CSC 282

Heaps, heapsort

Daniel Stefankovic,
Girts Folkmanis

Insertion / Merge / Heapsort

- Insertion sort:
  - $O(n^2)$
  - Sorts in place.
- Merge sort:
  - $O(n \lg n)$
  - Needs $O(n)$ extra space.
- Heap sort:
  - $O(n \lg n)$
  - Sorts in place!
  - In practice usually slower than Quicksort.
  - Not a stable sort.

Heaps, huh?

- Heap - data structure used in heapsort.
- Different concept that in memory management.
- Useful for priority queues as well.
- Heap (almost always) is a “nearly” complete binary tree – gets filled on all levels, except on lowest. Then – from left to right.
- Each node in the tree contains a value.

Heaps!

- Nodes of heaps satisfy **heap property**.
- **Max-heap property:**
  - For every node i:
    A[PARENT(i)] >= A[i]
  - Root - largest.
- **Min-heap property:**
  - For every node i:
  - A[PARENT(i)] <= A[i]
  - Root – smallest.
- We will use max-heaps.
- Example.

Implementing a heap

- Storage – array, with the following indexing:
  - PARENT(i) = floor(i/2)
  - LEFT-CHILD(i) = 2 * i
  - RIGHT-CHILD(i) = 2 * i + 1
  - Quick to calculate (by shifting).
- Example.

Height of a heap

- **Height of a node** – number of edges on the longest downward path to a leaf.
- **Height of the heap** – height of root node.
- Height = $\Theta(\lg n)$, where n – number of elements
- We will use heap-size[A] <= length[A] in our algorithms.

Our building blocks

- MAX- HEAPIFY(A, i) – fix a node i in heap A by “floating down” the value – O(lg n)
- BUILD- MAX- HEAP(A) – produces max-heap from unordered array – O(n)
- HEAPSORT(A) – sorts array in place – O(n lg n)

MAX- HEAPIFY(A, i)

- A – possibly “broken” heap, i - index.
- Assumption: children LEFT(i) and RIGHT(i) are max-heaps, A[i] might be smaller.
- Goal: “float down” the value at A[i] to it's correct place down to heap-size[A].
- Idea: determine the largest of A[i], A[LEFT(i)], A[RIGHT(i)]. If that is not A[i], swap, and recurse.
- Running time O(lg n). Example.

BUILD-MAX-HEAP(A)

- Goal: convert an array into a max-heap.
- Idea: "float up" values that are in wrong places.
- BUILD-MAX-HEAP(A):

```
heap-size[A] = length[A]
FOR i = floor(length[A]/2) DOWNTO 1
    MAX-HEAPIFY(A, i)
```

- Example.
- Running time: O(n), using clever summation (see the book for details).

HEAPSORT(A)

- BUILD-MAX-HEAP(A)
  FOR i = length[A] DOWNTO 2
    Exchange A[1] <-> A[i]
    heap-size[A] = heap-size[A] - 1
    MAX-HEAPIFY(A, 1)
- Idea: build a max-heap, pull out the top element, replace with last, re-run heapify, repeat. Example.
- Running time: O(n) + n-1 * O(lg n) = O(n lg n)

Priority queues

- **Priority queue** – set S of elements, each associated with a **key**.
- **Max-priority-queue:**
  - **INSERT(S, x)**
  - **MAX(S)**
  - **EXTRACT-MAX(S)**
- Example: scheduling jobs.

Priority queue implementation

- MAX(A): return A[1]
- EXTRACT- MAX(A):
  - Idea: remember the root, move the last element to first, shrink the heap by 1, heapify.
  - O(lg n)
- INSERT(A, x):
  - Idea: increase heap, add the element as last, move A[i] up while A[i] > A[PARENT(i)]