur only in $a\mathbf{bb}a$. So we can replace $f(bb) = 3$ with $f(bb) = 0$, and t still won't have any tandem repeats.

D. String without repetitions

Let's construct new string t, where $t_i = f(s_i s_{i+1})$, and $f(aa) = 0, f(ab) = 1, f(ba) = 2, f(bb) = 3$. $s = abbabaab\ldots$, so $t = 1321201\ldots$.

It is easy to see that there are no tandem repeats in t, because tandem repeat in t leads to a tandem repeat with an overlay in s, and that contradicts the theorem.

Also, every aa in s can occur only in $b\mathbf{aa}b$, and every bb can occur only in $a\mathbf{bb}a$. So we can replace $f(bb) = 3$ with $f(bb) = 0$, and t still won't have any tandem repeats.

So, t is infinite string without tandem repeats with 3 different characters. Cases $n \leqslant 3$ should be handled manually.

E. Beans Gathering

In this problem we are given that there are a_{j_k} beans in the cell j_k, and we are allowed to move i beans from the cell i to the cells $0, 1, \ldots, i-1$.

We are asked to determine is it possible to gather all the beans in the cell 0.

E. Beans Gathering

If there is a solution, then we can always do the operations in decreasing way (because we can swap two consecutive operations x and y if $x < y$ and x is before y).

E. Beans Gathering

If there is a solution, then we can always do the operations in decreasing way (because we can swap two consecutive operations x and y if $x < y$ and x is before y).

The solution is the implementation of this process. We can see that if the smallest used operation was with the i-th cell, then all cells $0, 1, \ldots, i-1$ were increased by the same value x.

We iterate over all the cels in decreasing order from the maximal j_k, keep the current value x that we need to add to all the cells, and check if $a_i + x$ is divisible by i.

E. Beans Gathering

Let A be the $\max(a_k)$, and n is the initial number of non-empty cells.

The solution works in $\mathcal{O}(\min(A, nt(nA)))$, where $t(nA)$ is is the maximal possible $i_k - i_{k-1}$ (maximal number of consecutive empty cells) such that there is a solution to the problem.

For the problem constraints ($n \leqslant 10^5; A \leqslant 10^{18}$) we have $t(nA) \leqslant 30$. We can see that if there is the largest cell i and k empty cells $i-1, \ldots, i-k$, then we need a_i to be divisible by $i(i-1)\ldots(i-k)/2^{k/2}$, which is growing very fast, but the total number of beans is limited by nA.

F. Reverse beans gathering

In this problem we need to find anti-lexicographically minimal test with positive answer for previous problem with total K beans.

F. Reverse beans gathering

Observation: for each i we have $a_i \leqslant i$ in answer. Proof: if for some i we have $a_i > i$, we can take $i + 1$ beans from cell i and put them to cell $i + 1$. New answer is anti-lexicographically smaller, and it is still solvable (we can replace first operation with the cell i to the operation with the cell $i + 1$).

F. Reverse beans gathering

Observation: for each i we have $a_i \leqslant i$ in answer. Proof: if for some i we have $a_i > i$, we can take $i + 1$ beans from cell i and put them to cell $i + 1$. New answer is anti-lexicographically smaller, and it is still solvable (we can replace first operation with the cell i to the operation with the cell $i + 1$).

Observation: for each $i > 0$, any operation either keeps the $\sum_{j=i}^{+\inf} a_j$ unchanged, or it decreases this sum by i.

F. Reverse beans gathering

$\sum_{j=2}^{+\inf} a_j$ is divisible by 2 (follows from second observation)

F. Reverse beans gathering

$\sum_{j=2}^{+\text{inf}} a_j$ is divisible by 2 (follows from second observation)

$a_1 = K - \sum_{j=2}^{+\text{inf}} a_j = K \mod 2.$

F. Reverse beans gathering

$\sum_{j=2}^{+\text{inf}} a_j$ is divisible by 2 (follows from second observation)

$$a_1 = K - \sum_{j=2}^{+\text{inf}} a_j = K \mod 2.$$

By induction we get that $a_i = (K - \sum_{j=1}^{i-1} a_j) \mod i + 1$.

This solution works in $\mathcal{O}(N)$, where N is maximal used cell in answer. We can see that $N = \mathcal{O}(\sqrt{K})$, so this is fast enough.

G. Equation

In this problem we are given a polynomial $P = a_N x^N + a_{N-1} x^{N-1} + \ldots + a_0$ and a prime number p.

We are asked to find the lowest non-negative x such that $P(x) = 0 \mod p$.

G. Equation

$P(x + p^2) = P(x) \mod p^2$, so we need to find $x < p^2$.

