Problem A. Array

Source file name:	Array.c, Array.cpp, Array.java, Array.py
Input:	`Standard`
Output:	`Standard`

In his backyard, Cimrman had created a breakthrough bionic solar array installment.

The array is made of separate cells arranged in a triangular layout. The layout consists of parallel rows, the first row contains one cell, and then the number of cells in each row is always one more than the number of cells in the immediately preceding row. The distances between adjacent cells in each row are the same and they are the same in all rows, and the distances between the consecutive rows are the same. The layout is also symmetrical along the axis perpendicular to the rows and going through the middle of the cell in the first row.

The cells at both ends of all rows and also the cell in the first row are the so-called elementary cells and they are all labelled by 1.

The labelling scheme for the remaining cells is more intricate. Each of the remaining cells in a particular row is connected to exactly two cells in the previous row. These two cells are the closest ones in the previous row to the given cell. The label of the given cell is equal to the sum of the labels of the cells in the previous row to which it is connected.

The cells are maintained by an automated pentacopter. When a cell needs a maintenance it sends out a request containing its label. The pentacopter has to locate the row of the cell when it is given only the label of the cell.

Input

The first input line contains one integer Q ($1 \leq Q \leq 10^5$), the number of maintenance requests. Each of the next Q lines contains an integer N ($1 \leq N \leq 10^9$), the label of a cell producing the request.

Output

Output Q lines, for each request print the smallest possible number of a row in the array on which the cell requesting a maintenance is located.

Example

Input	Output
5 1 3 66 330 10	1 4 13 12 6

Problem B. Canoes

Source file name:	Canoes.c, Canoes.cpp, Canoes.java, Canoes.py
Input:	`Standard`
Output:	`Standard`

Cimrman had built a whole fleet of special weather prediction molybdenum canoes. Each canoe was separately built in its own dry dock. Consequently, the shore is littered with dug out patterns of dry docks, some of them even intersect each other, some of them are separate.

A dry dock is a rectangle with width of one standard unit and its length is a few standard units, always at least two. Each dock runs either in vertical or horizontal direction. Consequently, each two docks run either in a parallel direction or in directions perpendicular to each other. The width of each canoe is the same as the width of the dock and the length of the canoe is just one unit shorter than the length of its dock.

Next week, a hurricane is expected in the area, and Cimrman wants each canoe to be put back to the dock in which it was created. However, it is not immediately clear whether such universal storage plan can be accomplished.

And, by the way, are there any square shaped canoes? Yes, Cimrman is capable of building square molybdenum canoes.

Input

The shore with docks is modelled as a rectangular grid, the size of its each elementary square is equal to one standard unit. The first input line contains three positive integers H, W, N ($1 \leq H, W \leq 500$, $1 \leq N \leq 250000$), giving the height of the grid, the width of the grid and the number of docks in the grid.

Each of the following N lines specifies one dock. The dock is defined by four entries separated by spaces. The first three entries X, Y, K are integers specifying the coordinates (X, Y) of one end of the dock and the dock length K (number of grid squares occupied by the dock). It holds that $1 \leq X \leq H$, $1 \leq Y \leq W$, and $2 \leq K \leq 500$. The fourth entry on a row is one of characters "`L`", "`R`", "`U`", "`D`" and it specifies in which direction runs the dock from its start coordinate ("`L`" - Left, "`R`" - Right, "`U`" - Up, "`D`" - Down). Moreover, no dock runs out of the bounds of the shore, e.g. for a dock which runs Down with one end on coordinates (X, Y), it additionally holds $1 \leq X + (K - 1) \leq H$ (and analogously for other directions).

Output

Output one line "`Yes`" if all canoes can be stored back in their docks or "`No`" otherwise.

