CS109A Notes for Lecture 3/4/96

Priority Queues

1. Model = set with *priorities* associated with elements.

   - Priorities are comparable by a $<$ operator, e.g., priorities could be real numbers.

2. Operations:

   a) *Insert* an element with a given priority.

   b) *Deletemax* = find and remove from the set the element with highest priority.

Example PQ Implementation

Linked list, sorted by priority. If n elements:

- Deletemax is $O(1)$ — just take head.
- Insert is $O(n)$ — on average you go halfway down the list; in worst case all the way.

Partially Ordered Tree

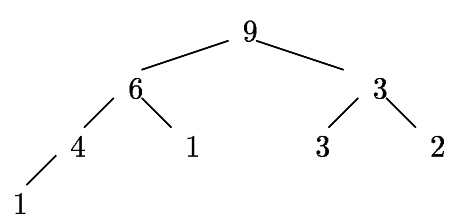
A better implementation of PQ.

- A binary tree with elements and their priorities at nodes, satisfying the *partially ordered tree (POT) property*:

   - The priority at any node $\geq$ the priority at either of its children.

Balanced POT

A POT in which all nodes exist, down to a certain level, and for one level below that, "extra" nodes may exist, as far left as possible.



Insertion on Balanced POT

1. Place new element at the leftmost possible place, e.g., as right child of 4 in tree above.

2. *Bubbleup* new node by repeatedly exchanging it with its parent if it has higher priority than parent.

   - ☐ Restores the POT property, because violations are fixed and no new ones are created (if i becomes the parent of j, then i is even bigger than the old parent of j).

   - ☐ Maintains the "balanced" property.

   - ☐ $O(\log n)$, because an n-node balanced tree has no more than $1 + \log_2 n$ levels.

Deletemax on Balanced POT

1. Select element at root.

2. Replace root element by rightmost element at lowest level.

3. *Bubbledown* root by repeatedly exchanging it with the larger of its children, as long as it is in a position where it is smaller than either of its children.

   - ☐ Restores the POT property, because violations are fixed and no new ones are created.

   - ☐ Preserves "balanced" property.

   - ☐ $O(\log n)$ because $O(1)$ work done at each step along a path of length at most $\log_2 n$.

Heap Data Structure

Represent a balanced POT by an array A.

- n nodes represented by $A[1]$ through $A[n]$.

   - ☐ $A[0]$ not used.

- Array holds elements of tree level-by-level.

Example: For our example POT: 9,6,3,4,1,3,2,1.

- $A[1]$ = root.
- A node represented by $A[i]$ has parent $A[i/2]$.

Bubbleup on Heap

At position i:

1. If $i = 1$, then done.
2. Otherwise, compare $A[i]$ with $A[i/2]$. If $A[i]$ is smaller, done. Else, swap and repeat at $i/2$.

Bubbledown on Heap

At position i, heap in $A[1]$ through $A[n]$.

1. If $2i > n$, then done.
2. Otherwise, see if $A[i]$ is smaller than $A[2i]$ or (if $2i + 1 \leq n$) $A[2i + 1]$. If not, done; if so, swap with larger and repeat there.

Insert on Heap

Put new element in $A[n + 1]$, add 1 to n, and bubbleup.

Deletemax on Heap

Take $A[1]$, replace it by $A[n]$, subtract 1 from n, and bubbledown.

Heapsort

1. *Heapify* array A by bubbling down $A[i]$ for $i = n/2, n/2 - 1, \ldots, 1$, in that order.
   - ☐ Takes $O(n)$ time total — argument is triangular sum trick as in FCS, p. 283.
2. Repeatedly deletemax, until all are selected. Yields elements in order of priority = reverse of sorted order.
   - ☐ $O(n \log n)$ for n deletemax's.