Problem A. A Bit Common

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	1024 megabytes

Given two integers n and m, among all the sequences containing n non-negative integers less than 2^m, you need to count the number of such sequences A that there exists a non-empty subsequence of A in which the bitwise AND of the integers is 1.

Note that a non-empty subsequence of a sequence A is a non-empty sequence that can be obtained by deleting zero or more elements from A and arranging the remaining elements in their original order.

Since the answer may be very large, output it modulo a positive integer q.

The bitwise AND of non-negative integers A and B, A `AND` B is defined as follows:

- When A `AND` B is written in base two, the digit in the 2^d's place ($d \geq 0$) is 1 if those of A and B are **both** 1, and 0 otherwise.

For example, we have 4 `AND` 6 = 4 (in base two: 100 `AND` 110 = 100).

Generally, the bitwise AND of k non-negative integers $p_1, p_2, \ldots, p_k$ is defined as

$$(\ldots((p_1 \texttt{ AND } p_2) \texttt{ AND } p_3) \texttt{ AND } \ldots \texttt{ AND } p_k)$$

and we can prove that this value does not depend on the order of $p_1, p_2, \ldots, p_k$.

Input

The only line contains three integers n ($1 \leq n \leq 5\,000$), m ($1 \leq m \leq 5\,000$) and q ($1 \leq q \leq 10^9$).

Output

Output a line containing an integer, denoting the answer.

Examples

standard input	standard output
2 3 998244353	17
5000 5000 998244353	2274146

Problem B. A Bit More Common

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	1024 megabytes

Given two integers n and m, among all the sequences containing n non-negative integers less than 2^m, you need to count the number of such sequences A that there exists **two different** non-empty subsequences of A in each of which the bitwise AND of the integers is 1.

Note that a non-empty subsequence of a sequence A is a non-empty sequence that can be obtained by deleting zero or more elements from A and arranging the remaining elements in their original order. **Two subsequences are different if they are composed of different locations in the original sequence.**

Since the answer may be very large, output it modulo a positive integer q.

The bitwise AND of non-negative integers A and B, A `AND` B is defined as follows:

- When A `AND` B is written in base two, the digit in the 2^d's place ($d \geq 0$) is 1 if those of A and B are **both** 1, and 0 otherwise.

For example, we have 4 `AND` 6 = 4 (in base two: 100 `AND` 110 = 100).

Generally, the bitwise AND of k non-negative integers $p_1, p_2, \ldots, p_k$ is defined as

$$(\ldots((p_1 \texttt{ AND } p_2) \texttt{ AND } p_3) \texttt{ AND } \ldots \texttt{ AND } p_k)$$

and we can prove that this value does not depend on the order of $p_1, p_2, \ldots, p_k$.

Input

The only line contains three integers n ($1 \leq n \leq 5\,000$), m ($1 \leq m \leq 5\,000$) and q ($1 \leq q \leq 10^9$).

Output

Output a line containing an integer, denoting the answer.

Examples

standard input	standard output
2 3 998244353	7
5000 5000 998244353	530227736

Problem C. Sum of Suffix Sums

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	1024 megabytes

Given an array which is initially empty, you need to perform q operations:

- Given two non-negative integers t and v, take out the element from the end of the array for t times and then append v to the end of the array. It is guaranteed that t does not exceed the length of the array before this operation.

After each operation, let $a_1, a_2, \ldots, a_n$ be the current array, find the **sum** of $s_1, s_2, \ldots, s_n$, where $s_i = a_i + a_{i+1} + \ldots + a_n$ is the sum of the suffix starting from position i.

Since the answers may be very large, output them modulo $1\,000\,000\,007$.

Input

The first line contains an integer q ($1 \le q \le 5 \times 10^5$), denoting the number of operations.

Each of the following q lines contains two non-negative integers t and v ($0 \le v \le 10^9$), describing an operation, where t does not exceed the length of the array before this operation.

Output

Output q lines, each of which contains an integer, denoting the answer.

Examples

standard input	standard output
5 0 1 0 2 1 3 0 6 2 100000	1 5 7 25 200001
1 0 1000000000	1000000000

Note

After the first operation, the current array is [1], the suffix sum array is [1], and the sum of the suffix sums is 1.

