

Homework 4

Assigned Nov 11th

Due: Nov 25th

The problem:

1. Implement a real number binary min Heap ADT class using C++
You need to have the following functions related to the Heap class.

- Insert(x, l): add an element x with label l (label is introduced for look up purpose).
- DeleteMin(): delete the minimal element in the heap and return it
- ConstructHeap(Vector<real> X, Vector<string> L): using a set of elements to construct a (initially empty) binary heap.
- Update(x,l): update the value of label l.
- Size(): return the number of elements in the heap
- isEmpty(): return true if the heap contains no element and false, otherwise.
- Print(): print out the elements of the heap in the order of the array entry of the elements

Your code should not assume that the maximal number of elements that can be inserted into a heap. Basic error handling should be there.

2. Based on YOUR heap class, implementing the dijkstra's algorithm for computing the single source shortest path problem for directed (non-negatively) weighted graphs. You should implement a function

- computeDistance(X): X is the index of the source node and computes the distances from X to all nodes in a graph.

How to test your code:

You should name your executables as "hadt" for the heap ADT class, "sssp" for the dijkstra's algorithm.

For the heap class executable hadt, it should accept an in-line parameter, which is the name of the testing file. For example, to run hadt, you should type:

```
$hadt htest
```

where htest is a testing file. You do not necessarily name your test file as htest.

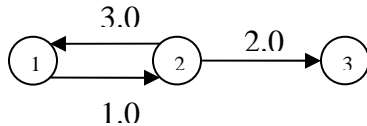
The testing file for the heap ADT class is a set of commands, one at each line. Each command is composed of the name of the related function and the related parameters (without parenthesis and comma, with the exception of the ConstructHeap function). For the ConstructHeap function, you should provide a set of real numbers and their labels. The values (real numbers) and the labels should be separated by a single comma. There can only be one ConstructHeap function and that function must be the first command in the test file. For example the following is a valid testing file for the heap class:

```
ConstructHeap 0 1.1 3.0 -2.0, A B C D
```

Print
Insert 3.1 E
Insert 2.1 F
Update 5.1 F
DeleteMin
Size
Print

For the executable “sssp”, it should expect two in-line parameters, the first one is the name of the testing file and the second one is the index of the source node. The testing file contains a single graph, in the adjacency list format. Specifically each line is a list of neighboring nodes (and the related edge weights) associated with a single node, starting with the index of the node, in the format of neighboring node index: weight. For example, the following is a valid testing file for the graph showing below.

```
1 2:1.0  
2 1:3.0 3:2.0
```



What to hand in:

- (1) Source codes with a makefile. If you do not use makefile before, the following is a good tutorial: <http://frank.mtsu.edu/~csdept/FacilitiesAndResources/make.htm>. Source code needs to be properly commented.
- (2) A report with the following components:
 - a. A brief discussion of the data structure that you use to implement the heap ADT. Justify your answer.
 - b. A brief discussion of the algorithm that you use to solve the problem 2.
 - c. Show your test files and your testing results. You may use typescript to capture Unix output. If you are not familiar with script and typescript, see <http://www.tech-faq.com/capture-unix-terminal-session.shtml>.

For example in Unix/Linux environment, assuming you have an executable named “hadt” and a test file “thadt”, you may recode the testing results in the following way:

```
$script          #start scripting  
$cat thadt      #show testing file contents  
$hadt thadt     #run your code  
$^D (ctrl-D)   #stop scripting
```

\$cat typescript #see the recorded results

d. Briefly discuss error handlings that you have implemented

Grading:

- Correctness of the code (whether it can compile, whether it produces the correct output when correct input files are provided). 50%
- Style of the coding including comments 10%
- Robust of the code with error handling 15%
- Report (style, completeness, clearness) 25%