Contents

1 Stacked Floating Mountains

The *floating mountains* of Pandora present a challenge for the human scientists, especially geologists and physicists, who have been trying to understand how such structures could exist. While exploring the mountains, the scientists stumbled across interesting *stacked* floating mountain structures, where different mountains appeared stacked above one other, with the larger mountains being higher up in the stack. The scientists were able to calculate the size of each mountain, and they made an interesting observation: that the sizes of the mountains formed a (generalized) *Fibbonacci* sequence.

A sequence of numbers: $x_1, x_2, ..., x_n$, is called a generalized Fibbonacci sequence if, for all $i > 2$,

$$x_i = x_{i-1} + x_{i-2}$$

The standard Fibbonacci sequence is simply a generalized Fibbonacci sequence with $x_1 = x_2 = 1$.

An example of generalized Fibbonacci sequence is: 2, 5, 7, 12, 19, ...

Your goal is to help the scientists verify this conjecture. Specifically, you are to write a program that, given a sequence of numbers, decides whether the sequence is a generalized Fibbonacci sequence or not.

Input/Output Format:

Input: The first line in the test data file contains the number of test cases, n. After that, each line contains one test case. The test case begins with the number of elements in the sequence, k, and then we have k numbers which form the sequence. Assume all numbers are ≥ 0, and that the numbers are all $< 2^{30}$.

Output: For each test case, you are to output "YES" (if the sequence is a generalized Fibbonacci sequence) or "NO" (if it is not).

Examples:

Input:	Output:
3	YES
6 1 1 2 3 5 8	NO
7 1 2 2 4 6 10 16	YES
4 2 10 12 22	

2 Chess Puzzle

Jake and Sully are playing around with a chessboard one night after working with their avatars all day. They decide it would be interesting to place some rooks on the chessboard in a way that no rook can threaten another rook. Since rooks move along rows and columns, this means two rooks may not be on the same row or column. Your goal is to write a program to determine whether any rooks are threatened.

Input/Output Format:

Input: Chessboards are 8x8 boards with positions between (1,1) and (8,8). The input begins with the number of chess boards. Each chessboard is on a separate line and begins with the number of rooks, followed by the column and row positions of each rook.

Output: For each chessboard, your program should output the words ”SAFE” or ”NOT SAFE” on a single line.

Examples:

Input:	Output:

2
3 1 1 2 6 8 8
2 2 3 1 3
	SAFE
	NOT SAFE

3 Life Connections

On Pandora all Navi are connected by friendships. After carefully mapping friendships between different Navi, Grace wants to measure the strength of the connection between pairs of Navi. She decides the way to calculate this is to treat Navi as nodes in a graph, and friendships between Navi as edges. Then the connection strength between two Navi can be defined as the number different shortest paths each could take to visit the other. Your goal is to help her calculate these values.

Given a list of connections between Navi and two Navi u and v, you must compute the number of different shortest paths between u and v. The length of the path is the number of Navi on the path. Two paths are different if they pass through at least one different Navi.

Input/Output Format:

Input: Connections between Navi are described beginning with the line "GRAPH BEGIN". Additional lines lists individual Navi (nodes), followed (on the same line) by their friends (edges). The line "GRAPH END" ends the list of connection descriptions. The next lines describe pairs of Navi for which answers need to be calculated, each on a single line. Following these lines, a new instance of the problem can be given, starting from scratch.

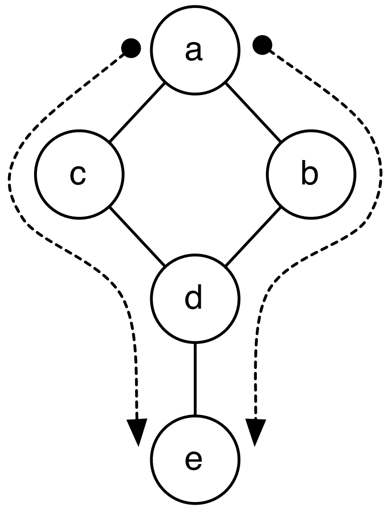
You may assume all Navi are connected (i.e., any Navi can reach another Navi by some path). Not all Navi will have their connections listed on a separate line: the friendships of some Navi may only be implied by the friendships given on other lines.

Output: Your output should consist of pairs of Navi in the same order as in the input, followed by the number of shortest paths between them, both on a single line.

