ТУЙМААДА XXIX Международная олимпиада

TUYMAADA The 29th International Olympiad

PROBLEMS AND SOLUTIONS

INFORMATICS

ИНФОРМАТИКА

ЗАДАЧИ И РЕШЕНИЯ

Yakutsk 2022
Якутск 2022

XXIX Международная олимпиада школьников по математике, физике и информатике «Туймаада». ИНФОРМАТИКА. – Якутск, 2022.

Сборник содержит задачи XXIX Международной олимпиады «Туймаада» по информатике, а также возможные варианты решений. Олимпиада проводилась 30 июня – 10 июля 2022 года в г. Якутске. Олимпиада прошла в смешанном очном и дистанционном форматах в два соревновательных дня, в первый из которых участникам было предложено решить четыре задачи, во второй – пять.

This booklet contains the problems and possible solutions of the 29th Tuymaada International Olympiad in informatics. The olympiad was held on June 30 – July 10, 2022 in Yakutsk, Russia. The olympiad was held both remotely and on-site, in two competition days. The participants were invited to solve four and five problems on the first and second day, respectively.

АВТОРЫ		*AUTHORS*
Юрий Саввич АНТОНОВ Северо-Восточный федеральный университет им. М. К. Аммосова	A	*Yuri ANTONOV* *M. K. Ammosov North-Eastern Federal University*
Сергей Геннадьевич ВОЛЧЁНКОВ Ярославский государственный университет им. П. Г. Демидова	B	*Sergei VOLCHENKOV* *P. G. Demidov Yaroslavl State University*
Никита Сергеевич СЕМЁНОВ студент, Высшая школа экономики	C	*Nikita SEMYONOV* *student, Higher School of Economics*
Семен Леонидович МЕКУМЯНОВ студент, Университет ИТМО	D	*Semyon MEKUMYANOV* *student, ITMO University*
Виталий Витальевич ИВАНОВ Яндекс.Маркет	E	*Vitalii IVANOV* *Yandex.Market*
Дьулустан Васильевич Никифоров Северо-Восточный федеральный университет им. М. К. Аммосова	F	*Dulustan NIKIFOROV* *M. K. Ammosov North-Eastern Federal University*
Василий Алквиадович БУЛАТОВ Яндекс Go	G	*Vasilii BULATOV* *Yandex Go*
Тимур Алексеевич ЗЫКОВ студент, Московский физико-технический институт	H	*Timur ZYKOV* *student, Moscow Institute of Physics and Technology*
ИНКОГНИТО	I	*INCOGNITO*

Problem statements

A. Granopodus

Time limit: 1 second
Memory limit: 256 megabytes

The planet Tau Ceti is home to a delicious fish species called granopodus. These fish have the shape of a convex polyhedron, the number of faces of which varies between 4 and 100.

A taucetizen called R2D2 has caught n granopoduses and wants to fry them. R2D2 can fit only m fish in their pan. It takes 1 minute to fry a single face of a granopodus. R2D2 has neatly separated their catch into heaps according to the number of faces of a fish.

What maximal number of fish can R2D2 manage to fry in k minutes?

Input

Input data is given in three lines. First line contains three numbers n, m, k ($n, m, k \leq 2500$). The second line contains the quantities of fish in each of the heap. Obviously, the sum of these numbers is n. The last line contains the number of faces for granopoduses in each of the heaps in the same order as on the second line.

Output

Output one integer — the number of fish.

Scoring

Each test is scored independently.

Example

standard input	standard output
2 3 4 2 4	2

B. Sum of digits

Time limit: 1 second
Memory limit: 256 megabytes

An inquisitive novice programmer Petya has noticed that the the sum of an integer number's digits behaves peculiarly as the number increases. Sometimes it increases uniformly with the number, but sometimes it falls sharply. For example, starting from the number 96, the digit sum changes in following way: 15, 16, 17, 18, 1, 2, 3, and so on. Petya also also dabbles in math and knows that among consecutive N integers there will always be one that is divisible by N. But for the digit sums of consecutive integers, this is not true. Thus, for example, among the numbers between 92 and 108 none has the sum of digits divisible for 10.

Petya is considering a number K and wants to find out the maximum number of consecutive integers, for which the sum of digits is not divisible for K. Help Petya to find the answer to this question.

Input
A single natural number K ($1 < K \leq 40$).

Output
Output on the first line the maximum possible number of consecutive integers with sum of digits not divisible by K. Output the first number of such sequence on the second line.

Scoring
Each test is scored independently.

Example

standard input	standard output
10	18 1

C. International Olympiad

Time limit: 1 second
Memory limit: 512 megabytes

Everybody loves olympiads in informatics. And everyone loves to participate in them. There is no unified international informatics olympiad on the planet Tau Ceti. But each country holds its own international olympiad for school students.

Taucetizens consider an olympiad to be international if, in addition to the hosting country, at least one student from another country participates in it. Every host country tries to minimize the costs and, accordingly, to minimize the number of foreign participants. However, in order not to lose the status of an international olympiad, it invites all members of the national team from only one country.

In total on Tau Ceti there are n active participants of the olympiads from m countries. The strength of a participant is determined by their rating on a reputable online programming contest site. No two participants have the same rating. Thus for any two participants we can determine who is stronger.

The hosts can not lose face, so they want the first place to be taken by a member of their team, and a member from the invited country to take the last place. Thus, the host country invites the team from another country if the strongest and weakest participants of the host country are stronger than the strongest and weakest participants of the invited team, respectively. If there are several options, the country with the smallest number of participants is chosen.

Pursuing various goals, the competitors may change their nationality, and thus can participate for a new country.

You need to handle q requests. Each request belongs to one of two types:

1. A country g wants to host an olympiad. You have to output the number of the country it will invite.

2. Participant number x changes their citizenship to country g.

Input

The first line contains three integer numbers n, m, q ($1 \le m \le n \le 1000$, $1 \le q \le 3 \cdot 10^5$) — the total number of participants on the planet, the number of countries and the number of requests, respectively.

The sacond line contains n country numbers describing the participants in the descending order of their ranking.

The following q lines describe the queries. Each query starts with an integer t ($1 \le t \le 2$) — the type of the query, followed by

- For the query of the first type, the number g — the number of the country wanting to host an olympiad;

- For the second query type, the numbers x and g — the number of the participant and their new country, respectively.

It is guaranteed that each country has at least one representative at any given time.

Output

For each query of the first type, print the number of the country that the hosts will invite. If there is more than one possible country, print any of them. If there is none — print -1.

Scoring

A group is solved correctly if each of the inputs it contains is solved correctly. Points are awarded only for correctly solved groups of inputs. Some subtasks may also require that all tests in the sample pass. For such subtasks, the letter S is indicated additionally.

Subtask	Constraints	Points	Required subtasks
1	$1 \le n,\ m,\ q \le 100,\ t = 1$	5	—
2	$1 \le q \le 1000,\ t = 1$	15	1
3	$1 \le q \le 1000$	20	S, 1, 2
4		60	S, 1, 2, 3

Examples

standard input	standard output
12 4 4 1 1 2 2 2 4 1 4 3 2 3 3 1 1 1 2 1 3 1 4	4 3 -1 3
12 4 6 1 1 2 2 2 4 1 4 3 2 3 3 2 12 1 2 9 4 1 1 1 2 1 3 1 4	-1 3 -1 3

D. Mex tree

Time limit: 3 seconds
Memory limit: 512 megabytes

Consider a tree of size n (a connected graph consisting of n vertices and $n-1$ edges). Initially all vertices are in off state. When a vertex is turned on, *mex* of all numbers in neighboring activated vertices is written to it. The *mex* of a list of numbers is defined as the minimum integer non-negative number not contained in the list. For example, *mex* of list $[3, 0, 0, 1]$ is 2, and for the list $[1, 2, 3]$ the mex is 0.

What is the maximal sum of numbers written in the vertices of the tree can be obtained after switching on all the vertices? It is known that the vertices are switched on one by one.

Input

The first line contains an integer n ($1 \le n \le 3 \cdot 10^5$) — the number of vertices in the tree.

The next $n-1$ lines contain pairs of integers u, v ($1 \le u, v \le n$) — the edges of the tree.

Output

Output a single integer, the maximal possible sum of numbers written in the vertices of the tree.

Scoring

For all subtasks, points are given only when all tests of that and required subtasks have passed correctly.

Subtask	Constraints	Points	Required subtasks
1	$n \leq 8$	8	–
2	$n \leq 13$	13	1
3	$n \leq 500$	14	1 – 2
4	$n \leq 5000$	15	1 – 3
5	$n \leq 10^5$	40	1 – 4
6	$n \leq 3 \cdot 10^5$	10	1 – 5

Examples

standard input	standard output
4 2 1 3 2 2 4	3
4 1 2 4 2 1 3	3
5 4 5 1 3 1 4 4 2	3

E. Planet exploration

Time limit: 1 second
Memory limit: 256 megabytes

An expedition of specially constructed rovers was sent to the planet Tau Ceti from Earth to explore a section of the planet's surface. It is represented by a rectangular cell field in which there are no isolated places.

The rovers will be delivered by one landing module. Therefore it is necessary to choose the optimal landing point of the module, that is, a place from which the path length to the most distant point of the studied area is minimal. The rovers can move only vertically and horizontally on the site.

Input

The first line contains numbers N, M ($1 \leq N, M \leq 50$) — the number of rows and columns of the cell grid. Rows and columns are numbered starting from zero. The next N lines each contain M numbers 0 or 1, where 0 means that the cell is passable, and 1 means not.

Output

Print two numbers, the row and column coordinates of the cell where the landing module should be dropped. If there are several such places, print any of them.

Scoring

Points for each test are awarded independently.

