# Programming Assignment 3: Search for connected components

For this assignment, you are asked to use two approaches, *Breath First Search* and *Depth First Search*, to find all connected components of undirected graphs in a given file.

**Program requirement:** The name of the program should be graphcc.java and I will compile and run your program on our Linux server as follows.

javac graphcc.java

java graphcc udGraphs.txt

where udGraphs.txt is the name of the input file. If your program fails to compile, you will get 0 point. I may test your program on a different graph file.

**Input:** The format of the input file is similar to the previous assignments’ inputs files except the graphs are undirected. Consider the following example, which is the beginning of file unGraphs.txt and the first graph: Note: since undirected graphs must be symmetric, if vertex *x* and vertex *y* is directly connected, only one of (*x, y*) and (*y, x*) is shown in the edge section.

20 undirected graphs in udGraphs.txt.

\*\* Note: Since (x,y) in E implies (y,x) in E, only one edge is listed.

\*\* G1: |V|=10 V={0,1,...,9} (u, v) E = {

( 0, 2)

( 0, 7)

( 2, 8)

( 3, 6)

( 3, 8)

( 4, 5)

( 6, 9) }

\*\* G2: |V|=10 V={0,1,...,9}

|  |
| --- |
| (u, v) E = {  ( 0, 7) |
| ( 1, 5) |
| ( 2, 5) |
| ( 3, 6) |

( 7, 8) }

**Output:** Both approaches should give the same results, but the order of visiting vertices may be different. You should list all vertices in one component in one line according to the order of visits. Special requirement,

during the course of computation, if there are more than one choice to proceed, always select the vertex with a small index. With this requirement, the order of visits will be unique for each graph. For example, the output should be as follows: (check graphs G1 and G2 and explain the results)

Connected components of graphs in udGraphs.txt

\*\* G1’s connected components: Breadth First Search:

0 2 7 8 3 6 9

1

4 5

Depth First Search:

0 2 8 3 6 9 7

1

4 5

\*\* G2’s connected components: Breadth First Search:

|  |  |
| --- | --- |
| 0 | 7 8 |
| 1 | 5 2 |
| 3 | 6 |
| 4 |  |
| 9 |  |

Depth First Search:

|  |  |
| --- | --- |
| 0 | 7 8 |
| 1 | 5 2 |
| 3 | 6 |
| 4 |  |
| 9 |  |

## There are two parts of submission: Programs (60%) and Reports (40%)

* Programs 60%, submission on Linux server. The score is based on the correctness and documentation of your programs.

1. Also, at the beginning of **every** method, class, or function that you developed on your own, put a few lines of comments with your name and date/time when you develop the code.
2. Do not declare any package for your programs. You can use any Java IDE on your local computer, but you have to transfer all programs and data files needed to our Linux server. Note that, some IDE may automatically add your programs/classes to some default package. You have to remove the package declaration after you transfer the program to a different location, otherwise, the grading script will fail to handle the special need and run into compilation problems.

You will lose significant points if you fail to follow the rules.

**Report 40%.** You have to write up a report and prepare it in pdf format. The report should include the following items:

1. 10%. The program code is not required in the report. Instead, the *direct output* of your programs on the required input, which is graphs.txt, should be included immediately after the cover page. You can use Unix redirect command, >, to redirect your program’s output to a text file, then copy and paste to your report. Don’t fetch the screen shot. The output on the report must be consistent with the result I get when I run your program on graphs.txt.
2. 10%. A summary of the methods, algorithms and data structures, and the difficulties, if any, the project has faced and how to solve them.
3. 10%. Both time and space efficiency should be analyzed in terms of big-O notations.

All are the same as the previous assignment except that **you don’t have to put output of your program on your report becuase I will run and exam your program directly on some different input files**.