**CS400/600: Data Structures and Algorithms**

**Programming Assignment #4**

**Due March 13th, 2010, 11:55pm**

## 1 Project Description

For this assignment, you will be working on the digraph which simulates the link structure of a set of web pages. Each link $v_i -> v_j$ is associated with a cost $w_{i,j}$. Figure 1 shows an example.

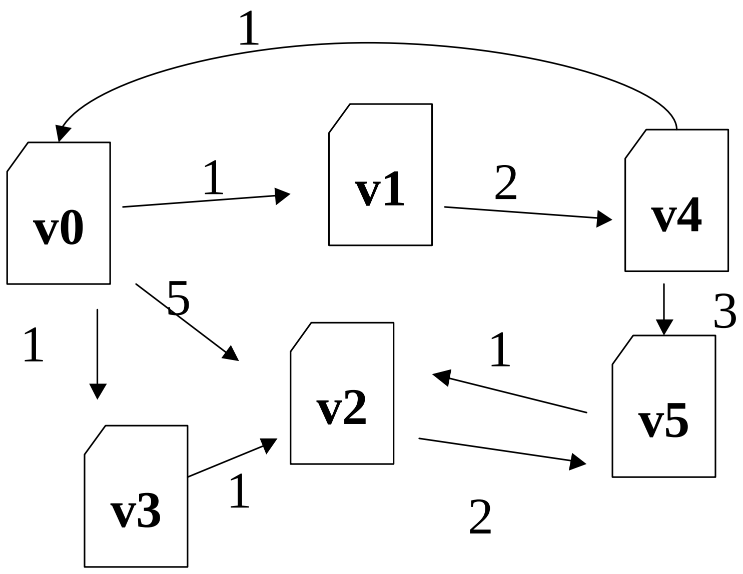

Figure 1: Web pages and hyperlinks.

The link structure is stored in a file. The file starts with the number of vertices at the first line, followed by the adjacency matrix of the link structure. Each element of the adjacency matrix is the cost $w_{i,j}$. '0' means no connection. The element (i, j) contains the value $w_{i,j}$, which is the cost for the the link $v_i -> v_j$. For example, the above link structure is stored as

```
6
0 1 5 1 0 0
0 0 0 0 2 0
0 0 0 0 0 2
0 0 1 0 0 0
1 0 0 0 0 3
0 0 1 0 0 0
```

The elements are separated by white space. You can assume the elements are all integers.

## 2 Tasks

- Task1 (20points): Read the input file, and store the graph into an adjacency list. Print out the adjacency structure in the following format. The numbers denote the ID of the corresponding vertex.

0 − > 1,2,3
1 − > 2,4
. . .

- Task2 (20points): Print out the sequence of vertices visited using a DFS (Depth First Search) graph traversal starting at vertex 0.
- Task3 (20points): Print out the sequence of vertices visited using a BFS (Breadth First Search) graph traversal starting at vertex 0.
- Task4 (40points): Implement Dijkstra's algorithm and find the shortest path from the first vertex ($v_0$) to other vertices and print out the shortest paths with the following format.
  0− > 1 cost: c1
  0− > 1− > 2 cost:c2
  . . .
  c1, c2 . . . $c_i$ are the costs of the shortest paths.

Your program should have a user interface that supports the following operations:

- Create Adjacency list of the graph: Reads the link structure file, and store it into an adjacency list. Print out the adjacency structure.
- DFS graph traversal.
- BFS graph traversal.
- Shortest Path – Implement Dijkstra's Algorithm.
- Quit.

## 3 Requirements

1. Your code should follow the Code Standards handed out during the first day of class. Your code will be graded according to its correctness (80 pts), and style/readability (20 pts). The grading policy is posted on webCT.
2. Your main program better create a user interface which the grader can use to test the functionality of your program. The file name for the main program should be lab4.cpp.
3. You must submit an ELECTRONIC COPY of your source program and report through webCT before the due date. If for some reason WebCT is unavailable, submit your source code by email to wlodarski.4 AT wright.edu. If you want, you can also cc to the instructor Meilin Liu, whose email address is meilin.liu AT wright.edu.
4. Submit all your source codes, makefile, README, and any other required files. You must explain your programs clearly in the README file. You must write down your name and email address in the README file.
5. The grader will test your programs under the schools UNIX environment, e.g., unixapps1.wright.edu. It is YOUR responsibility to make your programs workable and runnable by others under school's UNIX environment.
6. You can use the matrix for Figure 1 to test your algorithm. However, you should allow the grader to specify any matrix file as the input. The grader may use different matrix file for testing.
7. The programming assignment is individual. You must do the project by yourself. If you allow others to copy your programs or answers, you will get the same punishment as those who copy yours.