Algorithms and Data Structures

Grandma Alberta's crochet (`grandma`)

Problem text

Slides originali su: judge.science.unitn.it/slides/asd23/prog2.pdf

The legend of Nonna A.

Every once in a while, in our homes, we come across those delicate crochet works known as **doilies**. Some are simple and straightforward, others incredibly intricate, white or in the most varied colors. They are everywhere, decorating our furniture and adding a touch of elegance and tradition. But have you ever stopped to think about where they come from? How do they reach our homes?
The answer is shrouded in mystery and magic. There is a legend, handed down from mouth to mouth, which tells of an old lady known as Grandma A. No one knows her real name, nor where she lives, but it is said that she is the tireless creator of all these doilies. She used to teach algorithms, now she is retired and every night, while everyone is sleeping, Grandma A. works on her crochet, creating those small cotton masterpieces non-stop.

A meticulous grandmother

Grandma A. works at the same time at N_c doilies using N_g balls. Each ball can contribute to one or more doilies, and each doily can be made with one or more balls.[1] There may be already completed doilies that are not connected to any balls of yarn and balls that are not used to produce any doilies.

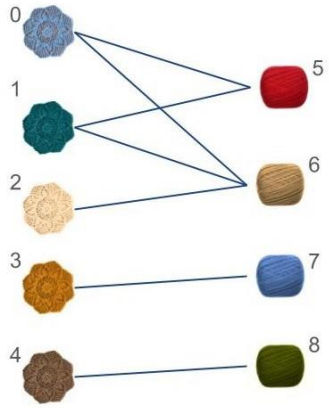
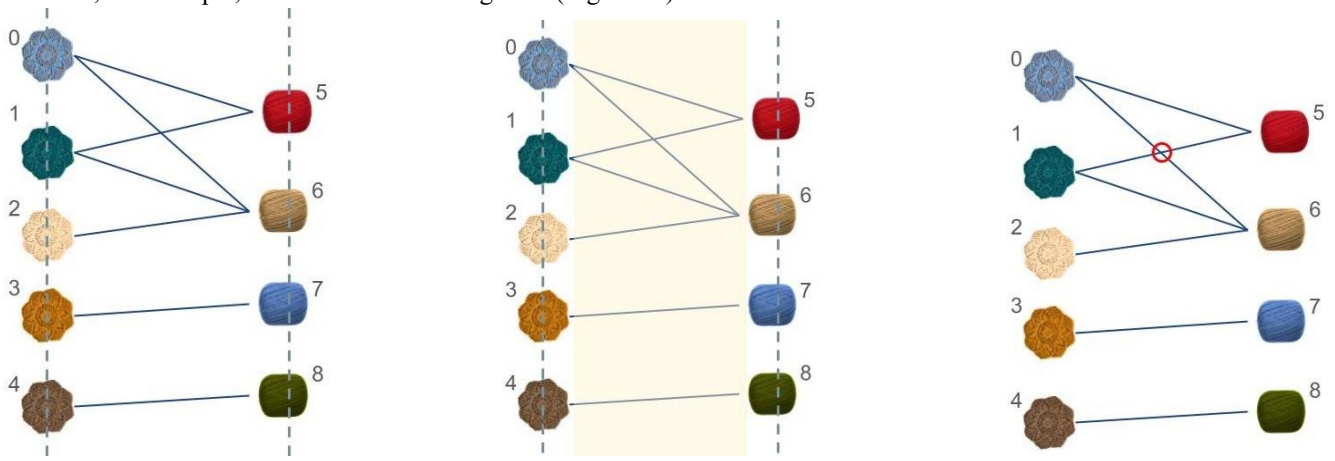


Figure 1: Example with 5 doilies (left) and 4 balls (right), each ball is tied by a thread to the doilies in which it is used. When complete, the doilies are not connected to any ball of yarn. There may be unused balls.

Having to manage this enormous amount of work and having to guarantee certain quality standards, Nonna A. can only have to be very organized and precise. As you can already see, grandma ordered all the doilies in progress on the left (including the completed doilies), and on the right all the balls (including the unused ones). Both doilies and balls are **ordered vertically on straight lines** (Figure 2a). In addition, the threads are very taut **(they are straight lines)** and **pass only and only within the area delimited by the vertical lines** that create the doilies and balls. In Figure 2b, this area is highlighted in yellow. But that's

[1] For the most crochet enthusiasts among you who are getting angry right now: these are magic balls.

not all! A closer eye will notice that grandma has placed doilies and balls of yarn in order to minimize the number of wire crossings! **A crossing occurs when two distinct threads meet at one point** (excluding when they touch balls or doilies). In this case, for example, the number of crossings is 1 (Figure 2c).