Examples

standard input	standard output
5 5 1 1 1 1 1 1 0 0 0 1 1 0 0 0 1 1 0 0 0 1 1 1 1 1 1	2 2
4 5 1 0 0 0 1 1 0 0 0 0 1 1 1 0 0 1 0 0 0 0	2 3

F. Table game

Time limit: 1 second
Memory limit: 256 megabytes

The playing field consists of n horizontal and m vertical rows. At the beginning of the game the chip is in the first cell of the first horizontal row. The goal of the game — move the chip to any cell of the last horizontal row.

Inside each horizontal row the player can freely move a piece, but for moving between adjacent cells the player receives exactly one penalty point.

The cell j of horizontal row i can be reached in one move from any cell of horizontal row $i-1$ that is not further away than r_{ij} cells (horizontally) from it, and in this case c_{ij} penalty points are given for this action.

The value of $r_{ij} = -1$ means that it is impossible to make a move to that cell.

Find the least number of penalty points required to move a piece to the last horizontal row.

Input

The first line contains two integers n and m ($0 < n, m \le 2 \cdot 10^5, n \cdot m \le 2 \cdot 10^6$).

The following $n-1$ lines each contain m integers r_{ij} ($-1 \le r_{ij} \le 10^5$).

The following $n-1$ lines each contain m integers c_{ij} ($0 \le c_{ij} \le 10^9$).

Output

Print a single number— the smallest possible number of penalty points required to reach the game goal.

If it is impossible to reach the last horizontal row, print the number —1.

Scoring

Points for the first subtask are awarded independently.

Points for the second subtask are awarded only if all the tests for this subtask and the first subtask have been successfully completed.

Subtasks	Constraints	Points	Required subtasks
1	$r_{ij} \leq 20$	40	—
2		60	1

Example

standard input	standard output
4 4 2 -1 2 0 1 1 3 -1 1 -1 0 2 3 3 1 1 4 1 3 4 2 2 2 2	4

G. Underwater cavern

Time limit: 2 seconds
Memory limit: 256 megabytes

This is an interactive problem

On the planet Tau Ceti, a group of oceanologists has sent a remote-controlled submarine on a deep-sea exploration mission. After another submersion, the submarine got stuck in an underwater cavern which is a multilevel cell grid maze. After many attempts to recover the submarine, the researchers have concluded that they could not solve that problem themselves. The submarine

is a pinnacle of engineering and one-of-a-kind technical marvel, so scientists beg you to help them to recover it.

The exit from the maze is located in the starting cell, but it is closed. The exit can be opened by finding all the artifacts scattered deep in the cave. The submarine can detect the presence of an artifact in the cell in which it is currently located and send readings from its sensors. Based on this data, you must find a way to retrieve the submarine. It can move between the cells of the maze in 6 directions — North, East, South, West, Up, Down. You can control the submarine by sending commands to it.

The scientists have also found that between two different points in the maze reachable using only horizontal commands there is only one simple path.

It is guaranteed that the total number of cells in the cave Q does not exceed 10000.

The artifact scanner has a limited amount of energy, so it can be used at most Q times.

Interaction Protocol

You start by receiving readings from the submarine's sensors indicating the presence of the walls of the cell in 6 directions. A sensor reading string has the following format: the first word is `WALLS`, then the sequence of 6 space-separate digits — either 1 or 0, meaning the presence or absence of the wall, respectively. The numbers are given in the following order: North, East, South, West, Up and Down.

Example - `WALLS 1 0 0 1 0 1`.

The submarine can accept one of these commands:

- `GO N` - move one cell North
- `GO E` - move one cell East
- `GO S` - move one cell South
- `GO W` - move one cell West
- `GO U` - move one cell Up
- `GO D` - move one cell Down

- `SCAN` - check for the presence of an artifact in the cell, and retrieve it if present
- `ESCAPE` - attempt to leave the cave

To communicate with the submarine, you send one of the commands, and the submarine responds in the following way:

- to the motion commands `GO N`, `GO E`, `GO S`, `GO W`, `GO U`, `GO D` with the sensor readings from the cell that the boat arrives at, in the format described above;
- to the `SCAN` command with the string `PICKED_UP` if an artifact has been found and picked up in the current cell, or `NOT_FOUND` if no artifact has been found;
- to the `ESCAPE` command with the string `SUCCESS` if the exit attempt was successful, and `FAILURE` otherwise.

Having successfully left the cave, the submarine will stop receiving any commands.

If the number of scanning requests is exceeded, the test system will display a `WRONG_ANSWER` verdict. Your solution can get an `IDLENESS_LIMIT_EXCEEDED` verdict if you do not output anything or if you forget to reset the output buffer. If an invalid command is entered, the solution will get the verdict `PRESENTATION_ERROR`.

To reset the output buffer, immediately after outputting the query and the end of line character in the program text, use

- `fflush(stdout)` or `cout.flush()` in C++;
- `System.out.flush()` in Java;
- `flush(output)` in Pascal;
- `stdout.flush()` in Python.

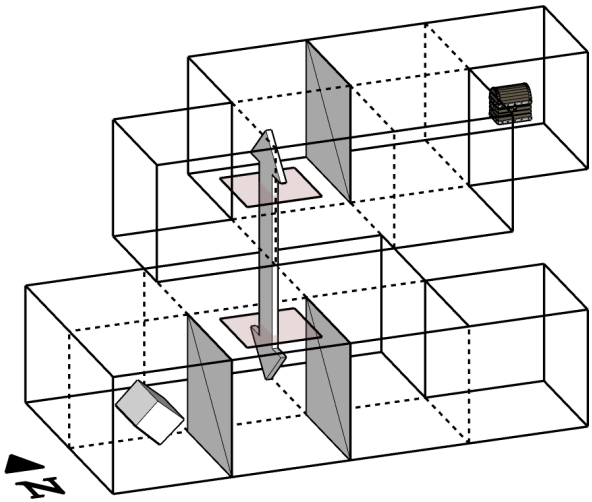
Scoring

Points for each subtask are awarded only if the solution passes all the tests of that subtask and all required subtasks.

Subtask	Constraints	Points	Required subtasks
1	The cave has only 1 level	20	
2	No additional restrictions	80	1

Note

An example of two-level cave:



Communication example:

```
WALLS 0 1 1 1 1 1
GO N
WALLS 1 0 0 1 1 1
GO E
WALLS 1 0 0 0 1 1
GO S
WALLS 0 1 1 1 0 1
GO U
WALLS 1 1 0 1 1 0
GO S
```

```
WALLS 0 0 1 0 1 1
SCAN
NOT_FOUND
GO E
WALLS 0 1 1 0 1 1
GO N
WALLS 1 0 0 1 1 1
GO E
WALLS 1 1 1 0 1 1
```

H. Numbers and sweet buns

Time limit: 2.5 seconds
Memory limit: 256 megabytes

Arthur loves programming very much! Today he came to school to brag to Merlin that he wrote a program that generates a random sequence of natural numbers that has not a single **beautiful** pair. A pair of numbers (a_i, a_j) is called beautiful if there are some integer pair (x, y) that meet the following requirement:

$$a_i x + a_j y = 1.$$

However, Merlin noticed an interesting thing. The prime numbers that **divide** the generated numbers do not exceed p_k, where p_k is the k-th number in the series of primes. Merlin informed about the problem to Arthur, but he didn't believe it and promised him to buy 2 buns for each unique pair of indeces of a beautiful pair of numbers found from the sequence.

Merlin loves buns very much, so he asked you for help. Calculate the maximum number of buns he can get. Output the answer modulo $10^9 + 7$.

Input

The first line contains a single integer n — the size of sequence ($1 \le n \le 3 \cdot 10^5$).

The second line contains n integers $a_1, a_2, \ldots, a_n$ whose prime divisors do not exceed p_k ($2 \le a_i \le 10^{12}$).

Output

Output the number of pairs of integers (a_i, a_j) that meet the given requirements modulo $10^9 + 7$.

Scoring

A group is solved correctly if each of the inputs it contains is solved correctly. Points are awarded only for correctly solved groups of inputs. Some subtasks may also require that all tests in the sample pass. For such subtasks, the letter S is indicated additionally.

Subtask	Constraints	Points	Required subtasks
1	$n \leq 1000, k \leq 21$	9	S
2	$n \leq 10000, k \leq 21$	11	S, 1
3	$k \leq 2$	12	—
4	$k \leq 15$	14	S, 3
5	$k \leq 21$	25	S, 1–4
6	$k \leq 35$	29	S, 1–5

Examples

standard input	standard output
4 46 8 69 3	6
4 3 3 5 5	8

Note

Every beautiful pairs of numbers from sample:

- $(46, 3)$: $1 * 46 + (-15) * 3 = 1$ (1 and 2)
- $(8, 3)$: $2 * 8 + (-5) * 3 = 1$ (2 and 4)
- $(8, 69)$: $26 * 8 + (-3) * 69 = 1$ (2 and 3)

In total, 2 buns were received for each pair. It can be proved that there are no more.

In the second sample, there are 4 pairs:

- $(3, 5)$: $2 * 5 + (-3) * 3 = 1$ (1 and 3, 1 and 4, 2 and 3, 2 and 4)

Note that even though the numbers are the same, the pairs are counted as distinct.

I. Football

Time limit: 0.8 seconds
Memory limit: 256 megabytes

N kids numbered from 1 to N met to play one football game, found you around and asked you to be their referee. You soon understand that there are M pairs of kids who intensely dislike each other. When trying to create the two teams **X** and **Y**, you realize that no matter how you assign each kid to one of **X** and **Y**, there will still be a pair of kids $\langle a, b \rangle$ who dislike each other in the same team, be it **X** or **Y**.

In order to arrange the two football teams so that there is no mutual dislike inside any of the two teams, you plan to pick one of the M pairs and reconcile the two kids so that there is no more dislike between the two kids in the pair. Which are the pairs $\langle a, b \rangle$ such that after removing the dislike between a and b (and between b and a), you could create the two teams such that there is no dislike between two members of the same team?