G. Equation

$P(x + p^2) = P(x) \mod p^2$, so we need to find $x < p^2$. Let $x = x_0 + x_1 p$, $0 \leqslant x_0, x_1 < p$.

G. Equation

$P(x + p^2) = P(x) \mod p^2$, so we need to find $x < p^2$. Let $x = x_0 + x_1 p$, $0 \leqslant x_0, x_1 < p$.

$P(x_0 + x_1 p) = 0 \mod p^2$,

$(\sum_{i=0}^{N} a_i x_0^i) + (\sum_{i=1}^{N} i a_i x_0^{i-1} x_1 p) = 0 \mod p^2$, so $\sum_{i=0}^{N} a_i x_0^i$ should be divisible by p.

G. Equation

We can iterate over possible values of x_0, check that $\sum_{i=0}^{N} a_i x_0^i = 0 \mod p$, and after that we can calculate the only possible value of x_1.

G. Equation

We can iterate over possible values of x_0, check that $\sum_{i=0}^{N} a_i x_0^i = 0 \mod p$, and after that we can calculate the only possible value of x_1.

As $x_1 p \sum_{i=1}^{N} i a_i x_0^{i-1} = -\sum_{i=0}^{N} a_i x_0^i \mod p^2$, and the right side is divisible by p,

G. Equation

We can iterate over possible values of x_0, check that $\sum_{i=0}^{N} a_i x_0^i = 0 \mod p$, and after that we can calculate the only possible value of x_1.

As $x_1 p \sum_{i=1}^{N} i a_i x_0^{i-1} = -\sum_{i=0}^{N} a_i x_0^i \mod p^2$, and the right side is divisible by p,

$x_1 \sum_{i=1}^{N} i a_i x_0^{i-1} = (-\sum_{i=0}^{N} a_i x_0^i)/p \mod p$, or

G. Equation

We can iterate over possible values of x_0, check that $\sum_{i=0}^{N} a_i x_0^i = 0 \mod p$, and after that we can calculate the only possible value of x_1.

As $x_1 p \sum_{i=1}^{N} i a_i x_0^{i-1} = -\sum_{i=0}^{N} a_i x_0^i \mod p^2$, and the right side is divisible by p,

$$x_1 \sum_{i=1}^{N} i a_i x_0^{i-1} = (-\sum_{i=0}^{N} a_i x_0^i)/p \mod p, \text{ or}$$

$$x_1 = (-\sum_{i=0}^{N} a_i x_0^i)/p \cdot (\sum_{i=1}^{N} i a_i x_0^{i-1})^{-1} \mod p.$$

G. Equation

We can iterate over possible values of x_0, check that $\sum_{i=0}^{N} a_i x_0^i = 0 \mod p$, and after that we can calculate the only possible value of x_1.

As $x_1 p \sum_{i=1}^{N} i a_i x_0^{i-1} = -\sum_{i=0}^{N} a_i x_0^i \mod p^2$, and the right side is divisible by p,

$$x_1 \sum_{i=1}^{N} i a_i x_0^{i-1} = (-\sum_{i=0}^{N} a_i x_0^i)/p \mod p, \text{ or}$$

$$x_1 = (-\sum_{i=0}^{N} a_i x_0^i)/p \cdot (\sum_{i=1}^{N} i a_i x_0^{i-1})^{-1} \mod p.$$

Answer is minimum of $x_0 + x_1 p$ by all such pairs x_0, x_1. Complexity is $\mathcal{O}(pN \log p)$, but with very small constant.

H. Competition

There are two teams, members of a team A have skill levels $a_1, a_2, \ldots, a_n$, and members of a team B have skill levels $b_1, b_2, \ldots, b_n$. We can choose a matching between participants from different teams in a way that maximizes score of team B, if a win gives two points, and a draw gives one point.

H. Competition

Let $a_i \leqslant a_{i+1}, b_i \leqslant b_{i+1}$.

Let f_j be a maximum number of members from team A, which can be outplayed by first j players of team B, and g_i be a maximum number of members from team A with indices $\geqslant i$, which can be outplayed by players of team B. These two arrays can be calculated in linear time by greedy algorithm with two pointers.

H. Competition

Let $a_i \leqslant a_{i+1}, b_i \leqslant b_{i+1}$.

Let f_j be a maximum number of members from team A, which can be outplayed by first j players of team B, and g_i be a maximum number of members from team A with indices $\geqslant i$, which can be outplayed by players of team B. These two arrays can be calculated in linear time by greedy algorithm with two pointers.

Note that $2f_n = 2g_1$ is maximum number of points that can be achieved without draws.