(a) Doilies and balls are arranged on two parallel straight lines.

(b) The wires may cross only within the yellow area between the two lines.

(c) The red circles highlight the crossings between the wires.

Figure 2: Example of possible arrangement of doilies and balls.

The joke of the grandchildren

Grandma A. has quite lively and mischievous grandchildren. And they don't know the legend of Grandma A. because they have never been interested in doilies. On the contrary, doilies have always been quite bored, they find them out of fashion! Wanting to play a joke on Grandma A., the grandchildren enter her workshop, changing the order of doilies and balls, which may not be great anymore! For example, there are 5 crossings in the example in Figure 3.

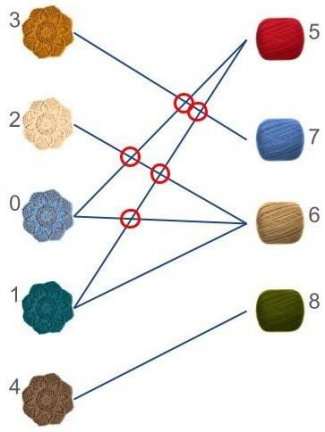


Figure 3: Situation after the grandchildren's joke.

In the confusion, they drop jam on the floor, and the **balls** are now **stuck in their position forever**. Thankfully, the **doilies** were saved, and **you can still move them**! Obviously Grandma A. is angry, and wants to teach her grandchildren a good lesson. And in fact it forces them to tidy everything up! Who knows, maybe this will lead them to appreciate doilies a bit...

Question

Your task is very simple: starting from the situation caused by the grandchildren, find a strategy that allows you to get as close as possible to the number of crossings of the original grandmother's organization. In other words, you need **to minimize the number of wire crossings** .
Remember that you can only reorder the doilies, the balls are stuck forever in their position because of the jam!

Examples

Example I

1. 5 doilies, 4 balls and 7 threads.

2. There is only one crossing of wires, and this number cannot be reduced any further.

3. A possible ordering of the doilies that obtains 1 crossing is the one proposed in Figure 4: **the input situation was already excellent** given the ordering of the balls.

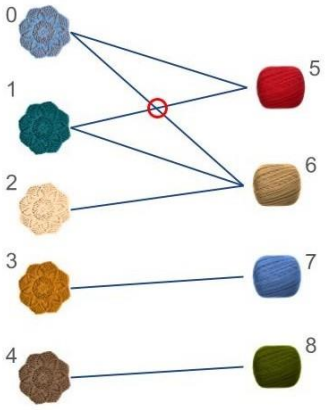


Figure 4: Solution for Example I: The input sorting is already great.

Example II

4. 4 doilies, 4 balls and 4 threads.

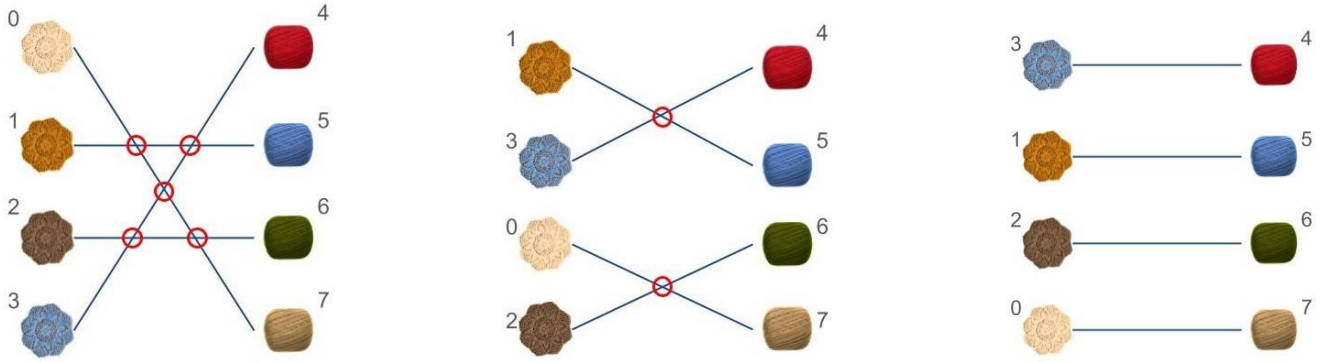
5. There are 5 wire crossings.