For instance, in the following example the strength of the connection between Navi a and e is 2, since there are 2 paths of length 3 (the shortest possible) from a to e ($a \rightarrow b \rightarrow d \rightarrow e$ and $a \rightarrow c \rightarrow d \rightarrow e$).

Example:

Example Input:	Example Output:
GRAPH BEGIN	a b 1
a b c	a c 1
b d	a d 2
c d	a e 2
d e	b c 2
GRAPH END	b e 1
a b	
a c	
a d	
a e	
b c	
b e	



4 Circle of Friends

After measuring the strength of friendships between different Navi, Grace wants to find groups of Navi who form close-knit friendships. A group of friends has strength k if each Navi in the group has at least k friends within the group. Your goal is to help Grace find the strongest, largest circle of friends for individual Navi.

Input/Output Format:

Input: Connections between Navi are described beginning with the line “GRAPH BEGIN”. Additional lines lists individual Navi, followed (on the same line) by their friends. The line ”GRAPH END” ends the list of connection descriptions. The next lines describe individual Navi to be analyzed, each on a single line. Following these lines, a completely new instance of the problem can be given, starting from scratch.

Some Navi may be only be listed as friends of other Navi (i.e., not all Navi will have their connections listed on a separate line).

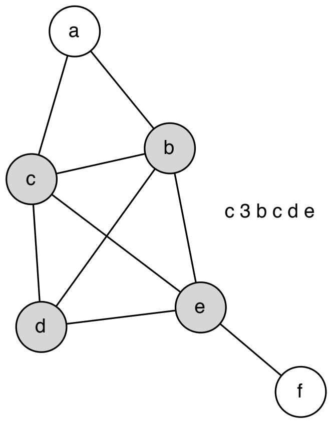
Output: Your output should consist of one line for each Navi analyzed, in the same order they were listed in the input. Each line should contain the name of the Navi, the largest k for which the Navi is a member of some group of friends of strength k, and all the friends in that group (in alphabetical order), including themselves. Every Navi in the group must know the initial Navi either directly or indirectly through some sequence of common friends (i.e., the friendship graph must be connected).

In the example below, Navi c is a member of a group of friends of strength 3: *bcde*. She is also a member of several groups of friends of strength 2 (*bcd*, *bce*, *cde*, ...) but because $3 > 2$, the group of strength 3 is the one that should be output.

Navi f is a member of several groups of strength 1 (ef, bef, def, ...) but the largest one is $abcdef$, so that is the one that should be output.

Example:

Input:	Output:
GRAPH BEGIN	a 2 a b c d e
a b c	b 3 b c d e
b c d e	c 3 b c d e
c d e	d 3 b c d e
e d f	e 3 b c d e
GRAPH END	f 1 a b c d e f
a	
b	
c	
d	
e	
f	



5 Floating Mountain Stability

After receiving your program from Problem 1, the scientists used it to try to verify their conjecture (that the sizes of stacked floating mountains formed a generalized Fibbonacci sequence). However, although they were able to verify the conjecture for a large number of cases, they discovered that there were stacked structures that did not satisfy the property. Further, they also discovered that floating mountains may have *negative weights* (they conjecture that this has to do with some unique properties of "Unobtainium").

The scientists now believe that the sizes of the stacked mountains did follow generalized Fibbonacci sequence property originally (when they were formed), but they believe that some of the mountains in the structures may have been destroyed or may have drifted apart. They further observed that at most 9 consecutive mountains in the stack may be removed without compromising the stability of the structure. They are now trying to verify this new conjecture.

You are to write a program for this purpose. Specifically, given a sequence of numbers, some of which may be negative, you must determine if the numbers are part of a generalized Fibbonacci sequence (let's call it the *original* sequence), such that all consecutive pairs of numbers in the input sequence are less than 10 apart (i.e., fewer than 9 items between any consecutive pair of numbers) in the original generalized Fibbonacci sequence.

As an example, the sequence: 0 6 16, follows this property because the numbers are from the following generalized Fibbonacci sequence:

0 2 2 4 6 10 16

and 0 & 6 are only 4 numbers apart in the generalized sequence.

As another example: the sequence -22 8 77 125, also satisfies the property. Here is the corresponding generalized Fibbonacci sequence:

37 -22 15 -7 8 1 9 10 19 29 48 77 125

Input/Output Format:

Input: The first line in the test data file contains the number of test cases, n. After that, each line contains one test case. The test case begins with the number of elements in the sequence, k ($k < 50$), and then we have k numbers which form the sequence. Assume the numbers are all $> -2^{30}$ and $< 2^{30}$.

