Problem A. The New President

Program:	`vote.(cpp|java)`
Input:	`vote.in`
Balloon Color:	`Gold`

Finally, it is time to vote for a new president and you are really excited about that. You know that the final results may take weeks to be announced, while you can't really wait to see the results.

Somehow you managed to get the preferences list for every voter (we don't care how you managed to get this piece of information!). Each voter sorted out all candidates starting by his most preferred candidate and ending with his least preferred one. When voting, a voter votes for the candidate who comes first in his preferences list. For example, if there are 5 candidates (numbered 1 to 5), and the preferences list for one voter is [3, 2, 5, 1, 4] and the current competing candidates in the voting process are candidates 2 and 4, the voter will vote for candidate number 2.

Here are the rules for the election process:

1. There are $\boldsymbol{C}$ candidates (numbered from 1 to $\boldsymbol{C}$), and $\boldsymbol{V}$ voters ($\boldsymbol{V}$ is always an odd number).
2. The election process consists of up to 2 rounds. All candidates compete in the the first round. If a candidate receives more than 50% of the votes, he wins, otherwise another round takes place, in which only the top 2 candidates compete for the presidency. The candidate who receives more votes than his opponent wins and becomes the new president.
3. You can safely assume that the given preferences will never cause a situation in which the second and the third candidates from the first round receive the same number of votes.
4. The voters' preferences are the same in both rounds, and each voter must vote exactly once in each round for a currently competing candidate according to his preferences.

Given the preferences lists, you need to write a program to figure out which candidate will win and in which round.

Input

Your program will be tested on one or more test cases. The first line of the input will be a single integer $\boldsymbol{T}$, the number of test cases ($1 \leq \boldsymbol{T} \leq 100$). Followed by the test cases, the first line of a test case contains two integers $\boldsymbol{C}$ and $\boldsymbol{V}$ separated by a single space. $\boldsymbol{C}$ and $\boldsymbol{V}$ ($1 \leq \boldsymbol{C}$, $\boldsymbol{V} \leq 100$) represent the number of candidates and voters respectively, followed by $\boldsymbol{V}$ lines each line contains $\boldsymbol{C}$ integers separated by a single space, representing the preferences list for a single voter (the first is his most preferable candidate while the last is his least preferable one). Each integer from 1 to $\boldsymbol{C}$ will appear exactly once in each line.

Output

For each test case, print on a single line two integers separated by a single space. The first integer is the ID of the winner candidate (a number from 1 to $\boldsymbol{C}$), the second integer is either 1 or 2 indicating whether this candidate will win in the first round or the second one respectively.

Examples

vote.in	Standard Output
2 2 3 2 1 1 2 2 1 3 5 1 2 3 1 2 3 2 1 3 2 3 1 3 2 1	2 1 2 2

Problem B. No Name

Program:	`strings.(cpp|java)`
Input:	`strings.in`
Balloon Color:	`Orange`

This is the most direct problem ever, you are required to implement some basic string operations like insert and substring.

In this problem $|S|$ means the length of the string S.

Note: We didn't find a good name for this problem.

Input

Your program will be tested on one or more test cases. The first line of the input will be a single integer T, the number of test cases ($1 \leq T \leq 100$). Followed by the test cases, each test case starts with a line containing a string S ($1 \leq |S| \leq 1{,}000{,}000$), followed by one or more lines each describing one of the following operations to perform on S (all indices are zero based, and the quotes are for clarity):

1. "I R X" means insert the string R ($1 \leq |R| \leq 1{,}000{,}000$) in S at index X ($0 \leq X \leq |S|$), when X equals $|S|$ this means append R at the end of S. For example, the result of inserting "xy" in "abc" at position 1 will be "axybc", and the result of inserting "xy" in "abc" at position 3 will be "abcxy", and the result of inserting "xy" in "abc" at position 0 will be "xyabc".
2. "P X Y" means print the substring of S from X to Y, inclusive ($0 \leq X \leq Y < |S|$). For example the substring from 0 to 2 of "abc" is "abc", and the string from 1 to 1 of "abc" is "b".
3. "END" Indicates the end of operations for the test case.

Strings S and R will consist of lower case English letters only ('a' to 'z'), and $|S|$ will never get greater than 1,000,000 as a result of the operations for any test case. Also, the total number of characters to be printed for any test case will not exceed 1,000,000.