6. Possible ordering of doilies are those in Figure 5.

Input/Output

Input: A file with 1+ M rows.

- The first row shows 3 whole numbers: Nc (`int`), Ng (`int`) and M (`int`), respectively the number of doilies, the number of balls and the number of threads joining balls and doilies.



(a) Initial situation: 5 crossings. (b) Intermediate solution: 2 crossings. (c) Optimal solution: 0 crossings.

Figure 5: Possible solutions for Example II.

1. The following M lines describe the wires. Each line i shows 2 integers c_i (`int`) and g_i (`int`), representing the identifiers of the doily and the ball connected by the i-th thread.

This file describes the situation *after* the grandchildren have changed the order of doilies and balls, i.e. it describes the situation on which you will have to act.

Output: A file with your solution proposals, each proposal consists of 3 lines:

2. The first line must show the value of K (`int`) identified, i.e. the number of crossings;
3. the next row must contain N_c values (`int`), i.e. the identifiers of the doilies in the order in which they make the number of K crossings identified;
4. the last line contains three asterisks;

Limits and assumptions

1. $1 < Nc < 5.000$.
2. $1 < Ng < 5.000$.
3. $1 < M < 100.000$.
4. $0 \leq K \leq 4.999.950.000$.
5. Doilies and balls are identified respectively by integers from 0 to $Nc - 1$ (doilies) and from Nc to $Nc + Ng - 1$ (balls).

   The index also represents their respective position in the legal system.
6. The threads are represented by pairs (c_i, g_j) where *there* is a doily and gj is a ball of yarn. There is no *ball already* connected to another ball of g_j. There is no doily *there* connected to another doily c_j
7. There are no double threads, i.e. there is at most one thread described by a pair (c,g) of doilies and balls.
8. There may be doilies and balls of yarn that have no connections.

Test cases

There are 20 test cases in total:

1. In 6 out of 20 cases, the doilies are already optimally ordered;
2. In the remaining cases there are no particular limitations.

Time and memory limits

The time and memory limits are:

1. Time limit: 3 seconds (timeout: *3.3* seconds).
2. Maximum memory: 32 MB.

Score

There are 20 test cases and each test case assigns a maximum score of 5 points for a theoretical maximum total of 100 points.

1. A solution is valid if it complies with all the requests. Invalid solutions score **zero points**. For example, in all the following cases the solution from **zero points**:
    1. The output format is not respected.
    2. the declared number of crossings (`K`) does not match the number of crossings obtained with the specified order.
    3. The declared doily ordering is not a correct permutation of the doilies (incorrect length, repeated balls, etc.).
2. The score is calculated using the **last solution** that ends with three asterisks.

The evaluation parameters are as follows:

1. `K`: the number of crossings of your output order;
2. `initK`: the number of crossings of the input order; • `minK`: lower bound of the minimum number of crossings obtainable.

For each test case for which your solution provides valid output within time and memory constraints, you will get the following P-score:

$$\mathcal{P} = 5 \times \begin{cases} 0 & \text{se } \texttt{minK} \leq \texttt{initK} < \texttt{K} \\ 1 & \text{se } \texttt{minK} = \texttt{initK} = \texttt{K} \\ \dfrac{\texttt{initK} - \texttt{K}}{\texttt{initK} - \texttt{minK}} & \text{se } \texttt{minK} \leq \texttt{K} < \texttt{initK} \end{cases}$$

Bound on the number of crossings

1. `minK` is a *lower bound* at the minimum number of crossings that can be obtained by rearranging the doilies. It may not seem possible to reach this bound, and therefore make 100 points. However, in cases for sufficiency, this bound is always achievable.
2. `initK` is the number of crosses of the trivial solution that is obtained by leaving the doilies in the order in which they are given to you. Although it is possible to "do worse", if your solution has a higher number of crossings, you will still receive **0 points**. In fact, it is the *upper bound* for the test case. If your solution has a number of crossings equal to `initK`, you will receive **5 points** if the initial solution was already optimal (`initK = minK`), or **0 points** if you can do better (`initK > minK`).

