Priority Queue for Hospital

Introduction & purpose:

As a programmer, I need to design a program for the local hospital in my town. My system can help the hospital process the pregnant people efficiently and sort patients in the appropriate sequence. In this project, I will use three methods of priority queue implementation to complete this system. They are: heap-based priority queue; linked list-based priority queue; STL-based priority queue. Then, I need to compare the time complexity of these three methods so that I can choose the best method for the hospital.

Procedure:

1.Heap based implementation

In this implementation, I used the data structure of heap. Firstly, I have a struct named 'patient'. It can store the data of patients' name, priority number and treatment number. In 'buildheap' function, I read file and store the datas into the array whose type is patient. In the function 'smallmain', I put all of the data into a heap, and pop the element with the highest priority each time. Function 'printqueue' is used to print the data after implementation. Function 'push' is used to put all the data into a proper heap. In the 'pop', I dequeue an element each time, and build a new heap in the function 'Minheapify' so that the next time I dequeue the element is the one with the highest priority. In this way, I can get a sequence of the patient according to their priority.

2.linked list-based priority queue

In this implementation, I used the data structure of linked list. Firstly, I have a struct named patient2. the struct can store the data of patients' name, patients; priority number and their treatment number. In addition, there are two pointers called *next and *parent. These two pointers are generally used in the implementation of linked list. In this implementation, I have 5 void functions besides the constructor and the destructor. In the 'buildarr' function, I read the file and draw the data into an array whose type is type is 'patient2'. Then, in the 'putinqueue' function, I put the array into a queue. The 'buildll' function can build a linked list. The data are sorted based on their priority in the linked list. After this implementation, we can get a linked list that has sorted the patients. Then, in the 'dequeue' function, the system will pop the element with the most priority. Finally, in the 'printqueue' function, it can print the patient and their corresponding priority number and treatment number.

3. STL based priority queue

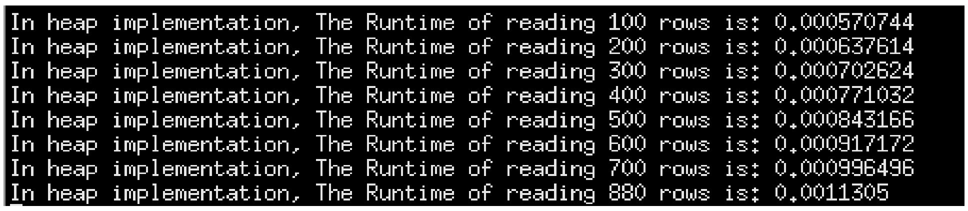
In this implementation, I used the data structure of STL (standard template library). In this implementation, I have a similar struct called 'patient3' containing the name of patients, the priority number and the treatment number of patients. And there are 4 functions in this implementation. The 'buildarr' function is used to store the data of the file into an array whose type is patient3, and the 'putinqueue' function can store the data of array into a STL priority queue. Then, the priority queue can sort the patients based on their priority number. Then, I use another priority queue to sort their treatment number if they have the same priority number. After the implementation, I can get three arrays that store the name, priority number and treatment number of those patients. ‘dequq11’ function and print functions have the same function as before.

Data:

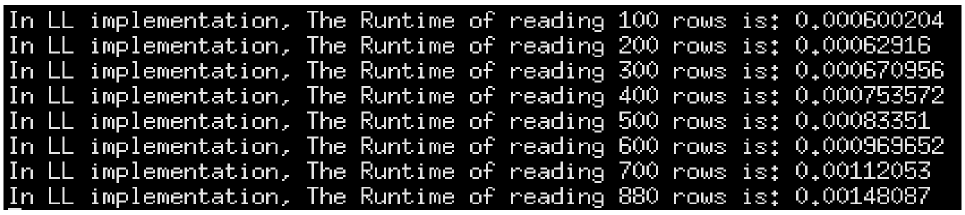
In this project, there are three very important data. They are: The name of pregnant people, their priority number and their treatment number. So, when we sort those patients, we need to compare their priority number first. If their priority numbers are the same, then compare their treatment number.

