Exam EDAF05

29 Aug 2013

Thore Husfeldt

Instructions

What to bring. You can bring any written aid you want. This includes the course book and a dictionary. In fact, these two things are the only aids that make sense, so I recommend you bring them and only them. But if you want to bring other books, notes, print-out of code, old exams, or today's newspaper you can do so. (It won't help.)

You can't bring electronic aids (such as a laptop) or communication devices (such as a mobile phone). If you really want, you can bring an old-fashioned pocket calculator (not one that solves recurrence relations), but I can't see how that would be of any use to you.

You can answer the questions in Swedish or English. Be neat and tidy. Most questions can be answered by a single word or sentence, some require a few sentences, and some require a well-chosen drawing or example.

Consistency

HateBook is a new social media site where you connect with people you hate.

The Consistency problem is to verify that HateBook satisfies the well-known rule that "the enemy of you enemy is your friend," in the following sense: If Alice hates Bob, and Bob hates Clare, then Alice does not hate Clare, and Clare does not hate Alice.

Input

The input is a complete specification of the social network HateBook. It contains n lines. Every line contains a HateBook member, followed by colon, followed by a comma-separated (possibly empty) list of people he or she hates. Let m denote the total number of connections in the input.

Output

Write `consistent` if the whole social network is consistent. Otherwise output a counterexample: three people that violate the rule.

Examples

Here are two examples with $n = m = 4$.

Input:
```
Juliet:  Alice, Bob
Bob:  Romeo
Alice:
Romeo:  Juliet
```
Output:
Romeo, Juliet, Bob

Input:
```
Juliet:  Alice, Bob
Bob:  Romeo
Alice:  Romeo
Romeo:  Clare
Clare:  Juliet
```
Output:
consistent

Loveletter

(See the Consistency problem for a definition of *HateBook*.)

Romeo does not really hate Juliet, it's just something he has to claim because of pressure from his family. In truth, he loves her with fiery passion. He wants to send her a love letter. The messaging system of HateBook only allows you to send letters to people you don't hate, so he can't send the letter directly. Instead, Romeo will split the letter in half, and send each half on separate paths through the HateBook network. (Why exactly two letters are better than one is explained in figure 1 below. It's a simple (but cute) cryptographic idea that has nothing to do with the exercise, so you can safely ignore it.)

Input

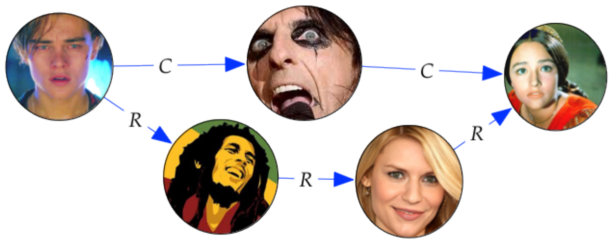
Same as for the Consistency problem. You can assume that `Romeo` and `Juliet` are both part of the network, and that Romeo hates Juliet.

Output

Two sequences of names of HateBook members, each starting in `Romeo` and ending in `Juliet`. No other person may appear on *both* sequences. If p follows q on a sequence then q may not hate p.

If this can't be done, write `impossible`.

Input:
```
Juliet:  Alice, Bob
Bob:  Romeo
Alice:  Claire
Romeo:  Juliet, Claire
Claire:  Claire
```
Output:
Romeo, Alice, Juliet
Romeo, Bob, Claire, Juliet



$$\begin{array}{rcccccccc}
 & \text{I} & \text{L} & \text{O} & \text{V} & \text{E} & \text{Y} & \text{O} & \text{U} \\
M = & 8 & 11 & 14 & 21 & 4 & 24 & 14 & 20 \\
R = & 7 & 3 & 20 & 18 & 24 & 1 & 11 & 7 \\
C = & 1 & 23 & 14 & 12 & 18 & 21 & 5 & 1
\end{array}$$

Figure 1: How to transform a message into two messages that are indistinguishable from random noise. M is the message (in fact, the encoding of "ILOVEYOU"). R is a string of random numbers between 0 and 25. C is the ciphertext, $C[i] = M[i] + R[i] \mod 26$. In particular, both C and R *by themselves* are random noise and can be freely sent through the network, provided nobody unwanted receives both of them. Only Juliet, who receives both messages, can recover M (by computing $M[i] = C[i] - R[i] \mod 26$.) *Warning:* Don't misinterpret the picture. The arrow from Romeo to Bob means that Romeo sends message R to Bob. It's not an edge in the social network.

Party

(See the Consistency problem for a definition of *HateBook*.)

In a fit of good will, all HateBook members arrange a party. You need to seat them around a large table such that nobody sits next to somebody they hate.

Input

Same as for the Consistency problem.

Output

A list of names $l_1, \ldots, l_n$ such that l_i does not hate l_{i+1}, l_{i+1} does not hate l_i (for each $i = 1, \ldots, n-1$), l_n does not hate l_1, and l_1 does not hate l_n.