Note 1: Since the number of crossings is such that $0 \le K \le 4.999.950.000$ you can use a variable of type `long int` to store it.

Note 2: You don't need to calculate the score of your solution – you'd need the bound `minK` you don't have. Your goal is in any case to **minimize the number of crossings**.

Examples (score)

In the example in Figure 4, the output

```
1
0 1 2 3 4
***
```

it is valid and excellent. In this case `minK` = `initK` = `K` = 1, so P = 5×1 = 5.0.

Invece, l'output

```
6
3 2 0 1 4
***
```

it would have a greater number of crossings than the starting situation. We are in the situation `minK` ≤ `initK` < `K`, so P = 0.

Instead, let's consider the example in Figure 5, and the output

```
2
1 3 0 2
***
```

with `minK` = 0, `initK` = 5 and `K` = 2. In this case we are in the `minK` situation ≤ `K` < `initK` and $\mathcal{P} = 5 \times \frac{5-2}{5-0} =$ 5×0.6 = 3.0.

Evaluation

For project evaluation:

1. Counts the score of the **last source** sent to the system;
2. The project is passed with a score of not less than 30 points;
3. There is a limit of 40 subpositions per group;

Sample datasets

You can find an equivalent dataset to practice with locally:
judge.science.unitn.it/slides/asd23/dataset_nonna.zip.
Note: The inputs provided in the sample dataset did not have an optimal solution. In the output files, you'll find the lower bound `minK` for each test case.

Compilation instructions

Below are instructions for testing your programs on various systems. Supposedly, the source with your code is called nonna.cpp file. The nonna.cpp, `grader.cpp`, and `grandma.h` files must be in the same folder.

GNU/Linux Systems

```
/usr/bin/g++ -DEVAL -std=c++11 -O2 -pipe -static -s -o nonna nonna.cpp grader.cpp
```

Mac OS X Systems

On Mac OS X systems, use the following compilation command:

```
/usr/bin/g++ -DEVAL -std=c++11 -O2 -pipe -o nonna nonna.cpp grader.cpp
```

Se ottente un errore del tipo: `use of undeclared identifier quick_exit`, sostituite in `grader.cpp` l'istruzione `quick_exit(EXIT_SUCCESS);` con `exit(EXIT_SUCCESS);`.

System Windows

For the Windows 10 system you can install the "Windows Subsystem for Linux".[2] You can then install the tools you need to use Visual Studio Code[3] or Visual Studio 2017[4] by following the relevant guides in the notes. When using this system, be careful where you save the files and what name you give them as you may have difficulty with paths that contain spaces and special characters.
Alternatively, or for systems prior to Windows 10 you can install *Cygwin5*, a fully POSIX-compatible environment for Windows. Again, there are guides to configure the common editors available on Windows to use the Cygwin environment, such as Visual Studio[5].
Once Cygwin is installed, you can simulate how much you start on arena by compiling your own source without including the `granna.h` header and the `grader grader.cpp`:

```
/usr/bin/g++ -DEVAL -std=c++11 -O2 -pipe -static -s -o nonna nonna.cpp
```

and run the command as:

```
timeout.exe 3 ./nonna
```

`timeout.exe` will stop the program after 3 seconds.

[2] https://docs.microsoft.com/en-us/windows/wsl/install-win10

[3] https://code.visualstudio.com/docs/cpp/config-wsl

[4] https://devblogs.microsoft.com/cppblog/targeting-windows-subsystem-for-linux-from-visual-studio/

[5]https://www.cygwin.com/

[5] https://devblogs.microsoft.com/cppblog/using-mingw-and-cygwin-with-visual-cpp-and-open-folder/

Example of input/output

File input.txt	File output.txt
3 5 1 0 0 0 0 1 1 2 2 2 2	6 1 0 2 ***

File input.txt	File output.txt
5 4 7 0 0 1 2 3 4 4	5 0 1 2 4 3 ***

File input.txt	File output.txt
5 6 1 3 0 0 10 3 3 4 1 1 2 0	12 1 3 4 0 2 ***

File input.txt	File output.txt
4 4 1 2 0 1 3 3 3 1 1 2 0	3 0 3 1 2 ***