Tiger Yu

Data Structure

May 4, 2018

Final Project

This project is asking us to build a priority system for a hospital due to the pregnancy problem in "my town." The purpose of this specific project is to help the doctors decide the order in which the pregnant women will be seen.

The job for me to do is to write up a program which could choose the order of the patients automatically from a given data. Also, we should create three different ways of solutions for this queue; the first one is using STL(standard template library), the second one is sorting the oder by using the concept of the linked list, and the third one is using the concept of binary minimum heap.

After finishing building up the system using theses three methods, we need to record the time data for analysis; the run time of enqueue and dequeue for 500 times with the size of data being 100 to 800.

I defined three "class" files, and three "cpp" files for each of them. The last "test.cpp" file is for testing and the output of each method which I used. The three methods matched with the name of "**stl**," "**linkedlist**," and "**heap**."

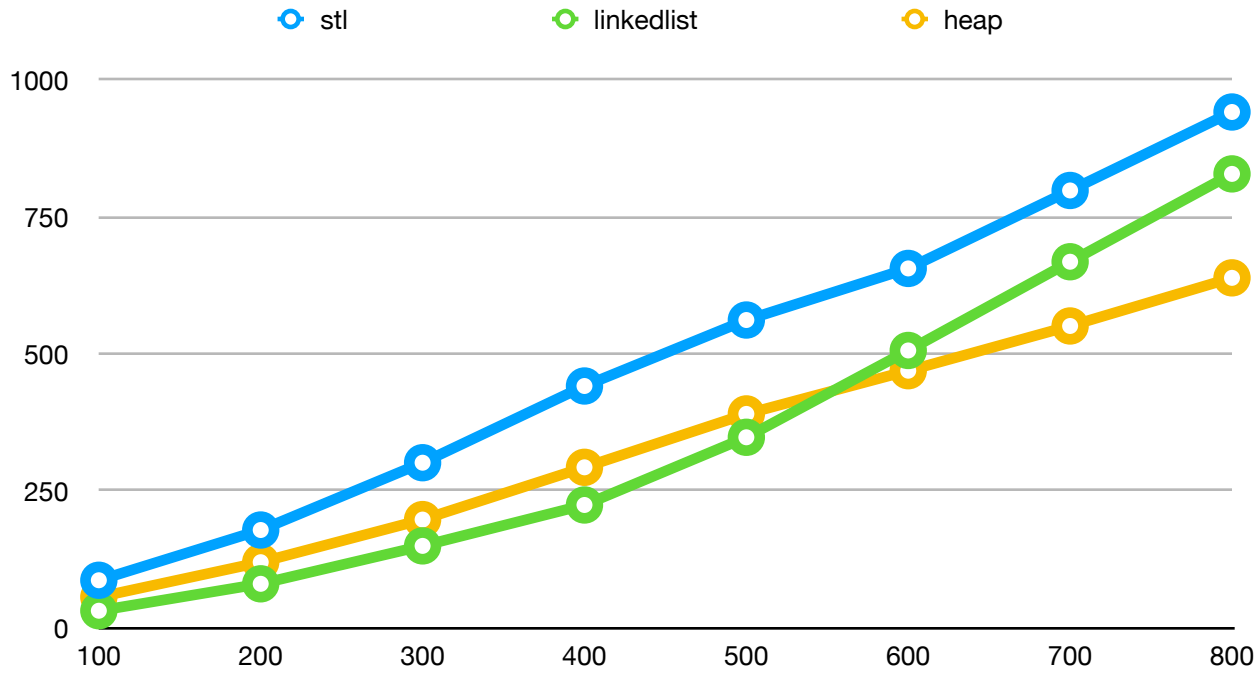
stl.hpp and stl.cpp are the functions I created for the method of STL. In the "class," I build the priority queue using the standard template library. Setting up the bool operator, it guarantees the conciseness and stability of arranging while inserting the new elements.

linkedlist.hpp and linkedlist.cpp are the functions I created for the method of sorting the order using linked lists. In this method, I used linked list as a representation and a method of creating the priority queue. A linked list queue uses a pointer pointing the head and another pointer pointing the end. This provided convenience for inserting and deleting. However, due to the characteristics of the linked list, I chose "insertion sort" as a sorting method. I did not use bubble sort or others because I believed "insertion sort" would work best in this type of special case.

heap.hpp and heap.cpp are the functions I created for the method of using the concept of binary minimum heap. The heap itself has could be sort “up” and “down.” Due to its’ complexity, at first, I used the school’s heap visualization software to understand the logic. Every time I added a data, the function will rearrange the oder. In this case, it would be for the pregnant women.

Final Project Data

	stl	linkedlist	heap
100	86.822	31.158	56.148
200	178.572	80.324	119.808
300	301.468	149.594	197.632
400	441.562	224.284	293.074
500	562.038	347.908	390.584
600	656.252	506.444	469.472
700	798.614	668.456	551.176
800	941.908	828.88	639.048



test.cpp is the test function I used for all three of the methods. Instead of writing a main function for each individual method, I thought this would be much faster, and it indeed saved me a lot of time. In the function itself, I included the other three "cpp" files, and when I was testing one of them, I would "comment" the other two; in case I have errors.

At last, after writing and testing out the three methods, I recorded the time of execution of using different size of the data and the time I ran the program. For recording the time, I used the library of "chrono," which calculated out each of the data and time I have. The unit being used was in microseconds, because the output of the time would likely to be zero if we use seconds here.

From my data showed above, we can see that the left side (time used in micro seconds), binary minimum heap used the least time. The second place would go to linked list, and the slowest of the three was STL. The result was a bit surprising because I expected the heap to be right in the middle of STL and linked list. The fact is that due to our computer, the time would vary every time even if we gave it the same data. And, our data of the patient would be too small for a big hospital, but the graph was indeed enough to show the trends of time using for each of the method.