After the second operation, the current array is $[1, 2]$, the suffix sum array is $[3, 2]$, and the sum of the suffix sums is 5.

After the third operation, the current array is $[1, 3]$, the suffix sum array is $[4, 3]$, and the sum of the suffix sums is 7.

After the fourth operation, the current array is $[1, 3, 6]$, the suffix sum array is $[10, 9, 6]$, and the sum of the suffix sums is 25.

After the fifth operation, the current array is $[1, 100\,000]$, the suffix sum array is $[100\,001, 100\,000]$, and the sum of the suffix sums is $200\,001$.

Problem D. XOR of Suffix Sums

Input file:	`standard input`
Output file:	`standard output`
Time limit:	6 seconds
Memory limit:	1024 megabytes

Given an array which is initially empty, you need to perform q operations:

- Given two non-negative integers t and v, take out the element from the end of the array for t times and then append v to the end of the array. It is guaranteed that t does not exceed the length of the array before this operation.

After each operation, let $a_1, a_2, \ldots, a_n$ be the current array, find the bitwise **XOR** of $s_1, s_2, \ldots, s_n$, where $s_i = a_i + a_{i+1} + \ldots + a_n$ is the sum of the suffix starting from position i.

Since the answers may be very large, output them modulo 2 097 152.

The bitwise XOR of non-negative integers A and B, $A \oplus B$ is defined as follows:

- When $A \oplus B$ is written in base two, the digit in the 2^d's place ($d \geq 0$) is 1 if exactly one of the digits in that place of A and B is 1, and 0 otherwise.

For example, we have $3 \oplus 5 = 6$ (in base two: $011 \oplus 101 = 110$).

Generally, the bitwise XOR of k non-negative integers $p_1, p_2, \ldots, p_k$ is defined as

$$(\ldots((p_1 \oplus p_2) \oplus p_3) \oplus \ldots \oplus p_k)$$

and we can prove that this value does not depend on the order of $p_1, p_2, \ldots, p_k$.

Input

The first line contains an integer q ($1 \leq q \leq 5 \times 10^5$), denoting the number of operations.

Each of the following q lines contains two non-negative integers t and v ($0 \leq v \leq 10^9$), describing an operation, where t does not exceed the length of the array before this operation.

Output

Output q lines, each of which contains an integer, denoting the answer.

Examples

standard input	standard output
5 0 1 0 2 1 3 0 6 2 100000	1 1 7 5 1
1 0 1000000000	1755648

Note

After the first operation, the current array is [1], the suffix sum array is [1], and the bitwise XOR of the suffix sums is 1.

After the second operation, the current array is $[1, 2]$, the suffix sum array is $[3, 2]$, and the bitwise XOR of the suffix sums is 1.

After the third operation, the current array is $[1, 3]$, the suffix sum array is $[4, 3]$, and the bitwise XOR of the suffix sums is 7.

After the fourth operation, the current array is $[1, 3, 6]$, the suffix sum array is $[10, 9, 6]$, and the bitwise XOR of the suffix sums is 5.

After the fifth operation, the current array is $[1, 100\,000]$, the suffix sum array is $[100\,001, 100\,000]$, and the bitwise XOR of the suffix sums is 1.

Problem E. Product of Suffix Sums

Input file:	standard input
Output file:	standard output
Time limit:	8 seconds
Memory limit:	1024 megabytes

Given an array which is initially empty, you need to perform q operations:

- Given two non-negative integers t and v, take out the element from the end of the array for t times and then append v to the end of the array. It is guaranteed that t does not exceed the length of the array before this operation.

After each operation, let $a_1, a_2, \ldots, a_n$ be the current array, find the **product** of $s_1, s_2, \ldots, s_n$, where $s_i = a_i + a_{i+1} + \ldots + a_n$ is the sum of the suffix starting from position i.

Since the answers may be very large, output them modulo 1 004 535 809.

Input

The first line contains an integer q ($1 \le q \le 1.3 \times 10^5$), denoting the number of operations.

Each of the following q lines contains two non-negative integers t and v ($0 \le v \le 10^9$), describing an operation, where t does not exceed the length of the array before this operation.

Output

Output q lines, each of which contains an integer, denoting the answer.