Example

Input	Output
5 3 4 1 1 4 D 3 1 3 R 5 1 2 U 5 3 3 L	Yes
5 5 5 1 1 5 R 1 1 5 D 5 5 5 L 5 5 5 U 5 3 5 U	No

Explanation

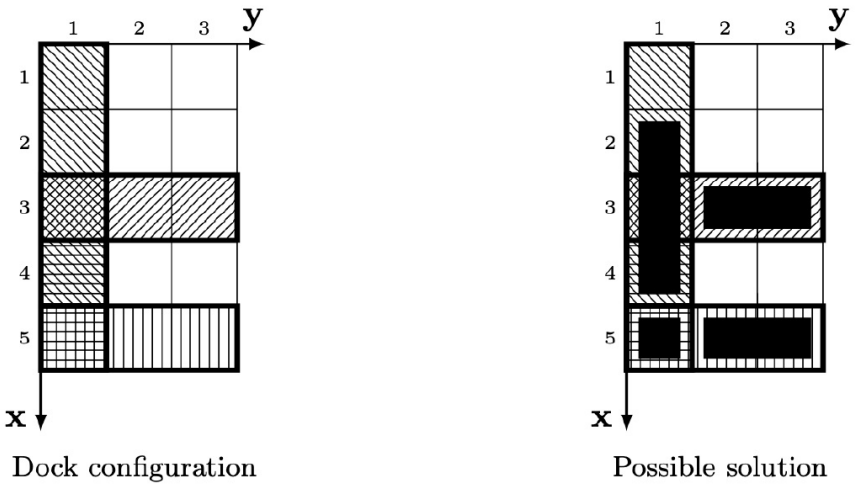

Figure 1: Illustration of Example Input 1.

Problem C. Earthquake

Source file name:	Earthquake.c, Earthquake.cpp, Earthquake.java, Earthquake.py
Input:	`Standard`
Output:	`Standard`

Cimrman has quite a number of collaborators. Tomorrow morning, he is planning to make phone calls to many of them and his assistant had prepared a list of phone numbers on a sheet of paper. Regrettably, an earthquake struck yesterday. The coffee and juice stored close to the list had spilled out and made stains on the list. Upon close examination of the affected list, the following turned out:

- No digit affected by a stain is readable.
- Each coffee stain covers exactly one digit in any of the numbers.
- There are at most two coffee stains on any of the numbers.
- Each juice stain covers one or more consecutive digits in any of the numbers.
- There is at most one juice stain on any of the numbers.
- No stain affects more than one number.
- No number has been affected by coffee and juice simultaneously.

Fortunately, Cimrman has an older list of his collaborators phone numbers. This older list was stored in the drawer and it survived the earthquake undamaged. Now, it will help to restore the damaged list. First, the assistant needs to know, to how many items on the old list may correspond each of the items on the damaged list.

Input

The first input line contains one integer N ($1 \leq N \leq 10^4$), the number of phone numbers in the older undamaged list. Each of the next N lines contains one phone number consisting of exactly 9 digits, leading zeros are allowed. All numbers are mutually different. Next, there is a line with one integer Q ($1 \leq Q \leq 3 \cdot 10^5$), the number of phone numbers on the damaged list. Each of the next Q lines contains one, possibly stained, phone number from the damaged list. A coffee stain is represented by a question mark ("`?`"), a juice stain is represented by an asterisk ("`*`"). There are no spaces in any numbers. The phone numbers are given in no specific order.

Output

Output Q lines, the i-th line should contain the number of items in the undamaged list which may correspond to the i-th item on the damaged list.

Example

Input	Output
2 728147956 606327482 2 72814?956 622629145	1 0
4 606200400 606200500 606300500 706200400 3 ?06200400 6*00 606?00?00	2 3 3

Problem D. Journals

Source file name:	Journals.c, Journals.cpp, Journals.java, Journals.py
Input:	`Standard`
Output:	`Standard`

There is a stack of journals on Cimrman's desk. The journals are printed in Cimrman's own printing house. Each journal front cover is charged slightly positively and the back cover is charged slightly negatively. Different charges of the front and back cover help the journals to stick better to each other when they are stacked in a single stack one upon another. When two adjacent journals in the stack are positioned in such a way that either their front covers or their back covers touch each other, the repulsion between the same charges in both covers makes the stack more prone to collapse.

Cimrman wants his stack to be arranged in such a way that no two adjacent journals in the stack produce the repulsion.

