Assignment 4

COS 212

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA
Denkleiers • Leading Minds • Dikgopolo tša Dihlalefi

Department of Computer Science
Deadline: 26/06/2020 at 23:00

Objectives:

- Implementing the graph data structure.
- Solving the Chinese Postman problem.

General instructions:

- This assignment should be completed individually, **no** group effort is allowed.
- Only the output of your Java program will be evaluated.
- Your code may be inspected to ensure that you've followed the instructions.
- Be ready to upload your assignment tasks well before the deadline, as no extensions will be granted.
- You are NOT allowed to import any Java packages except the I/O packages. Doing so you will receive a mark of 0. You must provide your own implementation of any additional data structures that you require.
- If your code does not compile you will be awarded a mark of 0.
- All submissions will be checked for plagiarism.
- Read the entire assignment before you start coding.
- For each task, you will be afforded three opportunities to upload your submissions for automarking.

Plagiarism:

The Department of Computer Science considers plagiarism as a serious offence. Disciplinary action will be taken against students who commit plagiarism. Plagiarism includes copying someone else's work without consent, copying a friend's work (even with consent) and copying material (such as text or program code) from the Internet. Copying will not be tolerated in this course. For a formal definition of

plagiarism, the student is referred to `http://www.library.up.ac.za/plagiarism/index.htm` (from the main page of the University of Pretoria site, follow the *Library* quick link, and then choose the *Plagiarism* option under the *Services* menu). If you have any form of question regarding this, please ask one of the lecturers, to avoid any misunderstanding. Also note that the OOP principle of code re-use does not mean that you should copy and adapt code to suit your solution.

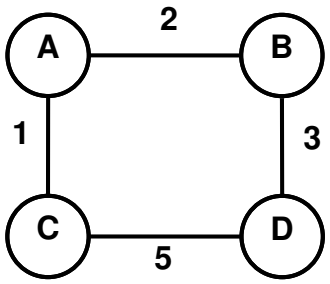
After completing this assignment:

Upon successful completion of this assignment you will have implemented your own graph data structure, as well as an algorithm to solve the Chinese Postman problem in Java.

Problem Description

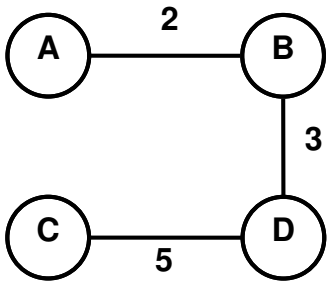
For this assignment, you will implement a solution to the Chinese Postman problem, a well-known problem in graph theory, introduced for the first time by Kwan Mei-Ko in 1960. The idea behind it is as follows: a postman has to go down **each street** in town to deliver the mail. What is the *most efficient* route through the town covering each street at least once, and returning the postman back to the post office? If each street is represented by an **undirected weighted edge** in a graph, and the post office location is the starting vertex in a graph, the problem can be reformulated in graph terminology as follows: find the shortest *cycle* or *circuit* starting at the given starting point, such that each edge is visited at least once.

The problem is easy when the graph is *traversable*, i.e. if it can be "traced" without lifting the pen from the paper and without ever retracing the same edge:



In the above graph, if the post office is at vertex **A**, then the postman can either take the A-B-D-C-A route, or A-C-D-B-A route. In both cases, the postman will travel down each edge (street) exactly once, thus the total cost of the journey will be $1 + 2 + 3 + 5 = 11$.

Now consider the following graph:

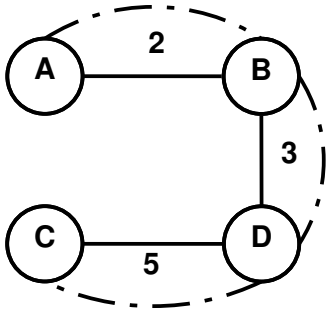


If the post office is at vertex **A**, then the only way to cover all edges and to return back home would be A-B-D-C-D-B-A. In other words, every edge has to be taken twice, with the total cost of the journey $2 + 3 + 5 + 5 + 3 + 2 = 20$.