H. Competition

We can look for the answer, where all draws are between the participants with equal score (if there are two draws between participants with skills x, y and $x < y$, we can replace these draws by one win, and still get 2 points).

Suppose there are some draws between participants with skill x, and participants with skill x are $a_{l_a}, \ldots, a_{r_a}$ and $b_{l_b}, \ldots, b_{r_b}$. We can iterate over the number of draws d, make draws between participants $(l_a, r_b), (l_a + 1, r_b - 1), \ldots, (l_a + d - 1, r_b - d + 1)$. In this case we can get the total score equals to $g_{r_b - d} + d + f_{l_a + d}$, and we should take maximum of this value by all possible pairs x, d.

Overall complexity is $\mathcal{O}(N \log N)$ due to sorting of input arrays.

I. The incircle

In this problem we are given a convex polygon, and we need to inscribe a circle with maximal radius.

I. The incircle

First solution: binary search of maximum radius.

How to check if we are able to inscribe a circle with radius R in polygon? It is simple, we can just shift all edges of polygon inside by R and check if the polygon is still non-empty. Each edge of polygon defines a semiplane $A_i x + B_i y + C_i \leqslant R$, and we need to check if the intersection of these semiplanes is non-empty. It can be done in $\mathcal{O}(n)$ time.

I. The incircle

Second solution: two ternary searches.

Let's define function $R(x, y)$ as minimum of all distances from point (x, y) to the edges of polygon,

$$R(x, y) = \min_{i=1}^{n}(A_i x + B_i y + C)$$

We can find a maximum of this function by two nested ternary searches by x and y, and it is the answer to the problem. We don't need to carry about points outside polygon, if we define $R(x, y) < 0$ to such points.

J. The dividing line

In this problem we are given a set of points, and a bunch of queries. For each query we need to determine if there are two points from this set, lying in a different half-planes from this line.

J. The dividing line

Let the query line be between points A and B. We need to find points P_1, P_2 such that $\overrightarrow{AB} \times \overrightarrow{AP_1} > 0$ and $\overrightarrow{AB} \times \overrightarrow{AP_2} < 0$.

J. The dividing line

Let the query line be between points A and B. We need to find points P_1, P_2 such that $\overrightarrow{AB} \times \overrightarrow{AP_1} > 0$ and $\overrightarrow{AB} \times \overrightarrow{AP_2} < 0$.

Let $\vec{n}$ be a normal vector to $\overrightarrow{AB}$. We will minimize/maximize scalar product with n instead of minimizing/maximizing vector product with $\overrightarrow{AB}$ (just for convenience).

J. The dividing line

Let the query line be between points A and B. We need to find points P_1, P_2 such that $\overrightarrow{AB} \times \overrightarrow{AP_1} > 0$ and $\overrightarrow{AB} \times \overrightarrow{AP_2} < 0$.

Let $\vec{n}$ be a normal vector to $\overrightarrow{AB}$. We will minimize/maximize scalar product with n instead of minimizing/maximizing vector product with $\overrightarrow{AB}$ (just for convenience).

We need to look only at the points on the convex hull of our set. Let's construct the convex hull and divide it to upper and lower parts by line between leftmost and rightmost points.

J. The dividing line

We can find the point with maximal scalar product with given vector $\vec{n}$ by ternary search in both parts of convex hull. Suppose the optimal point is in the upper part of convex hull. Then the function $f(i) = up_i \cdot \vec{n}$ is unimodular, and we can do a ternary search to find an optimal point.

J. The dividing line

We can find the point with maximal scalar product with given vector $\vec{n}$ by ternary search in both parts of convex hull. Suppose the optimal point is in the upper part of convex hull. Then the function $f(i) = up_i \cdot \vec{n}$ is unimodular, and we can do a ternary search to find an optimal point.

For each query we find most distant points in directions collinear to normal vector for this line in $\mathcal{O}(\log N)$ time, so the overall complexity is $\mathcal{O}(M \log N)$, which pass the time limit.

K. Stringangulation

In this problem we are given a circular string s and we need to count the number of ways to divide it into 3 strings a, b, c, so

$$a + b > c, b + c > a, c + a > b$$

K. Stringangulation

Each partition can be specified by three integers $0 \leqslant i < j < k < |s|$, where i, j, k correspond to the starts of the strings b, c, a.

K. Stringangulation

Each partition can be specified by three integers $0 \leqslant i < j < k < |s|$, where i, j, k correspond to the starts of the strings b, c, a.

First, we need to calculate the $lcp_{i,j}$ — longest common prefix of i-th and j-th cyclic shifts of s. Knowing $lcp_{i,j}$ we are able to compare substrings of string in $\mathcal{O}(1)$ time.

