Assignment 2: Algorithm Analysis and Graphs

COSC 3020: Algorithms and Data Structures

Lars Kotthoff
larsko@uwyo.edu

26 March 2018

Instructions

In your code, you may *not* use library functions (standard library or otherwise) unless explicitly stated.

Solve the following tasks. You may work in teams of up to two people. For the theoretic part, submit a PDF with your solution, for the practical part, submit the C++ file(s) to WyoCourses. The name and student ID of *all* partners must be clearly visible on each page of your PDF and in each source code file. Only one partner needs to submit. You have until Friday, 06 April 2018, 23:59h.

I will test your code on Linux with `clang++`. Please test it on the lab machines, where you have the same environment – if your code does not compile, you may get no points for this part. If you submit a file other than a PDF or if your code is not in a separate file you may get no points for it.

1 Theory vs. Practice (9 points)

- List at least 3 reasons why asymptotic analysis may be misleading with respect to actual performance in practice. (3 points)
- Suppose finding a particular element in a binary search tree with 1,000 elements takes 5 seconds. Given what you know about the asymptotic complexity of search in a binary search tree, how long would you guess finding the same element in a search tree with 10,000 elements takes? Explain your reasoning. (3 points)
- You measure the time with 10,000 elements and it takes 100 seconds! List at least 3 reasons why this could be the case, given that reasoning with the asymptotic complexity suggests a different time. (3 points)

2 Graph Properties (6 points)

- Prove that if two graphs A and B have the same number of nodes and are completely connected, they must be isomorphic. (3 points)
- Prove that if two graphs A and B are isomorphic they do *not* need to be completely connected. (3 points)

You need to give complete, formal proofs – state *all* your assumptions and make sure that you've explained every step of your reasoning.

Hint: A good way to start is by writing down the definitions for everything related to what you want to prove.

3 Ford-Fulkerson – Augmenting Paths (15 points)

When we talked about the Ford-Fulkerson algorithm to find the maximum flow through a graph, I mentioned the "find an augmenting path" function. You're going to implement this function. Given a directed graph (the residual graph), a start, and an end node, the function returns a path (a list of nodes, starting at the start node and ending at the end node), or a special value if there is no such path.

I'm not giving you the prototype of the function – you are free to implement it whatever way you like, as long as it conforms to the above specification. In particular, you are free to choose whatever graph representation you like, and return whatever data type you like from the function. However, you *must* explain in comments what choices you made and why.

You may use any functions from the C++ standard library, but not other libraries (in particular, you are not allowed to use any Boost libraries).

Test your code by calling the function with a few (at least two) different inputs. Implement functions that can print an input graph and an output path (the graph can be printed as an adjacency list or matrix).

Hint: Think about the different kinds of search you can do in a graph. Which one is most appropriate here? How would you implement it?

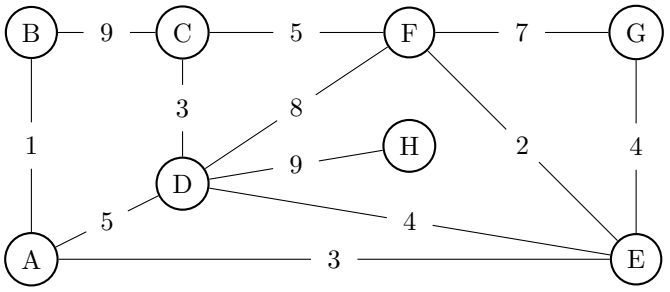
To get full points, your code style and design must be good. This means in particular avoiding unnecessary steps and good object-oriented encapsulation and abstraction.

3.1 Peer Grading

We'll do peer grading for this part, meaning that you'll have a look at and evaluate each other's code. To do this, go to `http://fish20.cs.uwyo.edu` (this will only work on the university network) and create an account. Enter code `d0a0f9cc` to enroll in the course. Select the course, and click "Create my submission" in the "Assignment 2 augmenting paths code" row. Upload your `.cpp` file there – the deadline is the same as for the assignment. After the

deadline, I will assign you the code of three of your classmates to have a look at and answer questions on.

4 Kruskal's Algorithm (3 points)



Run Kruskal's algorithm on the above graph to produce a minimum spanning tree (MST). The weights are the numbers drawn on top of each edge. Draw the minimum spanning tree Kruskal's algorithm finds, and indicate the order in which nodes are added to it. You may break ties arbitrarily.