**2-1444. Elephpotamus**

**Input**

The number of pumpkins on the lawn *N* (3 ≤ *N* ≤ 30000).

In the next *N* lines, the coordinates of the pumpkins are given in the order corresponding to their numbers.

It is guaranteed that there are no two pumpkins at the same location and there is no straight line passing through all the pumpkins.

**Output**

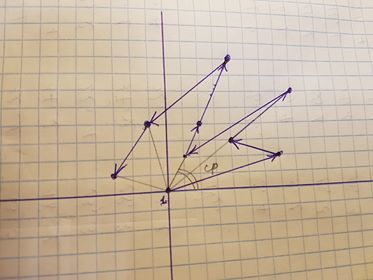
In the first line write the maximal number *K* of pumpkins that can be fed to the elephpotamus. In the next *K* lines, output the order in which the animal will eat them, giving one number in a line. The first number in this sequence must always be 1.

**Notes:**

* The elephpotamus never cross their path.
* The path must always start with the pumkin No.1

**Solution:**

1. Let’s build a coordinate system with the pumkin No.1 is the center.
2. Sort all remaining points in anticlockwise order according to their angle value(which is converted from tan value) and their coordinates.
3. Points with smaller angle go to the front
4. Point with the same angle value but closer to pumkin No.1 go to the front.
5. Our path will go in anticlockwise order and from closer to further, which look pretty much like in this pic



**3-1067. Disk tree**

Your task is to write a program that will help Bill to restore his state of the art directory structure by providing nicely formatted directory tree.

**Input**

*N* (1 ≤ *N* ≤ 500) that denotes a total number of distinct directory paths.

Then *N* lines with directory paths follow. Each path is listed once and consists of a number of directory names separated by a back slash ("**\**").

**Output**

Write to the output the formatted directory tree.

**Solution:**

I use a 2 dimentional linked list to store our formatted tree. Each node has 2 child node

* A node to the next name in the current directory
* A node to its subdirectories

Since all the path start from ~HOME directory, for each path we will start from the root of our tree and search for names.

Names and shall be listed in lexicographic order. So we only search while the current string is smaller than our directory name. If it doesn’t exist in our tree yet, add it into the tree.

**3-1494 Monobilliards**

One has to pocket successively the balls with numbers 1, 2, …, *N* into the only pocket (exactly in this order).

While Mr. Chichikov was playing, the inspector several times came up to the table and took out from the table's pocket the last of the pocketed balls.

In the end it turned out that Chichikov had pocketed all the balls and the inspector had taken out and inspected them.

**Input**

The first line contains the number of billiard balls *N* (1 ≤ *N* ≤ 100000). In the next *N* lines there are the numbers of the balls in the order in which the inspector took them out from the pocket.

**Output**

Output the word "Cheater" if Chichikov could not pocket all the *N* balls in the right order, otherwise output "Not a proof".

**Solution**

While Mr. Chichikov was playing, the inspector several times came up to the table and took out from the table's pocket the **last** of the pocketed balls. So this work in a LIFO way, which is similar to stack.

Let’s call

* a[] is the list of pocketed balls in the table’s pocket, a[] work as a stack.
* a[i] is the last element of array a[].
* k is the number of the ball that the inspecter took.

For each k,

1. if k>a[i] then add the next number until we meet k into a[], then pop(k) from the stack.
2. If k=a[i] then pop(k).
3. If k<a[[i] then Chichikov cheated.

**3-1521. War game 2**

**Problem**

In accordance with this scheme, the war games are divided into **N** phases; and **N** soldiers, successively numbered from 1 to **N**, are marching round a circle one after another, i.e. the first follows the second, the second follows the third, ..., the (**N**-1)-th follows the **N**-th, and the **N**-th follows the first.

At some phase, the circle is left by a soldier, who is marching **K** positions before the one, who left the circle at the previous phase.

A soldier, whose number is **K**, leaves the circle at the first phase.

**Input**

The only line contains the integer numbers **N** (1 ≤ **N** ≤ 100000) and **K** (1 ≤ **K** ≤ **N**).

