Lecture 14: Queues

CS 62

Fall 2018

Alexandra Papoutsaki & William Devanny

Queue ADT

If you're from the UK, you call a "line" (like waiting in line) a queue

Queue: linear data structure that stores arbitrary objects

Objects are inserted at the end of the queue and removed at the front following the FIFO principle (First-In First-Out)

Operations:

- enqueue (at end/tail)
- dequeue (from front/head)

Like a stack the only way we manipulate the data is by adding and removing items, the difference is where we add and remove the items

Queue Interface

```
public interface Queue<E> extends Linear<E> {
  //same as add(E item)
  public void enqueue(item); //add item to tail of queue
  //same as remove()
  public E dequeue(); //remove item from head of queue
  //same as get() and getFirst()
  public E peek(); //return reference to head
  public boolean empty();
  public int size();
}
```

Queue Applications

- Scheduling tasks
- Which process/job should run next (e.g., printer)

- Modeling real world phenomena (lines show up in lots of places, e.g., call centers)

- Search (Breadth First Search)
  - VS Depth First Search when using stacks
  - More on that in Graphs

Implementing Queues with Linked Lists

- Where should we add and remove items?
  - Doesn't matter but we do need a doubly linked list
  - We will add them at the tail and remove them from the head

`QueueList` in `structure5` uses a circular linked list

Runtime of the different operations (assuming doubly linked list):

- `enqueue()`: $O(1)$
- `dequeue()`: $O(1)$
- `peek()`: $O(1)$
- `empty()`: $O(1)$

Implementing Queues with ArrayLists

Where should we add and remove items?

If we add to the back, then remove will be expensive
If we add to the front, then add will be expensive
We will add them at the back and remove them from the front

Look at `QueueVector` in `structure5`

Runtime of the different operations:

- `enqueue()`: $O(1)$
- `dequeue()`: $O(n)$
- `peek()`: $O(1)$
- `empty()`: $O(1)$

Dequeue

- Steque:
  - Add and remove from one end. Only add from other.

- java.util.Deque: Double-Ended Queue
  - Can add or remove from either end.
  - Resizable array implementation
  - Faster than Java Stack class when used as stack, faster than LinkedList (doubly-linked) when used as queue.