Input:

```
Juliet:  Alice, Bob
Bob:  Romeo
Alice:  Claire
Romeo:  Juliet, Claire
Claire:  Claire
```

Output:
impossible

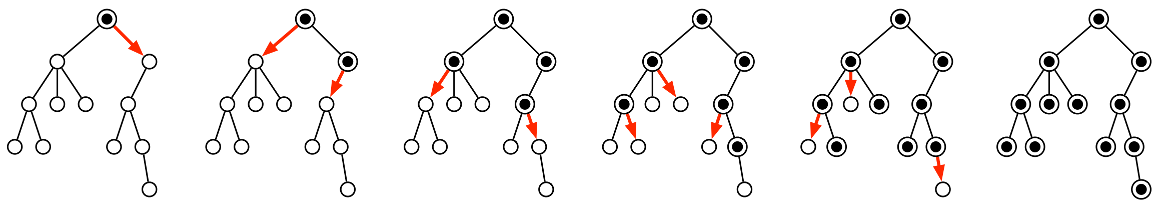
Input:

```
Juliet:  Bob, Romeo
Bob:  Alice
Romeo:  Claire
Alice:  Bob
Claire:  Claire, Alice
```

Output:
Juliet, Alice, Romeo, Bob, Claire

Spread

Suppose we need to distribute the message "I dislike you" to all the nodes in a rooted tree of n nodes. Initially, only the root node knows the message. In a single round, any node that knows the message can forward it to at most one of its children. Design an algorithm to compute the minimum number of rounds required for the message to be delivered to all nodes.


A message being distributed through a tree in five rounds.

Input

Each line contains two integers from $\{1, \ldots, n\}$ denoting an edge in the tree.

Output

The minimum number of rounds it takes to pass the message.

Input:

```
1 2
1 3
2 4
2 5
2 6
3 7
4 8
4 9
7 10
7 11
11 12
```

Output:

```
5
```

Fixpoint

Let A be an array of n integers, $A[0], \ldots, A[n-1]$. The method

```
public static int fixpoint(int[] A)
```

returns i ($0 \leq i \leq n-1$) such that $A[i] = i$, or -1 if no such i exists.

Design an algorithm for this problem faster than linear time.

Exam Questions

There are five exam questions in this set (on page 7 and 8), corresponding to the five algorithmic problems on pages 2–6. Answer them on a separate piece of paper.

1.

(a) (*1 pt.*) What is the running time of the following piece of code in terms of n?

```
for i = 1 to n:
    print i
endfor
for j = 1 to n:
    print j
endfor
```

2. One of the problems in the set can be solved using divide-and-conquer.

(a) (*1 pt.*) Which one?

(b) (*2 pt.*) Describe the algorithm, for example by writing it in pseudocode.

(c) (*1 pt.*) Formulate a recurrence relation that describes the asymptotic running time of your algorithm. State the running time of your algorithm in terms of the original parameters.

3. One of the problems can be efficiently solved using standard graph traversal methods (such as the ones used for breadth-first search, depth-first search, shortest paths, connected components, etc.), without using more advanced design paradigms such as dynamic programming or network flows.

(a) (*1 pt.*) Which one?

(b) (*1 pt.*) Very briefly explain the graph problem you will solve. In particular, tell me what the graph is (What are the vertices? How many are there? What are the edges? How many are there? Are the edges directed? Are there weigths on the edges?) Draw a small example graph.

(c) (*1 pt.*) Describe your algorithm, for example in pseudocode. You are welcome to use existing algorithms from the course book.

(d) (*1 pt.*) State the running time of your algorithm.

4. One of the problems is solved by dynamic programming.

(a) (*1 pt.*) Which one?

(b) (*3 pt.*) Following the book's notation, we let $\mathrm{OPT}(i)$ denote the value of a partial solution. (Maybe you need more than one parameter, like $\mathrm{OPT}(i, j)$. Who knows?) Give a recurrence relation for OPT, including relevant boundary conditions and base cases.

(c) (*1 pt.*) State the running time and space of the resulting algorithm.

5. One of the problems in the set is easily solved by a reduction to network flow.

(a) (*1 pt.*) Which one?

(b) (*3 pt.*) Describe the reduction. Be ridiculously precise about which nodes and arcs there are, how many there are (in terms of size measures of the original problem), how the nodes are connected and directed, and what the capacities are. Do this in general (use words like "every node corresponding to a giraffe is connected to every node corresponding to a letter by an undirected arc of capacity the length of the neck"), and also draw a small, but *complete* example for an example instance. What does a maximum flow mean in terms of the original problem, and what size does it have in terms of the original parameters?

(c) (*1 pt.*) State the running time of the resulting algorithm, be precise about which flow algorithm you use. (Use words like "Using Bellman–Ford (p. 5363 of the textbook), the total running time will be $O(n^{17} \log^{-3} \epsilon + \log^2 m)$.")[1]

[1]This part merely has to be correct. There is no requirement about choosing the cleverest flow algorithm.

6. We will show that Spread belongs to NP.

(a) (*1 pt.*) Is Spread a decision problem? Answer "yes" or "no". If "no", describe the decision version of Spread: what are the inputs, what are the outputs?

(b) (*1 pt.*) Describe a certificate for Spread. In particular, give an example of such a certificate for a small instance. How long is this certificate in terms of the instance size?

(c) (*1 pt.*) Describe very briefly how your certificate can be checked. In particular, what is the running time of that procedure?

7. One of the problems in the set is NP-complete.[2]

(a) (*1 pt.*) Which problem is it? (Let's call it P_1.)

(b) (*1 pt.*) The easiest way to show that P_1 is NP-hard is to consider another NP-hard problem (called P_2). Which one?

(c) (*1 pt.*) Do you now need to prove $P_1 \leq_P P_2$ or $P_2 \leq_P P_1$?

(d) (*3 pt.*) Describe the reduction. Do this both in general and for a small but complete example. In particular, be ridiculously precise about what instance is *given*, and what instance is *constructed* by the reduction, the parameters of the instance you produce (for example number of vertices, edges, sets, colors) in terms of the parameters of the original instance, what the solution of the transformed instance means in terms of the original instance, etc.

[2]If P = NP then *all* these problems are NP-complete. Thus, in order to avoid unproductive (but hypothetically correct) answers from smart alecks, this section assumes that P $\neq$ NP.