EECS 2011 M: Fundamentals of Data Structures

Suprakash Datta
Office: LAS 3043

Course page: `http://www.eecs.yorku.ca/course/2011M`
Also on Moodle

Adaptable Priority Queues

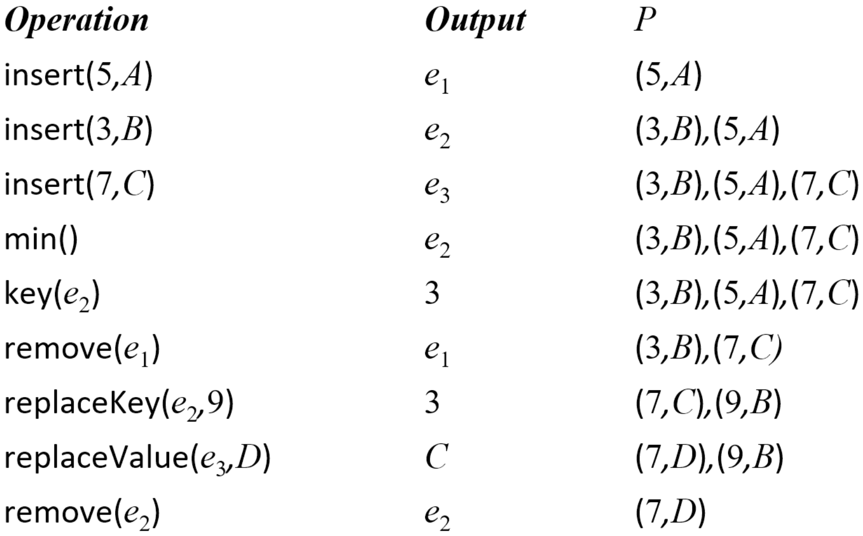
Ch. 9.5

Motivation: Suppose we want to

- remove an element that is not the minimum, given its value?
- remove an element that is not the minimum, given its key?
- update the value of an element?
- update the key of an element?

Note: Some slides in this presentation have been adapted from the authors' slides.

Adaptable Priority Queue - Trace



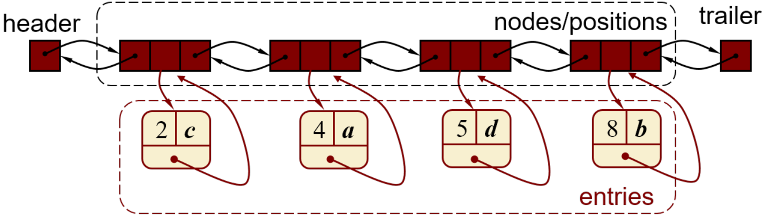
Operation	***Output***	P
insert(5,A)	e_1	(5,A)
insert(3,B)	e_2	(3,B),(5,A)
insert(7,C)	e_3	(3,B),(5,A),(7,C)
min()	e_2	(3,B),(5,A),(7,C)
key(e_2)	3	(3,B),(5,A),(7,C)
remove(e_1)	e_1	(3,B),(7,C)
replaceKey(e_2,9)	3	(7,C),(9,B)
replaceValue(e_3,D)	C	(7,D),(9,B)
remove(e_2)	e_2	(7,D)

Locating Entries

- In order to implement the operations remove(e), replaceKey(e,k), and replaceValue(e,v), we need a fast way of locating an entry e in a priority queue.
- We can always just search the entire data structure to find an entry e, but this takes $\Omega(n)$ time.
- Using location-aware entries, this can be reduced to $O(1)$ time.
- A location-aware entry identifies and tracks the location of its (key, value) object within a data structure

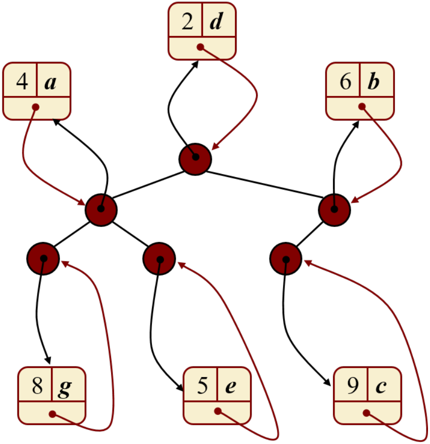
List Implementation

- A location-aware list entry is an object storing
  - key
  - value
  - position (or rank) of the item in the list
- In turn, the position (or array cell) stores the entry
- Back pointers (or positions) are updated during swaps



Heap Implementation

- A location-aware list entry is an object storing
  - key
  - value
  - position of the entry in the underlying heap
- In turn, each heap position stores an entry
- Back pointers (or positions) are updated during swaps



Performance

- Improved times thanks to location-aware entries are highlighted in red

Method	Unsorted List	Sorted List	Heap
size, isEmpty	$O(1)$	$O(1)$	$O(1)$
insert	$O(1)$	$O(n)$	$O(\log n)$
min	$O(n)$	$O(1)$	$O(1)$
removeMin	$O(n)$	$O(1)$	$O(\log n)$
remove	$\boldsymbol{O}(1)$	$\boldsymbol{O}(1)$	$\boldsymbol{O}(\log \boldsymbol{n})$
replaceKey	$\boldsymbol{O}(1)$	$O(n)$	$\boldsymbol{O}(\log \boldsymbol{n})$
replaceValue	$\boldsymbol{O}(1)$	$\boldsymbol{O}(1)$	$\boldsymbol{O}(1)$