Input

The first line contains number T of tests ($1 \leq T \leq 100$). Each of the T int the subsequent lines, has the following format:

- first line contains numbers N and M, separated by a space;
- next M lines each contain two numbers a, b ($1 \leq a, b \leq N$), separated by a space, describing a relationship of mutual dislike between kids number a and b.

The pairs are considered to be numbered from 0 to $M - 1$.

Output

Print answers for each of the test T test consecutively, with each answer consisting of one or two lines as follows:

- first line of each description contains the number of values on the second line or 0 if the second line would have been empty;

- second line contains, in ascending order, all numbers i from 0 to $M - 1$ such that, if we were to remove the pair number i, the remaining relationships of dislike would be such that you could create two teams with all N kids such that there is no dislike between two members of the same team. If this second line would be empty, just omit it.

Scoring

Let Q be sum of the T numbers M in the input. Let G be the undirected graph of N vertices and M edges described by the relationships of mutual dislike.

For all tests:

- $3 \leq N \leq M \leq 50\,000$;
- $3 \leq Q \leq 150\,000$;
- G is connected.

For all subtasks, points are given only when all tests of that and required subtasks have passed correctly.

Subtask	Constraints	Points	Required subtasks
1	$Q \leq 500$	20	—
2	G contains a single cycle	10	—
3	$Q \leq 30\,000$	40	1
4	No other constraints	30	1, 2, 3

Example

standard input	standard output
3 3 3 1 2 2 3 3 1 4 6 1 2 1 3 1 4 2 3 2 4 3 4 7 8 1 2 2 3 3 4 4 5 5 6 6 7 6 3 7 1	3 0 1 2 0 4 0 1 5 7

Solutions

A. Granopodus

Let r be the maximum number of fish that can be fried in k minutes. Consider two cases:

1. The frying pan can fit n fish. Then the number r is equal to the number of fish with faces not exceeding the number k.

2. The number n is greater than m. In this case, consider the number $z = m \times k$. Sort the fish in ascending order of the number of faces. Let t_i be the number of fish in the pile with number i. Find sequentially in ascending order i the sums $t_i \times i$ and t_i, denoting these sums by p_i and q_i, respectively.

   Consider the differences $z - p_i$. If these differences are non-negative even when we consider the last pile, then $r = n$. Suppose $z - p_i$ becomes negative or equal to zero when the last heap has not yet been considered. If the difference is zero, then r equals q_i. Otherwise, we return to the previous p_i and q_i. Let us fix these p_i and q_i. To emphasize this, we denote these quantities by p and q respectively. Let us now sequentially add i to p (we denote the number of addition operations by f), and one to q until $z - (p + f \times i)$ is less than or equal to zero. If it is equal to zero, then $r = q + f$, but if it is less than zero, then $r = q + f - 1$.

Another way to look at this problem is to prove that if the sum of faces does not exceed mk and the number of faces of each fish does not exceed k, then they can be cooked in time.

Consider pairs (p, t) denoting the period from $(t - 1)$th to tth minute at position p on the frying pan. Sort them lexicographically and give these pairs to each fish consecutively. That is, the pairs $(1, 1), (1, 2), \ldots, (1, g_1)$ will go to the first fish with g_1 faces, $(1, g_1 + 1), \ldots, (1, k), (2, 1), \ldots$ will go to the second fish. Distribution is impossible only when some fish is at different positions at the same time. But since the number of faces of any fish does not exceed k, this cannot happen in our distribution. Hence, we must choose the fish in increasing order of the number of faces.

B. Sum of digits

1. If the series of such numbers is completely in the same group, you can find it this way. Consider the residues from dividing by K all numbers from 0 to 99999. Find the largest row of consecutive numbers that do NOT contain any residue R. Suppose there are L such numbers. Then before each number of this series, add a prefix with the sum of digits equal to $K - R$. No number in this series has a sum of digits divisible by K.

2. If series we seek begins in one group and ends in the next group, then by varying the number of nines in the barrier, as well as different prefixes (digits preceding the nines of the barrier) it is possible to find this series with a little exhaustive search. Note that it is sufficient to consider prefixes that have different remainders modulo K. The number of nines in the barrier can also be limited to K.

   Note. An analysis of all the resulting solutions showed that the prefix can be taken to be zero everywhere.

C. International Olympiad

To process the queries for optimal country search queries in the first subtask, run through all the countries. For each country, search for all participants, the number of participants, the strongest and the weakest member of the team. Among all matching countries, find the country with the minimum number of participants.

To solve the second subtask, precalculate the answers for each country. Answer the query by printing the saved answer.

To solve the third subtask, we will put the indices of each country's participants into a `set`. When a participant changes citizenship, we will remove their index from the old set and add it to the new one. In this way we can answer queries of the second type in $O(\log n)$. Also, we can check in a constant amount of time whether the next candidate is suitable. Thus we can answer a query of the first type in $O(n)$.

There is a way to answer a query of the first type in $O(\log^2 n)$. Denote by x_i and y_i the indices of the strongest and weakest participant from country i ($1 \le x_i, y_i \le n$). Then country i can invite country j if $x_i < x_j$ and $y_i < y_j$. Plot m

points corresponding to each country with coordinates (x_i, y_i) in a Cartesian coordinate system. The value at the point will be the number of participants in the corresponding country. Then the optimal candidate for country i is the point with the minimum value in the rectangular area $(x_i < x \le n,\ y_i < y \le n)$. To find this point efficiently, we can build a *2D segment tree* on the region $(1 \le x \le n,\ 1 \le y \le n)$, which will allow us to answer both types of queries for $O(log^2\ n)$. The final asymptotic is $O(q \cdot \log^2 n)$.

D. Mex tree

For $O(n!)$ it is enough to enumerate all possible ways to switch the vertices on.

For $O(3^n)$ let us iterate over all possible correct ways to choose vertices with value 0. A subset is considered correct if no two vertices from the subset are neighbors, and every vertex not from the subset is a neighbor of some vertex from the subset. In this case, the sum for the current set is equal to the answer for the forest consisting of unselected vertices, plus the size of that forest.

Note that the values in the vertices do not exceed $\log_2(n)$.

Let us calculate the answer using dynamic programming. Root the tree at some vertex. In $dp[v][up][cur]$ we will store the maximum sum for a subtree with root v, ancestor value up, and value in vertex cur. To calculate the answer, we need to have neighbors with values from 0 to $cur - 1$. To do this, we will calculate the auxiliary dynamics over subsets $subDP[bitmask]$, where the ones indicate that a given value is taken. The value in the child does not exceed $\log_2(size_u) + 1$, where $size_u$ — the size of the subtree u. If we order the children by size and for each child update $subDP$ in $O(size_u)$, we get a solution for $\widetilde{O}(n^2)$ (omitting logarithms). We can ignore the largest child in $subDP$ and just go through its value, then we get a solution working for $O(n \log^3 n)$ with a small enough constant.

E. Planet exploration

1. For each cell with value 0 do the following:
   (a) Using BFS find the minimum distances to each zero cell from the selected cell

(b) Select the maximum value from the obtained distances and store it for the selected cell

2. Select the minimum value from the obtained values of the maximum distances and display its coordinates

F. Table game

The problem is solved by dynamic programming. Let $dp[i][j]$ be the minimal number of penalty points to reach the cell j of row i. In row $i = 0$, $dp[i][j] = j$.

Next, we do the following: first, set $dp[i][j]$ be equal to the minimal $dp[i-1][j']$ among all j' such that $|j' - j| \le r[i][j]$, plus $c[i][j]$. The cells that cannot be reached must have $dp[i][j] = \infty$. Then we go from left to right, updating each $dp[i][j]$ to account for horizontal movement, and walk from right to left in the same way.

For the first subtask, a simple implementation of this algorithm is enough.

For the second subtask, one needs to use a data structure that allows to quickly handle requests of minimum over ranges of numbers (where you need to find the minimal $dp[i][j']$) — such as sparse table, segment tree, Fenwick tree. For each new row, the structure needs to be built anew.

The time complexity of the algorithm is $O(nm \log m)$.

G. Underwater cavern

By reading the statements, you can understand that you can traverse the cavern using simple DFS and should output **SCAN** command on every cell exactly once. The most difficult part is to carefully implement the interaction with the system and store every visited cell not to overflow the scanning limit.

H. Numbers and sweet buns

Formally it is required to find all pairs of coprime numbers.

For the first subtask, it's enough to brute through all pairs and check the coprimality using the Euclidean algorithm, with complexity $\mathcal{O}(n^2 \log(a_i))$.

For the second subtask, we also iterate over all pairs, but now store the numbers as bitmasks, where the ith bit in the mask is 1 if $p_i | a$ (p_i divides a), and 0 otherwise. Complexity of $\mathcal{O}(n^2)$.

For further improvement, we need to use slightly different approach. Note that if all prime divisors do not exceed $p_2 = 3$, then the numbers are of the form $a_i = 2^k * 3^l$. The numbers a_i and a_j are coprime if and only if, without loss of generality, one has the form $a_i = 2^k$, the other $b_j = 3^l$. Let's keep two counters that count the number of such numbers and at the end we will output the product of these counters.

