Educational Codeforces Round 73 (Rated for Div. 2)

A. 2048 Game

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are playing a variation of game 2048. Initially you have a multiset s of n integers. Every integer in this multiset is a power of two.

You may perform any number (possibly, zero) operations with this multiset.

During each operation you choose two **equal** integers from s, remove them from s and insert the number equal to their sum into s.

For example, if $s = \{1, 2, 1, 1, 4, 2, 2\}$ and you choose integers 2 and 2, then the multiset becomes $\{1, 1, 1, 4, 4, 2\}$.

You win if the number 2048 belongs to your multiset. For example, if $s = \{1024, 512, 512, 4\}$ you can win as follows: choose 512 and 512, your multiset turns into $\{1024, 1024, 4\}$. Then choose 1024 and 1024, your multiset turns into $\{2048, 4\}$ and you win.

You have to determine if you can win this game.

You have to answer q independent queries.

Input

The first line contains one integer q ($1 \le q \le 100$) – the number of queries.

The first line of each query contains one integer n ($1 \le n \le 100$) — the number of elements in multiset.

The second line of each query contains n integers $s_1, s_2, \ldots, s_n$ ($1 \le s_i \le 2^{29}$) — the description of the multiset. It is guaranteed that all elements of the multiset are powers of two.

Output

For each query print YES if it is possible to obtain the number 2048 in your multiset, and NO otherwise.

You may print every letter in any case you want (so, for example, the strings yEs, yes, Yes and YES will all be recognized as positive answer).

Example

input
6 4 1024 512 64 512 1 2048 3 64 512 2 2 4096 4 7 2048 2 2048 2048 2048 2048 2048 2 2048 4096
output
YES YES NO NO YES YES

Note

In the first query you can win as follows: choose 512 and 512, and s turns into $\{1024, 64, 1024\}$. Then choose 1024 and 1024, and s turns into $\{2048, 64\}$ and you win.

In the second query s contains 2048 initially.

B. Knights

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input

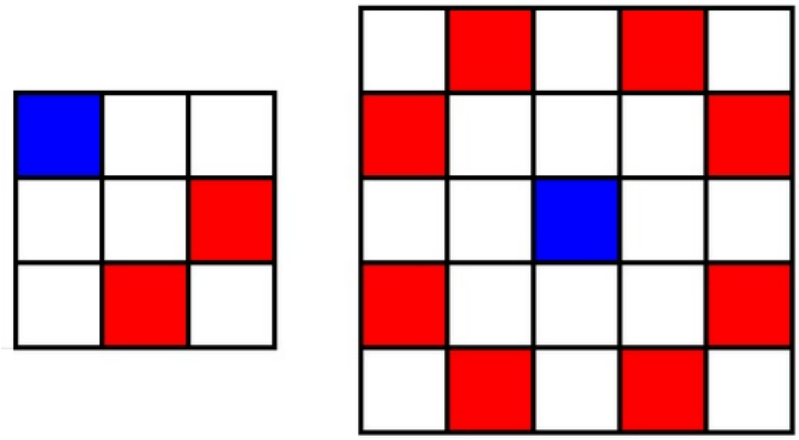
output: standard output

You are given a chess board with n rows and n columns. Initially all cells of the board are empty, and you have to put a white or a black knight into each cell of the board.

A knight is a chess piece that can attack a piece in cell (x_2, y_2) from the cell (x_1, y_1) if one of the following conditions is met:

- $|x_1 - x_2| = 2$ and $|y_1 - y_2| = 1$, or
- $|x_1 - x_2| = 1$ and $|y_1 - y_2| = 2$.

Here are some examples of which cells knight can attack. In each of the following pictures, if the knight is currently in the blue cell, it can attack all red cells (and only them).



A *duel of knights* is a pair of knights of **different** colors such that these knights attack each other. You have to put a knight (a white one or a black one) into each cell in such a way that the number of duels is maximum possible.

Input
The first line contains one integer n ($3 \le n \le 100$) — the number of rows (and columns) in the board.

Output
Print n lines with n characters in each line. The j-th character in the i-th line should be `W`, if the cell (i, j) contains a white knight, or `B`, if it contains a black knight. The number of duels should be maximum possible. If there are multiple optimal answers, print any of them.

Example

input
3
output
WBW BBB WBW

Note
In the first example, there are 8 duels:

1. the white knight in (1, 1) attacks the black knight in (3, 2);
2. the white knight in (1, 1) attacks the black knight in (2, 3);
3. the white knight in (1, 3) attacks the black knight in (3, 2);
4. the white knight in (1, 3) attacks the black knight in (2, 1);
5. the white knight in (3, 1) attacks the black knight in (1, 2);
6. the white knight in (3, 1) attacks the black knight in (2, 3);
7. the white knight in (3, 3) attacks the black knight in (1, 2);
8. the white knight in (3, 3) attacks the black knight in (2, 1).