To set the stack into appropriate order he can repeat a single operation consisting of three moves. In the first move (which may be empty), Cimrman puts aside some number of journals from the top of the stack without changing their order. Next, he takes a pile of one or more journals from the new top of the stack, flips this pile over, and puts it back on the stack. Finally, he moves back onto the stack the journals removed from the stack in the first step. Again, no change of the journal order appears in this move.

Obviously, Cimrman wants to perform as few operations as possible. The number of journals moved or flipped in one operation is not important, the journals paper is light enough.

Input

The journals' front and back covers are represented by signs plus or minus ("+" or "-") in the input. The single line of input contains K ($1 \leq K \leq 10^5$) plus signs and K minus signs without spaces between them. The input corresponds to the original orientation of the journals on the stack.

Output

Print the minimal number of operations to be performed to achieve a stack where no two adjacent journals produce the repulsion.

Example

Input	Output
`+-+-+--+-+`	`1`
`+--++-`	`1`

Problem E. Mower

Source file name:	Mower.c, Mower.cpp, Mower.java, Mower.py
Input:	`Standard`
Output:	`Standard`

Cimrman's newest lawn mower can juggle 17 ping-pong paddles and it can also play 2 electric glass violins simultaneously.

To get his invention approved internationally, Cimrman has to play a game against the Vice-Chair of the Patent Office and Cimrman has to win.

The rectangular lawn on the Patent Office field is divided into squares. The lawn is completely unmown. The mower starts at a square selected by the Patent Office and this square is considered to be already mown.

Then players take turns, the first player is Cimrman, the next player is the Vice-Chair. In each turn, the player sends a remote control command to the mower which then moves itself to one of the squares sharing an edge with the last mown square. The mower immediately mows the entire square to which it had been moved and then it awaits a command of the next player's move. In a legal move, a player can send the mower only to one of yet unmown squares. They cannot send the mower either outside the lawn or to any already mown square. The player who cannot make a legal move loses and the other player wins.

Input

The input consists of a single line with four space separated numbers W, H, X, Y ($1 \leq W, H \leq 10^9$, $1 \leq X \leq W$, $1 \leq Y \leq H$). These values describe the width and the height of the Patent Office lawn expressed in the number of squares, and the coordinates of the square where the mower starts.

Output

Output a single line with either `Win` if Cimrman can win the game no matter how well it is played by his opponent, or `Lose` otherwise.

Example

Input	Output
6 1 4 1	Win
4 3 4 2	Win
1 1 1 1	Lose

Problem F. Needle

Source file name:	Needle.c, Needle.cpp, Needle.java, Needle.py
Input:	`Standard`
Output:	`Standard`

Another unexpected invention of Cimrman, a marvel of microminiaturization, is the self-balancing reactive needle.

It is capable of keeping itself upright and move itself freely on a flat surface in this orientation without being supported by any external device. While moving, it can draw extremely thin lines and curves. The trace the needle leaves is in fact only few molecules wide. However, the drawing surface should be extremely clean.

Another needle experiment is in process. The drawing surface contains experimental points which are divided into so-called clouds. The clouds arrangement is specific: Each cloud contains at least three points. The area of the convex hull of each cloud is positive. The intersection of convex hulls of any two clouds is empty.

The needle has to travel between two additional points S and T. None of these two points lies inside the boundary of any cloud. The boundary of the cloud is considered to be the boundary of the convex hull of the cloud.

The needle has to travel from S to T along a shortest possible route. It should not travel inside the boundary of any cloud, the surface there may be unsuitable for the needle movement. On the other hand, the needle can travel along any part of the boundary of any cloud.

Input

The first input line contains five numbers N, S_x, S_y, T_x, T_y, the number of clouds, and the coordinates $(S_x,\ S_y)$ of the start point S and the coordinates $(T_x,\ T_y)$ of the target point T, respectively. Next N lines contain description of cloud points in the following format. The i-th line contains an integer number c_i, the number of points in the i-th cloud. Next on the line, there are $2 \cdot c_i$ numbers $x_{i,j}$, $y_{i,j}$ for $3 \leq j \leq c_i$, the coordinates of the points in the i-th cloud. All coordinates are integers in the range -2000 to 2000. All input points are pairwise distinct. It is guaranteed that $1 \leq N \leq 200$, and also

$$\sum_{i=1}^{N} c_i \leq 500.$$

Output

Output a single number, the length of the needle's shortest path from S to T, avoiding interiors of boundaries of all clouds. Your answer must be precise to 4 decimal digits.

