**COMP 7612 Foundations of Computing  Spring 2014**

Programming Project – NP-complete problem and polynomial time reduction

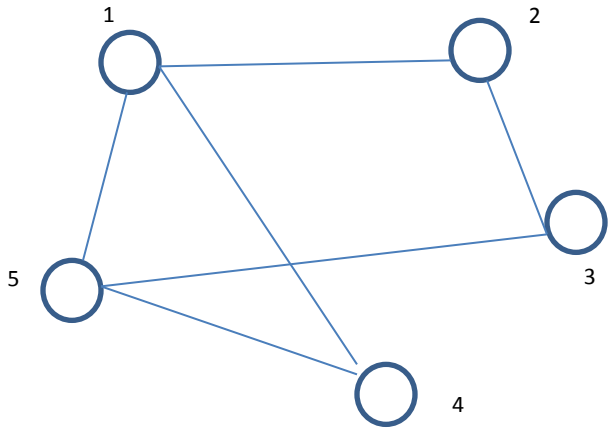
The goal of this project is to implement a NP-complete program (using brute-force algorithm) and also try to code a polynomial-time reduction. The problem is divided into 3 parts.

The following problem is NP-complete:

(Hamiltonian Completion – Decision version)

You are given an undirected graph G and a number k. Can I add at most k edges to G such that the new graph has a Hamiltonian cycle?

1. Write a program that given an instance of the Hamiltonian completion problem (decision version), return YES or NO **correctly.** Your program should read a text file that contains the detail about the longest cycle problem. The file will have the following format:
    a. The first line contains three numbers, the number (n) of vertices of the graph. (The vertices are labeled from 1 … n); the second number is the number of edges (e); the third number is the maximum number of edges to be added (k).
    b. Each subsequent line contains an edge, represented by 2 numbers; denoting the vertices that the edge connects, and the third one is the weight of the edge.
    - For example, to find if there is a Hamiltonian Completion of the following graph with 1 additional edge, the input will be as follows:




```
5 6 1
1 2
1 5
4 5
4 1
3 5
2 3
```

The name of the text file to be read is to be passed as a command line parameter when you run the program. You can assume the file to be in correct format. If the program fails because the input

format is wrong (for instance, if there is a set with an element 11 in the above example), you are not responsible for that. (I.e. you can get away with minimal checking of input). Notice that the two vertices of an edge can be listed in any order.

2. Using the program you have written, find the solution to the *Hamiltion completion number problem.* That is, given a graph G, you will return the smallest number of edges you need to add in order to form a Hamiltonian Cycle.

   You should use the same format for the input as of problem 1. However, in your program, you should ignore the value of k that is given.

3. Using the program that you have written for part 1 and/or 2, solve an instance of the following problem:

   (Path Cover) Given a graph G and a number k, can you form k distinct paths in G (no common edge and no common vertices), such that every vertex in G belongs to exactly one path?

   Once again, you are to read the graph from a file (which should be supplied as a common line parameter). The format of the input is the same as part 1 and 2. (Hint: If a graph G has a path cover of size k, how do you connect them to form a Hamiltonian Cycle?)

Grading of the project:

You MUST use either C, C++ or Java. If you use C or C++, you are to ensure that your program compile with no problems using GNU g++/gcc. (Do not just compile it using turbo C++, for instance).

Your grade is divided into the following portions. Full mark is 100 points, but you may earn as much as 160 points.

- Base grade (125 points)
  - Base grade Hamiltonian Completion (decision version) (45 points): If your program is correct for Hamiltonian Completion (decision version)
  - Base grade Hamiltonian Completion number (35 points): If your program is correct with Hamiltonian Completion Number
  - Base grade Path Cover (35 points): You MUST reduce Path Cover to Hamiltonian Cycle
  - Documentation (10 points): You should document your program (at the very least, describe all your classes and functions).
  - Additions/subtraction
    - +7% maximum if you use javadoc or doxygen for documentation
    - -25% if your program has a pre-defined upper limit on your problem size. (i.e. your program can only handle graphs with at most a certain number of edges/vertices). Of course, you will not be penalized if the physical memory limit is reached.
    - -everything if I detect ANY kind of plagiarism. It will be reported to the higher authorities. And you will get an F for the class (regardless of what you do in other part of the class)
    - -50% if your program does not read the input in the required format

    - Notices that full marks will only be awarded if the program does not fail under multiple legal inputs. If the program breaks for some legal input, the grades will be substantially affected. (As mentioned, if the input is not of the right format, you are not responsible).

- Bonus grades (35 points): Running time for longest cycle
    - The correct programs for part 2 will be timed (the programs will be divided into two groups: Java and C/C++). The 4 fastest ones in each group will receive bonus ranging from 5 – 35 points. Notice that if there are ties, the bonus will be split.

- Groups of 2
    - You are welcomed to work as groups of 2. However, all the bonus grades for your group will be split between the two members. And the maximum base grade you can earn is 100.

Handing in the project

The project is due 5:00pm**, 4/29 (Tue).** You are required to hand in your project via the drop box in e-courseware class web site. You should hand in ONLY the source code (your documentation should be embedded in your source code). You should also mention your name(s) inside your source code. ***Any group handing in anything more than the source code will have the score subtracted by 25%.***

If you are in the group of 2, only one of you needs to upload the source code.

Using existing code

If you want to use any existing source code, you need to upload to the dropbox the class(es) you plan to use and where do you plan to download it from, by **4/18 (Fri) 5:00pm**. I will need to approve that. Otherwise, it will be treated as plagiarism.

There are two kinds of code I will allow you to use

- Any Pre-defined class that implements a undirected graph
- Any existing Hamiltonian Cycle algorithm.