C. Perfect Team

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You may have already known that a standard ICPC team consists of exactly three members. The perfect team however has more restrictions. A student can have some specialization: coder or mathematician. **She/he can have no specialization, but can't have both at the same time.**

So the team is considered perfect if it includes at least one coder, at least one mathematician and it consists of exactly three members.

You are a coach at a very large university and you know that c of your students are coders, m are mathematicians and x have no

specialization.

What is the maximum number of full perfect teams you can distribute them into?

Note that some students can be left without a team and each student can be a part of no more than one team.

You are also asked to answer q independent queries.

Input
The first line contains a single integer q ($1 \le q \le 10^4$) — the number of queries.

Each of the next q lines contains three integers c, m and x ($0 \le c, m, x \le 10^8$) — the number of coders, mathematicians and students without any specialization in the university, respectively.

Note that the no student is both coder and mathematician at the same time.

Output
Print q integers — the i-th of them should be the answer to the i query in the order they are given in the input. The answer is the maximum number of full perfect teams you can distribute your students into.

Example

input
6 1 1 1 3 6 0 0 0 0 0 1 1 10 1 10 4 4 1
output
1 3 0 0 1 3

Note
In the first example here are how teams are formed:

1. the only team of 1 coder, 1 mathematician and 1 without specialization;
2. all three teams consist of 1 coder and 2 mathematicians;
3. no teams can be formed;
4. no teams can be formed;
5. one team consists of 1 coder, 1 mathematician and 1 without specialization, the rest aren't able to form any team;
6. one team consists of 1 coder, 1 mathematician and 1 without specialization, one consists of 2 coders and 1 mathematician and one consists of 1 coder and 2 mathematicians.

D. Make The Fence Great Again

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You have a fence consisting of n vertical boards. The width of each board is 1. The height of the i-th board is a_i. You think that the fence is *great* if there is no pair of adjacent boards having the same height. More formally, the fence is great if and only if for all indices from 2 to n, the condition $a_{i-1} \neq a_i$ holds.

Unfortunately, it is possible that now your fence is not great. But you can change it! You can increase the length of the i-th board by 1, but you have to pay b_i rubles for it. The length of each board can be increased any number of times (possibly, zero).

Calculate the minimum number of rubles you have to spend to make the fence great again!

You have to answer q independent queries.

Input
The first line contains one integer q ($1 \le q \le 3 \cdot 10^5$) — the number of queries.

The first line of each query contains one integers n ($1 \le n \le 3 \cdot 10^5$) — the number of boards in the fence.

The following n lines of each query contain the descriptions of the boards. The i-th line contains two integers a_i and b_i ($1 \le a_i, b_i \le 10^9$) — the length of the i-th board and the price for increasing it by 1, respectively.

It is guaranteed that sum of all n over all queries not exceed $3 \cdot 10^5$.

It is guaranteed that answer to each query will not exceed 10^{18}.

Output
For each query print one integer — the minimum number of rubles you have to spend to make the fence great.

Example

input
3 3 2 4 2 1 3 5 3 2 3 2 10 2 6 4 1 7 3 3 2 6 1000000000 2
output
2 9 0

Note
In the first query you have to increase the length of second board by 2. So your total costs if $2 \cdot b_2 = 2$.

In the second query you have to increase the length of first board by 1 and the length of third board by 1. So your total costs if $1 \cdot b_1 + 1 \cdot b_3 = 9$.

In the third query the fence is great initially, so you don't need to spend rubles.

E. Game With String

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Alice and Bob play a game. Initially they have a string $s_1, s_2, \ldots, s_n$, consisting of only characters . and X. They take alternating turns, and Alice is moving first. During each turn, the player has to select a contiguous substring consisting only of characters . and replaces each of them with X. Alice must select a substing of length a, and Bob must select a substring of length b. It is guaranteed that $a > b$.

For example, if $s =$ ...X.. and $a = 3$, $b = 2$, then after Alice's move string can turn only into XXXX... And if it's Bob's turn and the string $s =$ ...X.., then after Bob's move the string can turn into XX.X.., .XXX.. or ...XXX.

Whoever is unable to make a move, loses. You have to determine who wins if they both play optimally.

You have to answer q independent queries.

Input
The first line contains one integer q ($1 \le q \le 3 \cdot 10^5$) — the number of queries.

The first line of each query contains two integers a and b ($1 \le b < a \le 3 \cdot 10^5$).

The second line of each query contains the string s ($1 \le |s| \le 3 \cdot 10^5$), consisting of only characters . and X.

It is guaranteed that sum of all $|s|$ over all queries not exceed $3 \cdot 10^5$.

Output
For each test case print YES if Alice can win and NO otherwise.

You may print every letter in any case you want (so, for example, the strings yEs, yes, Yes and YES will all be recognized as positive answer).

Example

input
3 3 2 XX......XX...X 4 2 X...X.X..X 5 3 ....X..X
output
YES NO

YES

Note
In the first query Alice can select substring $s_3 \dots s_5$. After that s turns into XXXXX...XX...X. After that, no matter what move Bob makes, Alice can make the move (this will be her second move), but Bob can't make his second move.