Example

Input	Output
1 0 0 10 0 3 4 2 7 3 5 -4	11.8770543023

Explanation

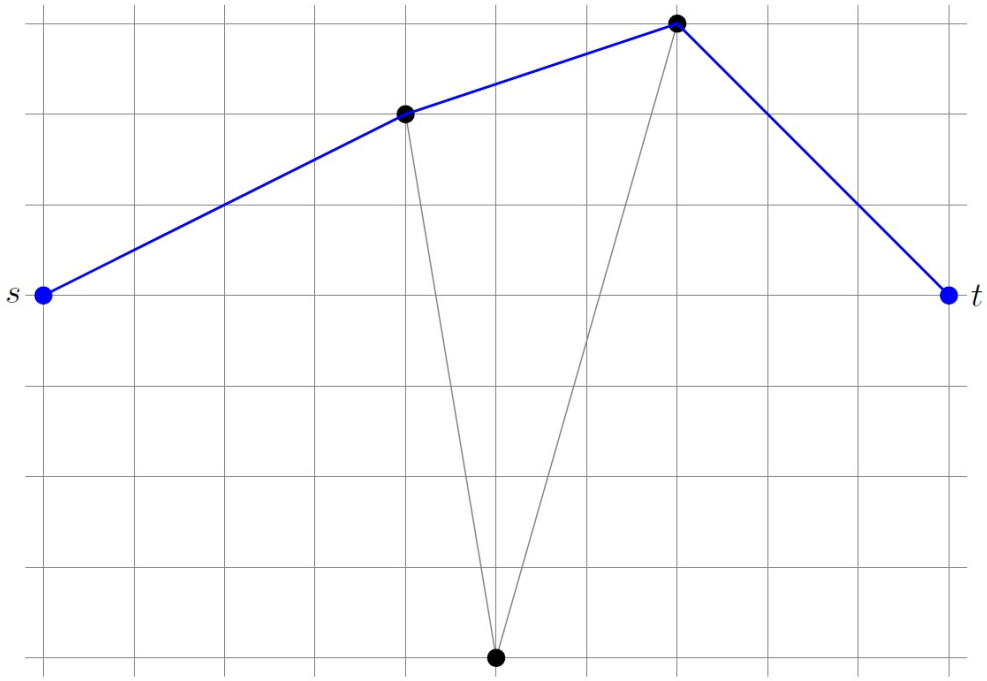

Figure 2: Depiction of Example Input 1 and its optimal solution (in blue/bold).

Problem G. Patio

Source file name:	Patio.c, Patio.cpp, Patio.java, Patio.py
Input:	`Standard`
Output:	`Standard`

Cimrman wants to make a square patio floor using tiles of two colours, red and blue. The patio floor should look like this (with the colors slightly faded in time):

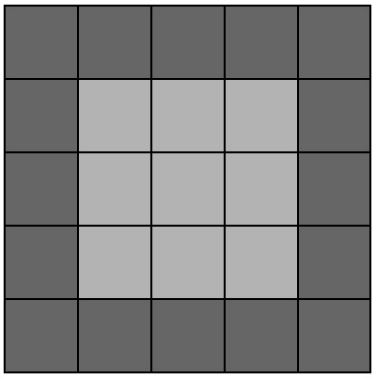

Figure 3: One of Cimrman's perfect patios.

More specifically, the patio must have a square shape. Tiles of one of the colours are used as the border of the square. The border must be exactly one tile thick. The tiles of the other colour are used to fill the rest of the square. Also, the side of the square must consist of at least 3 tiles.

Cimrman has a long file of square red tiles and blue tiles, the size of all tiles is the same. From this file, Cimrman is going to take some tiles to use them on the floor. Manipulating the file is clumsy, so Cimrman wants the tiles to be taken easily from the file, meaning the taken tiles have to form one contiguous subsequence in the file.

