CSCI 102 assignment 8 – Heaps!

Nov 27

1 Modify-able heaps

In this section you will implement a Heap such that you can change the priority of entries in the heap.

- If we want to change the priority of entries, we need to be able to refer to them! Modify `Heap<V>` so that `add` returns the `Entry<Integer, V>` of the data you just added.
- Now that you can access a particular entry in the heap, add a method to `Heap<V>`, `void changePriority(Entry<Integer, V> entry, Integer new_priority)` that changes the priority at entry `entry`. Make sure to bubble up or down so that we still have a heap!!!
- Add a `main` method that adds `"A", "B", "C", "D", "E", "F", "G"` to the heap at priority `3, 8, 6, 1, 4, 100, 2`. Modify the priority of `"F"` to `0` and then output and print all elements in the heap.

2 In-place heap sort

In this section you will implement Heap sort in-place.

- Implement `Integer[] HeapSort(Integer[] arr)` by:

  1) treating `arr` as an array that stores the values in the heap followed by the values that have not been added to the heap. Thus, to put the `i`-th number in the heap, we know the first `i-1` entries are a heap, and we add the `i`-th entry by bubbling it up. Write the heap so that the **largest** element is at the root.

  2) Next we need to remove everything from the heap. We start by swapping the root of the heap with the last element of `arr` and bubbling it down. Now the root of the heap is the second largest element in the array, so we swap it with the second last element of the array and bubble down. We continue until the array is fully sorted.

- Add a `main` method that sorts then prints the array `[6, 7, 2, 4, 3, 8, 1, 5]`.

Please submit your code and answers to the questions in a zipped folder on Brightspace by Dec 5.