For the fourth subtask, we store the number of numbers with *mask*. Then iterate over just the subpatterns of the inversion. Complexity of $\mathcal{O}(3^k + nk)$. Using sum over subsets with dynamic programming (SOSDP), we can improve it to $\mathcal{O}(nk + k \cdot 2^k)$

Let $a_k, a_{k+1}, \dots, a_l$ is some sequence of numbers and $R(a_i)$ – amount of numbers divisible by a_i. Then according to the inclusion–exclusion principle:

$$R(\neg a_k, \neg a_{k+1}, \dots, \neg a_l) =$$
$$= N - \sum_{k \le i \le l} R(a_i) + \sum_{k \le i < j \le l} R(a_i, a_j) - \dots (-1)^{l-k+1} R(a_k, a_{k+1}, \dots, a_l).$$

In other words, this is the amount of coprime numbers with our sequence $a_k, a_{k+1}, \dots, a_l$. Let's sort all our masks, create such an array R and at each iteration we will recalculate the answer by submasks.

```
std::sort(mask.begin(), mask.end());
uint64_t ans = 0;
uint64_t l = 0;
for(int i = 0; i < n; ++i) {
    l++;
    if (i + 1 < n && mask[i] == mask[i + 1])
      continue;
    uint64_t m = mask[i];
    ans += (i - l + 1) * l;
    for (uint64_t submask = m; submask > 0;
         submask = (submask - 1) & m) {
        if (__builtin_popcountll(submask) % 2 == 1) {
            ans -= R[submask] * l;
        }
        else {
            ans += R[submask] * l;
```

```
17          }
18          R[submask] += l;
19      }
20      l = 0;
21  }
22  std::cout << (ans * 2) % mod;
```

I. Football

G is not bipartite, so in the case of subtask 2 we know *G* has a single **odd** cycle. Removing any of the edges of this cycle makes *G* a tree, which is always bipartite. Removing any of the edges outside the cycle maintains the odd cycle of *G*, which makes *G* still not bipartite. Thus, a solution to subtask 2 is to write up the edges that form the cycle.

For other subtasks, build the Depth-First-Search tree *T* of *G* by starting a DFS from any of the nodes of *G*, say node 1, and note that because *G* is undirected, every edge of *G* either belongs to *T* or is a *back-edge*. For our purposes, it's useful to denote a *back-edge* to be ***black*** when it connects two vertices that have the same height modulo 2 in *T*, and ***white*** when it connects two vertices that have different heights modulo 2 in *T*. We also say a *back-edge* (c, d) traverses an edge (a, b) in *T* if (a, b) is on the path in *T* from *c* to *d*. Then the index of an edge (a, b) should be in the output if and only if one of the below is true:

- it's the only black edge in the tree, or;
- it's an edge of *T* which is traversed by **all** ***black*** edges and **not** traversed by **any** ***white*** edges.

These properties can be checked in $O(1)$ per input edge after an $O(N + M)$ precomputation that includes one DFS, leading to a linear solution in both time and memory.

Задачи

A. Граноподусы

Ограничение по времени: 1 секунда
Ограничение по памяти: 256 мегабайт

На планете Тау Кита водится очень вкусная рыба — граноподус, тело которой имеет форму выпуклого многогранника. Число граней у разных рыб может варьироваться от 4 до 100.

Таукитянин R2D2 выловил n граноподусов и хочет их пожарить. На его сковородке умещается m рыб. Известно, что каждую грань одной рыбы нужно жарить ровно одну минуту. R2D2 аккуратно разложил улов на кучки из рыб с одинаковым количеством граней.

Какое максимальное количество рыб R2D2 сможет приготовить за k минут?

Формат входных данных

Входные данные записаны в трёх строках.

В первой строке — три числа n, m, k ($n, m, k \leqslant 2500$), во второй — числа, означающие количество рыб в кучках. Понятно, что сумма чисел этой строки равна n. И, наконец, в третьей строке содержатся числа, означающие количество граней рыбы из соответствующей кучки второй строки.

Формат выходных данных

Вывести одно искомое число — ответ.

Система оценивания

Баллы за каждый тест начисляются независимо.

Пример входного и выходного файлов

стандартный ввод	стандартный вывод
2 3 4 2 4	2

B. Сумма цифр

Ограничение по времени: 1 секунда
Ограничение по памяти: 256 мегабайт

Любознательный начинающий информатик Петя заметил, что сумма цифр числа изменяется с ростом числа по не совсем понятному закону — иногда она равномерно растёт, а иногда резко уменьшается. Например, начиная с числа 96 она изменяется так: 15, 16, 17, 18, 1, 2, 3, и т. д. Петя немного интересуется математикой, и знает, что среди N подряд идущих натуральных чисел обязательно найдётся число, делящееся на N. А вот про сумму цифр подряд идущих натуральных чисел этого сказать нельзя. Так, например, среди чисел от 92 до 108 нет ни одного, сумма цифр которого делится на 10.

Петя взял для исследований некоторое число K и решил определить, сколько подряд идущих натуральных чисел может иметь сумму цифр, не делящуюся на K. Помогите Пете найти ответ на этот вопрос.

Формат входных данных

Одно натуральное число K ($1 < K \leqslant 40$).

Формат выходных данных

Выведите в первой строке наибольшее возможное количество последовательных натуральных чисел с суммой цифр, не делящейся на K, во второй — первое число последовательности.

Система оценивания

Баллы за каждый тест начисляются независимо.

Пример входного и выходного файлов

стандартный ввод	стандартный вывод
10	18 1

C. Международная олимпиада

Ограничение по времени: 1 секунда
Ограничение по памяти: 512 мегабайт

Все любят олимпиады по информатике. И все любят в них участвовать. На планете Тау Кита нет единой общей международной олимпиады по информатике. Зато каждая страна проводит свою собственную международную олимпиаду школьников.

Таукитяне считают олимпиаду международной, если в ней, помимо страны организатора, участвует хотя бы один школьник из другой страны. Каждая страна-организатор старается минимизировать издержки, то есть минимизировать количество иностранных участников. Однако, чтобы не потерять статус международной олимпиады, она приглашает всех членов сборной команды только из одной страны.

Всего на Тау Кита есть n активных участников олимпиад из m стран. Сила участника определяется рейтингом на авторитетном сайте с соревнованиями по программированию. У каждого участника — свой уникальный рейтинг. Таким образом можно однозначно определить, кто из двух участников сильнее.

Хозяйка олимпиады не может ударить в грязь лицом, поэтому она хочет, чтобы первое место занял участник ее сборной, а последнее место — участник из приглашенной страны. Таким образом, страна-организатор приглашает сборную из другой страны, если сильнейший и слабейший участники из страны-организатора сильнее сильнейшего и слабейшего участника приглашенной сборной соответственно. Если вариантов несколько, из них выбирается страна с наименьшим количеством участников.

Преследуя разные цели, некоторые участники могут поменять гражданство, и, следовательно, могут выступать за новую страну.

Вам требуется обработать q запросов двух типов:

1. Страна g хочет провести олимпиаду. Нужно вывести номер страны, которую она пригласит.

2. Участник под номером x меняет гражданство на страну g.

Формат входных данных

В первой строке заданы три целых числа n, m, q ($1 \leqslant m \leqslant n \leqslant 1000$, $1 \leqslant q \leqslant 3 \cdot 10^5$) — общее количество участников на планете, количество стран и количество запросов соответственно.

В второй строке даны n чисел — номера стран, за который выступают участники, в порядке убывания их рейтинга.

В следующих q строках описаны запросы. Каждый запрос начинается с целого числа t ($1 \leqslant t \leqslant 2$) — типа запроса.

- Для первого запроса далее дано число g — номер страны, проводящей олимпиаду;

- Для второго запроса — числа x и g — номер участника и его новая страна соответственно.

Гарантируется, что в любой момент времени у каждой страны есть хотя бы один представитель.

Формат выходных данных

Для каждого запроса первого типа выведите номер страны, которую пригласит заданная страна. Если таких несколько, выведите любую. Если такой нет — выведите -1.

Система оценивания

Баллы за каждую подзадачу начисляются только в случае, если все тесты для этой подзадачи и необходимых подзадач успешно пройдены. Для некоторых подзадач может также требоваться, чтобы были пройдены все тесты из условия. Для таких подзадач дополнительно указана буква У.

Подзадача	Ограничения	Баллы	Необходимые подзадачи
1	$1 \leqslant n,\ m,\ q \leqslant 100,\ t = 1$	5	—
2	$1 \leqslant q \leqslant 1000,\ t = 1$	15	1
3	$1 \leqslant q \leqslant 1000$	20	У, 1, 2
4		60	У, 1, 2, 3

Примеры входного и выходного файлов

стандартный ввод	стандартный вывод
12 4 4 1 1 2 2 2 4 1 4 3 2 3 3 1 1 1 2 1 3 1 4	4 3 -1 3
12 4 6 1 1 2 2 2 4 1 4 3 2 3 3 2 12 1 2 9 4 1 1 1 2 1 3 1 4	-1 3 -1 3

D. Мекс дерево

Ограничение по времени: 3 секунды
Ограничение по памяти: 512 мегабайт

Дано дерево размера n (связный граф, состоящий из n вершин и $n-1$ ребра). Изначально все вершины находятся в выключенном состоянии. При включении вершины в неё записывается *mex* всех чисел в соседних активированных вершинах. *mex* списка чисел — это минимальное целое неотрицательное число, не содержащееся в списке. Например, *mex* списка $[3, 0, 0, 1]$ равен 2, а списка $[1, 2, 3]$ равен 0.

Какую максимальную сумму чисел, записанных в вершинах дерева, можно получить после включения всех вершин? Известно, что вершины включаются поочередно.

Формат входных данных

В первой строке записано целое число n ($1 \leqslant n \leqslant 3 \cdot 10^5$) — количество вершин в дереве.

Следующие $n-1$ строк содержат пары целых чисел u, v ($1 \leqslant u, v \leqslant n$) — рёбра дерева.

Формат выходных данных

Выведите целое число — максимально возможную сумму чисел, записанных в вершинах дерева.

Система оценивания

Баллы за каждую подзадачу начисляются только в случае, если все тесты для этой подзадачи и необходимых подзадач успешно пройдены.