Before Cimrman starts the construction, he needs to know how many suitable subsequences of tiles are there in the file.

Input

The input consists of two lines. The first line contains integer N ($1 \leq N \leq 2 \cdot 10^5$), the length of the file of tiles. The second line contains string of N characters, representing the file of tiles. Only two characters appear in the string, "`X`" represents a blue tile and "`O`" represents a red tile.

Output

Output the number of contiguous subsequences in the file from which Cimrman can construct a nice square patio floor.

Example

Input	Output
9 XXXOXXXXX	1
10 XOXXXXXXXX	2

Problem H. Robots

Source file name:	Robots.c, Robots.cpp, Robots.java, Robots.py
Input:	`Standard`
Output:	`Standard`

Cimrman's robot producing company Cimrman Dynamics is testing their new model, **L**ong **R**ange **A**utonomous **T**rekking **B**ot (**LRATB**). **LRATB** has to find its way from village S to village F in the region. For environmental and safety reasons, **LRATB** moves only at night and each night its movement is limited to relocation from one village to some adjacent village along a cycling path between the villages.

Another company, Overseas Dynamics, is testing their similar product in the same area. Their humanoid robot called AtlasTiger also travels only at night and it always relocates itself from a village to an adjacent one in one night.

Cimrman Dynamics programmers have no clue about the planned route of the competing company robot. They suspect that if the two robots stay for the whole day (from dawn till dusk) in one village, the signal interference may damage their own robot navigation systems. Thus, they want to plan the sequence of visited villages in such way that the two robots will never meet in the same village in one day. They know only the location of AtlasTiger on the first day of testing and know nothing about its further movements. Sometimes, if it helps the strategy, **LRATB** may stay overnight in a village without leaving it at all. If the two robots meet briefly at night travelling the same path in opposite directions, it poses no risk to their navigation.

Now, Cimrman Dynamics programmers want to plan the path of **LRATB** from village S to village F so that it takes the shortest possible time and it is guaranteed that no signal interference will occur, no matter how AtlasTiger will organise its own movements. Both robots start their journeys at the dawn of the same day in their corresponding starting villages (and spend the day there by charging batteries etc).

Input

The first line of input contains five integers N, M, F, T, S ($1 \leq N \leq 10^5$, $0 \leq M \leq 2 \cdot 10^5$, $0 \leq F \leq N-1$, $0 \leq T \leq N-1$, $0 \leq S \leq N-1$). N is the number of villages, M is the number of paths between pairs of villages, F is the label of the destination village, T is the label of village in which AtlasTiger is originally located, S is the label of village in which **LRATB** is originally located. The villages are labelled 0, 1, ..., $N-1$. Each of the next M lines contain a description of one path between adjacent villages, the description consists of the labels of the villages separated by space. No path connects a village to itself, no two paths connect the same villages.

Output

Output either minimal number of nights for **LRATB** to reach destination or output string death if it is not possible, that is, if there is no path that can guarantee interference-free journey for **LRATB** from S to F.

Example

Input	Output
5 4 0 1 4 0 1 1 2 2 3 3 4	4
5 5 0 1 4 0 1 1 2 1 3 2 3 3 4	death
3 2 0 2 1 2 1 0 1	1
4 3 0 1 3 2 1 2 3 0 1	4

Problem I. Shamans

Source file name:	Shamans.c, Shamans.cpp, Shamans.java, Shamans.py
Input:	`Standard`
Output:	`Standard`

Cimrman has gathered a number of important shamans from local tribes. He wants to conduct an experiment in which he will investigate whether an intense drumming can create resonance effects in the jungle that will suppress growth of unwelcome species. He is going to select a group of shamans most suitable for the experiment.

Each of the shamans who will take part in the experiment must get a piece of parchment made of a sacred ant-eater skin. Shamans demand that each of them must get a piece of the same shape and size.

Cimrman had prepared the parchment in such a way that its division can be easily modelled by a computer algorithm. In particular, the parchment consists of square blocks of identical size. Two adjacent blocks always share the full length of their edges. At some places, the parchment can be divided into two pieces by applying a single cut along only one edge between two adjacent blocks. Only these places on the parchment may be used for subsequent cutting.

