CSE 830: Fall 2011 Project

Due: Tuesday, November 22nd 2011, 10:20am

The real reason we study algorithms is to find fast ways for solving problems. Analysis lets us see how well we are doing in broad strokes, but washes away questions about constants, implementation difficulty, and performance in practice. For this project, you must design and implement a *fast* program that requires some form of combinatorial search.

The Problem:

You are hired by the U.S. Postal Service to help them implement cost-cutting measures. You are given the map of an area where each town currently has a post office and you must determine the largest number of post offices that can be shut down such that each town either keeps its post office or else has a neighboring town that keeps its post office.

The USPS represents this problem as a graph where the n towns are the vertices, and the m edges denote neighboring towns. Your goal is to find the maximum set of vertices such that if a vertex is in the set, at least one of its neighbors is not.

This problem is NP-complete, meaning that it is exceedingly unlikely that you will be able to find an algorithm with polynomial worst-case running time (you will prove it NP-Complete on homework #5). However, since the goal of the problem is to find a subset of vertices in the graph, a branching program that iterates through all 2^n possible subsets of vertices and tests whether all of the vertices in the set are directly connected to one not in the set is an easy $O(n\ m\ 2^n)$ algorithm. But the goal of this assignment is to find as practically good an algorithm as possible.

Input and Output:

There are a variety of data files available in the directory ~cse830/Public/project/. Each graph is in a format such that the first line gives the number of vertices (n) and the second line is the number of edges (m). The next m lines each contain a pair of numbers on them indicating the two vertices that are connected by each edge. Vertices are numbered from 0 to n-1. Please keep an eye on this directory because I will add extra sample files based on questions that I receive.

Your program **must:**

- Generate (or include if it's a scripting language) an executable named "solve-USPS".
- Accept a single command-line argument of the filename containing the graph to solve (you may assume that this file contains a simple graph). Do not prompt for any additional input.
- Output the number of vertices in the solution set, followed by a colon (':') and then a list the specific subset of vertices the solution includes. Do not clutter the output with extra information.

For example:

```
> ./solve-USPS rand-25-60
15 : 0 2 3 4 6 7 8 11 12 13 15 18 19 22 23
```

This result would indicate at most 15 post-offices could be closed in "rand-25-60", and it provides the specific IDs. You do not need to worry if your solution is unique as long as it is correct.

Implementation:

You may use any major programming language, but your program must run under Linux on the CSE cluster (adriatic.cse.msu.edu). Start early enough to go through several iterations. Use a simple branching approach first; don't get fancy until you have something that works. It is possible to get a program working in under 100 lines. Don't forget to compile with optimizations for a free speedup.

Grading:

You will be graded on how fast and clever your program is, not on style. Incorrect programs will receive no credit. A program will be deemed incorrect if it does not find a subset corresponding to a minimum legal set of vertices for multiple examples. As a rough guide, if you can consistently handle graphs with ~18 vertices, you'll get a 70, ~25 vertices will get you an 80, ~35 will get you a 90, and ~45 will get you 100. If your program can handle graphs with 55 vertices, you'll get an extra credit, and if you make a significant effort and get to 70 vertices, I'll drop your lowest grade on the first four homework assignments. Your program will have a total of 60 seconds to run on each test graph. A (compiled) example of an efficient program can be found in ~cse830/Public/project/solve-USPS-ex that you may compare your results to.

Other details:

Everyone *must* do this question individually. Just this once, you are not allowed to work with your partners. The idea is to think about the problem from scratch and on your own. If you do not completely understand the problem definition, you don't have the slightest chance of producing a working program. Don't be afraid to ask for clarification or explanation!

You are to e-mail me:

- A listing of your program.
- Instructions on how to compile your program.
- A brief description of your algorithm including any interesting optimizations, and the time it takes on sample data files. Be sure to indicate the largest test files your program could handle in 60 seconds or less of wall clock time.

Good luck!