Подзадача	Ограничения	Баллы	Необходимые подзадачи
1	$n \leqslant 8$	8	—
2	$n \leqslant 13$	13	1
3	$n \leqslant 500$	14	1 – 2
4	$n \leqslant 5000$	15	1 – 3
5	$n \leqslant 10^5$	40	1 – 4
6	$n \leqslant 3 \cdot 10^5$	10	1 – 5

Примеры входного и выходного файлов

стандартный ввод	стандартный вывод
4 2 1 3 2 2 4	3
4 1 2 4 2 1 3	3
5 4 5 1 3 1 4 4 2	3

E. Освоение планеты

Ограничение по времени: 1 секунда
Ограничение по памяти: 256 мегабайт

На планету Тау Кита с Земли отправили экспедицию специально построенных планетоходов для исследования участка поверхности планеты. Этот участок представлен прямоугольным клеточным полем, в котором нет изолированных мест.

Планетоходы доставляются на планету одним посадочным модулем. Необходимо выбрать оптимальную точку посадки модуля, то есть такое место, от которого длина пути до максимально удаленной точки исследуемого участка является минимальной. Планетоходы по участку могут перемещаться только по вертикали и по горизонтали.

Формат входных данных

В первой строке заданы числа N, M ($1 \leqslant N, M \leqslant 50$) — количество строк и столбцов клеточного поля. Строки и столбцы нумеруются, начиная с нуля. В последующих N строках записаны по M чисел 0 или 1, где 0 — клетка проходима, 1 — иначе.

Формат выходных данных

Выведите два числа — координаты (строку и столбец), куда следует высадить модуль. Если таких мест несколько, то выведите любой из них.

Система оценивания

Баллы за каждый тест начисляются независимо.

Примеры входного и выходного файлов

стандартный ввод	стандартный вывод
5 5 1 1 1 1 1 1 0 0 0 1 1 0 0 0 1 1 0 0 0 1 1 1 1 1 1	2 2
4 5 1 0 0 0 1 1 0 0 0 0 1 1 1 0 0 1 0 0 0 0	2 3

F. Настольная игра

Ограничение по времени: 1 секунда
Ограничение по памяти: 256 мегабайт

Игровое прямоугольное клеточное поле состоит из n горизонтальных и m вертикальных рядов. В начале игры фишка находится в первой клетке первого горизонтального ряда. Цель игры — переместить фишку в любую клетку последнего горизонтального ряда.

Внутри каждого горизонтального ряда игрок может свободно передвигать фишку, но за переход между соседними клетками игроку начисляется ровно одно штрафное очко.

В клетку j горизонтального ряда i можно сделать ход из любой клетки предыдущего горизонтального ряда $i-1$, которая расположена не далее, чем на r_{ij} клеток по горизонтали от нее. За такое действие начисляется c_{ij} штрафных очков.

Значение $r_{ij} = -1$ означает, что в данную клетку невозможно сделать ход из предыдущего ряда.

Определите, с каким наименьшим числом штрафных очков можно фишку переместить на последний горизонтальный ряд.

Формат входных данных

В первой строке даются два целых числа n и m ($0 < n, m \leqslant 2 \cdot 10^5$, $n \cdot m \leqslant 2 \cdot 10^6$).

В следующих $n - 1$ строках даны по m целых чисел r_{ij} ($-1 \leqslant r_{ij} \leqslant 10^5$).

В следующих $n - 1$ строках даны по m целых чисел c_{ij} ($0 \leqslant c_{ij} \leqslant 10^9$).

Формат выходных данных

Вывести единственное целое число — наименьшее возможное количество штрафных очков за достижение цели игры.

Если невозможно переместить фишку на последний горизонтальный ряд, то вывести число -1.

Система оценивания

Баллы за первую подзадачу начисляются независимо.

Баллы за вторую подзадачу начисляются только в случае, если все тесты для этой подзадачи и первой подзадачи успешно пройдены.

Подзадача	Ограничения	Баллы	Необходимые подзадачи
1	$r_{ij} \leqslant 20$	40	—
2		60	1

Пример входного и выходного файлов

стандартный ввод	стандартный вывод
4 4 2 -1 2 0 1 1 3 -1 1 -1 0 2 3 3 1 1 4 1 3 4 2 2 2 2	4

G. Подводная пещера

Ограничение по времени: 2 секунды
Ограничение по памяти: 256 мегабайт

Это интерактивная задача

Группа океанологов планеты Тау Кита отправила дистанционно-управляемую подводную лодку в глубоководную исследовательскую миссию. При очередном погружении лодка оказалась в подводной пещере, представляющей собой многоэтажный клеточный лабиринт. После множества тщетных попыток учёные поняли, что сами они не смогут вернуть её. Эта подводная лодка — единственное в своем роде чудо инженерной мысли, поэтому ученые очень просят вас помочь вернуть лодку.

Выход из лабиринта расположен в стартовой клетке, но он закрыт. Его можно открыть, отыскав все артефакты, расположенные в глубине пещеры. Подводная лодка может обнаружить наличие артефакта в той клетке, в которой она находится, и посылать показания своих датчиков. Основываясь на этих данных, вы должны найти способ вернуть подводную лодку. Она может двигаться между клетками лабиринта в 6 направлениях — на Север, Восток, Юг, Запад, Вверх, Вниз. Вы можете управлять подводной лодкой, посылая ей команды.

Ученые также установили, что между разными двумя точками лабиринта, достижимыми с использованием исключительно команд горизонтального перемещения, существует только один простой путь.

Гарантируется, что общее количество клеток в пещере Q не превышает 10000.

Сканер артефактов имеет ограниченное количество энергии, поэтому его можно применить максимум Q раз.

Протокол взаимодействия

Вначале вы получаете показания датчиков подводной лодки, указывающие на наличие стен в 6 направлениях.

Показания датчиков представляют собой строку следующего формата — первое слово `WALLS`, затем последовательность из 6 цифр, разделенных пробелами — либо 1, либо 0. Они показывают наличие или отсутствие стены соответственно. Цифры следуют в следующем порядке, показывая наличие стены в определенном направлении: Север, Восток, Юг, Запад, Вверх и Вниз.

Пример: `WALLS 1 0 0 1 0 1`.
Подводная лодка может принять одну из следующих команд:

- `GO N` — переместиться на одну ячейку на Север
- `GO E` — переместиться на одну ячейку на Восток
- `GO S` — переместиться на одну ячейку на Юг
- `GO W` — переместиться на одну ячейку на Запад
- `GO U` — переместиться на одну ячейку Вверх
- `GO D` — переместиться на одну ячейку Вниз
- `SCAN` — проверить наличие артефакта и, при наличии, подобрать его
- `ESCAPE` — попытаться покинуть пещеру

Общение с подводной лодкой осуществляется следующим образом — вы должны послать одну из команд, в ответ подводная лодка отвечает следующим образом:

- на команды движения `GO N`, `GO E`, `GO S`, `GO W`, `GO U`, `GO D` — отправляются значения датчиков из клетки, в которую лодка переместилась, в ранее описанном формате;
- на команду `SCAN` отправляется `PICKED_UP`, если в текущей клетке артефакт был найден и подобран, или `NOT_FOUND`, если артефакт не был обнаружен;
- на команду `ESCAPE` отправляется `SUCCESS`, если попытка выхода была успешной, иначе будет отправлено `FAILURE`.

Успешно покинув пещеру, подводная лодка перестанет принимать какие-либо команды.

При превышении количества запросов на сканирование системой тестирования будет выведен вердикт `Неверный ответ/WRONG_ANSWER`. Ваше решение может получить вердикт `Решение зависло/IDLENESS_`

`LIMIT_EXCEEDED`, если вы ничего не выведете или забудете сбросить буфер вывода. Если введена некорректная команда, решение получит вердикт `Ошибка презентации/PRESENTATION_ERROR`.

Для того, чтобы сбросить буфер вывода, сразу после вывода запроса и символа конца строки в тексте программы используйте:

- `fflush(stdout)` или `cout.flush()` в C++;
- `System.out.flush()` в Java;
- `flush(output)` в Pascal;
- `stdout.flush()` в Python.

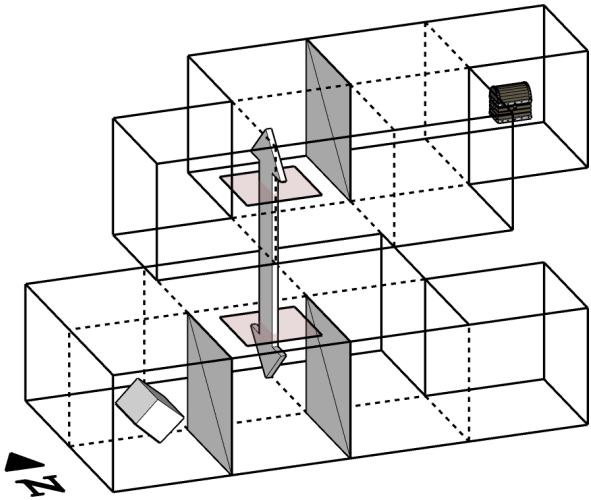
Система оценивания

Баллы за каждую подзадачу начисляются только в случае, если все тесты для этой подзадачи и необходимых подзадач успешно пройдены.

