Project 3: Graph modeling and graph algorithms

COT 4400, Spring 2019

Due April 23, 2019

1 Overview

This project requires you to model the problem below as graph and then use a known graph algorithm to solve the problem. **You are not allowed to use the internet or consult any references. This is an individual project. This policy will be strictly enforced.**

This problem is based on the classic puzzle of how you can measure out a specific amount of water using buckets of different sizes and a hose. Initially, all of the buckets are empty, and at each point, the only legal moves are: filling up a bucket, dumping out a bucket, or transferring the water in one bucket into another. Note that a transfer will either empty out the bucket it is pouring from or fill up the bucket it is pouring into, whichever happens first.

For example, suppose you want to measure out 4 gallons of water using a 3-gallon bucket and a 5-gallon bucket. You might fill up the 5-gallon bucket (5 gal, 0 gal), transfer the 3 gallons to the 3-gallon bucket (2 gal, 3 gal), dump the 3-gallon bucket (2 gal, 0 gal), transfer 2 gallons to the 3-gallon bucket (0 gal, 2 gal), fill the 5-gallon bucket again (5 gal, 2 gal), transfer 1 gallon to the 3-gallon bucket (4 gal, 3 gal), and finally dump the 3-gallon bucket (4 gal, 0 gal). This is the most efficient solution, which takes 7 moves.

This project will be a twist on the original problem. Instead of describing how to make a given amount of water, your problem is to identify the amount of water that takes the most moves to make. The amount of water in a given set of buckets is the sum of the water in all buckets. In the previous example, the two buckets can have from 0 up to 8 gallons of water between them, and 4 gallons takes the greatest number of moves to make.

2 Modeling the problem

Before you write a program to solve this problem, you will first write a report describing (in English and pseudocode) how you will solve this problem. This report should answer two basic questions:

1. What type of graph would you use to model the problem input (detailed in the Section 3.1), and how would you construct this graph? (I.e., what do the vertices, edges, etc., correspond to?) Is the graph directed or undirected? Weighted or unweighted? Vertex-labelled and/or edge-labelled?

2. Give pseudocode for an algorithm to compute the amount of water that takes the most moves to reach. Your pseudocode should involve a graph algorithm we have discussed this semester.

3 Coding your solution

In addition to the report, you should implement your algorithm in C++ or Java so that it can find the most difficult amount of water to make for a set of buckets. Your code may be in C++ or Java, but it must compile and run on the C4 Linux Lab machines.

Your code may be split into any number of files. In addition, you are allowed to make use of any built-in library, and C++ users may use the Boost library in their implementations. Boost is a free, open-source library with a rich collection of mathematical functions, including several that deal with graphs. You may read more about Boost at `www.boost.org`.

3.1 Input format

Your program should read its input from the file `input.txt`, in the following format. The first line of the input has one positive integer, n, representing the number of buckets. The next line contains n positive integers, representing the size of each bucket. The bucket sizes will be given in decreasing order.

The number and size of the buckets will be such that your graph will never have more than 1 million vertices or 10 million edges.

3.2 Output format

Your program should write its output to the file `output.txt`, in the following format. You should write one line with 4 positive integers, separated by spaces.

The first two integers will be the number of vertices and the number of edges in the graph, respectively. Note: these numbers should be the number of vertices and edges in the *entire* graph.

The third and fourth integers should be the most difficult amount of water to make and the number of moves that amount takes, respectively. If there is a tie for the amount that takes the greatest number of moves, output the smallest amount of water that takes this many moves.

4 Submission

You must submit a zip archive containing 1) your report (described in Section 2) as a PDF document, 2) your code (described in Section 3), and 3) a README file describing how to compile and run your code to Canvas. If your code requires more than a simple command to compile and run then you must also provide a Makefile and/or shell script. A simple command might be something like:

```
g++ *.cpp -o buckets
```

If you are using Boost in your solution, you must provide a Makefile and/or shell script that uses the environment variable `$BOOST_HOME` (pointing to the Boost installation directory) to compile your code.

As this is an *individual* project, your project report and code will be checked for plagiarism.

5 Grading

Report	**50 points**
Graph model	20
Using correct graph algorithm	10
Pseudocode	20
Code	**50 points**
README file	5
Follows input and output specs	10
Compiles and is correct	30
Good coding style	5

Note: if your code is unreasonably slow, you will lose points for both your algorithm design and your correct output grade.