Note: Make sure to use fast IO operations, because the input and output files are very large.

Output

For every "P X Y" operation in the input, print one line with the corresponding substring.

Examples

strings.in	Standard Output
1 acm I ac 3 P 0 3 I x 3 I xxxx 6 I pc 6 P 0 11 END	acma acmxacpcxxxx

Problem C. Encrypted Password

Program:	`password.(cpp|java)`
Input:	`password.in`
Balloon Color:	White

Encrypting passwords is one of the most important problems nowadays, and you trust only the encryption algorithms which you invented, and you have just made a new encryption algorithm.

Given a password which consists of only lower case English letters, your algorithm encrypts this password using the following 3 steps (in this given order):

1. Swap two different characters of the given password (you can do this step zero or more times).
2. Append zero or more lower case English letters at the beginning of the output of step one.
3. Append zero or more lower case English letters to the end of the output of step two.

And the encrypted password is the output of step three.

You have just finished implementing the above algorithm and applied it on many passwords. Now you want to make sure that there are no bugs in your implementation, so you decided to write another program which validates the output of the encryption program. Given the encrypted password and the original password, your job is to check whether the encrypted password may be the result of applying your algorithm on the original password or not.

Input

Your program will be tested on one or more test cases. The first line of the input will be a single integer $\boldsymbol{T}$, the number of test cases ($1 \leq \boldsymbol{T} \leq 100$). Followed by the test cases, each test case is on two lines. The first line of each test case contains the encrypted password. The second line of each test case contains the original password. Both the encrypted password and the original password are at least 1 and at most 100,000 lower case English letters (from 'a' to 'z'), and the length of the original password is less than or equal the length of the encrypted password.

Output

For each test case, print on a single line one word, "YES" (without the quotes) if applying the algorithm on the original password may generate the encrypted password, otherwise print "NO" (without the quotes).

Examples

password.in	Standard Output
3 abcdef ecd cde ecd abcdef fcd	YES YES NO

Problem D. Kids Love Candies

Program:	`candy.(cpp|java)`
Input:	`candy.in`
Balloon Color:	`Green`

Your son's birthday is coming soon (assume that you have a son), and you promised to make the best party ever for him. He will be very happy if he can invite all his friends to this party (he has many friends), but unfortunately you can't invite everyone because you have a limited number of candies, and you want everyone to be happy.

As we all know, kids love to eat a lot of candies of the same type, let's say a kid will be happy only if he can eat at least $\boldsymbol{K}$ candies of the same type.

Given $\boldsymbol{K}$, and the number of available candies of each type, calculate the maximum number of kids where you can make all of them happy by giving each one at least $\boldsymbol{K}$ candies of the same type.

Input

Your program will be tested on one or more test cases. The first line of the input will be a single integer $\boldsymbol{T}$, the number of test cases ($1 \leq \boldsymbol{T} \leq 100$). Followed by the test cases, each test case is on two lines. The first line of each test case contains two integers $\boldsymbol{N}$, the number of different candies ($1 \leq \boldsymbol{N} \leq 100$), and $\boldsymbol{K}$, the minimum number of candies which will make a kid happy as described above ($1 \leq \boldsymbol{K} \leq 100$). The second line of each test case contains $\boldsymbol{N}$ integers, separated by a single space, which are the available number of candies of each type. There will be at least 1 candy and at most 100 candies of each type.

Output

For each test case, print on a single line one integer, the maximum number of kids you are asked to calculate as described above.

Examples

candy.in	Standard Output
2 3 2 4 5 7 3 8 4 5 7	7 0

Problem E. The Swapping Game

Program:	`game.(cpp|java)`
Input:	`game.in`
Balloon Color:	`Silver`

Last year you invented a game which can be played using a board and a die (singular of dice), this year you invented another new game which can be played using a single string of some letters.

The game starts with a string of N lower case English letters ('a' to 'z'), and you can only swap two different characters in this string, and you can make this step zero or more times. Your goal is to reach the lexicographically smallest string after doing zero or more moves.

But there are some constraints on the final string. For each position, it must be a letter of some given letters (the given letters are not necessary the same for each position). For example, the first letter must be 'a' or 'b', the second letter must be 'b' or 'c', and so on.

Note that these constraints are on the final string only, which means you can make moves which cause invalid strings to reach a valid string after some more moves.

