UNIVERSITY OF WASHINGTON, TACOMA
TCSS 343: Design and Analysis of Algorithms
Homework #6A, B: **Graphs, Prim's Algorithm**
Written part (6A) due: Tuesday, November 24, 2015, 10:15am
Program (6B) due: Tuesday, December 8, 2015, 10:15am

Autumn 2015 November 17, 2015

Written homework is due at the *beginning* of class on the day specified. Any homework turned in after the deadline will be considered late. **Late homework policy: Late homework will not be accepted on the programming project (6B). It would be extremely unwise to turn in the written part (6A) late.**

Homework #6A (due Tuesday, November 24) (TO BE SOLVED INDIVIDUALLY)

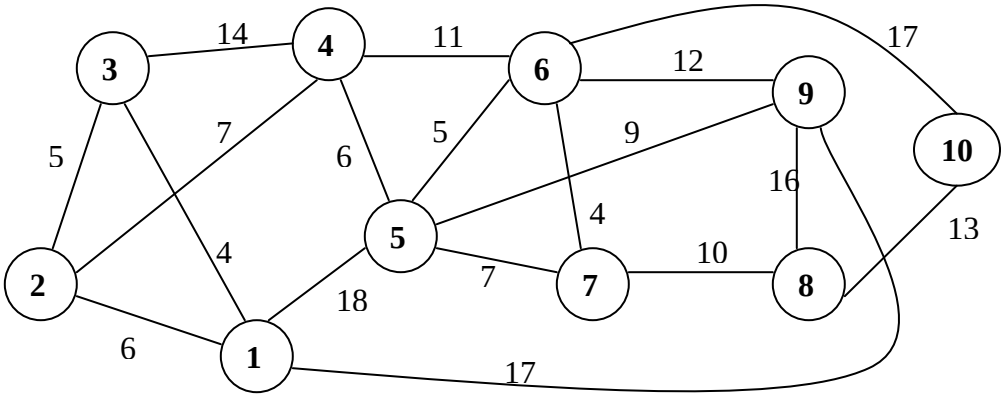
1. Apply the bottom-up dynamic programming algorithm to the following instance of the 0-1 knapsack problem. Also, show how you compute which items are in the optimal set. Show all of your work.

item	weight	value
1	4	15
2	6	19
3	1	5
4	2	4
5	3	14

Capacity W = 9.

2. For the graph below, show what happens in Prim's algorithm to find a minimum spanning tree, starting from vertex 4. Show the order in which the vertices are added to the set of known vertices (the book calls them tree vertices) and an indication for why that vertex was chosen.

   It would be useful to make several copies of the graph below. Each graph would represent the state of the algorithm after each iteration. On each graph, 1) indicate the value at each node after each iteration of the algorithm, 2) show which nodes are in the known set, and 3) indicate which edges are in the minimum spanning tree being built.



* * *

Suggested problems (highly recommended, but not to be turned in):

1. 8.1.1, 8.1.2, 8.1.5, 8.1.6.
2. 8.2.1, 8.2.3, 8.2.4.
3. 8.3.1, 8.3.5, 8.3.6.
4. Apply the OptimalBST algorithm to determine the optimal binary search tree containing the following items and their probabilities:
   a: 0.3
   b: 0.25
   c: 0.15
   d: 0.3
   Show your work (for example, show the table of partial results).
5. 8.3.2a,b.

6. 9.1.1.
7. 9.1.2, 9.1.3.
8. 9.1.9, 9.1.10, 9.1.11, 9.1.12, 9.13.
9. 9.2.1, 9.2.2, 9.2.4, 9.2.5.
10. 9.3.1, 9.3.2, 9.3.3, 9.3.7.
11. 9.4.1.

Homework 6B is on the next page.

Homework #6B: Trip Planner (due Tuesday, December 8)
GROUP PROJECT

Learning Objectives

- Reading, understanding, and using a graph ADT
- Prim's algorithm and its implementation (heap)
- More practice with debugging and constructing good test cases

The power grid (the infrastructure and wires needed to provide electricity) in Tacoma has been destroyed by a huge fire. An emergency plan has been put into place, but it is expensive and so therefore cannot be a long-term solution. You are part of the reconstruction team. You are one of the software engineers who need to figure out a good way to connect the power grid.

You have decided that the best way to do this is to first list all the places that need power. Each one of these places will be represented by a vertex in a graph. Making a direct power connection between some pairs of places will be impossible (because connecting them would require digging up parts of the city that can't be disrupted or because it would require construction that is far too expensive). Of the remaining possible connections, you would have to estimate how much it would cost to connect the two places.

Fortunately, someone has already provided all of this information to you. Your task is to come up with a set of connections such that there is an electrical path from every vertex to every other vertex. The head of your software design team has decided that Prim's minimum spanning tree algorithm is the best way to solve this problem. You should implement the algorithm in a way that is asymptotically optimal for Prim's algorithm (otherwise, what's the point?).

You are provided with code that implements a simple graph ADT. You will need to write code that implements Prim's algorithm. You should allow the user to specify a graph file that has place and cost data (terminal input is fine – you do not need to write a GUI). As output, you should list the set of edges in the minimum spanning tree your algorithm generates. Also, you should output the total cost of the minimum spanning tree.

Tips

Familiarize yourself with the graph ADT code before doing anything. Notice that when the graph data is read in, it returns a hash table that maps labels to vertices. (Why is that necessary?) Then read and understand how Prim's algorithm works (that is, do homework 6A). Only then will you be ready to write code.

Starter Code

Here is the java code to start with:

Vertex.java Edge.java SimpleGraph.java GraphInput.java InputLib.java KeyboardReader.java

You should generate your own test cases and test them on your code.

Software Engineering

Understand the graph ADT code and Prim's algorithm before writing code. You will waste valuable time if you don't. (Thought questions: Is the graph ADT code written in a way you would have implemented it? Why or why not? Are there object-oriented principles you think are not being adhered to in the code?)

As usual, comment your code and use good programming practices (good variable names and good indentation), so that it is easy to figure out what it is doing when/if you detect a bug.

What to Turn In

- Electronically turn in the code you created. You should not turn in the starter code, since you should not have changed it.
- Electronically turn in two test files, containing graph data with a non-trivial number of vertices (at least 10) and a non-trivial number of edges (at least 25). Think carefully about your test cases, because they should illustrate that your algorithm works and will therefore be useful in debugging.
- Please Zip your code (just the .java files and the test cases – no .class files, please) with the test cases into one file and submit it to the Moodle site.
- Also, turn in a printout of your code.
- Finally, each person in your group will turn in a report (at most one or two pages, hard copy) that described how you approached the problem, what troubles you had, and what you learned. Your report should also describe what contributions each person in your group made to the project.

Grading

This programming assignment (part 6B) counts for 30 homework points, the same as 3 normal homework problems. It will be graded on functionality, code organization, readability of code, quality of test cases, and the quality of your lab report.

Bonus

1. (Up to 10 points) Make a **non-trivial** improvement to your application. A GUI would be the obvious thing to do (3 points), but for more of a challenge, come up with some sort of feature in this application that involves modifying or adding to the algorithm. For example, displaying the graph somehow and showing each edge as it is added (by coloring it or making the line thicker) to the final minimum spanning tree would be challenging. Describe what you did and why you think it is a non-trivial improvement in your report.