CSCE 221 Cover Page

Assignment #5

First Name Alan Last Name Achtenberg UIN 321000608

User Name alan76 E-mail address alanachtenberg@neo.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 on Aggie Honor System Office website: `http://aggiehonor.tamu.edu/`

Type of sources				
People				
Web pages (provide URL)				
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 Alan Achtenberg Date 11/9/2013

Assignment 5

due nov 9 at 11:59 pm.

Program Description:

The program main.cpp tests two different implementations of a PriorityQueue, one with a user defined Binary Heap as the underlying container and the other

with a std template libary list container. Each implementation uses the item and locator class defined in Item.h as well as different implementations of the

functions min(), remove(loc), CreatePriorityQueue(filename, loc array), isEmpty(), replaceKey(loc,k), and insertItem(k,x,loc).

Purpose:

To get better experience with creating templated types as well as to understand the heap and queue structures and how the locator can be implemented to

improve access times.

Data Stuctures:

Item<T> stores an item of type T and a int key associated with its order k, for the locator style a locator pointer was added

Locator<T> structure that contains a pointer to a Item<T>

PQcomp<T> comparison object to compare items in a standard list, returns the comparison key1-key2

BinaryHeap<T,Comp> binary heap to contain priority queue items<T> and order is maintained by Comparison object <Comp>

list<T> std template libary doubly linked list

PriorityQueue<T> simple implementation of a priority queue, not used in final program but was an important step

AdaptPriorityQueuelist<T> locator based Priority Queue with a list as a container

AdaptPriorityQueueheap<T> locator based Priority Queue with a heap as a container

Except for list the above structures are user defined adts that allowed for the complete definition of a PriorityQueue not included are several default types

as well as std library defined types.

Algorithms:

remove(loc) can be described as remove the item pointed to by locator loc. in pseudo code the algortihm is as follows

loc.item.key=smallest value; //O(1)

resort heap or list//O(nlogn) or O($\log_2$n)

call delete min //internal calls list case O(1) pop_front() heap case O($\log_2$n) walkdown()

Remove loc for the list based implementation is O(nlogn) because the sort for a list is also O(nlogn) for the heap case the implementaion is O($\log_2$n)

replaceKey(loc,k) can be described as replace key of item pointed to by loc. in pseudo code the algortihm is as follows

loc.item.key=k; //O(1)

resort heap or list// O(nlogn) or O($\log_2$n)

Just as above the list is bounded by sort to O(nlogn) and the heap is bounded by sort O($\log_2$n)

CreatePriorityQueue(filename, locators) can be described open filename read all items(includes a key and elem) initialize locator list with items and store in

Queue. in pseudo code the algortihm is as follows

vector<int> keys

vector<ElemType> elems

int counter=0;

create temp vars string and int for file reads;

ifstream in;

While in.good(){

++counter;

in.readkey
keys.pushback(key);
in.readElem
Elems.pushback(elem);}
for(int i=0;i<counter;++i)
insertItem(keys[i],elems[i],locators[i]

The CreatePriorityQueue is bounded by O(n^2logn) for list or O(nlog2n) for heap because there are n elements that need to be read and inserted and insert doesnot
have O(1) complexity as stated above

The running time for the heap is consistently faster because to sort and resort the heap is faster or as fast than list and many other implementations of array sort

Real life applications of a priority queue include a online waiting list where someone may cancel their order, higher keys represent later customers. Or a
warehouse loading bay keys represent the priority and elems are the name of the object to be loaded. Or even something such as a traffic Routing service prefered
routes are placed on a priority queue in order but when one becomes inavailable or the priority(key) changes based on real time traffic updates the priority queue
always reflects the prefered route.

Program:
The Program tests each type of AdaptPriorityQueue by first calling create list and displaying both list and Then the program recreates the queues with teh
flights information as follows <key=distance elem=# of stops and locator is the hash function first letter minus 'A'. after creating the list and displaying one the
other updates the key of LAX and is then displayed by the program

To compile and run this program simply place all files in the same directory and run the command g++-4.7 -std=c++11 main.cpp
and then the command ./a.out

Input specs.
no required user input. The file format must be
key elem
key2 elem
etc..
key is an int