Given the initial string and the constraints on each position, your task is to write a program to find the lexicographically smallest valid string after making zero or more moves.

Note: When comparing two different strings of the same length, the lexicographically smaller one is the one with a smaller letter on the first place where they differ.

Input

Your program will be tested on one or more test cases. The first line of the input will be a single integer T, the number of test cases ($1 \leq T \leq 100$). Followed by the test cases, each test case starts with a line containing the initial string S which consists of N lower case English letters ($1 \leq N \leq 100$). Followed by N lines each line contains a string C_i which consists of L_i distinct lower case English letters ($1 \leq L_i \leq 5$) which are the valid letters for the i^{th} position in the final string. Each letter in each C_i will appear at least once in S.

Output

For each test case, print on a single line the lexicographically smallest valid string you can get after zero or more moves. If there is no such valid string print "NO SOLUTION" (without the quotes).

Examples

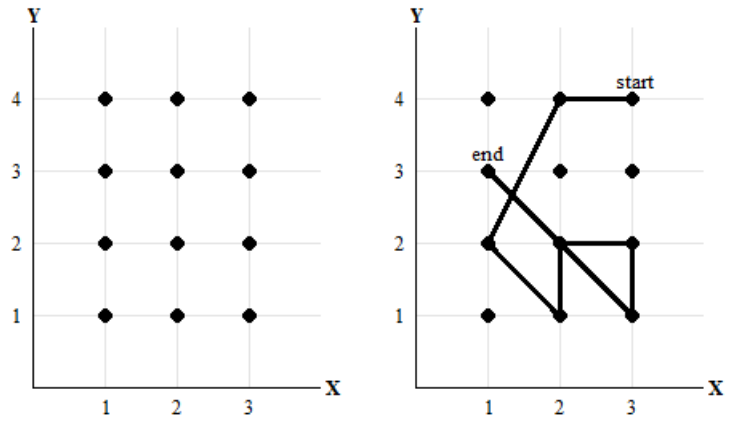
game.in	Standard Output
2 abcde abcde a abcde abcde abcde abcde ab ab ab abcde abcde	bacde NO SOLUTION

Problem F. Lock Pattern

Program:	`pattern.(cpp|java)`
Input:	`pattern.in`
Balloon Color:	`Purple`

One of the ways to lock some phones, is the lock pattern. To unlock your phone you have to draw a secret pattern on a grid of some points, the pattern will be some line segments connecting these points.

Your phone pattern grid consists of four rows with three points in each row. The following image on the left is the representation of the grid, it can be modeled as a 2D plane with X and Y coordinates for each point, for example the top left point is (1,4) and the bottom right point is (3,1). The image on the right is a valid pattern, which connects the following points in this order: (3,4) (2,4) (1,2) (2,1) (2,2) (3,2) (3,1) (1,3).



A valid pattern has the following properties:

1. A pattern can be represented using the sequence of points which it's touching for the first time (in the same order of drawing the pattern), a pattern going from (1,1) to (2,2) is not the same as a pattern going from (2,2) to (1,1).
2. For every two consecutive points A and B in the pattern representation, if the line segment connecting A and B passes through some other points, these points must be in the sequence also and comes before A and B, otherwise the pattern will be invalid. For example a pattern representation which starts with (3,1) then (1,3) is invalid because the segment passes through (2,2) which didn't appear in the pattern representation before (3,1), and the correct representation for this pattern is (3,1) (2,2) (1,3). But the pattern (2,2) (3,2) (3,1) (1,3) is valid because (2,2) appeared before (3,1).
3. In the pattern representation we don't mention the same point more than once, even if the pattern will touch this point again through another valid segment, and each segment in the pattern must be going from a point to another point which the pattern didn't touch before and it might go through some points which already appeared in the pattern.
4. The length of a pattern is the sum of the Manhattan distances between every two consecutive points in the pattern representation. The Manhattan distance between two points (X_1,Y_1) and (X_2,Y_2) is $|X_1 - X_2| + |Y_1 - Y_2|$ (where $|X|$ means the absolute value of X). The length of the pattern in the above image is: $1 + 3 + 2 + 1 + 1 + 1 + 4 = 13$.
5. A pattern must touch at least two points.