Examples

standard input	standard output
5 0 1 0 2 1 3 0 6 2 100000	1 6 12 540 959277719
1 0 1000000000	1000000000

Note

After the first operation, the current array is [1], the suffix sum array is [1], and the product of the suffix sums is 1.

After the second operation, the current array is $[1, 2]$, the suffix sum array is $[3, 2]$, and the product of the suffix sums is 6.

After the third operation, the current array is $[1, 3]$, the suffix sum array is $[4, 3]$, and the product of the suffix sums is 12.

After the fourth operation, the current array is $[1, 3, 6]$, the suffix sum array is $[10, 9, 6]$, and the product of the suffix sums is 540.

After the fifth operation, the current array is $[1, 100\,000]$, the suffix sum array is $[100\,001, 100\,000]$, and the product of the suffix sums is 10 000 100 000, where the remainder divided by 1 004 535 809 is 959 277 719.

Problem F. Career Path

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	1024 megabytes

Little Q has X years of work experience and is currently unemployed. He wants to plan the career path for the next N years and then retire. More specifically, assume that today is the first day of the 1-st year, and Little Q will officially retire on the last day of the N-th year.

There are M companies in total, the i-th of which will be founded on the first day in the L_i-th year and go bankrupt after paying all the employees' payment on the last day of the R_i-th year. That is, Little Q can start working for the company on the first day from the L_i-th year to the R_i-th year.

Each company's fiscal year is from the first day to the last day of one year. An employee who starts on the first day and is still employed on the last day in one year will receive all the payments as a reward for working hard for a full year. Since Little Q is highly skilled, every company is willing to hire him at any time. However, Little Q only works for at most one company in a year and considers changing jobs after receiving all the payments on the last day of the year and being able to start at a different company on the first day of the next year. If he chooses to work for a company in a year, he will gain a year of work experience.

Assume that, on the first day of some year, Little Q has Y years of work experience in total, and has already worked for Z full years continuously since the latest time being newly employed by the i-th company, and he will work for the i-th company in this full year:

- He will receive a payment of $A_i \times Y + B_i$ as a signing fee on the last day of this year if he is just newly employed, i.e. $Z = 0$.
- He will receive a payment of $C_i \times Y + D_i$ as annual cash salary on the last day of the year. If the company goes bankrupt just on the last day of the year, he will also receive another cash payment of $\frac{(Z+2)(C_i \times Y + D_i)}{12}$ as compensation and then resign from the company.
- He will receive a payment of $E_i \times Z + F_i$ as annual bonus on the last day of the year if the company doesn't go bankrupt on this day.
- He will receive $G_i \times Y + H_i$ units of unvested stock shares as annual stock bonus on the last day of the year. The stock shares will be vested in equal portions upon Little Q along with the payments in each of the following I_i years after the year receiving. Resigning from the company will automatically give up all the unvested stock shares, and sell all vested shares forcedly.

The stock price for the i-th company in the j-th year is $P_{i,j}$ per share, that is, sell p shares of stock of the i-th company in the j-th year will receive a payment of $P_{i,j} \times p$, where p could be any positive real number. The price is updated on the first day of every year and remains valid for the entire year. The stock price is 0 if the company has not yet been founded or has already gone bankrupt. The only way to gain stock shares is from annual stock bonus, but he can choose to sell vested stock shares at any time at the current price as long as he is still working at this company. Little Q must sell all the remaining vested stock shares at that year's price when he resigns from the company.

Additionally, the i-th company has non-compete restrictions, which means Little Q cannot work at the companies indexed in range $[U_i, V_i]$ in the next year if he resigns from the i-th company before the day he retires or the day the company goes bankrupt:

- If there are surviving companies indexed in range $[U_i, V_i]$ but Little Q chooses to take a gap year in the next year, he will receive a one-time payment of $J_i \times W + K_i$ for the non-compete compensation since he has already worked for the company for W full years continuously since the latest time

being newly employed by the i-th company. The gap year will not be counted in the number of years of work experience.

- Otherwise, he receives no non-compete compensation and cannot work in companies indexed in range within $[U_i, V_i]$ in the next year.

You need to help Little Q design a career path that maximizes his total income for the next N years. The unsold stock shares will not be counted in Little Q's income.

