CSC 220 Data Structures

Priority Queues and Heaps I – API and complexities
Parkland College Spring 2016
20160307
Priority Queues

- FIFO, but with ability to “jump the line” with a priority
- Pre-emptive multitasking Oses
- Load balancing servers
- Priority mail/packaging
- Priority boarding
- Triage
- Common helper data structure
API

- `priority_queue_base.py`
- **keys must be orderable**
  - `__lt__()`
  - relies on `_key` members having meaningful `<`
- arbitrary that min is most important
  - could just as easily say max is most important
  - most important to be consistent
Linked List Implementations

• `unsorted_priority_queue.py`
  • enqueue as regular queue
  • linear search to dequeue

• `sorted_priority_queue.py`
  • linear search to enqueue
  • dequeue as regular queue

• complexity chart in notes/textbook
Heaps

• binary tree with following conditions
  • either top heavy
    • recursively: subtree’s highest value at root.
  • or bottom heavy
    • recursively: subtree’s lowest value at root.
• a complete binary tree
  • filled in by levels from left to right
  • guarantees height is floor(log n).
Heap Implementation

- `heap_priority_queue.py`
  - min heap (could have just as easily been a max heap)
  - enqueue with up-heap bubbling
    - (diagram on board, notes, textbook)
  - dequeue with down-heap bubbling
    - (diagram on board, notes, textbook)
- almost always array form, “binary tree” is implicit
- $O(\log n)$/amortized $O(\log n)$ enqueues/dequeues
- Python’s `heapq`