Unfortunately you forgot your phone's pattern, but you remember the length of the pattern and a set of points S which are not touched by the pattern for sure (the points which are not in S might and might not be touched by the pattern), and you decided to try all the valid patterns which satisfy what you remember. Before doing this, you have to write a program to calculate for you the number of different valid patterns.

Input

Your program will be tested on one or more test cases. The first line of the input will be a single integer T, the number of test cases ($1 \leq T \leq 100$). Followed by the test cases, the first line of a test case contains two integers L and N separated by a single space. L ($1 \leq L \leq 1{,}000$) is the length of the pattern (as described above), and N ($0 \leq N \leq 12$) is the number of points that you are sure they are not touched by the pattern, followed by N lines each line contains two integers X ($1 \leq X \leq 3$) and Y ($1 \leq Y \leq 4$) separated by a single space, representing the coordinates of one of the points which are not touched by the pattern for sure, the N points are distinct.

Output

For each test case, if there are no valid patterns according to what you remember, print on a single line "BAD MEMORY" (without the quotes), otherwise print the number of different valid patterns.

Examples

pattern.in	Standard Output
3 1 0 2 10 1 1 1 2 1 3 2 1 2 2 2 3 2 4 3 2 3 3 3 4 1 3 1 4 2 4 3 4	34 BAD MEMORY 24

Problem G. Archery

Program:	`archery.(cpp|java)`
Input:	`archery.in`
Balloon Color:	`Red`

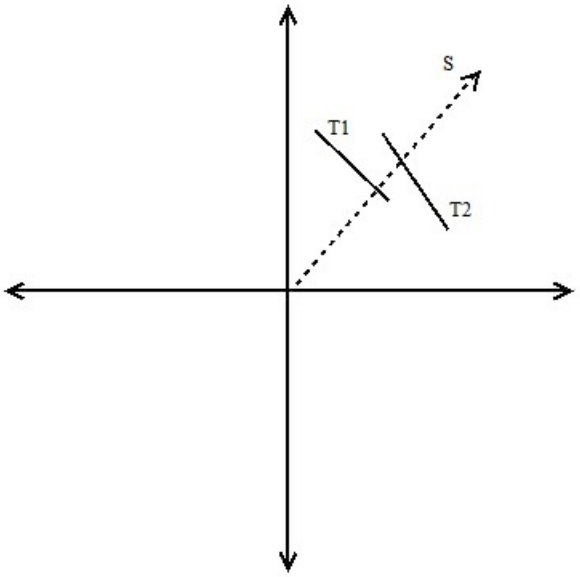
Last summer you were watching all the matches in the 2012 Olympics in London. One of the interesting sports is archery (it's the game of propelling arrows with the use of a bow to hit a target), but in this problem we are dealing with a new type of archery.

In this new type of archery, the player has arrows that can penetrate any target and can go to infinity (the same arrow may hit more than one target), and there will be a lot of targets around the player everywhere, and the targets may intersect and/or overlap with each others.

From the top view you can model the targets as line segments and the player as a point at the origin (point (0,0) is the origin), also there will be no target which intersects with the player's position.

You are really interested to calculate the expected number of targets this player can penetrate using one arrow, if he will shoot his arrow in a random direction (there are infinite number of different directions, and each direction has the same probability to be used for the random shoot).

For example, the following figure explains the first sample test case, where the player is at the origin, and there are two targets T_1 with end points (1,5) and (3,3), and T_2 with end points (3,5) and (6,2), you can notice that there is a region where the player can shoot an arrow and penetrate the two targets, and there are two regions where he can penetrate only one target, and the last region he will not penetrate any target.



Note that a target can be hit at any point between its 2 end points (inclusive).

Input

Your program will be tested on one or more test cases. The first line of the input will be a single integer $\boldsymbol{T}$, the number of test cases ($1 \leq \boldsymbol{T} \leq 100$). Followed by the test cases, each test case starts with a line containing one integer $\boldsymbol{N}$ ($1 \leq \boldsymbol{N} \leq 100$) representing the number of targets in the game. Followed by $\boldsymbol{N}$ lines, the $\boldsymbol{ith}$ line contains 4 integers separated by a single space X_1 Y_1 X_2 Y_2 (-100 $\leq X_1, Y_1, X_2, Y_2 \leq$ 100) representing the $\boldsymbol{ith}$ target end points (X_1,Y_1) and (X_2,Y_2).

