Programming Assignment 2

CSCI4041 – Spring 2010

Released: 16 April 2010
Due: 7 May 2010

Assignment Purpose

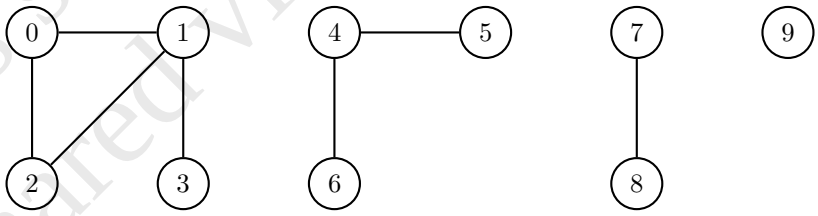
The goal of this assignment is to gain a practical understanding of some basic data structures for disjoint sets (Chapter 21) and some elementary graph algorithms (Chapter 22). You will write a program (in c/c++/java) called *Graphtool* which can read in a text file containing a set of nodes and edges, and answer queries about the disjoint sets and graphs that exist. There are 3 specific queries that you must handle:

1. For a pair of vertices, return whether they are connected or not.
2. For a given vertex, return all vertices that it is connected to.
3. For a given vertex, return the list of all vertices it is directly connected to (its adjacency list).

Input Format

Graph Input

Your program must take a filename as a command-line argument and read that file to get the graph structure. The format of the files is very simple. The first line in the file will contain the number of vertices n in the graph. The rest of the file will contain all the edges that exist in the graph. An edge consists of 2 vertex numbers (*int*s) on a single line separated by a single space.



For example, the file *sample.input* (mirroring the figure above – figure 21.1 (a) in the text) is as follows:

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

Your program should be invoked as follows (for c/c++):
`./Graphtool sample.input`
OR (for java)
`java Graphtool sample.input`
In addition to the file *sample.input*, we will be providing several other graph files that you can use to test your code on more substantial data sets. These files contain data about the connectivity of roadways in the Minneapolis area (see course website).

User Input

Your program must read its queries from standard input, so that when the program is run, it will wait for keyboard input. Your program should read each line of input from standard input, and expect the following format:

`<Command number> <Vertex or space-separated vertex list>`

Where the command numbers correspond to those in the query list above.

Examples

Given the file above, the query:
`> 1 4 8`
is querying whether vertex 4 is in the same set as vertex 8 (in this case the query should print:
`false`
The query
`> 2 0`
queries for the set of vertices connected to vertex 0, and should print
`0 1 2 3`
(order not important). The query
`> 3 2`
should print
`2 0 1`
(order not important).

For the more complex test cases, here are some examples you can use to test your code. There are 7 different files with indices representing road types.

Minneapolis '0' Set

The query
`> 1 1490 1491`
should return `true`. Any other value substituted for 1491 should return `false` (assuming it's in the correct range). The query
`> 3 189`
should return
`189 188 1358` (order not important).

Minneapolis '4' Set

The query
`> 2 1`
should return:
`0 1 4 2 3 5 6 7 8 9 10 32 14 15 16 17 11 12 13 31 18 19 30 33 34 23 24 26 27 28 29 35 36 37 38 39 40 41`
and:
`> 3 35`
should return:
`35 23 36`

Minneapolis '3' Set

The query
`> 2 9674`
should return:
`9674 9675 9676 9677 9678 9679 9681 9682 9685`
The query:
`> 3 37721`
should return:
`37721 37720 37429 37722 37430`

Organization/Submission Format

Your code may be organized however you like, but must compile to a program called Graphtool (please note exact spelling and case). As in the last assignment, your submission must include a Makefile for easy organization-independent compiling. This time you only need one target, Graphtool (or Graphtool.class if you're using java), which will compile your source code using your compiler/language of choice and put the output in the same directory as the Makefile. Again, as in last assignment, please use the submit tool to submit a tarball/zipped directory containing your source code and Makefile.

If you have questions about this assignment please email TA Tim Miller (tim.miller@gmail.com) or Prof. Mokbel (mokbel@cs.umn.edu)