Input

The first line contains three integers X, N, and M.

Then $2M$ lines follow, where each of the M companies is described by two lines. For the i-th company:

The first line consists of 15 integers: A_i, B_i, C_i, D_i, E_i, F_i, G_i, H_i, I_i, U_i, V_i, J_i, K_i, L_i, and R_i. It is guaranteed that $I_i > 0$, $1 \le U_i \le V_i \le M$, and $0 \le L_i \le R_i \le N$.

The second line consists of N integers $P_{i,1}, P_{i,2}, \ldots, P_{i,N}$, representing the stock prices for the following N years. It is guaranteed that the stock price is 0 if the company has not yet been founded or has already gone bankrupt. This line would be empty if $N = 0$.

The stock option prices are integers in the range $[0, 8\,000]$, and all other input values are integers in the range $[0, 100]$.

Output

Output a line containing a single real number, indicating the maximum total income for the next N years.

Your answer will be considered correct if its absolute or relative error does not exceed 10^{-6}. Formally speaking, suppose that your output is a and the jury's answer is b, your output is accepted if and only if $\frac{|a-b|}{\max(1,|b|)} \le 10^{-6}$.

Examples

standard input	standard output
5 10 2 3 1 2 48 1 6 2 8 4 2 2 1 24 1 7 1 1 2 2 3 3 4 0 0 0 1 5 5 25 0 10 3 10 5 1 1 2 10 3 10 0 0 0 1 3 1 3 1 3 1	1338.933333333333
5 10 2 3 1 2 48 1 6 2 8 4 2 2 1 24 1 6 1 1 2 2 3 3 0 0 0 0 1 5 5 25 0 10 3 10 5 1 1 2 10 3 10 0 0 0 1 3 1 3 1 3 1	1247.500000000000
5 0 2 3 1 2 48 1 6 2 8 4 2 2 1 24 0 0 1 5 5 25 0 10 3 10 5 1 1 2 10 0 0	0.000000000000

Note

For the first sample case:

Little Q works for the 1-st company in the first 7 years and receives

- a signing fee of 16;

- cash salaries of 448 in total, along with a compensation of $\frac{140}{3}$ since the company goes bankrupt in the 7-th year;
- annual bonuses totaling 51;
- 168 units of stock shares in total, among which 98 units are vested and all sold at a price of 4.

Then Little Q works for the 2-nd company in the last 3 years.

Problem G. 3/2 Square Strings

Input file:	`standard input`
Output file:	`standard output`
Time limit:	4 seconds
Memory limit:	1024 megabytes

Given two strings $S = s_1 s_2 \cdots s_{|S|}$ and $T = t_1 t_2 \cdots t_{|T|}$, where $|S|$ denotes the length of S, and $|T|$ denotes the length of T, you need to count the number of quadruples (p, q, u, v), such that $1 \le p \le q \le |S|$, $1 \le u \le v \le |T|$, and $t_u t_{u+1} \cdots t_v$ is a square string where the first half is identical to $s_p s_{p+1} \cdots s_q$.

Recall that a square string is a string of even length in which the first half is identical to the second half. For example, "`aaaa`" and "`abcabc`" are square strings, while "`aaa`" and "`abcabd`" are not.

Since the answer may be very large, output it modulo 998 244 353.

Input

The input consists of two lines, where the first line contains the string S, and the second line contains the string T.

It is guaranteed that both S and T consist only of lowercase English letters and their lengths do not exceed 2×10^5.

Output

Output a line containing an integer, denoting the answer.

Examples

standard input	standard output
ababab ababaa	8
aaaaaaaa aaaaaaaa	114

Note

For the first sample case, the 8 quadruples are $(1, 1, 5, 6)$, $(1, 2, 1, 4)$, $(2, 3, 2, 5)$, $(3, 3, 5, 6)$, $(3, 4, 1, 4)$, $(4, 5, 2, 5)$, $(5, 5, 5, 6)$, and $(5, 6, 1, 4)$.

Problem H. World Finals

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	1024 megabytes

The ICPC World Finals are coming. Due to some reasons, the 46th and 47th World Finals will be held simultaneously. For the teams qualified in both competitions, they should choose one to take part in.

