Programming Assignment 5 Report

Jaiden Gerig Sec:505

Problem Description:

Implement a Minimum Priority Queue first as a sorted/unsorted array, and again as a Binary Heap.

Purpose of the Assignment:

The purpose of the assignment was to demonstrate a knowledge of both the 2 most common implementations of a Minimum Priority Queue, as well as to prepare for the final project of this class.

Data Structures Description:

By doing this assignment I Implemented a Priority Queue first using an unsorted array of items which held a pair, and the address of their repective Locator, while their Locator held the address of their respective item, the locators were all located in a Vector and had an index corresponding to their Element. next I implemented this priority queue again using a Binary Heap data structure which was implemented using an array and a similar system of Items and Locators.

Algorithm Description:

Array implementation: insertItem(Item) inserts an Item into the end of the array and happens in O(1) time as the array is unsorted. min() returns the locator of the item with the minimum key and happens in O(n) time as the array is unsorted. remove(loc) removes an element based on the locator passed to it and happens in O(n) time as the array must be shifted to fill the gap, the number of comparisons made by this function decreases as the priority queue becomes smaller. replaceKey(loc,int) replaces the key of the item which corresponds to the locator and runs in O(1) time due to the locator structure and does not make any comparisons in this implementation as min is determined when asked for. createPriorityQueue() builds a priority queue from arrays passed to it and runs in O(n) time as it simply inserts items into the end of the array. getItem(loc) returns an item and runs in O(1) time due to the locator structure.

Binary Heap implementation: InsertItem(Item) inserts an Item into the heap and happens in O(Log(n)) time due to the need to walk the item up. min() returns the locator of the item with the minimum key and happens in O(1) time as the item is always at the tope of the binary heap. remove(loc) removes an element based on the locator passed to it and happens in O(Log(n)) time as the item is walked down the heap and then removed from it, the number of comparisons made by this function decreases as the priority queue becomes smaller. replaceKey(loc,int) replaces the key of the item which corresponds to the locator and runs in O(Log(n)) time due to the locator structure, and the need to walk up and down the heap to ensure it is in the right place, this function makes slightly more comparisons as it has to walk up and down the heap.. createPriorityQueue() builds a priority queue from arrays passed to it and runs in O(nlog(n)) time as it inserts the items into the binary heap. getItem(loc) returns an item and runs in O(1) time due to the locator structure.

Program Organization:

In the array implementation, the main data structure is located in PriorityQueue.h along with the data structures it is composed of, while main.cpp holds all the tests of the data structure. in the heap implementation, BinaryHeap.h holds the heap implementation along with the locator structure, whereas PriorityQueue.h holds the prioirty queue data structure along with the data structures it is composed of, main.cpp holds all the tests of the data structure.

How to compile + run my program:

(1) type "make all" to compile (2) type "./Main" to run tests

---

C++ OO or Generic Programming features:

This progam uses templates for generic programming to paramaterize the type of the Priority Queue..

Real World Applications:

This could easily be used by any scheduling software for flights, it could be used by hospitals to prioritize care of patients, and it could also be used by servers for ensuring a correct order of operations using timestamps as keys.