Output: For each test case, you are to output "STABLE" (if the sequence satisfies the property) followed by the first five elements of the generalized Fibbonacci sequence (beginning with the first number in the input sequence), or "UNSTABLE" (if it does not). If multiple generalized Fibbonacci sequences are possible, select the sequence with the smallest gap (i.e., number of missing numbers) between the first two numbers.

Examples:

Input:	Output:

3
3 0 6 16
4 -22 8 77 125
4 1 1 1 1
	STABLE 0 2 2 4 6
	STABLE -22 15 -7 8 1
	UNSTABLE

6 Aim It Right!

There are many similarities between the human and the Navi cultures and lifestyles. For instance, their sports and boardgames bear similarities to different sports and games on Earth. But at the same time, there are significant differences (e.g., instead of water polo, they play air polo on their Banshees). There is also a version of Billiards, which is quite similar to our version, but with the difference that their Billiards table has a large circular hole right at the center (see figure). Of course, a person who hits the ball into that hole immediately loses.

Jake and Tsu'tey start a friendly game of Billiards which soon turns into a not-so-friendly game with possibly the leadership of the Omaticaya at stake. In each round, they take turns. At each turn, an impartial judge places the cue ball at one location on the table (call it point **A**) and another ball somewhere else (call it point **B**). The person whose turn it is, must hit the ball at point B from point A with minimum number of rebounds off the walls of the table (see figure).

Your goal is to help Jake win the game and retain his leadership (otherwise he will have to try to tame Toruk once again, and only so many times you can survive that). Given the input parameters, you are to find the minimum number of rebounds needed to get from point A to point B (or declare that it is not possible to do so). Assume the board is perfectly frictionless and the walls are infinitely elastic. So a ball hit at an angle θ_1 will rebound in the opposite direction at exactly the same angle.

Input/Output Format:

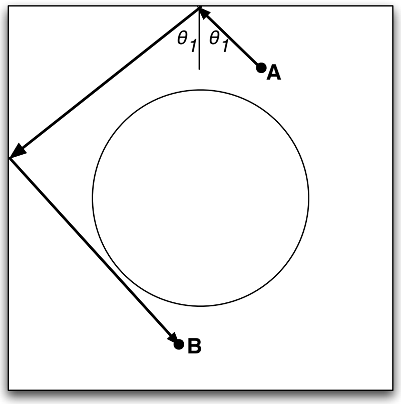
Input: The Billiards table is a square with dimensions 100 by 100. Assume that the center of the table (and hence the center of the hole at the middle) is at (0, 0). You are given the x and y coordinates of the points A and B, and also the radius of the hole in the middle (r). The first line of the input file is the number of test cases. Each line contains 5 integers: A_x, A_y, B_x, B_y, r, where (A_x, A_y) are the coordinates of point A, and (B_x, B_y) are the coordinates of point B. The value r denotes the radius. You can assume that neither point A nor point B are in the hole, and that $r < 50$.

Output: For each test case, you are to find the minimum number of rebounds needed to reach from point A to point B. If it is less than 10, output on a single line "REBOUNDS" followed by the number of rebounds needed. If it is not possible to hit point B from point A in < 10 rebounds, output on a single line "NOT POSSIBLE".

Examples:

Input:	Output:

2
16 -14 38 -27 13
21 44 -38 -17 38
	REBOUNDS 2
	NOT POSSIBLE



7 Navi Navigation

The Navi villages on Pandora are part of gigantic hometrees. Hometrees specialize in producing different types of fruits that Navi like to eat. Neytiri's mother Mo'at asks her to calculate the shortest path between two given hometrees that allows Mo'at to collect every different type of fruit exactly once. Your goal is to help Neytiri calculate these paths. Warning: since there are many hometrees on Pandora, you will not be able to simply examine all possible paths and select the least expensive.

Input/Output Format:

Input: Paths between hometrees are described beginning with the line "GRAPH BEGIN". Additional lines lists individual hometrees (nodes), the type of fruit produced by the hometree, the distance to the neighboring hometrees, followed (on the same line) by their neighboring hometrees (edges). The line "GRAPH END" ends the list of connection descriptions. The next lines describe pairs of hometrees for which answers need to be calculated, each on a single line. Following these lines, a completely new instance of the problem can be given, starting from scratch.

