UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Department of Computer Science

COS212: Practical 8

Release: Monday 29 April 2019
Deadline: Friday 3 May 2019, 18:00

Objectives

This practical has the following objectives:

- To implement a variation of Selection Sort by using a Graph instead of an array for input.
- To navigate and manipulate the underlying graph data structure.

Instructions

Complete the task below. Certain classes have been provided for you in the *files* zip archive of the practical. You have also been given a main file which will test some code functionality, but it is by no means intended to provide extensive test coverage. You are encouraged to edit this file and test your code more thoroughly. Remember to test boundary cases. Upload the given source files with your changes and any additional Classes in a zip archive before the deadline. Please comment your name **and** student number in at the top of each file. **This practical will be checked for plagiarism**.

You are required to upload at least the following files in a zip archive:

- `Main.java` - Ignored by Fitchfork, but may be considered in case of any queries
- `Tests.Java` - Ignored by Fitchfork. Required by `Main.java`
- `GraphSelectionSort.java` - Input and output from this class will be marked by Fitchfork.
- `Edge.java` - Required by `GraphSelectionSort.java`. You may add helper functions to this class.
- `Vertex.java` - Required by `Edge.java`. You may add helper functions to this class.

You may create additional Classes to assist in the implementation of this practical. Your `makefile` will be overwritten by Fitchfork, all given source files will still be compiled. It is recommended to make use of the `Test` function provided by Fitchfork to verify that your code compiles on Fitchfork. The `Test` function will execute the `makefile` you provide using the command: `make`

Introduction

Selection sort is a simple sorting algorithm that identifies elements out of place and moves them to their final position. The lowest value in the array is swapped with the element in the first position. The next lowest value is then found and swapped with the element in the second position, and so on. See section 9.1.2 in the textbook for more information on Selection sort using arrays.

However for this practical you will be using a Graph as the input data structure instead of an array. Your code will take as input an array of edges, representing an unweighted, directed graph. Each vertex will contain a value which will be sorted. Instead of swapping elements in an array, the vertex with the lowest value needs to be removed from the graph, and the value of the vertex appended to an array of sorted values.

Task 1: [48]

Implement the following methods in the *GraphSelectionSort* Class according to the given specification:

`GraphSelectionSort(Edge[] edges)`

Initialize the internal state using the given array of Edges.

`Integer[] getSorted()`

Return the Array of sorted values.
In total the output of this function contributes 18 marks.

`Edge[] getEdges()`

Return an array of Edges that are still in the graph.
In total the output of this function contributes 22 marks.

`void doSortIteration()`

Perform one iteration of the custom Selection Sort. That is, remove the vertex from the graph with the lowest value, and append the value to the array of sorted values. The result returned by the function `Edge[] getEdges()` should reflect the state of the new graph.

`Boolean isSorted()`

Return true if all all elements are sorted and the graph contains no vertices. In total the output of this function contributes 8 marks.

You may use your own helper functions to assist in implementing the specification. However you may not modify any of the given method signatures.

Submission

You need to submit your source files on the Assignment website (assignments.cs.up.ac.za). All methods need to be implemented (or at least stubbed) before submission. Place all the source files including a makefile in a zip or tar/gzip archive named uXXXXXXXX.zip or uXXXXXXXX.tar.gz where XXXXXXXX is your student number. There should be no folders in your archive. You have 1 week to finish this practical, regardless of which practical session you attend. Upload your archive to the *2019 Prac 8* slot on the Assignment website. Submit your work before the deadline. **No late submissions will be accepted!**