As we know, lzr010506's team is double-qualified and should make a choice. To make a wiser choice, lzr010506 looked up the qualified lists for two competitions and trained a magic model to predict the results for all participants among the two competitions. Moreover, a result contains the number of solved problems and the time penalty. The more solved problems, the better the result is, and if two teams solved the same number of problems, the result with the lower time penalty is better.

Now, lzr010506 wants to know the best possible ranking if the actual results are all the same as predicted and that the competition choices of the double-qualified teams can be arbitrarily arranged by him.

Input

The first line contains one integer $n\,(1 \le n \le 10^5)$, denoting the number of teams qualified in the 46th World Finals.

Next n lines each contain one string $S\,(1 \le |S| \le 10)$ and two integers $p, t\,(1 \le p, t \le 10^9)$, denoting the name, the predicted number of solved problems, and time penalty of one team in the 46th World Finals respectively.

Next one line contains one integer $m\,(1 \le m \le 10^5)$, denoting the number of teams qualified in the 47th World Finals.

Next m lines each contain one string $S\,(1 \le |S| \le 10)$ and two integers $p, t\,(1 \le p, t \le 10^9)$, denoting the name, the predicted number of solved problems, and time penalty of one team in the 47th World Finals respectively.

It is guaranteed that:

- the team names only contain digits and English letters;
- the team names in one competition are different from each other;
- no two teams have the same predicted number of solved problems and the time penalty simultaneously in one competition;
- the same names among two qualified name lists refer to the same team in real;
- "lzr010506" appears in both two qualified name lists.

Output

Output one line containing one integer, denoting the best possible ranking of lzr010506's team.

Example

standard input	standard output
5 pku 10 1513 thu 8 1195 lzr010506 8 1234 MIT 9 816 ntu 8 1325 4 mipt 9 1143 ntu 7 962 lzr010506 9 1523 pku 9 1068	2

Note

There are three double-qualified teams: lzr010506, pku, ntu.

One possible arrangement is that pku takes part in the 46th World Finals while lzr010506 and ntu take part in the 47th World Finals. As a result, lzr010506's team could get rank 2 in the 47th World Finals. According to the arrangement, we can get the boards of the two competitions as follows:

- The board of the 46th World Finals:

1. pku, solved 10, penalty 1513
2. MIT, solved 9, penalty 816
3. thu, solved 8, penalty 1195

- The board of the 47th World Finals:

1. mipt, solved 9, penalty 1143
2. **lzr010506, solved 9, penalty 1523**
3. ntu, solved 7, penalty 962

Problem I. Mirror Maze

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	1024 megabytes

There is an $n \times m$ mirror maze, where there is a mirror on each grid. The mirrors are in one of the following four types:

- "-", the light from above or below will be reflected back, the light from left or right will continue going forward without being reflected, respectively;
- "|", the light from left or right will be reflected back, the light from above or below will continue going forward without being reflected, respectively;
- "/", the light from left, right, above, below will be reflected to go above, below, left, right, respectively;
- "\", the light from left, right, above, below will be reflected to go below, above, right, left, respectively.

Now there are q light sources. Little G, the believer of the light, wants to know the numbers of different mirrors the emitted light will be reflected by within sufficient time for each light source.

Input

The first line contains two integers $n, m\,(1 \le n, m \le 1\,000)$, denoting the size of the mirror maze.

Each of the following n lines contains a string of length m, where the j-th character in the i-th line $S_{i,j}\,(S_{i,j} \in \{|, -, /, \backslash\})$ denotes the mirror on grid (i, j).

The next line contains an integer $q\,(1 \le q \le 10^5)$, denoting the number of light sources.

Each of the following q lines contains two integers $u\,(1 \le u \le n)$, $v\,(1 \le v \le m)$ and a string $dir\,(dir \in \{above, below, left, right\})$, denoting that a light source is on grid (u, v) and emits light going along the dir direction. Specifically, the light will **not** be influenced by the mirror on grid (u, v) initially.

Output

Output q lines each containing one integer, denoting the number of different mirrors the emitted light will be reflected by within sufficient time for each light source.