We can calculate all values $lcp_{i,j}$ in $\mathcal{O}(n^2)$ time by simple dynamic programming (for $s_i \neq s_j$ we have $lcp_{i,j} = 0$, and $lcp_{i,j} = 1 + lcp_{i+1,j+1}$ otherwise, with the case of equal cyclic shifts).

K. Stringangulation

Each partition can be specified by three integers $0 \leqslant i < j < k < |s|$, where i, j, k correspond to the starts of the strings b, c, a.

First, we need to calculate the $lcp_{i,j}$ — longest common prefix of i-th and j-th cyclic shifts of s. Knowing $lcp_{i,j}$ we are able to compare substrings of string in $\mathcal{O}(1)$ time.

We can calculate all values $lcp_{i,j}$ in $\mathcal{O}(n^2)$ time by simple dynamic programming (for $s_i \neq s_j$ we have $lcp_{i,j} = 0$, and $lcp_{i,j} = 1 + lcp_{i+1,j+1}$ otherwise, with the case of equal cyclic shifts).

Let $f_{i,j}$ be 1 if $s_i s_{i+1} \dots s_{j-1} > s_j s_{j+1} \dots s_{i-1}$ and 0 otherwise. This function is in fact a check if $a + b > c$ for partition, where $a + b = s_i s_{i+1} \dots s_{j-1}$ and $c = s_j s_{j+1} \dots s_{i-1}$.

K. Stringangulation

Let's iterate over the position of substring a — fix indices i, k ($i < k$). Now we need to find a number of possible values j, which will divide string $s_i s_{i+1} \dots s_{k-1}$ to strings b, c.

K. Stringangulation

Let's iterate over the position of substring a — fix indices i, k ($i < k$). Now we need to find a number of possible values j, which will divide string $s_i s_{i+1} \dots s_{k-1}$ to strings b, c.

Let's keep values $f_{i,j}$ in two bitsets: bitset $b_{i,j} = f_{i,j}$ will help us to check if $a + b > c$, and bitset $c_{i,j} = f_{j,i}$ will help us to check if $c + a > b$.

K. Stringangulation

Let's iterate over the position of substring a — fix indices i, k ($i < k$). Now we need to find a number of possible values j, which will divide string $s_i s_{i+1} \ldots s_{k-1}$ to strings b, c.

Let's keep values $f_{i,j}$ in two bitsets: bitset $b_{i,j} = f_{i,j}$ will help us to check if $a + b > c$, and bitset $c_{i,j} = f_{j,i}$ will help us to check if $c + a > b$.

First, we check if $f_{i,k}$ is true ($b + c > a$). Next, we take `and` of three bitsets: b_k, c_i, and bitset d where $d_j = 1$ iff $i < j < k$. We can find their `and` in $\mathcal{O}(n/w)$ time and bitcount is the number of appropriate j-s.

Overall complexity is $\mathcal{O}(n^3/w)$, where $w = 64$ for our purposes.

L. Building a cube

In this problem we need to construct a cube with vertices on a distance $d_1, d_2, \ldots, d_8$ from a given plane $Ax + By + Cz + D$.

L. Building a cube

Suppose all vertices of a cube are in one half-space from a given plane. Later we will reduce the problem to this case.

L. Building a cube

Suppose all vertices of a cube are in one half-space from a given plane. Later we will reduce the problem to this case.

Let d_1 be a distance to closest vertex A, and $d_2 \leqslant d_3 \leqslant d_4$ be a distance to its neighbours B, C, D (we can brute force a permutation of d_i). Let $d_2 - d_1 = x, d_3 - d_1 = y, d_4 - d_1 = z$ be a lengths of projections of vectors from A to its neighbours on normal vector $\vec{n}$ to a given plane. As lengths of projections of parallel segments to one line are equal, there should be distances $d_1 + x + y, d_1 + x + z, d_1 + y + z, d_1 + x + y + z$ among the d_5, d_6, d_7, d_8, otherwise there is no solution.

L. Building a cube

Suppose all vertices of a cube are in one half-space from a given plane. Later we will reduce the problem to this case.

Let d_1 be a distance to closest vertex A, and $d_2 \leqslant d_3 \leqslant d_4$ be a distance to its neighbours B, C, D (we can brute force a permutation of d_i). Let $d_2 - d_1 = x, d_3 - d_1 = y, d_4 - d_1 = z$ be a lengths of projections of vectors from A to its neighbours on normal vector $\vec{n}$ to a given plane. As lengths of projections of parallel segments to one line are equal, there should be distances $d_1 + x + y, d_1 + x + z, d_1 + y + z, d_1 + x + y + z$ among the d_5, d_6, d_7, d_8, otherwise there is no solution.

