Lecture 14

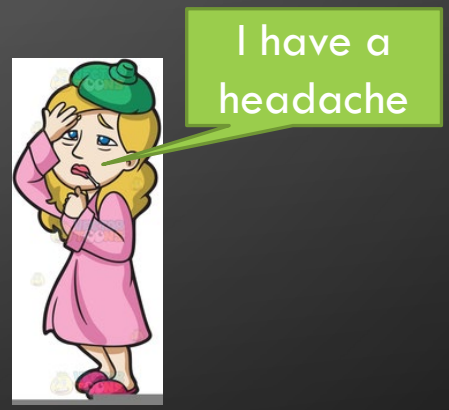
PRIORITY QUEUES

Outline

- **Recall Queues:**
  - **First in First Out (FIFO)**
    - **Double ended queues, access from the head or tail**
    - **Converse of a Stack (LIFO)**
- What is the Priority Queue ADT?
- Applications of Priority Queues

What is a *Priority* Queue

- Priority Queue is a variation based on Sorting (Order)
  - Add to the queue in priority *position*
  - This complicates the insertion
- Let's consider a hospital ER….



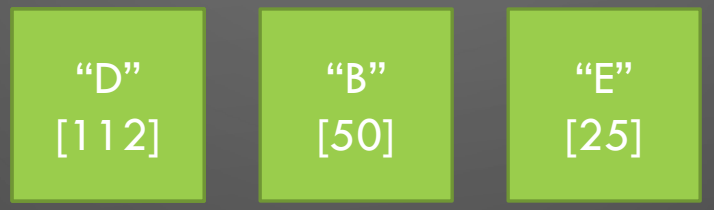
Priority Queue ADT

```
// type T must be orderable (support operator<)
template <typename T>
class AbstractPriorityQueue
{
public:

  // return true if the priority queue is empty
  virtual bool isEmpty() = 0;

  // insert item into the queue
  virtual void add(const T& item) = 0;

  // remove highest priority item from the queue
  virtual void remove() = 0;

  // get the item with the highest priority from the queue
  virtual T peek() = 0;
};
```

Example

- Priority queue, *q* contains strings with associated priority values, *k*
  - q.add(k, string)
  - What are the contents of the queue after each operation? (larger is higher priority)

1. q.add(98, “A”)
2. q.add(50, “B”)
3. q.add(131, “C”)
4. q.remove()
5. q.remove()
6. q.add(112, “D”)
7. q.add(25, “E”)



Example uses of Priority Queue

- Process Scheduling
  - Operating System handles multiple processes
  - You can weight the processes (nice number) to give some processes higher priority
  - Starved processes will increase in priority until they run then return to original priority
- Network routing and switching
  - Need to guarantee a particular QoS for packets to reach their destination

Complexity

Operation	Array		Linked-List	
	Best	Worst	Best	Worst
add	O(1)	O(n)	O(1)	O(n)
remove	O(1)	O(1)	O(1)	O(1)
peek	O(1)	O(1)	O(1)	O(1)

Challenge 1

Assume that you have two Dragon Eggs and we have a building made of 50 floors. How can we find the exact floor that the egg will break, if it was thrown from, in the minimum number of trials.

Challenge 2

Given an array and a number k where

$$1 \leq k \leq n$$

and n is the number of array elements. How can we find the k'th smallest element in the given array in the shortest time (on average). It is given that all array elements are distinct and that the array is not sortedin a given order.

Assignment/Homework

- Reading pp. 445-470
- ICE 6 due on Today