You may assume any hometree can reach any other hometree by some path. Each hometree will be listed at least once as the first item on some line between the GRAPH BEGIN and GRAPH END. The same hometree can be listed more than once with different distance values, but it must always have the same type of fruit assigned to it. Individual connections can appear at most once. It is valid to list only a hometree and its color (specifying no new connections).

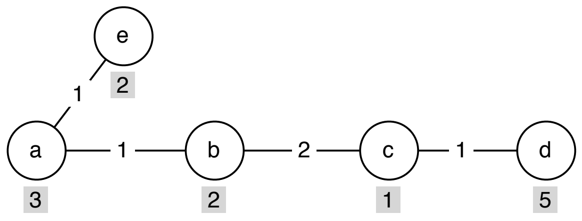
Fruit names will be integers. Not all integers have to be used, however. Your path need only try to collect fruits that at least one tree grows.

Output: Your output should consist of pairs of hometrees in the same order as in the input, followed by the length of the shortest path between them that collects each type of fruit exactly once. If such a path does not exist, you should output "NONE".

In the first example below, the path $a \rightarrow b \rightarrow c \rightarrow d$ collects all the fruit types (1, 2, 3, and 5) and the path has length 4.0. No good path exists between a and c, however: the path $a \rightarrow b \rightarrow c \rightarrow d \rightarrow c$ would collect all the fruit types, but it collects fruit 1 twice!

Examples:

Input:	Output:
GRAPH BEGIN a 3 1 b e b 2 2 c c 1 1 d d 5 e 2 GRAPH END a d a c GRAPH BEGIN e 1 2 f e 1 3 g f 2 g 2 h 3 4 g f GRAPH END h e	a d 4.0 a c NONE h e 6.0



Shaded boxes give the type of fruit grown at each node.

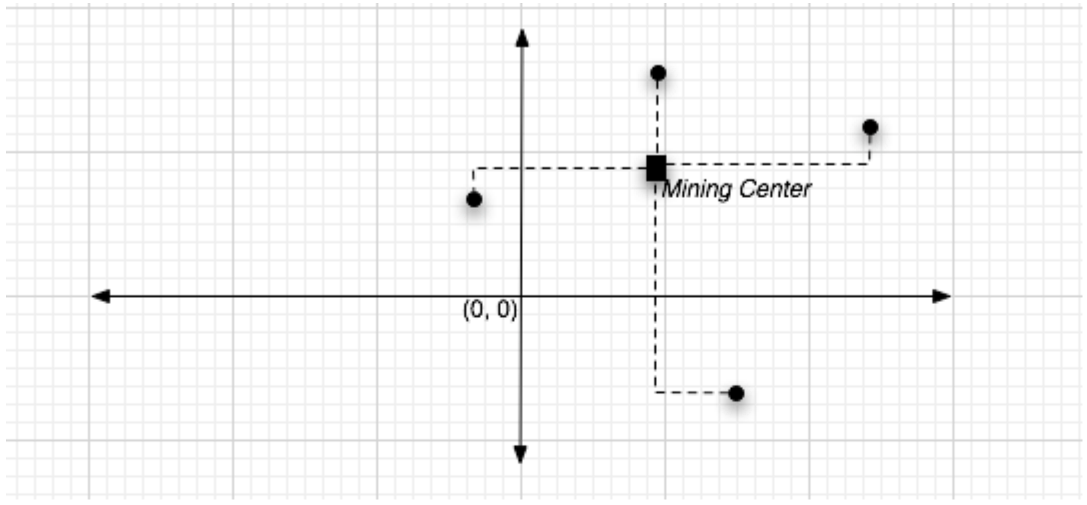
8 Locate Mining Center

As part of a peace treaty with the Navi, the humans are allowed mine for Unobtainium in a remote, deserted area of Pandora. The scientists have identified many possible excavation sites, and are now trying to figure out where to place the *Mining Center*. The placement of the mining center is further constrained by the fact that the robots that will carry the Unobtainium from the excavation sites to the mining center can only walk along prespecified Grid lines (see figure). The goal is now to find the location of the mining center so that the maximum distance to the excavation sites is minimized. You are to write a program for finding that location.