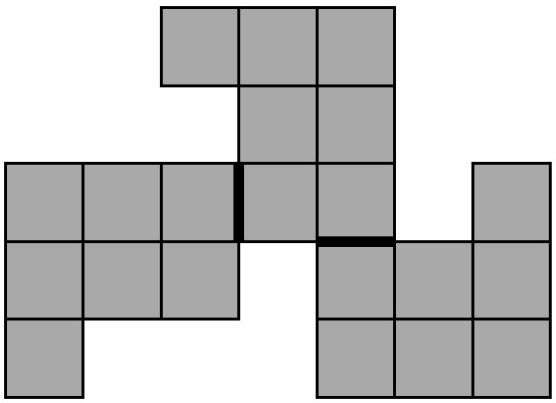

Figure 4: Optimal solution for the first example input. Using the two thick cuts, three identical parchment pieces can be created.

First, the most senior shaman will cut off his piece of parchment using a single cut along the edge of one block. Then the second most senior shaman will cut off his piece of parchment in the same way, and so on, in the order of decreasing seniority of the shamans. The last shaman to get a piece of the parchment will take, without any cutting, the remaining part of the parchment, after all previous shamans had cut and taken their pieces. All pieces cut off must be of the same shape and size, and each shaman must get exactly one piece of the parchment. Also, the shape and size of the final remaining piece of the parchment, taken by the last shaman, must be the same as the shape and size of all previous pieces. Rotation is allowed when comparing the pieces. Flipping a piece is not allowed, as the reverse side does not look the same as the front side.

Given the shape and size of the whole parchment, Cimrman's goal is to employ the maximum possible number of shamans.

Input

The first input line contains integers N and M ($1 \leq N, M \leq 300$). The next N lines contain a matrix of N rows and M columns. Each element is either a dot '.' representing empty space or a hash symbol '#' representing the parchment. The parchment is guaranteed to be connected (one piece) and non-empty.

Output

Output the maximum number of shamans that can take part in the experiment.

Example

Input	Output
5 7 ..###.. ...##.. #####.# ###.### #...###	3
7 5 .##.. ##### ...## ...#. ...#. ...## ...##	1

Problem J. Transmitter

Source file name:	Transmitter.c, Transmitter.cpp, Transmitter.java, Transmitter.py
Input:	`Standard`
Output:	`Standard`

There is one radio transmitter on each floor in the Cimrman Labs main building. The transmitters are going to communicate with Cimrman Lunar Base on the far side of the Moon. Cimrman needs a strong signal, therefore more transmitters will work simultaneously to attain good connection characteristics.

However, for other practical reasons, the working transmitters have to be located in a single block of floors, that is, there should not be any floor with unused transmitter anywhere between other two floors with used transmitters.

Immediately after the start of the transmission, at the beginning of each second, each transmitter sends out also the so-called coordination signal (CoSi) on a particular coordination frequency. A separate sequence of coordination frequencies is given for each transmitter. The transmitter sends out CoSi on i-th frequency in the sequence at the beginning of i-th second. The number of seconds in which CoSi is sent out is equal to the length of the sequence. When the sequence of frequencies is exhausted the transmitter stops sending out CoSi, but it still continues to do its main work.

The transmitters often work in pairs to boost each other performance. The performance quality of a pair of transmitters is equal to the maximum length in seconds of a time interval immediately after the start of the transmission in which both transmitters in the pair send out CoSi on the same frequency. This interval ends when the transmitters in the pair send out a CoSi on different frequencies or when any of them stops sending out CoSi altogether.

When a group of more than two transmitters work together the performance quality is calculated as the sum of performance qualities of all possible pairs of transmitters in the group.

Now, Cimrman wants to choose a group of transmitters located in a contiguous block of floors in such a way that the performance quality of the group is at least a given predefined value K.

Help Cimrman calculate the number of such possible groups of transmitters.

