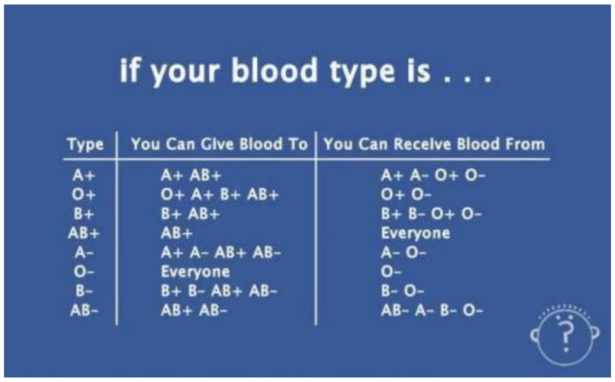
Introduction to Analysis of Algorithms	Homework 4
4820 Fall 2020	Due Tuesday, October 13, 2020

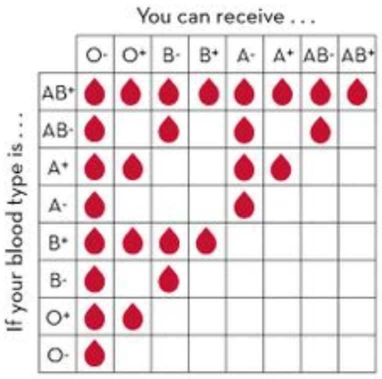
Instructions. Submit a .pdf file with your typeset solutions to Questions 1 and 2 and a text file with your source code for Question 3 in a .zip file to CMS (just the source code please, do not include the compiled object or other compilation artifacts). Remember that when asked to design an algorithm, you must also prove correctness and analyze its running time. All problems are worth 10 points.

1. K&T Ch. 7 Ex. 8(a). Let's make it a little more realistic by including the Rh factor (+/−). In addition to the rules for the antigens A and B stated in K&T, the rule for the Rh factor is that an Rh-positive person cannot donate blood to an Rh-negative person. Besides that, the Rh factor imposes no other constraints and is independent of the rules for the other antigens. This gives eight blood types O+, O–, A+, A–, B+, B–, AB+, AB–. Here are two charts showing who can donate blood to whom:



if your blood type is . . .

Type	You Can Give Blood To	You Can Receive Blood From
A+	A+ AB+	A+ A- O+ O-
O+	O+ A+ B+ AB+	O+ O-
B+	B+ AB+	B+ B- O+ O-
AB+	AB+	Everyone
A-	A+ A- AB+ AB-	A- O-
O-	Everyone	O-
B-	B+ B- AB+ AB-	B- O-
AB-	AB+ AB-	AB- A- B- O-



You can receive . . .

If your blood type is . . .

	O-	O+	B-	B+	A-	A+	AB-	AB+
AB+	●	●	●	●	●	●	●	●
AB-	●		●		●		●	
A+	●	●			●	●		
A-	●				●			
B+	●	●	●	●				
B-	●		●					
O+	●	●						
O-	●							

Figure 1: *Blood type charts.*

Reformulate the problem to account for these eight blood types instead of just the four mentioned in K&T, and give an efficient algorithm for determining whether the hospital has enough supplies on hand to satisfy the projected week's demand.

2. Give an efficient algorithm for the s, t*-connectivity problem*: Given a directed graph $G = (V, E)$, nodes $s, t \in V$, $s \neq t$, and a number $k \geq 0$, decide whether there exist k edge-disjoint paths from s to t, and find them if so. A set of paths is *edge-disjoint* if no pair of them share an edge, but they may share nodes.

3. A certain underdeveloped country contains a remote region with n villages and m roads, each road connecting two villages. However, the villages in the region are currently not fully connected by roads. It is desirable to be able to get from any village in the region to any other village on a sequence of roads, possibly passing through other villages along the way, but there are currently some pairs of villages for which this is not possible, as there is no sequence of roads that connects them.

   As a road engineer for the country, you have been commissioned with the task of building enough new roads so that every village in the region is accessible to every other village by a sequence of roads. But you have been given a limited budget and told to construct only as many roads as necessary. What is the minimum number of roads you need to build?

   In this programming exercise, you will implement a program to answer this question. The input to your program will be a representation of the current road system for the region. Your program should output a single number, namely the minimum number of new roads needed for full connectivity.

Input format As before, the input to your program will be provided as an ASCII character stream on stdin. It will consist of several lines separated by a newline character (`'\n'`, `0x0A`).

- The first line will contain two non-negative integers n m separated by a space, denoting the number of villages and the number of currently existing roads, respectively. We will take the names of the villages to be the numbers from 0 to $n-1$, inclusive.
- Each of the next m lines will contain two integers u v separated by a space, denoting a road between villages u and v. The roads are assumed to be bidirectional, meaning that traffic may travel in either direction.

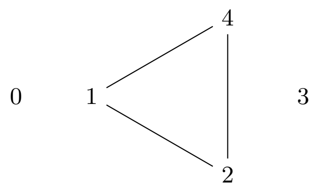
Output format Your program should produce a single integer terminated with a newline.

We guarantee that any input we provide to test your code will conform to the format described above, so you do not need to check for format errors. If n and m are the integers provided on the first line, there will be exactly m roads, every road will be listed exactly once, and every village name u in a road specification will satisfy $0 \le u \le n-1$.

We will test your code on large instances of the problem, so make sure your implementation is efficient.

Important points For full credit, your implementation must use only $O(n)$ storage, where n is the number of villages. This means that you do not have enough space in memory to store all the roads at once, as m could be as big as quadratic in n. In particular, a solution based on depth-first or breadth-first search may not be feasible. However, there is another approach that you can use; see the 9/14 and 9/16 lectures and K&T §4.6. There is also some relevant information on the handouts page that you might find useful.

Example The road system



might have the input representation

```
5 3
1 4
1 2
4 2
```

and should generate the output

```
2
```

You may use any of the following languages: Python, Java, C, C++, or OCaml. Please format your code attractively and provide copious comments. As before, we will grade your program using an autograder. The autograder can be accessed at https://cs4820.cs.cornell.edu/, code named *Villages*. The source code submitted to CMS must match the latest version uploaded to the autograder.