Let the length of cube edge be a. $x = a \cos \angle(\overrightarrow{AB}, \vec{n})$, $y = a \cos \angle(\overrightarrow{AC}, \vec{n})$, $z = a \cos \angle(\overrightarrow{AD}, \vec{n})$. As $\overrightarrow{AB}, \overrightarrow{AC}, \overrightarrow{AD}$ are mutually perpendicular, $x^2 + y^2 + z^2 = a^2$, so we know the length of edge of cube.

L. Building a cube

Let $\vec{X} = \overrightarrow{AB}/a, \vec{Y} = \overrightarrow{AC}/a, \vec{Z} = \overrightarrow{AD}/a$. We know that $\vec{X} \cdot \vec{n} = x/a, \vec{Y} \cdot \vec{n} = y/a, \vec{Z} \cdot \vec{n} = z/a, \vec{X} \cdot \vec{Y} = \vec{X} \cdot \vec{Z} = \vec{Y} \cdot \vec{Z} = 0, \vec{X} \cdot \vec{X} = \vec{Y} \cdot \vec{Y} = \vec{Z} \cdot \vec{Z} = 1$.

L. Building a cube

Let $\vec{X} = \overrightarrow{AB}/a, \vec{Y} = \overrightarrow{AC}/a, \vec{Z} = \overrightarrow{AD}/a$. We know that $\vec{X} \cdot \vec{n} = x/a, \vec{Y} \cdot \vec{n} = y/a, \vec{Z} \cdot \vec{n} = z/a, \vec{X} \cdot \vec{Y} = \vec{X} \cdot \vec{Z} = \vec{Y} \cdot \vec{Z} = 0, \vec{X} \cdot \vec{X} = \vec{Y} \cdot \vec{Y} = \vec{Z} \cdot \vec{Z} = 1$.

We can construct the matrix H with rows $\vec{X}, \vec{Y}, \vec{Z}$. $H\vec{n} = (x/a, y/a, z/a) = \vec{p}$.

This matrix is orthogonal, so $|\vec{p}| = |\vec{n}| = 1$, and this matrix is a turn/symmetry operator.

L. Building a cube

Let $\vec{X} = \overrightarrow{AB}/a, \vec{Y} = \overrightarrow{AC}/a, \vec{Z} = \overrightarrow{AD}/a$. We know that $\vec{X} \cdot \vec{n} = x/a, \vec{Y} \cdot \vec{n} = y/a, \vec{Z} \cdot \vec{n} = z/a, \vec{X} \cdot \vec{Y} = \vec{X} \cdot \vec{Z} = \vec{Y} \cdot \vec{Z} = 0, \vec{X} \cdot \vec{X} = \vec{Y} \cdot \vec{Y} = \vec{Z} \cdot \vec{Z} = 1$.

We can construct the matrix H with rows $\vec{X}, \vec{Y}, \vec{Z}$. $H\vec{n} = (x/a, y/a, z/a) = \vec{p}$.

This matrix is orthogonal, so $|\vec{p}| = |\vec{n}| = 1$, and this matrix is a turn/symmetry operator.

If $\vec{n} = \vec{p}$, we can take $H = E$, otherwise we can take a matrix, which makes symmetry translation of points to a plane, perpendicular to vector $\vec{p} - \vec{n}$. $(\vec{p} - \vec{n}) \cdot \vec{t} = 0$, where $\vec{t}$ is a point of the plane.

$H\vec{t} = \vec{t} - 2(\vec{t} \cdot (\vec{p} - \vec{n})) \cdot (\vec{p} - \vec{n})/|\vec{p} - \vec{n}|^2$

L. Building a cube

We can construct matrix H by applying previous transform to standard basis vectors.

L. Building a cube

We can construct matrix H by applying previous transform to standard basis vectors.

Now we can take A as an arbitrary point at distance d_1 from a plane, points B, C, D as $A + a\vec{X}$, $A + a\vec{Y}$, $A + a\vec{Z}$, and the rest of points of cube in a obvious way.

L. Building a cube

We can construct matrix H by applying previous transform to standard basis vectors.

Now we can take A as an arbitrary point at distance d_1 from a plane, points B, C, D as $A + a\vec{X}$, $A + a\vec{Y}$, $A + a\vec{Z}$, and the rest of points of cube in a obvious way.

We solved a problem with an assumption that all points are in one half-space from a given plane. We can brute force the subdivision of points to two halfspaces and multiply distances in one half-space by -1 and solve our version of a problem.