CS 260 100P
Heaps and Heapsorting

Revision Date: January 10, 2013

Printable version

WARNING

This problem will change; the new problem will require roughly the same effort as this one.

Topics to learn

Please read up on the following subjects:

- arrays
- binary trees
- heaps
- partial ordering
- heapsort

Tasks to perform

For this project. you will write two versions of a heapsort program. The first program uses an array-based implementation of a heap. The second uses a binary tree implementation of a heap. You may need auxiliary data structures to implement the second program so that it has the same runtime complexity as the first program. THESE AUXILIARY DATA STRUCTURES SHOULD BE AS HIGH-LEVEL AS POSSIBLE.

I/O

Use command-line arguments to supply data and other information to your programs.

Provide a mini-interpreter that responds to the following commands:

r XXX
- load the data from file XXX into the tree

A
- choose an array based implementation

T
- choose a tree-based implementation

s
- sort the data in ascending order.

v
- visualize the tree using graphviz

t
- test the tree or array for correctness

Use the mini-interpreter from the previous project as a starting point for this version.

Demonstrating your problem

You should be prepared to discuss and modify any portion of your code. You should also be prepared to justify each of your design choices in solving the problem. You should demonstrate your programs using large amount of data. Have convincing sample data ready when demonstrating. The output of your programs should make it clear that both versions have the same runtime complexity.

You should be able to describe the auxilliary data structure needed for the tree-based version of heap-sort in high-level terms (e.g. priority-queue, binary search tree, etc.).