**Output**

You should output the numbers of soldiers as they leave the circle. The numbers should be separated by single spaces.

**Solution:**

This problem demand updating the remaining position between all 2 sodiers after every phases, which is a pretty big work.

We will use Interval tree to deal with this assignment. Each node of the tree will be the number between soldier i and soldier j.

Let’s call p is the soldier who left in the previous phase. We use binary search to search for the next soldier who will leave in the current phase by comparing the number of soldier between soldier No.X and p with k. Update interval tree after every phase.

**4-1080. Map Coloring**

We consider a geographical map with *N* countries numbered from 1 to *N* (0 < *N* < 99). Write a program which determines whether it is possible to color the map only in two colors — red and blue in such a way that if two countries are connected their colors are different. The color of the first country is red.

**Input**

On the first line is written the number *N*.

On the following *N* lines, the *i*-th line contains the countries to which the *i*-th country is connected, the last one which is 0 and marks that no more countries are listed for country *i*.

If a line contains 0, that means that the *i*-th country is not connected to any other country, which number is larger than *i*.

**Output**

0 corresponds to red color, and 1 — to blue color. If a coloring is not possible, output the integer −1.

**Solution:**

We use DFS to color those country. 1- for blue color, (-1)-for red color (instead of 0).

Because may be not all the country are connected, we need to check each country whether it’s colored. If it’s not colored, paint it red. Starting with country No.1.

**Paint(k,code):**

Where:

* k is the number of current country
* Code is the color: 1- for blue color, (-1)-for red color.

Here is what we gonna do in this method:

1. Mark this country is colored(code).
2. Determine the color of the next country.
3. For each country which is connected to k:
   1. If its color == nextColor continue;
4. If its color != 0 then there is no way to solve the problem. Exit
5. Else paint it with nextColor.

**4-1160. Network**

There will be N hubs in the company. each hub must be accessible by cables from any other hub (with possibly some intermediate hubs).

a plan of hub connection, that the maximum length of a single cable is minimal.

**Input**

The first line contains two integer: N - the number of hubs in the network (2 ≤ N ≤ 1000) and M — the number of possible hub connections (1 ≤ M ≤ 15000).

**Output**

Output first the maximum length of a single cable in your hub connection plan (the value you should minimize). Then output your plan: first output P - the number of cables used, then output P pairs of integer numbers - numbers of hubs connected by the corresponding cable. Separate numbers by spaces and/or line breaks.

**Solution:**

I use Kruskal algorithm to find minimun-spanning-tree.

First, sort all the cable according to their length from shorter to longer.

Following this order, add these edge until all the hubs are connected(when we added n-1 cables into the tree)

**4-1450. Russian Pipelines**

**Problem**

N transfer station. For each of M pipelines the numbers of stations A[i] and B[i], which are connected by this pipeline, and its profitability C[i] are known.

Each two stations are connected by not more than one pipeline.

The pipelines are unidirectional. More over, if it is possible to transfer the gas from the station X to the station Y (perhaps, through some intermediate stations), then the reverse transfer from Y to X is impossible. gas arrives to the starting station number S and should be dispatched to F.

find a route to transfer the gas from the starting to the final station. A profitability of this route should be maximal.

the gas transfer between the starting and the final stations may appear to be impossible...

**Input**

The first line contains the integer numbers N (2 ≤ N ≤ 500) and M (0 ≤ M ≤ 124750). Each of the next M lines contains the integer numbers A[i], B[i] (1 ≤ A[i], B[i] ≤ N) and C[i] (1 ≤ C[i] ≤ 10000) for the corresponding pipeline. The last line contains the integer numbers S and F (1 ≤ S, F ≤ N; S ≠ F).

**Output**

If the desired route exists, you should output its profitability. Otherwise you should output "No solution".

**Solution:**

I use Ford-Bellman algorithm to find the maximum profit of every station. The starting point is S.

The maximum number of edges in the path between X and Y is n-1. So we just need to loop n-1 times, each time we go through all the egdes (a,b,w) and update best profit value of all station using this formular:

prf[b] = max(prf[b], prf[a] + w);