Connected Components Warmup

2015-01-30, rev. 2d1984d

Description

This is the first assignment in BADS/SGDS. It is supposed to be very
simple; the main objective is to form groups, set up your program-
ming environment, download the textbook's code, find a way to
include the book's libraries in you java environment. The algorithmic
problem itself is very simple and basically already solved in section
1.5 of the textbook: Form pairwise connections between sites and at
the end report if sites 0 and 1 are in the same component.

Input

The input contains an integer (the number of sites) followed by a
sequence of pairs of integers (connections), like the book's example
files `tinyUF.txt`, `mediumUF.txt`, and `largeUF.txt`.

The contents of `tinyUF.txt`:

```
10
4 3
3 8
6 5
9 4
2 1
8 9
5 0
7 2
6 1
1 0
6 7
```

Output

The string `true` if 0 and 1 belong to the same connected component
after forming all connections in the input. Otherwise `false`.

Example

```
% java UFW < tinyUF.txt
true
```

Requirements

Your class has to be called UFW and reside in a file called `UFW.java`.
Your class must inlude a single, static main method and nothing
else. It should refer to one of the union–find data structures from
the books; download that code and put it into the same directory
as `UFW.java`. Do not modify that file. Your solution must use as
much code from the book as possible—the exercise can be solved
by copy–paste and writing one new line of code. You have to use
`StdOut.println` for output and `StdIn.readInt()` for input. Your pro-
gram must communicate with standard input and output from the
command line exactly as shown.

Deliverables

Upload exactly two files:

1. the source code of `UFW.java`

2. A report in PDF. Use the report skeleton on the next page.

Tips

The main difficulty in this exercise is to convince your java setup to find the `StdOut` and `StdIn` classes in the book's standard library. This is the main point of this exercise, so take it seriously.

In fact, before starting with the exercise itself, try to make the following program run:

```
public class Hello
{
  public static void main (String[] args)
  {
    StdOut.println("Hello world!");
  }
}
```

Once that works, change `Hello.java` so that reads you name (in a text file) from Standard Input and returns it to you. The functionality should be like this:

```
% cat myname.txt
Bob
% java Hello < myname.txt
Hello Bob!
```

Do *not* avoid the issue by simply copying `StdOut.java` into you current directory, or using `System.out` and `System.in`. There are several suggestions for how to set up the book's standard library on the book website, under *Web Resources, Code*. You are *strongly encouraged* to use the setups called Windows Command Prompt (manual) or Mac OS X Terminal (manual) or Linux Command Line (manual), depending on which architecture you are on.

Report: Connected Components Warmup

by Alice Cooper and Bob Marley[1]

Results

input file	0 connected to 1?
`tinyUF.txt`	yes
`mediumUF.txt`	[...]
`largeUF.txt`	[...]

Methods

For the union–find data structure we used [...][2] from the textbook. [...][3]

[1] Complete the report by filling in your names and the parts marked [...]. Remove the sidenotes in your final hand-in.

[2] Replace by `QuickFindUF.java` or whatever you actually used.

[3] Add a single sentence motivating your choice of data structure. For instance "Using blabla instead turned out to take more that 5 blabla on instance blabla." Or "It didn't make much of a difference." Or "We didn't have time to try anything else".