Подзадача	Ограничения	Баллы	Необходимые подзадачи
1	Пещера имеет 1 уровень	20	
2	Дополнительных ограничений нет	80	1

Замечание

Пример двухуровневой пещеры:



Пример коммуникации:

```
WALLS 0 1 1 1 1 1
GO N
WALLS 1 0 0 1 1 1
GO E
WALLS 1 0 0 0 1 1
GO S
WALLS 0 1 1 1 0 1
GO U
WALLS 1 1 0 1 1 0
GO S
WALLS 0 0 1 0 1 1
SCAN
NOT_FOUND
GO E
WALLS 0 1 1 0 1 1
GO N
WALLS 1 0 0 1 1 1
GO E
WALLS 1 1 1 0 1 1
```

H. Числа и булочки

Ограничение по времени: 2.5 секунд
Ограничение по памяти: 256 мегабайт

Артур очень любит программирование! Сегодня он пришел в школу похвастаться Мерлину, что написал программу, которая генерирует совершенно случайную последовательность натуральных чисел, где нет ни одной **красивой** пары. Пара чисел (a_i, a_j) называется красивой, если найдется такая пара целых чисел (x, y), для которых выполнено следующее:

$$a_i x + a_j y = 1.$$

Однако Мерлин заметил интересный факт: простые числа, которые **делят** генерируемые числа, не превосходят p_k, где p_k — это k-е по порядку число в ряду простых чисел. Мерлин сразу же заявил об этом Артуру, но тот не поверил ему и пообещал за каждую уникальную пару индексов красивой пары чисел из последовательности купить ему 2 булочки.

Мерлин очень любит булочки, поэтому обратился за помощью к вам. Посчитайте, какое максимальное количество булочек он может получить. Ответ выведите по модулю $10^9 + 7$.

Формат входных данных

В первой строке задано число n ($1 \leqslant n \leqslant 3 \cdot 10^5$) — длина последовательности.

Во второй строке задана сама последовательность $a_1, a_2, \ldots, a_n$ простые делители которых не превосходят p_k ($2 \leqslant a_i \leqslant 10^{12}$).

Формат выходных данных

Выведите ответ на задачу — количество красивых пар в заданной последовательности по модулю $10^9 + 7$.

Система оценивания

Баллы за каждую подзадачу начисляются только в случае, если все тесты для этой подзадачи и необходимых подзадач успешно пройдены. Для некоторых подзадач может также требоваться, чтобы были пройдены все тесты из условия. Для таких подзадач дополнительно указана буква У.

Подзадача	Ограничения	Баллы	Необходимые подзадачи
1	$n \leqslant 1000, k \leqslant 21$	9	У
2	$n \leqslant 10000, k \leqslant 21$	11	У, 1
3	$k \leqslant 2$	12	—
4	$k \leqslant 15$	14	У, 3
5	$k \leqslant 21$	25	У, 1–4
6	$k \leqslant 35$	29	У, 1–5

Примеры входного и выходного файлов

стандартный ввод	стандартный вывод
4 46 8 69 3	6
4 3 3 5 5	8

Замечание

Все красивые пары из первого примера:

- $(46, 3)$: $1 \cdot 46 + (-15) \cdot 3 = 1$ (1 и 2)
- $(8, 3)$: $2 \cdot 8 + (-5) \cdot 3 = 1$ (2 и 4)
- $(8, 69)$: $26 \cdot 8 + (-3) \cdot 69 = 1$ (2 и 3)

Всего за каждую пару получено по 2 булочки. Можно доказать, что красивых пар больше нет.
Все красивые пары из второго примера:

- $(3, 5)$: $2 \cdot 5 + (-3) \cdot 3 = 1$ (1 и 3, 1 и 4, 2 и 3, 2 и 4)

Обратите внимание, что несмотря на то, что числа одинаковы, пары считаются различными.

I. Футбольные команды

Ограничение по времени: 0.8 секунд
Ограничение по памяти: 256 мегабайт

N детей в футболках с номерами от 1 до *N* собрались играть в футбол и попросили вас быть их судьей. Вскоре вы поняли, что есть *M* пар детей $\langle a, b \rangle$, таких что дети с номерами *a* и *b* испытывают друг к другу неприязнь. Назовем такую пару «нехорошей».

Вы попытались составить две команды **X** и **Y**, но поняли, что независимо от того, как вы назначите каждого ребенка в одну из команд **X** или **Y**, всегда найдется по крайней мере одна «нехорошая» пара детей, попавших в одну команду, будь то **X** или **Y**.

С целью составить две команды так, чтобы ни в одной из них не было взаимной неприязни, вы решили выбрать одну из *M* «нехороших» пар и помирить детей в этой паре. Каковы «нехорошие» пары $\langle a, b \rangle$, обладающие тем свойством, что после примирения детей *a* и *b*, вы сможете составить две команды так, что ни в одной из них не будет «нехороших» пар?

Формат входных данных
Первая строка содержит число T — количество тестовых наборов данных ($1 \leqslant T \leqslant 100$). Каждый из T тестовых наборов записан в последующих строках:

- первая строка содержит числа *N* и *M*, разделенные пробелом;
- каждая из следующих *M* строк содержит разделенные пробелом два числа *a* и *b* ($1 \leqslant a, b \leqslant N$), означающих «нехорошие» пары детей с номерами *a* и *b*.

Подразумевается что сами пары пронумерованы от 0 до $M - 1$..

Формат выходных данных
Выведите ответ на каждый из T тестовых наборов данных в следующем виде:

- первая строка каждого ответа содержит количество значений во второй строке, либо 0, если вторая строка будет пуста;

- вторая строка, если она присутствует, содержит все числа i от 0 до $M - 1$ такие, что после удаления пары с номером i, оставшиеся «нехорошие» пары не помешают создать две команды из N детей, не содержащие без взаимной неприязни внутри команды. Если вторая строка была бы пуста, выводить ее не нужно.

Система оценивания

Пусть Q — сумма T чисел M из входных данных. Пусть G — неориентированный граф из N вершин и M ребер, описывающий отношения неприязни.

Для всех тестов:

- $3 \leqslant N \leqslant M \leqslant 50\,000$;
- $3 \leqslant Q \leqslant 150\,000$;
- G является связным.

Баллы за каждую подзадачу начисляются только в случае, если все тесты для этой подзадачи и необходимых подзадач успешно пройдены.

Подзадача	Ограничения	Баллы	Необходимые подзадачи
1	$Q \leqslant 500$	20	—
2	G содержит один цикл	10	—
3	$Q \leqslant 30\,000$	40	1
4	Дополнительных ограничений нет	30	1, 2, 3

Пример входного и выходного файлов

стандартный ввод	стандартный вывод
3 3 3 1 2 2 3 3 1 4 6 1 2 1 3 1 4 2 3 2 4 3 4 7 8 1 2 2 3 3 4 4 5 5 6 6 7 6 3 7 1	3 0 1 2 0 4 0 1 5 7

Решения

A. Граноподусы

Пусть r — максимальное количество прожаренных рыб. Рассмотрим два случая:

1. Количество рыб n помещается в сковородку. Тогда число r равно числу рыб с гранями, не превышающими числа k.
2. Число n больше m. В этом случае рассмотрим число $z = m \times k$. Отсортируем рыб по возрастанию числа граней. Пусть t_i — число рыб в кучке с номером i. Найдем последовательно в порядке возрастания i суммы $t_i \times i$ и t_i, обозначив эти суммы соответственно через p_i и q_i.
   Рассмотрим разности $z - p_i$. Если эти разности будут неотрицательны даже, когда мы рассмотрим последнюю кучку, то $r = n$. Пусть $z - p_i$ станет отрицательным или равным нулю, когда последняя кучка еще не рассмотрена. Если разность равна нулю, то r равна q_i. В противном случае возвращаемся к предыдущим p_i и q_i. Зафиксируем эти p_i и q_i. Чтобы подчеркнуть это, обозначим их соответственно через p и q. Будем теперь к p последовательно прибавлять i (количество операций сложения обозначим за f), а к q — единицу до тех пор, пока $z - (p + f \times i)$ не станет меньше нуля или равно нулю. В случае, если оно равно нулю, $r = q + f$, если же оно меньше нуля, то $r = q + f - 1$.

Альтернативное решение. Докажем, что если сумма граней не превосходит $m \cdot k$ и количество граней каждой рыбы не превосходит k, то их можно успеть приготовить.

Рассмотрим пары (p, t) — период с $t - 1$ по t минуту на позиции p. Отсортируем их лексикографически и будем выдавать эти пары каждой рыбе по порядку. Т.е. пары $(1, 1), (1, 2), \ldots, (1, g_1)$ отойдут первой рыбе с g_1 гранями, $(1, g_1 + 1), \ldots, (1, k), (2, 1), \ldots$ отойдут второй рыбе. Распределение невозможно только в том случае, когда рыба находится на разных позициях в один и тот же момент времени. Но так как количество граней не превосходит k, такого не может быть при нашем распределении. Отсюда видно, что надо выбирать рыбы по возрастанию количества граней.

B. Сумма цифр

Заметим, что в каждой группе из 100 000 чисел, которая начинается с числа ... 00000 и оканчивается числом ... 99999, есть все числа с суммой цифр от $X + 0$ до $X + 45$, где X – сумма впереди идущих цифр, т.е. есть число, сумма цифр которого делится на K. Значит, искомый ряд чисел либо полностью лежит в пределах одной такой группы, либо начинается в одной группе, переходит барьер (число, оканчивающаяся пятью девятками) и заканчивается во второй группе. Оба эти случая и надо рассмотреть.

1. Если искомый ряд таких чисел полностью лежит в одной группе, то найти его можно так. Рассмотрим остатки от деления на K всех чисел от 0 до 99999. Найдём наибольший ряд идущих подряд чисел, среди которых НЕ встречается какой-либо остаток R. Пусть таких чисел L. Тогда допишем перед каждым числом этого ряда префикс, сумма цифр которого равна $K - R$. Ни одно число полученного ряда не будет иметь сумму цифр, делящуюся на K.

2. Если искомый ряд начинается в одной группе, а заканчивается во второй, то, варьируя количество девяток в барьере, а также разные префиксы (цифры, предшествующие девяткам барьера) можно небольшим перебором найти этот ряд. Заметим, что достаточно рассмотреть префиксы, дающие различные остатки по модулю K. Количество девяток в барьере тоже можно ограничить числом K.

   Замечание. Анализ всех получившихся решений показал, что префикс везде можно взять нулевым.