Result:

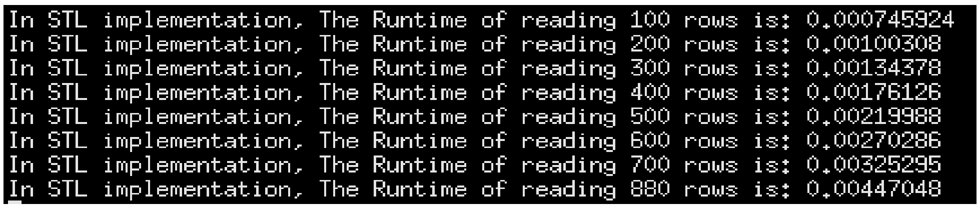
After running 8 tests 500 times each, I got some data of the runtime of these three implementations. Below is the screen shot of the runtime output:



```
In heap implementation, The Runtime of reading 100 rows is: 0.000570744
In heap implementation, The Runtime of reading 200 rows is: 0.000637614
In heap implementation, The Runtime of reading 300 rows is: 0.000702624
In heap implementation, The Runtime of reading 400 rows is: 0.000771032
In heap implementation, The Runtime of reading 500 rows is: 0.000843166
In heap implementation, The Runtime of reading 600 rows is: 0.000917172
In heap implementation, The Runtime of reading 700 rows is: 0.000996496
In heap implementation, The Runtime of reading 880 rows is: 0.0011305
```

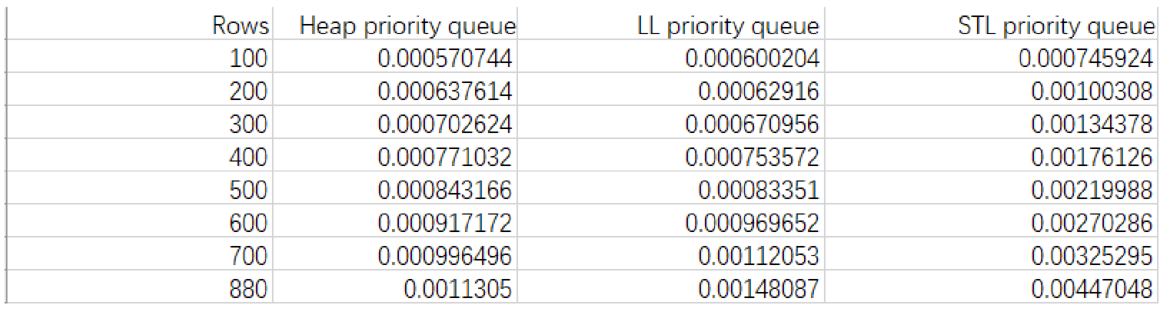


```
In LL implementation, The Runtime of reading 100 rows is: 0.000600204
In LL implementation, The Runtime of reading 200 rows is: 0.00062916
In LL implementation, The Runtime of reading 300 rows is: 0.000670956
In LL implementation, The Runtime of reading 400 rows is: 0.000753572
In LL implementation, The Runtime of reading 500 rows is: 0.00083351
In LL implementation, The Runtime of reading 600 rows is: 0.000969652
In LL implementation, The Runtime of reading 700 rows is: 0.00112053
In LL implementation, The Runtime of reading 880 rows is: 0.00148087
```



```
In STL implementation, The Runtime of reading 100 rows is: 0.000745924
In STL implementation, The Runtime of reading 200 rows is: 0.00100308
In STL implementation, The Runtime of reading 300 rows is: 0.00134378
In STL implementation, The Runtime of reading 400 rows is: 0.00176126
In STL implementation, The Runtime of reading 500 rows is: 0.00219988
In STL implementation, The Runtime of reading 600 rows is: 0.00270286
In STL implementation, The Runtime of reading 700 rows is: 0.00325295
In STL implementation, The Runtime of reading 880 rows is: 0.00447048
```

Then, I put these data into a table so that we can see those data more clearly:



Rows	Heap priority queue	LL priority queue	STL priority queue
100	0.000570744	0.000600204	0.000745924
200	0.000637614	0.00062916	0.00100308
300	0.000702624	0.000670956	0.00134378
400	0.000771032	0.000753572	0.00176126
500	0.000843166	0.00083351	0.00219988
600	0.000917172	0.000969652	0.00270286
700	0.000996496	0.00112053	0.00325295
880	0.0011305	0.00148087	0.00447048

In a nutshell, we can find that the best implementation is the heap-based priority queue since it takes the shortest mean runtime. This is because, in the heap implementation, the time complexity is log2(n), and in the linked list implementation, the time complexity is n. When n >= 1, the heap implementation is much better than the other one. Therefore, the result of the runtime I got is the same as the expected one!