What is the main difference between the first and the second graph? It is the presence/absence of the edge connecting A and C. Because there is no connection between A and C in the second graph, both A and C are vertices of *odd degree* (degree of a vertex is the number of edges incident to the vertex).

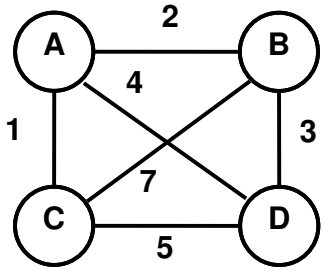
It can in fact be shown that every vertex in a traversable graph must have an even degree, otherwise the graph is not traversable. Traversable graphs are also referred to as *Eulerian graphs* (see Section 8.10.1 of the prescribed textbook).

How do you traverse a non-traversable graph? By travelling down some of the edges twice. If you represent each "travelling" action as an edge, you will notice that the resulting graph will indeed contain vertices of even degrees only:

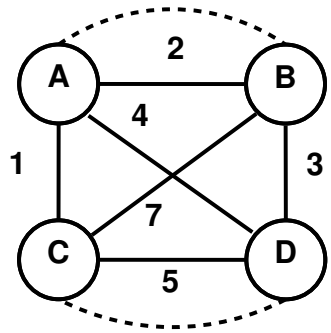


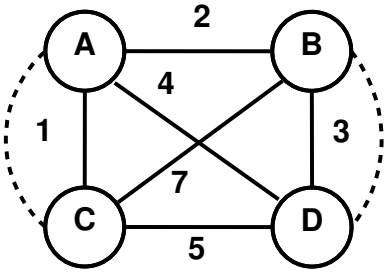
Now, the degree of A and C is 2, and the degree of B and D is 4. The graph has become traversable by adding a path from A to C in addition to the existing edges. In general, the only way to make sure the postman visits every street and gets back to the post office is to "**pair**" the odd-degree vertices and find a path between them, thus virtually turning them into even-degree vertices. The number of odd-degree vertices in a connected graph is **always even**, thus you can always separate them into non-overlapping pairs.

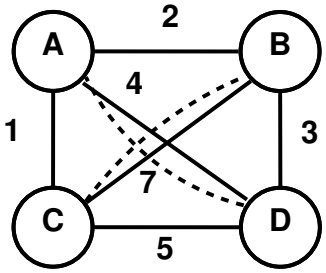
However, some weighted edges may be cheaper than others, thus not all pairings are equally good. Consider the following graph:



Every vertex in this graph is of an odd degree. Thus, we need to add alternative paths to make sure all vertices are of even degree. We need to "pair" the odd-degree vertices. The possible pairings are: (1) AB, CD, (2) AC, BD, (3) AD, BC.







Which pairing is the best? The one that offers the shortest total path, of course. To find out which pairing is the best, we need to calculate the shortest path between the vertices of each pair. For AB, CD and AC, BD direct links are the shortest paths between the paired vertices. For AD, BC, however, the shortest path from A to D is the direct edge, but the shortest path from B to C is via A. Now, path(AB) + path(CD) = 2 + 5 = 7; path(AC) + path(BD) = 1 + 3 = 4; path(AD) + path(BC) = 4 + 3 = 7. Thus, AC, BD is the wisest pairing.

Chinese Postman algorithm

The algorithm for finding the optimal Chinese postman route can be summarised as follows:

1. Find all odd-degree vertices.
2. Find all possible non-overlapping pairings of odd vertices.
3. For each pair {u,v}, find the edges that make up the shortest path between u and v.
4. Choose the set of pairings that minimises total travelling distance.
5. Create a copy of the original graph, and add to it the edges identified in Step 4.
6. The length of an optimal Chinese postman route is the sum of all edges + the total length found in Step 4.

Now that the existing graph has been converted to a traversable, i.e. Eulerian graph, a Eulerian circuit starting at the given vertex will return an optimal path of minimum length. For Eulerian circuit algorithm, refer to Section 8.10.1. For further information on the Chinese Postman problem, refer to Section 8.10.1.1.