In the second query Alice can not win because she cannot even make one move.

In the third query Alice can choose substring $s_2 \dots s_6$. After that s turns into .XXXXX.X..X, and Bob can't make a move after that.

F. Choose a Square

time limit per test: 6 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Petya recently found a game "Choose a Square". In this game, there are n points numbered from 1 to n on an infinite field. The i-th point has coordinates (x_i, y_i) and cost c_i.

You have to choose a square such that its sides are parallel to coordinate axes, the lower left and upper right corners belong to the line $y = x$, and all corners have integer coordinates.

The score you get is the sum of costs of the points covered by the selected square minus the length of the side of the square. Note that the length of the side can be zero.

Petya asks you to calculate the maximum possible score in the game that can be achieved by placing exactly one square.

Input
The first line of the input contains one integer n ($1 \le n \le 5 \cdot 10^5$) — the number of points on the field.

Each of the following n lines contains three integers x_i, y_i, c_i ($0 \le x_i, y_i \le 10^9, -10^6 \le c_i \le 10^6$) — coordinates of the i-th point and its cost, respectively.

Output
In the first line print the maximum score Petya can achieve.

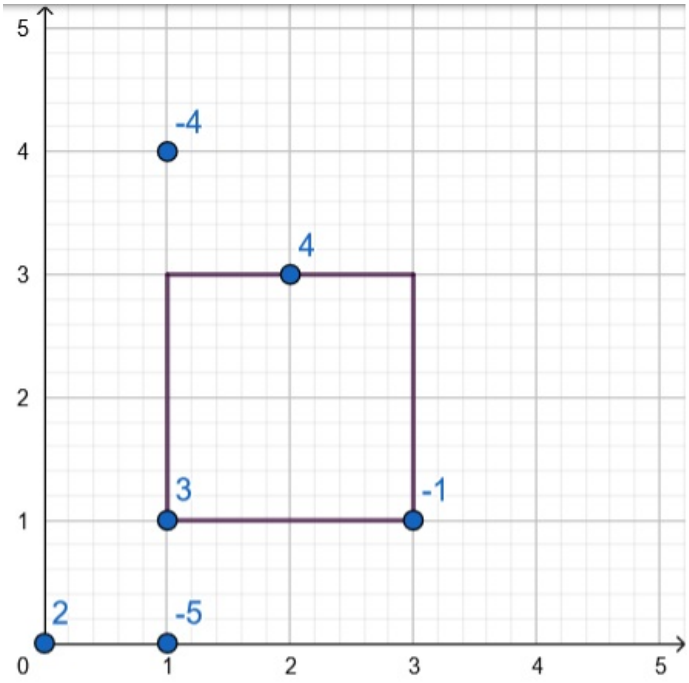
In the second line print four integers x_1, y_1, x_2, y_2 ($0 \le x_1, y_1, x_2, y_2 \le 2 \cdot 10^9, x_1 = y_1, x_2 = y_2, x_1 \le x_2$) separated by spaces — the coordinates of the lower left and upper right corners of the square which Petya has to select in order to achieve the maximum score.

Examples

input
6 0 0 2 1 0 -5 1 1 3 2 3 4 1 4 -4 3 1 -1
output
4 1 1 3 3

input
5 3 3 0 3 3 -3 0 2 -1 3 1 3 0 0 -2
output
0 1 1 1 1

Note
The field corresponding to the first example:



G. Graph And Numbers

time limit per test: 3.5 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

You are given an undirected graph with n vertices and m edges. You have to write a number on each vertex of this graph, each number should be either 0 or 1. After that, you write a number on each edge equal to the sum of numbers on vertices incident to that edge.

You have to choose the numbers you will write on the vertices so that there is at least one edge with 0 written on it, at least one edge with 1 and at least one edge with 2. How many ways are there to do it? Two ways to choose numbers are different if there exists at least one vertex which has different numbers written on it in these two ways.

Input

The first line contains two integers n and m ($1 \le n \le 40$, $0 \le m \le \frac{n(n-1)}{2}$) — the number of vertices and the number of edges, respectively.

Then m lines follow, each line contains two numbers x_i and y_i ($1 \le x_i, y_i \le n$, $x_i \ne y_i$) — the endpoints of the i-th edge. It is guaranteed that each pair of vertices is connected by at most one edge.

Output

Print one integer — the number of ways to write numbers on all vertices so that there exists at least one edge with 0 written on it, at least one edge with 1 and at least one edge with 2.

Examples

input
6 5 1 2 2 3 3 4 4 5 5 1
output
20

input
4 4 1 2 2 3 3 4 4 1
output
4

input
35 29 1 2

```
2 3
3 1
4 1
4 2
3 4
7 8
8 9
9 10
10 7
11 12
12 13
13 14
14 15
15 16
16 17
17 18
18 19
19 20
20 21
21 22
22 23
23 24
24 25
25 26
26 27
27 28
28 29
29 30
```

output

```
34201047040
```