

CS 14: Data Structures and Algorithms

Lab 7: Heaps

Peter H. Fröhlich
phf@cs.ucr.edu

Wagner Truppel
wagner@cs.ucr.edu

Out on: February 17, 2003

Due by: February 23, 2003

Instructions: Remember to follow the instructions and policies for assignments given on the course website. This week’s assignment consists of **two** separate parts:

- an **at home** programming exercise (1)
- an **at home** written exercise (2)

While there is no mandatory “in lab” exercise, we encourage you to attend lab anyway and work on the “at home” parts. The “at home” parts are due **before 8:00 pm** on the date given above. You have to use the department’s electronic turnin service for **everything**, including the **written** part. Put your solution for the latter into the **text file** `solution-7.txt` and turn it in together with the other files.

1 At-Home: Implementing Heaps

Your task for this week is to implement the **heap** data structure discussed in the lecture. Your implementation should be in the form of a **template** class `MinHeap` in a file `minheap`.

If available, your TA may provide a test driver for your code. Otherwise you’re on your own to ensure that your implementation works correctly.

Since a heap maintains a **set** of elements, it should adhere to ADT **Set** and provide the following basic operations:

```
void add( const T &t );
void rem( const T &t );
bool has( const T &t ) const;
bool empty() const;
```

Furthermore, the two **heap-specific** operations (the reason for using this data structure in the first place) must be provided as well:

```
T min() const;
void rem_min();
```

For a heap of size n , the operations `min()` and `empty()` must execute in $O(1)$ time, the operations `add()` and `rem_min()` must execute in $O(\log n)$ time, and the operations `rem()` and `has()` must execute in $O(n)$ time (or faster if you can make them faster). Your implementation should also use only $O(n)$ space.

You **can** provide an `Iterator` or add the operation `int size() const;` if you want, but neither is required for this assignment. Note that there is **no a-priori limit** on the size of the heap, so your constructor should not take any arguments and your implementation should be able to “grow and shrink” as necessary. Finally, your implementation should constrain the type parameter `T` as little as possible.

2 At-Home: Heaps of Problems

Answer the following questions in **one or two sentences** each (the shorter the better):

1. The only difference between a `MinHeap` and a `MaxHeap` are the names of two operations as well as the use of “greater than” instead of “less than” on the type parameter `T`. How would you organize an implementation of **both** classes?

Hint: This is a fairly open-ended question with no single “correct” answer but lots of “wrong” ones.

2. How can we implement a **single** data structure that supports `min()` and `max()` in $O(1)$ time, `rem_min()` and `rem_max()` in $O(\log n)$ time, and that still retains all the “other qualities” required in part 1 above?
3. Heaps can be used to implement **priority queues** which entail a slightly different set of operations than required in part 1 above. How would you design the C++ interface for a class `PriorityQueue`?

Answer **in your own words** if you want to avoid being accused of cheating! If you use sources **other** than the assigned books for your answer, be sure to include a **reference** as well!