CSCE 221 Cover Page

Programming Assignment #5

Due **November 18** by midnight to CSNet

First Name: Quinn Last Name: Nguyen UIN: 524002419

User Name:quinnminh_nguyen

E-mail address: quinnminh_nguyen@tamu.edu

Please list all sources in the table below including web pages which you used to solve or implement the current homework. If you fail to cite sources you can get a lower number of points or even zero, read more: Aggie Honor System Office

Type of sources			
People	Peer TA		
Web pages (provide URL)	http://www.cplusplus.com/		
Printed material			
Other Sources			

I certify that I have listed all the sources that I used to develop the solutions/codes to the submitted work.

"**On my honor as an Aggie, I have neither given nor received any unauthorized help on this academic work.**"

Your Name: Quinn Nguyen Date: 11/18/2017

Report

Purpose of the assignment: We implement a minimum binary heap by using vector to determine which element has highest priority. Next, we implement a priority queue based on Binary Heap that receives an input of a vector of objects and organize them into a binary heap according to their priority. Lastly, we apply it to CPU scheduling. Our priority queue receives a vector of jobs input and then put them into Binary Heap order based on priority. We process each of the job based on their priority.

Instruction to compile:

+There are 3 files:

. BinaryHeap.h: Implement Binary Heap and item class

.PriorityQueue.h: implement Priority Queue

.Main.cpp: get inputs and utilize priority queue.

+When compiling:

Use “make run” to compile. The program will ask for the input file. The output for the CPU operation will be cpu-jobs-output.txt. The output will process the current min node and come to the next min node till the end of the queue

Phases of the CPU:

Phase 1: Binary Heap operations:
deleteMin():delete the minimum data(first element of the vector). It takes O(logn) time complexity because we have to walk down every time to re-balance the tree.
IsEmpty(): check if the heap is empty. It takes O(1) time because it only checks if the Item vector is empty.
Insert(): insert an element by performing walk up. It takes O(logN) time because it iterates through the tree height.
The test results are included in the tar file.

Phase 2:
Priority Queue operations:
removeMin(): delete the first element of the queue. It calls the deleteMin function from Binary heap. Therefore, it also takes O(logN) time for each element.
isEmpty(): check if the queue is empty. It checks if the binary heap is empty. Therefore, it takes O(1) time.
Insert(): insert an element into the queue. It calls insert() of the binary heap class. Therefore, it also takes O(logN) time complexity.
The test results are included in the tar file.

Phase 3:
Reading N elements from the input will take O(N) time complexity. **Then insert N elements into the priority queue** will take O(NlogN) because every insertion takes O(logN).

Next, we have to process each job at a time. The time it takes to process each job is O(k) where k is the length of each job. Therefore, the **time complexity to process all of the jobs** is O(N*k). The test results are included in the tar file.