Specifically, you are given the x- and y-coordinates of the excavations sites, $(x_1, y_1), (x_2, y_2), ...(x_n, y_n)$ (n denotes the number of excavation sites). You are to find the coordinates for the Mining Center, (x_0, y_0), so that the maximum of the distances between the mining center and the excavation sites is minimized. All coordinates must be integers. The distance metric to be used is the *rectilinear* distance (also called *Manhattan* distance). Specifically, the rectilinear distance between (x_0, y_0) and (x_i, y_i) is:

$$|x_i - x_0| + |y_i - y_0|$$

Further, if there are multiple locations which qualify, then you must find the location that is closest to the origin by Euclidean distance metric. Since origin is at (0, 0), this corresponds to minimizing $\sqrt{x_0^2 + y_0^2}$.



In other words, given the input $(x_1, y_1), ..., (x_n, y_n)$, your goal is to find (x_0, y_0) such that:

$$\max_i(|x_i - x_0| + |y_i - y_0|)$$

is minimized, and if there are multiple answers, then $\sqrt{x_0^2 + y_0^2}$ is the smallest among all such answers.

Input/Output Format:

Input: The first line in the test data file contains the number of test cases, n. After that, each line contains one test case. The test case begins with the number of mines, followed by the x and y coordinates of each mine. All coordinates must be integers. Very large coordinate values may be used (million+), so brute force methods will not work.

Output: For each test case, you are to output a line beginning with "LOCATION", followed by the x and y coordinates of the mining center. All coordinates must be integers.

Examples:

Input:	Output:

2
4 100 0 40 0 -10 0 20 0
3 245 692 -772 -647 330 526
	LOCATION 45 0
	LOCATION -121 -120

Practice 1 Checkerboard Rows

Colonel Quaritch is playing checkers one day, and decides it would be interesting to write a program to calculate the maximum number of pieces on a single row.

Input/Output Format:

Input: Checkerboards are 8x8 boards with positions between (1,1) and (8,8). The input begins with the number of boards. Each board is on a separate line and begins with the number of pieces, followed by the column and row positions of each piece.

Output: For each checkerboard, your program should output the maximum number pieces on any one row.

Examples:

Input:	Output:
2	1
3 1 1 2 6 8 8	3
4 2 3 1 3 1 4 5 3	

Practice 2 Mining Maps

Administrator Selfridge is analyzing possible mining routes on Pandora. He has collected some data in the form of a graph. Your goal is to help him collect some information about each graph.

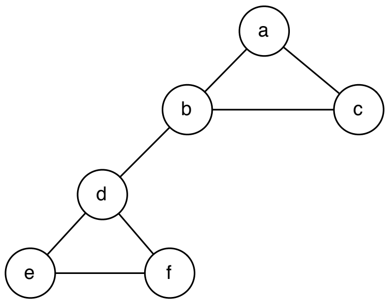
Input/Output Format:

Input: Connections between mines are described beginning with the line "GRAPH BEGIN". Additional lines lists individual mines (nodes), followed (on the same line) by their neighboring mines (edges). The line "GRAPH END" ends the list of path descriptions. The entire problem may be repeated as desired, starting from scratch each time. Some mines may appear only as neighboring mines, without being described on a separate line.

Output: Your output should consist one line (for each graph analyzed), consisting of "NODES" followed by the number of nodes, followed by "EDGES" and the number of edges in the graph.

Examples:

Input:	Output:
GRAPH BEGIN a b b c GRAPH END GRAPH BEGIN a b c b c d d e f e f GRAPH END	NODES 3 EDGES 2 NODES 6 EDGES 7



Practice 3 Hauling Ore

Administrator Selfridge is analyzing possible mining routes on Pandora. He has collected some data in the form of a graph. The latest ore carriers can visit exactly 3 mining camps. Your goal is to help him find out which mines may be visited from other mines with 2 stops (but not fewer).

Input/Output Format:

Input: Connections between mines are described beginning with the line "GRAPH BEGIN". Additional lines lists individual mines (nodes), followed (on the same line) by their neighboring mines (edges). The line "GRAPH END" ends the list of path descriptions. The next lines list mines for which answers need to be calculated, each on a single line. Following these lines, a completely new instance of the problem can be given, starting from scratch.

Some mines may appear only as neighboring mines, without being described on a separate line. Mine names can be arbitrary strings, as long as they don't contain any whitespace.

Output: Your output should consist one line (for each mine analyzed), consisting of the name of the mine, followed by the mines, in alphabetical order, that can be visited with exactly 2 stops but not fewer, starting from the given mine.

Examples:

Input:	Output:
GRAPH BEGIN	d
a b c	e f
b c d	d
d e f	a c
GRAPH END	b f
a	b e
b	
c	
d	
e	
f