Output

For each test case, print on a single line, a single number representing the expected number of targets the player can penetrate using one arrow, rounded to five decimal places.

Examples

archery.in	Standard Output
2 2 1 5 3 3 3 5 6 2 8 3 0 0 3 0 3 -3 0 -3 0 0 -3 0 -3 3 0 3 3 -3 3 -3 3 -3 -3 -3 -3 3 -3 3 -3 3 3	0.20636 2.00000

Problem H. Connecting Islands

Program:	`city.(cpp|java)`
Input:	`city.in`
Balloon Color:	`Black`

The civil war in the Bytelandian ocean has ended. The two Bytelandian islands have decided to unite forming the United Kingdom of Bytelandia. One of the first concerns of the newly chosen government is transportation.

Prior to the war each of the 2 islands had a perfect network of transportation. On any given island, every city was connected by a train to every other city. Connections are bidirectional. During the war, some train lines were disrupted and thus, the trains on these lines were suspended. On any island, no more than 15 trains were suspended. In addition to rerunning the disrupted trains, the government has to form inter-island ferry connections. The ferries will connect every city $\boldsymbol{X}$ to every city $\boldsymbol{Y}$, given that $\boldsymbol{X}$ and $\boldsymbol{Y}$ are not on the same island. Commuting between cities on the same island will only be via trains.

In Bytelandia, everything is binary, including train and ferry ticket prices. A ticket costs either 0 or 1 units of currency. The government decided not to change any ticket prices for trains that were not disrupted during the war. It will assign ticket costs to the trains suspended during the war and to all ferry lines. Priding themselves in being strong mathematicians, Bytelandians have decided to design the ticket system such that for any 3 distinct cities $\boldsymbol{X}$, $\boldsymbol{Y}$ and $\boldsymbol{Z}$,

$C(\boldsymbol{X}, \boldsymbol{Y}) + C(\boldsymbol{Y}, \boldsymbol{Z}) \geq C(\boldsymbol{X}, \boldsymbol{Z})$

where $C(\boldsymbol{X}, \boldsymbol{Y})$ is the ticket price of getting from $\boldsymbol{X}$ to $\boldsymbol{Y}$ (by train if they belong to the same island or by ferry otherwise).

You are given a description of the Bytelandian islands, your goal is to figure out a costs assignment if it is possible.

Input

Your program will be tested on one or more test cases. The first line of the input will be a single integer $\boldsymbol{T}$, the number of test cases ($1 \leq \boldsymbol{T} \leq 100$). Followed by the test cases, each test case is the description of both islands (one after the other). The description of one island starts with one line containing one integer C_k represents the number of cities in this island, followed by C_k lines each one contains C_k characters ($1 \leq C_k \leq 100$). Each character is '0', '1' or 'x'. The $\boldsymbol{jth}$ character of the $\boldsymbol{ith}$ line is the price of the train ticket from city $\boldsymbol{i}$ to city $\boldsymbol{j}$ on that island. A price 'x' means that it is one of the suspended train lines. Each C_k by C_k table is guaranteed to be symmetric (table[$\boldsymbol{i}$][$\boldsymbol{j}$] = table[$\boldsymbol{j}$][$\boldsymbol{i}$] for each different $\boldsymbol{i}$ and $\boldsymbol{j}$), and each table will have a diagonal of '0's (table[$\boldsymbol{i}$][$\boldsymbol{i}$] = '0' for each different $\boldsymbol{i}$), also each table will contain no more than 30 'x's (15 different trains, because each train will be mentioned twice in the table).

Note that the input might contain 3 different cities in the same islands and the trains connecting them are not suspended, and these 3 cities don't satisfy the described condition above.

Output

For each test case, if there is no way to do the costs assignment satisfying the conditions described above, output on a single line "NO" (without the quotes). Otherwise, print "YES" (without the quotes) on the first line followed by $C_1 + C_2$ lines each one contains $C_1 + C_2$ characters (C_1 is the number of cities in the first island, and C_2 is the number of cities in the second island). The $\boldsymbol{jth}$ character on the $\boldsymbol{ith}$ line is the cost of getting from city $\boldsymbol{i}$ to city $\boldsymbol{j}$ (by train if they belong to the same island or by ferry otherwise). In the output, the cities on the first island are numbered from $\boldsymbol{1}$ to C_1 (inclusive) while the cities in the second island are numbered from $C_1 + \boldsymbol{1}$ to $C_1 + C_2$ (inclusive). If there is more than one correct solution, print anyone of them.