C. Международная олимпиада

Чтобы обработать запросы поиска оптимальной страны в первой подзадаче, переберем все страны. Для каждой страны переберем всех участников, найдем количество участников, сильнейшего и слабейшего члена сборной. Среди всех подходящих стран найдем страну с минимальным количеством участников.

Для решения второй подзадачи предподсчитаем ответы на запросы для каждой страны. Отвечая на запрос, будем выводить сохраненный

ответ.

Для прохождения третьей подзадачи поместим индексы участников каждой страны в `set`. При смене участником гражданства будем удалять его индекс из старого сета и добавлять в новый. Таким образом, мы научились отвечать на запросы второго типа за $O(\log n)$. Так можно за константное время проверить, подходит ли очередной кандидат. На запрос первого типа мы отвечаем за $O(n)$.

Научимся отвечать на запрос первого типа за $O(\log^2 n)$. Обозначим через x_i и y_i индексы сильнейшего и слабейшего участника из страны i ($1 \leqslant x_i,\ y_i \leqslant n$). Тогда страна i может пригласить страну j, если $x_i < x_j$ и $y_i < y_j$. Построим декартову систему координат и обозначим на ней m точек, соответствующих каждой стране, с координатами (x_i, y_i). Значением в точке будет количество участников в соответствующей стране. Тогда оптимальный кандидат для страны i — это точка с минимальным значением в прямоугольной области ($x_i < x \leqslant n,\ y_i < y \leqslant n$). Чтобы эффективно находить эту точку, построим *двумерное дерево отрезков* на области ($1 \leqslant x \leqslant n,\ 1 \leqslant y \leqslant n$). Таким образом можно отвечать на запросы изменения и минимума за $O(\log^2 n)$. Итоговая асимптотика — $O(q \cdot \log^2 n)$.

D. Мекс дерево

Для $O(n!)$ достаточно перебрать все возможные способы включить вершины.

Для $O(3^n)$ переберем все возможные корректные способы выбрать вершины со значением 0. Подмножество считается корректным, если никакие две вершины из подмножества не являются соседями, и каждая вершина не из подмножества является соседом какой-нибудь вершины из подмножества. В этом случае сумма для текущего множества равна ответу для леса, состоящего из не выбранных вершин, плюс размер этого леса.

Заметим, что значения в вершинах не превышают $\log_2(n)$.

Вычислим ответ с помощью динамического программирования. Подвесим дерево за какую-нибудь вершину. В $dp[v][up][cur]$ будем хранить максимальную сумму для поддерева с корнем v, значением предка — up, значением в вершине — cur. Чтобы посчитать ответ, нам нужно, чтобы были соседи со значениями от 0 до $cur-1$. Для этого мы посчитаем вспо-

могательную динамику по подмножествам $subDP[mask]$, где единичные биты обозначают, что данное значение взято. Значение в ребенке не превышает $\log_2(size_u)+1$, где $size_u$ — размер поддерева u. Если упорядочить детей по размеру и для каждого ребенка обновить $subDP$ за $O(size_u)$, то мы получим решение за $\widetilde{O}(n^2)$ (опуская логарифмы). Можно не учитывать самого большого ребенка в $subDP$, а просто перебирать его значение, тогда мы получим решение, работающее за $O(n \log^3 n)$ с достаточно маленькой константой.

E. Освоение планеты

1. Для каждой клетки со значением 0 выполнить следующее:
   (a) используя BFS, найти минимальные расстояния до каждой нулевой клетки от рассматриваемой;
   (b) выбрать из полученных расстояний максимальное значение и запомнить его для рассматриваемой клетки.
2. Выбрать из полученных максимальных расстояний минимальное значение и вывести координаты клетки.

F. Настольная игра

Для решения задачи используется метод динамического программирования. Пусть $dp[i][j]$ — минимальное количество штрафных очков для достижения клетки j ряда i. В ряде $i = 0$ $dp[i][j] = j$.

Далее выполняем следующие действия. Сначала $dp[i][j]$ приравняем минимальному значению $dp[i-1][j']$ среди всех j' таких, что $|j'-j| \leqslant r[i][j]$, плюс $c[i][j]$. Клетки, до которых невозможно добраться, должны иметь $dp[i][j] = \infty$. Затем необходимо пройтись слева направо, обновляя $dp[i][j]$ по горизонтальным перемещениям, и так же пройтись справа налево.

Для первой подзадачи простой реализации этого алгоритма вполне хватает.

Для второй подзадачи необходимо использовать структуру данных, позволяющую быстро обрабатывать запросы минимума по диапазонам

чисел (там где надо находить минимальный $dp[i][j']$) — такие, как разреженная таблица, дерево отрезков, дерево Фенвика. Для каждого нового ряда необходимо заново строить структуру.

Временная сложность алгоритма — $O(nm \log m)$.

G. Подводная пещера

Прочитав условие задачи, можно понять, что пройти всю пещеру можно, используя метод обхода в глубину (DFS). При обходе требуется выводить команду `SCAN` для каждой клетки ровно один раз. Самое сложное — это аккуратно реализовать взаимодействие с системой и помечать каждую посещенную клетку, чтобы не выйти за пределы лимита сканирования.

H. Числа и булочки

Формально требуется найти все пары взаимно простых чисел.

Для первой подзадачи достаточно перебрать все пары и проверять взаимную простоту через алгоритм Евклида. Сложность алгоритма — $\mathcal{O}(n^2 \log(a_i))$.

Для второй подзадачи тоже перебираем все пары, но уже храним числа как битовые маски, где i-й бит в маске равен 1, если $p_i|a$, иначе 0. Сложность этого варианта — $\mathcal{O}(n^2)$.

Далее мы отказываемся от перебора. Заметим, что если все простые делители не превосходят $p_2 = 3$, то числа имеют вид $a_i = 2^k \cdot 3^l$. Числа a_i и a_j взаимно просты тогда и только тогда когда, без ограничения общности, один имеет вид $a_i = 2^k$, другой $b_j = 3^l$. Заведем два счетчика, которые считают количество таких чисел, и в конце выведем произведение этих счетчиков.

Для решения четвертой и пятой подзадачи храним количество чисел с *mask*. Чтобы определить количество взаимно простых чисел с *mask*, можно его инвертировать и далее работать с ним. Сложность $\mathcal{O}(3^k + nk)$. Можно применить подсчет сумм по подмножествам при помощи динамического программирования (SOSDP) и таким образом улучшить асимптотику до $\mathcal{O}(nk + k \cdot 2^k)$.

Пусть $a_k, a_{k+1}, \ldots, a_l$ — какой-то набор чисел и $R(a_i)$ — количество чисел, делящихся на a_i. Тогда по формуле включений-исключений верно

следующее:

$$R(\neg a_k, \neg a_{k+1}, \dots, \neg a_l) =$$
$$= N - \sum_{k \leqslant i \leqslant l} R(a_i) + \sum_{k \leqslant i < j \leqslant l} R(a_i, a_j) - \dots (-1)^{l-k+1} R(a_k, a_{k+1}, \dots, a_l).$$

Другими словами, это количество чисел, взаимно простых с нашим набором $a_k, a_{k+1}, \dots, a_l$. Отсортируем все наши маски, заведем такой массив R, и на каждой итерации будем пересчитывать ответ по подмаскам.

```
std::sort(mask.begin(), mask.end());
uint64_t ans = 0;
uint64_t l = 0;
for(int i = 0; i < n; ++i) {
    l++;
    if (i + 1 < n && mask[i] == mask[i + 1])
      continue;
    uint64_t m = mask[i];
    ans += (i - l + 1) * l;
    for (uint64_t submask = m; submask > 0;
         submask = (submask - 1) & m) {
        if (__builtin_popcountll(submask) % 2 == 1) {
            ans -= R[submask] * l;
        }
        else {
            ans += R[submask] * l;
        }
        R[submask] += l;
    }
    l = 0;
}
std::cout << (ans * 2) % mod;
```

I. Футбольные команды

Согласно условиям задачи, граф G, описывающий взаимоотношения детей, не является двудольным.

Рассмотрим подзадачу 2: поскольку граф G не двудольный и в нем присутствует только один цикл (согласно ограничениям подзадачи), то получается, что этот цикл состоит из нечётного числа рёбер. Удаление любого ребра из этого цикла превращает граф G в дерево и, следовательно, в двудольный граф. Удаление других ребер не разрывает нечётный цикл, и граф остается не двудольным. Таким образом, для решения подзадачи 2 нужно вывести все рёбра единственного цикла.

Для решения других подзадач нужно на базе графа G с помощью поиска в глубину (Depth-First Search, DFS) построить дерево T. В качестве корня дерева можно выбрать любую вершину (например, 1). При этом каждое ребро в G либо принадлежит дереву T, либо является *замыкающим цикл ребром (back-edge)*.

Для каждой вершины в дереве нужно отметить расстояние (количество рёбер) от корня дерева до неё (то есть высоту вершины). Также отметим *замыкающие цикл рёбра (back-edge)* как «черные» в случае, если высоты обеих соединяемых им вершин имеют одинаковые чётности, либо как «белые» в противном случае. Таким образом, получится, что «черные» рёбра образуют циклы с нечётным количеством рёбер, а «белые» — с чётным.

Будем говорить что *замыкающее цикл ребро* (c, d) обходит другое ребро (a, b) в дереве T, если (a, b) входит в путь от c до d (поскольку c и d входят в дерево, то такой путь только один).

После этого становится понятным, что ответом на задачу, являются:

- «черное» ребро в графе, если оно одно;
- ребро в дереве T, которое обходится всеми «черными» рёбрами и ни одним «белым» ребром (иными словами, это ребро входит во все чётные циклы, но не входит ни в один нечётный цикл).

Все эти свойства могут быть проверены алгоритмом со сложностью $O(1)$ на каждое ребро после предварительных вычислений со сложностью $O(N + M)$, которые включают в себя обход графа поиском в глубину (DFS). Таким образом, можно получить алгоритм с линейной сложностью решения по времени и памяти.

