[CS302-Data Structures] Homework 2: Priority Queues

Instructor: Kostas Alexis
Teaching Assistants: Shehryar Khattak, Mustafa Solmaz, Bishal Sainju
Fall 2018 Semester

Introduction: In this assignment, your goal is to design and implement a Priority Queue and further discuss aspects related to computational efficiency. In terms of how to implement the Priority Queue, you are requested to rely on using SortedLists.

The following header file for the class SL_PriorityQueue is provided to you.

```
#ifndef PRIORITY_QUEUE_
#define PRIORITY_QUEUE_

#include "PriorityQueueInterface.h"
#include "LinkedSortedList.h"
#include "PrecondViolatedExcept.h"
#include <memory>

template<class ItemType>
class SL_PriorityQueue : public PriorityQueueInterface<ItemType>
{
private:
      // Ptr to sorted list of items
      std::unique_ptr<LinkedSortedList<ItemType>> slistPtr;

public:
      SL_PriorityQueue();
      SL_PriorityQueue(const SL_PriorityQueue& pq);
      ~SL_PriorityQueue();

      bool isEmpty() const;
      bool enqueue(const ItemType& newEntry);
      bool dequeue();

      // @throw  PrecondViolatedExcept if priority queue is empty
      ItemType peekFront() const throw(PrecondViolatedExcept);
}; // end SL_PriorityQueue
#include "SL_PriorityQueue.cpp"
#endif
```

Exercise 1 (20/100). Design the ADT SL_PriorityQueue in the sense of a) writing down all the functionalities it should provide, b) write down the methods you will need (declarations) and provide a UML diagram, c) create stubs of all the methods.

Exercise 2 (60/100). Provide the implementation file of the SL_PriorityQueue. Use of smart pointers is not necessary. You can modify the header file accordingly. To verify for grading purposes the operation of the Priority Queue take the string "Abigail", assign to each character a priority key corresponding to the place of the character in the alphabet and use the Priority Queue to present it in sorted order. Present the result in your report (word/pdf file) and through a Log file.

Exercise 3 (20/100). You utilize the SL_PriorityQueue class in the framework of an application. For example, consider that you do an event-driven simulation. In particular, you simulate a pool table. As collisions suddenly change the motion of the colliding balls they have to be tackled carefully. One way will be to implement a simulation that does very small increments in time to check when a collision happens. A better approach is to check for all possible collisions and dump these "events" into a priority queue. The priority queue is ordered by the time each collision takes place. Doing this is much more efficient. Explain why you need a priority queue and not a normal (not value-based) queue. What would be the computational cost of using a normal queue instead?

Optional Bonus (+25/100): A very well-known graph search algorithm, the A*, can be implemented using Priority Queues. Provide a 1-page summary of the A* algorithm and its implementation.