Task 1

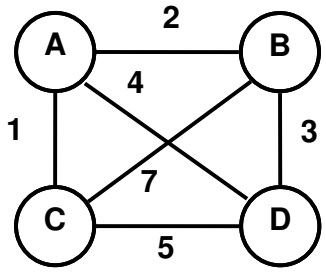
For this task, you will construct a graph from an input file. Download the `assignment4Code.zip` archive, and unzip it in an appropriate directory. The archive contains the following files:

- Graph.java
- Main.java
- graph.txt

The implementation of the graph data structure is left up to you. Note that for the purpose of this assignment, all graphs will be connected, undirected, and weighted, where every weight will be a positive integer. The constructor that will be used for marking takes only one argument: a `String` parameter. The string is the name of the file which contains a representation of a graph which you must construct. The file can be assumed to be placed in the same directory as your code. File `graph.txt` is an example input file:

```
4
A,B,2
A,C,1
A,D,4
B,C,7
B,D,3
C,D,5
```

The first integer represents the total number of nodes. This will be useful if you decide to use an adjacency matrix for graph representation. The letters represent the labels of the vertices, and the numbers represent the weights of the edges between the vertices. The graph described above would look as follows:



Vertex labels can be strings of varied length instead of single characters. I.e., `Arcadia,Richard,22` is a valid entry. The labels and the edge weights will always be comma-separated. You may import any relevant I/O libraries/packages in your assignment to aid you.

Submit Task 1 for marking only when the following functions have been implemented:

- `public Graph(String f)`
- `public Graph clone()`
- `public boolean reconstructGraph(String fileName)`
- `public int numEdges(String u, String v)`
- `public int getDegree(String u)`
- `public boolean changeLabel(String v, String newLabel)`
- `public String depthFirstTraversal(String v)`

Refer to the comments in the supplied code for more details on the functions listed above.

Task 2

For this task, you will calculate the total cost of the optimal Chinese postman route, and construct an optimal traversable graph based on the given graph. Refer to problem description for an idea of the algorithm. Submit Task 2 for marking only when the following functions have been implemented:

- `public String getOdd()`
- `public String getPath(String u, String v)`
- `public int getChinesePostmanCost()`
- `public Graph getChinesePostmanGraph()`

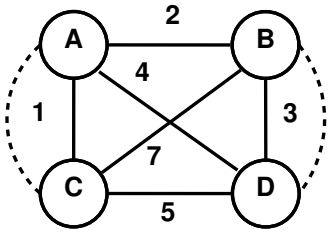
Refer to the comments in the supplied code for more details on the functions listed above.

Task 3

This task builds on top of the previous two tasks. The goal of this task is to construct the optimal route that the Chinese postman has to follow. To obtain the path, solve the Chinese postman problem by adding the necessary edges to the graph. Now that the graph is traversable, construct the Eulerian cycle of the graph, starting at the given "post office" vertex. Pseudocode for obtaining an Eulerian cycle is given in Section 8.10.1 (refer to the Fleury algorithm).

Typically, more than one alternative optimal paths exist. For this task, choose the path that adheres to the alphabetical order as far as possible. I.e., when there is a choice between vertices, choose in alphabetical order. The path should reflect the order in which the nodes are visited.

For the following graph:



Between two optimal routes, `A,B,C,D,B,D,A,C,A` and `A,B,C,A,C,D,B,D,A`, the latter one should be preferred.

Submit Task 3 for marking only when the following function has been implemented:

- `public String getChinesePostmanRoute(String v)`

Refer to the comments in the supplied code for more details.

Submission Instructions:

You must create your own makefile and submit it along with your Java code. Your code should be compiled with the following command:

```
javac *.java
```

Your makefile should also include a `run` rule which will execute your code if typed on the command line as in `make run`.

For your submission, you must zip your Java code together with the makefile, with no folders/sub-folders. Your Java archive should be named sXXXass4.tar.gz, or sXXXass4.zip, where XXX is your student/staff number. Separate upload links are provided for the three tasks. Each link grants 3 uploads. Every upload is a marking opportunity, do not use it to test your code. Test the code thoroughly, with multiple test cases, before submitting it for marking.

Good luck!