

## 858L: Homework #1 — Fall 2009

**Due:** September 21, 2009, by the start of class.

For this assignment, you should write two programs in the Python programming language. You should work in groups of 2. If you are not familiar with Python, you can find a tutorial here:

<http://docs.python.org/tutorial/>

A helpful library to use is NetworkX, which can be found at <http://networkx.lanl.gov/>. NetworkX lets you read and manipulate graphs. You can use any portion of it except that you should write the algorithms for the following problems without using the “Centrality” functions or the “Clique” functions.

Create two Python files, one for each of the problems below. Turn in your assignment by emailing me ([carlk@cs](mailto:carlk@cs)) a zip file with name “name1\_name2.zip”, where name1 and name2 are the names of the members of your group. The zip file should contain your two .py files.

**Problem 1.** Write a Python program that will respond to the following command:

```
python count_paths.py graphfile u v
```

This should read the graph file given by `graphfile` and then print the number of paths between the node named `u` and the node named `v`. It should not print the actual paths, just the count. The output should consist of a single line, with the format:

```
u v N
```

where `u` and `v` are the node names and `N` is the number of paths.

The graph should be considered to be undirected. The `graphfile` will specify the edges by giving two node names (strings) on each line. For example, the following specifies a triangle graph:

```
a b
b c
a c
```

If there are more than 2 words on a line, your program should *ignore* the extra words.

**Problem 2.** Write a Python program that will respond to the following command:

```
python bipart.py graphfile
```

This should read the graph file given by `graphfile` (same format as in Problem 1). Suppose  $V$  is the set of nodes in the graph. Your program should compute a bipartition  $U, W$  of the nodes such that  $U \cup W = V$  and  $U \cap W = \emptyset$  and the number of edges that cross between  $U$  and  $W$  is as large as possible. It is an NP-hard problem to find the bipartition with the largest number of edges. Use whatever algorithms, techniques, references, tricks you want to find the largest bipartition possible.

The output should list on the 1st line the nodes in  $U$  in alphabetical order. It should list the nodes in  $W$  in alphabetical order on the 2nd line. The 3rd line should list the number of edges that cross between  $U$  and  $W$ .

**Extra Credit.** (a) The group that finds the largest number of crossing edges in fewer than 30 minutes in a test set I will keep hidden will get an extra credit bonus. (b) The group with the largest value of  $\frac{\# \text{ edges}}{\# \text{ seconds} + 1}$  will also get an extra credit bonus.