CSCI 241
Scott Wehrwein
The Priority Queue ADT
Goals

Understand the purpose and interface of the Priority Queue ADT.
Priority Queue
Queue vs Priority Queue

add (enqueue):
insert an item into the queue

remove (dequeue):
remove the first item to be inserted

add (add):
insert an item into the queue

remove (poll):
remove the highest-priority item in the queue
What is this good for?

Systems - CPU scheduling, Virtual memory paging

<table>
<thead>
<tr>
<th>Process</th>
<th>Arrival Time</th>
<th>Burst Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1(0, 1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P2(0, 1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P3(0, 1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P4(3, 1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P5(3, 2)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P6(3, 3)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P7(7, 3)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P8(7, 2)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P9(7, 1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P10(13, 1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P11(13, 2)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P12(13, 3)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P13(17, 1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P14(17, 2)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Wait time**
- **Burst time**
What is this good for?

Computer Graphics - Mesh simplification, Collision detection
What is this good for?

AI - Robot Navigation and Path Planning
What is this good for?

Statistics (largest M values)
Computer Science (bin packing, graph search)

In fact, you'll use your Priority Queue (A3) to implement a graph algorithm (A4)!
Priority Queue

Like a Queue, but:

• Each item in the queue has an associated priority which implements Comparable
• Removing an element (poll) returns the item with the “highest priority”
  defined in this class as: the element with the “smallest” associated priority value

Take note:
this is easy to get mixed up

Ties are broken arbitrarily
Priority Queue: Java

interface PriorityQueue<E> {
  boolean add(E e); // insert e
  E peek(); // return min element
  E poll(); // remove/return min element
  void clear();
  boolean contains(E e);
  boolean remove(E e);
  int size();
  Iterator<E> iterator();
}

E represents the value and is also Comparable.
  i.e., determines the priority
interface PQ<V, P extends Comparable<P>> {
    boolean add(V v, P p); // add v w/ priority p
    V peek(); // return highest-priority val
    V poll(); // remove/return highest-priority val
    void clear();
    boolean contains(V val);
    void changePriority(V v, P newP)
    int size();
}