Example

standard input	standard output
2 3 /\- \/| 2 1 2 below 2 2 right	4 2

Note

- For the first light, it will be reflected by the mirrors at $(2, 2), (2, 1), (1, 1), (1, 2)$ repeatedly and stay in the mirror maze.
- For the second light, it will be reflected by $(2, 3)$ and go back to $(2, 2)$, then reflected by $(2, 2)$, go below, and get away from the mirror maze.

Problem J. 2D Travel

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	1024 megabytes

There is a robot in the 2D rectangular region $W = \{(x, y) \mid 0 \le x \le n, 0 \le y \le m\}$. Assume that the robot is at (x, y) currently, it can move in the following four directions:

- U : go from (x, y) to $(x, y + 1)$ if $(x, y + 1) \in W$;
- D : go from (x, y) to $(x, y - 1)$ if $(x, y - 1) \in W$;
- L : go from (x, y) to $(x - 1, y)$ if $(x - 1, y) \in W$;
- R : go from (x, y) to $(x + 1, y)$ if $(x + 1, y) \in W$.

An instruction contains a character c of "U", "D", "L", "R" and an integer $t\,(1 \le t \le 10^9)$, indicating that the robot moves in the direction c for t times consecutively under the instruction.

Given a list of instructions of length k, there are q queries, you should report the final position and the length of the moving route if the robot starts moving from the given position (x, y) and does the l-th, $(l + 1)$-th, $(l + 2)$-th, ..., r-th instructions one by one for each query.

Input

The first line contains four integers $n, m\,(1 \le n, m \le 10^9)$, $k, q\,(1 \le k, q \le 10^5)$, denoting the size of the rectangular region, the number of instructions, and the number of queries respectively.

Each of the following k lines contains a character c of "U", "D", "L", "R" and an integer $t\,(1 \le t \le 10^9)$, denoting an instruction.

Each of the following q lines contains four integers $x\,(0 \le x \le n)$, $y\,(0 \le y \le m)$, $l, r\,(1 \le l \le r \le k)$, denoting a query.

Output

For each query, output a line containing three integers u, v and w, denoting the coordinates of the final position and the length of the moving route.

Example

standard input	standard output
3 2 5 2 L 2 D 2 R 1 R 1 U 2 1 1 1 3 2 1 2 5	1 0 3 3 2 4

Note

- For the first query, the moving route is $(1, 1) \xrightarrow{L} (0, 1) \xrightarrow{L} (0, 1) \xrightarrow{D} (0, 0) \xrightarrow{D} (0, 0) \xrightarrow{R} (1, 0)$, where the final position is $(1, 0)$ and the length of the moving route is 3.
- For the second query, the moving route is $(2, 1) \xrightarrow{D} (2, 0) \xrightarrow{D} (2, 0) \xrightarrow{R} (3, 0) \xrightarrow{R} (3, 0) \xrightarrow{U} (3, 1) \xrightarrow{U} (3, 2)$, where the final position is $(3, 2)$ and the length of the moving route is 4.

Problem K. Matching Problem

Input file:	standard input
Output file:	standard output
Time limit:	4 seconds
Memory limit:	1024 megabytes

Given a simple undirected connected graph G, where the number of edges equals the number of vertices, and an unrooted tree H, determine whether H can be produced by erasing vertices or edges and relabeling vertices from G.

Input

The first line contains one integer $T\,(1 \le T \le 1\,000)$, denoting the number of test cases.

For each test case:

The first line contains one integer $n\,(3 \le n \le 1\,000)$, denoting the number of vertices as well as edges of graph G.

The next n lines each contain two integers $u, v\,(1 \le u, v \le n)$, denoting the edges of graph G. It is guaranteed that graph G contains no self-loops or multiple edges.

The next one line contains one integer $m\,(1 \le m \le n)$, denoting the number of vertices of tree H.

The next $m-1$ lines each contain two integers $u, v\,(1 \le u, v \le m)$, denoting the edges of tree H.

It is guaranteed that the sum of n among all test cases does not exceed 1 000.

Output

Output T lines each containing one string, "Yes" if H can be produced from G, or "No" if H cannot be produced from G.

Example

standard input	standard output
2 6 1 2 1 3 2 3 1 4 2 5 3 6 5 1 2 2 3 3 4 4 5 4 1 2 2 3 3 4 4 1 4 1 2 1 3 1 4	Yes No