Input

The first line of input contains two integers N, K ($1 \leq N \leq 10^6$, $1 \leq K \leq 10^9$), the number of transmitters and the predefined performance quality value. Each of the next N lines contains a string of lowercase letters. Each string represents the sequence of CoSi frequencies of a particular transmitter, each letter represents CoSi frequency in one second. The i-th string represents the CoSi frequencies of the transmitter on the i-th floor.

The same letters across the input represent the same frequencies, different letters represent different frequencies. The sum of lengths of all strings is guaranteed to be at most 10^6.

Output

Output the number of groups of transmitters which satisfy Cimrman's demands and have performance quality at least K.

Example

Input	Output
4 3 set stop setting state	3
5 6 a rating rating b c	6

Problem K. Volcanoes

Source file name:	Volcanoes.c, Volcanoes.cpp, Volcanoes.java, Volcanoes.py
Input:	`Standard`
Output:	`Standard`

Cimrman is going to visit all his artificial prairie volcanoes he has built in the previous few years. He is going to travel in his one-of-a-kind volcanology terrain vehicle.

Unfortunately, the vehicle has been currently damaged by an accidental meteorite strike. It can travel in only three possible directions, directly north, south or east. Cimrman decided this should not be a major problem, he is planning to organise his journey in such a way that the vehicle always travels in one of these three directions. Another peculiarity of the vehicle is that it can change its direction immediately.

Cimrman can start his journey at any point on the prairie. He wants the journey to be as short as possible.

Input

The first line of input contains one integer N ($0 < N \leq 10^5$), the number of volcanoes. Each of the next N lines contains coordinates of one volcano. A volcano is represented as a point on a plane, first its x coordinate is given, then follows the y coordinate.

The point coordinates are two integer values in the range between -10^6 and 10^6 inclusive. Direction of positive x-coordinate corresponds to the eastward direction in the terrain.

Output

Print the length of a shortest journey which visits all the volcanoes. Assume the journey starts at the first volcano visit and ends at the last volcano visit.

Example

Input	Output
4 2 3 3 2 1 1 5 5	10
5 1 2 5 4 3 6 7 8 1 1	17

Problem L. Wagon

Source file name:	Wagon.c, Wagon.cpp, Wagon.java, Wagon.py
Input:	`Standard`
Output:	`Standard`

Cimrman devised and built a railroad freight wagon capable of carrying a single construction crane. It has a specially built mechanism which automatically collapses and restores the crane again as the train moves through tunnels, narrow corridors in woods etc.

As any other active inventor, Cimrman is always short of money and he typically uses his current inventions in various unexpected ways to make him money for investments in his future creations.

This time he devised another unorthodox scheme. He attached the empty crane wagon to one of the regular cross country trains. Only one journey of the train is available to Cimrman and his crane wagon and he cannot change the direction of travel throughout the journey.

There is a list of crane types which can be sold and bought in each of the cities on the train journey. Often, the prices differ in various cities. Thus, Cimrman plans to buy a crane in one of the cities on the train path, load it on the train and in some of the next cities sell it with profit. He may repeat this action more times, each time buying a crane in a city and selling it in some of the next cities on the journey. Each time, he can transport only one crane. Also, he can travel between cities without carrying a crane if he considers it to be profitable.

Cimrman's private budget at the beginning of the journey is big enough to buy any crane in any city.

Cimrman wants to achieve maximum possible profit from the whole journey.

Input

The first line contains integer N ($1 \leq N \leq 10^5$), the number of cities on the train path.

For each of N cities, there is an input block starting with line containing one integer M ($1 \leq M \leq 10$), the number of crane types for sale in that city.

Next in the block, there are M lines, each contains two integers I, P ($0 \leq I \leq 10^9$, $0 \leq P \leq 10^4$), the ID of crane type for sale and its price. The order of the blocks is the same as the order of the cities on the train journey. The IDs are mutually different in a block and the price applies to both buying and selling the crane. In any city, Cimrman is allowed to both sell and buy only the types of cranes with the listed IDs in that city.

Output

Output one integer - the biggest profit Cimrman can achieve using his scheme.

Example

Input	Output
5 1 1 3 1 1 2 1 1 3 1 1 5 1 1 1	3
4 2 1 2 2 1 2 1 3 2 5 1 1 4 1 2 6	5