№ #	Имя, фамилия *Name*	Страна *Country*	Команда *Team*	1-й день / *1st Day*				2-й день / 2nd Day					ИТОГО *TOTAL*	НАГРАДА *AWARD*
				A	B	C	D	E	F	G	H	I		
1	Artem Kutuzov	Russia	Specialiazed Educational and Scientific Centre Of Ural Federal University named after B. N. Yeltsin	100	100	100	100	100	100	100	71	100	**871**	**ABS, HML1**
2	Alexander Reznichenko	Russia		100	100	100	8	100	100	100	100	100	**808**	**1DG, HML2**
3	Yu Zheyuan	Singapore	NUS High School of Math and Science	100	100	100	35	100	100	100	71	100	**806**	**1DG**
4	Alexandru Luchianov	Romania	Liceul Teoretic International de Informatica Bucuresti	100	100	100	0	100	100	100	100	100	**800**	**1DG,**
5	Rares Felix Tudose	Romania		100	100	100	35	100	100	100	100	60	**795**	**1DG**
6	Andrei Robert Ion	Romania		87	100	100	21	100	100	100	71	100	**779**	**1DG**
7	Altair Ashurov	Kazakhstan	Kazakhstan National team	63	73	100		100	100	100	71	100	**707**	**2DG**
8	Artem Abaturov	Russia	Specialiazed Educational and Scientific Centre Of Ural Federal University named after B. N. Yeltsin	100	100	100	21	100	100	100	46	30	**697**	**2DG**
9	Tudor Stefan Magirescu	Romania	Liceul Teoretic International de Informatica Bucuresti	100	85	100	8	100	100	100	71	30	**694**	**2DG**
10	Matthew Allan	Indonesia	Wardaya College	100	100	100	21	100	100	100	71	0	**692**	**2DG**
11	Alexandru Peticaru	Romania	Liceul Teoretic International de Informatica Bucuresti	100	100	100	8	100	100	0	71	100	**679**	**2DG**
12	Ramazan Perdebai	Kazakhstan	Karagandy region	63	85	100	21	100	100	100	71	30	**670**	**2DG**
13	Matvey Goncharov	Kazakhstan	Pavlodar region	100	100	100	8	100	100	100	46	0	**654**	**2DG**
14	Vasiliy Parnikov	Russia	I National team of the Republic of Sakha (Yakutia)	100	100	100	8	100	100	100	21	20	**649**	**2DG, SP**
15	Ruslan Ibodullaev	Kazakhstan	Akmola region	100	65	100	21	100	100	100	32	20	**638**	**3DG**
16	Temirlan Zharylgamyssov	Kazakhstan	city of Nur-Sultan	63	85	100	21	100	100	100	32	30	**631**	**3DG**
17	Mansur Taimas	Kazakhstan	Atyrau region	100	65	40	21	100	100	100	71	30	**627**	**3DG**
18	Timur Luzgov	Russia	Specialized Educational and Scientific Center IT Lyceum of Kazan Federal University	87	65	100	8	100	100	100	35		**595**	**3DG, SPO**

19	Egor Golikov	Russia	Specialiazed Educational and Scientific Centre Of Ural Federal University named after B. N. Yeltsin	100	100	40	0	100	100	100	32	20	**592**	**3DG**
20	Arslan Galyaviyev	Russia	Specialized Educational and Scientific Center IT Lyceum of Kazan Federal University	76	100	100	8	100	100	0	32	20	**536**	**3DG**
21	Nursultan Yerlanuly	Kazakhstan	NIS PhM Taldykorgan	100	65	40	8	100	100	0	71	30	**514**	**3DG**
22	Arman Issayev	Kazakhstan	city of Nur-Sultan	100	55	40	21	100	100		32	30	**478**	**3DG**
23	Dastan Mukhametkaliyev	Kazakhstan	Specialiazed lyceum №39 named after S.A.Hodzhikov	100	65	0		100	100	100	9		**474**	**3DG**
24	Bulat Khayev	Russia	Specialized Educational and Scientific Center IT Lyceum of Kazan Federal University	61	64	100	8	100	10	100	0	30	**473**	**3DG**
25	Kanysh Mukhametkarim	Kazakhstan	VKO	63	65	100	8	100	100	0	0	30	**466**	**3DG**
26	Nathan K Poernama	Indonesia	Wardaya College	63	65	40	21	100	100	20	21	30	**460**	**3DG**
27	Karim Suleymanov	Russia	Specialized Educational and Scientific Center IT Lyceum of Kazan Federal University	100	68	40		100	24	100	9	10	**451**	**3DG**
28	Amir Kigash	Kazakhstan	city of Nur-Sultan	100	55	40	8	100	15	100	32	0	**450**	**3DG**
29	Yerzhan Amangeldin	Kazakhstan		100	70	40	8	100	100		21	0	**439**	**HMLP**
30	Makhambet Turgali	Kazakhstan	Atyrau region	63	65	40	8	100	100		32	30	**438**	**HMLP**
31	Dinmukhamed Oralkhanov	Kazakhstan	VKO	63	71	100	8	40	100	0	32	20	**434**	**HMLP**
32	Yegor Trussov	Kazakhstan	Pavlodar region	63	47	40	8	100	100	0		20	**378**	
33	Nurislam Syrgabaev	Kazakhstan	Kyzylorda region	63	63	40	8	100	31		32	20	**357**	
34	Zhantore Galymzhan	Kazakhstan	NIS ChB Aktau	75	65	100	0	100	4		9		**353**	
35	Bolashak Kulmukhambetov	Kazakhstan	Aqtobe region	12	57	40	0	100	3	100			**312**	
36	Arsen Schumilov	Russia	II National team of the Republic of Sakha (Yakutia)	48	65	20		95	19		20		**267**	**SP**
37	Georgiy Karnaukhov	Russia	Specialized Educational and Scientific Center IT Lyceum of Kazan Federal University	63	15	40	8	100		0	21	20	**267**	

38	Bogdan Shakaryan	Kazakhstan	Kostanay region	51	65	40		100	4				**260**	
39	Nikifor Pavlov	Russia	I National team of the Republic of Sakha (Yakutia)	50	65	40	0	80			20	0	**255**	
40	Vladislav Pavlovskiy	Russia		63	65	0		80			9		**217**	
41	Denis Makarov	Russia	II National team of the Republic of Sakha (Yakutia)	37	61	20		75	3		21		**217**	
42	Damir Mindaev	Russia		25	65	20		75			9	20	**214**	
43	Andreas Surya Tanjung	Indonesia	Wardaya College	63	33	0		80	26		9	0	**211**	
44	David-Andrei Stancu	Romania	Liceul Teoretic Național	51	65	0		75	6		0		**197**	
45	Oleg Chabykin	Russia	II National team of the Republic of Sakha (Yakutia)	48		20		100			9	20	**197**	
46	Artem Viktorov	Russia	State budget non-standard educational institution of the Penza region "Provincial Lyceum"	63		0		100	26	0			**189**	
47	Aital Tuprin	Russia	II National team of the Republic of Sakha (Yakutia)	63	39	0		75	11			0	**188**	
48	Miron Makarov	Russia	I National team of the Republic of Sakha (Yakutia)	63	65	20		35					**183**	
49	Nurgeldi Yessengeldi	Kazakhstan	NIS ChB Aktau	63	33	40		20	19		0		**175**	
50	Muhammad Hamiz Ghani Ayusha	Indonesia	Wardaya College	12	24			70					**106**	
51	Timur Blamykov	Russia	Specialized Educational and Scientific Center IT Lyceum of Kazan Federal University					80					**80**	
52	Assemai Bakhytzhan	Kazakhstan	Kyzylorda region	13	24			30	10		0		**77**	
53	Kauas Assemay	Kazakhstan	Zhambyl region	13	18	0	0	40	4	0	0		**75**	
54	Nikita Malyshev	Russia	Specialized Educational and Scientific Center Engineering Lyceum-Boarding Kazan National Research Technical University named after A.N.					65	0	0			**65**	
55	Ivan Kislitsyn	Russia	Specialized Educational and Scientific Center of the Southern Federal District	37		5			19	0			**61**	

56	Ruslan Nasibyllin	Russia	Specialized Educational and Scientific Center Engineering Lyceum-Boarding Kazan National Research Technical University named after A.N.	0	53	0			0				**53**	
57	Nurassyl Maratov	Kazakhstan	NIS ChB Shymkent	25	25								**50**	
58	David Bakti Lodianto	Indonesia	Wardaya College		37								**37**	
59	Iliya Tkachev	Russia	Specialized Educational and Scientific Center of the Southern Federal District		32								**32**	
60	Alikhan Aralbek	Kazakhstan	NIS ChB Aktau	0	24				0				**24**	
61	Aliya Zairova	Kazakhstan	Almaty region				0						**0**	

Jury chairwoman: N. Nikolaeva

Jury secretary: V. Obudov 06.07.2022

НАГРАДЫ / *AWARDS :*

1DG — диплом 1 степени / *1st degree diploma,*

2DG — диплом 2 степени / *2nd degree diploma,*

3DG — диплом 3 степени / *3rd degree diploma,*

HML1 — почетная грамота за полное решение задач 1 дня / *Honourable mention letter "For the perfect result of Day 1",*

HML2 — почетная грамота за полное решение задач 2 дня / *Honourable mention letter "For the perfect result of Day 2",*

HMLP — почетная грамота за многообещающие результаты / *Honourable mention letter "For the promising results",*

SP — спецприз Якутского отделения Дальневосточного центра математических исследований / *special prize of the Yakutsk branch of The Far Eastern Center for mathematical research*

SPO — *спецприз за успешное выступление среди очных участников / special prize "For successful performance among on-site participants"*

Contents | Содержание