Note that you can't change the costs given in the input, and the output table must be symmetric.

Examples

city.in	Standard Output
2 3 011 10x 1x0 2 01 10 2 0x x0 4 010x 1010 0100 x000	YES 01101 10011 10011 01101 11110 NO

Problem I. Contest Hall Preparation

Program:	`hall.(cpp|java)`
Input:	`hall.in`
Balloon Color:	`Blue`

Before the ACM-ACPC regional contest, the site director and the volunteers were very busy preparing for the contest. One of their tasks is to assign a table for each team such that no two teams from the same university are adjacent to each other. The site director decided not to waste his time doing this task and asked the judges to do it. The judges thought this could be a good problem to be used in the contest problems set. As they were very busy preparing for the contest, the judges decided to solve part of the problem and ask the contestants to solve the rest.

The judges will generate a number of layouts for the teams assignment to the tables and will ask the contestants to write a program to check whether each of these layouts is valid or not. If a layout is not valid the program should count how many different universities have at least two of their teams sitting adjacent to each other. "Well, you may use those solutions for the next year's contest", said by the chief judge ***Ahmed Aly*** to the site director.

The contest hall can be represented as a 2D grid of N rows with M cells in each row. Each cell in the grid is either occupied by a team or empty. There could be up to 8 teams adjacent to a single team. A team may have less than 8 adjacent teams if it is seated next to a hall edge or some of its adjacent cells are empty.

For example, in the layout shown in the following figure, team E has 7 adjacent teams, named A, B, C, D, F, G and H, while the adjacent teams to team A are B, D and E.

A	B	C
D	E	F
G	H	

Input

Your program will be tested on one or more test cases. The first line of the input will be a single integer T, the number of test cases ($1 \leq T \leq 100$). Followed by the test cases, each one starts with a line containing 2 integers separated by a single space N and M ($1 \leq N, M \leq 100$) representing the dimensions of the hall, followed by N lines each line contains M integers separated by a single space, representing the tables assignment in this row. Each integer represents the university ID of the team assigned to this table or -1 if it is empty. All universities IDs are positive integers not greater than 100.

Output

For each test case, print on a single line one integer, the number of different universities having at least two of their teams adjacent to each other.

Examples

hall.in	Standard Output
3 3 3 1 2 3 2 2 2 1 1 1 3 3 1 2 3 3 -1 1 2 -1 2 3 3 1 2 3 -1 1 5 1 2 4	2 0 1

Problem J. Math Homework

Program:	`math.(cpp|java)`
Input:	`math.in`
Balloon Color:	`Pink`

Yes, your teacher gave you another hard math homework, and you have to finish it before its deadline.

This homework is about the division operation, and it's a practice for the division by small numbers. You are asked to count the non-negative numbers which consist of exactly N digits (leading zeros are allowed), and they satisfy some division requirements, for example let's say you want to count the numbers which consist of 2 digits and they are divisible by 6 and not divisible by 5, these are the numbers which satisfy these requirements: 06, 12, 18, 24, 36, 42, 48, 54, 66, 72, 78, 84 and 96.

Note that zero is divisible by any positive number (check the third sample test case).

So, you decided to write a program to solve this homework for you, because N can be really large.

Input

Your program will be tested on one or more test cases. The first line of the input will be a single integer T, the number of test cases ($1 \leq T \leq 1{,}000$). Followed by T lines, each line represents one test case, and consists of an integer N ($1 \leq N \leq 10^{18}$) which is the length of the numbers you are asked to count (again, leading zeros are allowed) followed by a space then a string of 6 digits (each digit is '0', '1' or '2'), the i^{th} digit (the left most digit is the digit number 1) is '0' if the numbers shouldn't be divisible by i, and it's '1' if the numbers should be divisible by i, and it's '2' if the numbers can be divisible or not divisible by i.

Output

For each test case, print on a single line one integer, the count of the numbers you are asked to count as described above, since the result may be very large, print it modulo 1,000,000,007 ($10^9 + 7$).

Examples

math.in	Standard Output
4 2 222201 1 111001 1 111111 2 222222	13 1 1 100