CS163 – Data Structures & Algorithms

Lab 09
Shortest path algorithms
Sorting algorithms

Cảm ơn thầy Trần Duy Quang đã cung cấp template cho môn học

Department of Software Engineering-FIT-VNU-HCMUS

1 Notes

Create a single solution/folder to store your source code in a week.

Then, create a project/sub-folder to store your source code of each assignment.

The source code in an assignment should have at least 3 files:

- A header file (.h): struct definition, function prototypes/definition.
- A source file (.cpp): function implementation.
- Another source file (.cpp): named YourID_Ex01.cpp, main function. Replace 01 by id of an assignment.

Make sure your source code was built correctly. Use many test cases to check your code before submitting to Moodle.

2
Content

In this lab, we will review the following topics:

- Shortest path algorithms.
- Sorting algorithms

3

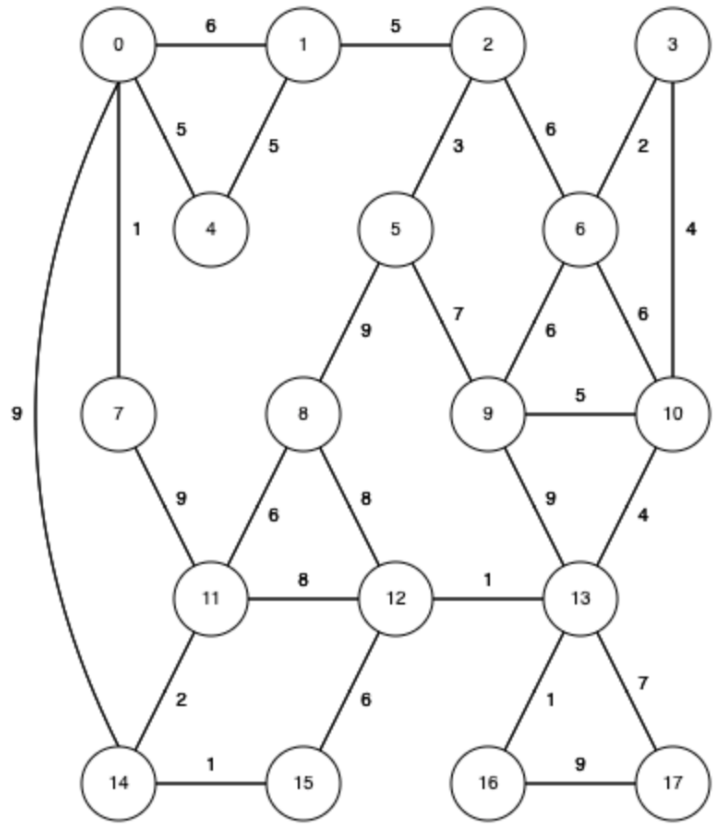
Assignments

H: YY: 08

3.1. Assignment 1 – Paper assignment

Given the below weighted graph, you are asked to demonstrate step by step how to:

1. Find the shortest paths from vertex 0 to all other vertices using Dijkstra's algorithm. For each pair of vertices, write down the total weight and list of vertices on the path.
2. Does Dijkstra's algorithm work for negative weight edges? Why or Why not? Give an example.



3. Suppose we have an undirected graph with weights that can be either positive or negative. Do Prim's and Kruskal's algorithim produce a MST for such a graph?
4. Consider the problem of computing a maximum spanning tree, namely the spanning tree that maximizes the sum of edge costs. Do Prim and Kruskal's algorithm work for this problem (assuming of course that we choose the crossing edge with maximum cost)?

3.2. Assignment 2 – Dijkstra

1. Implement Dijkstra’s algorithm. You are allowed to use built-in vector, stack and queue classes in C++.

Use rand() function to generate an array of 10.000 integer numbers and save them into a text file. Use this file as input for the following sorting algorithms.
For each algorithm, compute its running time on the above array. Create a table to compare the running time of all below algorithms.

3.3. Assignment 3 – Selection Sort

Implement the Selection Sort. Test it on the 10.000-element array.

3.4. Assignment 4 – Insertion Sort

Implement the Insertion Sort. Test it on the 10.000-element array.

3.5. Assignment 5 – Counting Sort

Implement the Counting Sort. Test it on the 10.000-element array.

3.6. Assignment 6 – Merge Sort

Implement the Merge Sort. Test it on the 10.000-element array.

3.7. Assignment 7 – Quick Sort

Implement the Quick Sort. Test it on the 10.000-element array.

3.8. Assignment 8 – Radix Sort

Implement the Radix Sort. Test it on the 10.000-element array.