ADTs – PriorityQueue

PriorityQueue	maintain removal order when there are out-of-order additions	• insert(x,p) – insert elt x with priority p • findMin() or findMax() – find elt with min/max priority • deleteMin() or deleteMax() – remove (and return) elt with min/max key note: a PQ is typically either a min-PQ or a max-PQ – it does not support both min and max operations simultaneously

Basic Implementation of PriorityQueue

We need some kind of collection to hold the keys/elements in the PQ.

There are two basic collections
- array
- linked list

and two basic ways elements can be ordered within those collections
- not sorted
- sorted

Basic Implementation of PriorityQueue

operation	array - unsorted	array - sorted	linked list - unsorted	linked list - sorted
find min				
insert				
remove min				

Basic Implementation of PriorityQueue

operation	array - unsorted	array - sorted	linked list - unsorted	linked list - sorted
find min	**O(n)** – search	**O(1)** – in slot 0	**O(n)** – search	**O(1)** – at head
insert	**O(1)** – add at end	**O(n)** – binary search + shift	**O(1)** – add at head	**O(n)** – sequential search
remove min	**O(n)** – search + delete (swap)	**O(n)** – shift	**O(n)** – search + delete	**O(1)** – at head

Can we avoid (some) searching and shifting?
– store min location (update on insert, remove)
– circular array or reverse sorted array

Basic Implementation of PriorityQueue

operation	array - unsorted	array – reverse sorted	linked list - unsorted	linked list - sorted
find min	**O(1)** – store index of min	**O(1)** – in last slot	**O(1)** – store node with min	**O(1)** – at head
insert	**O(1)** – add at end	**O(n)** – binary search + shift	**O(1)** – add at head	**O(n)** – sequential search
remove min	**O(n)** – delete (swap) + update min index	**O(1)** – in last slot	**O(n)** – update min node	**O(1)** – at head

Tradeoff: fast insert or fast remove, but not both.

Can we do better?