Heapsort

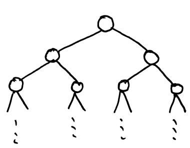
Chapter 6

Review of Binary Trees

- What is a binary tree?
- What is the depth of the node?
- What is the height of a node?
- What is the height of the tree?
- What is a complete binary tree?

Facts about Perfect Binary Trees

DEPTH	# OF NODES
0	1
1	2
2	4
⋮	⋮
h	2^h



TOTAL: $\sum_{i=0}^{h} 2^i = 2^{h+1} - 1$

Complete Binary Trees

- Nodes at depth h (the lowest level) are as far left as possible
- What is the relationship between the height and the number of nodes?

Heaps

- A ***heap*** is an complete binary tree
- Extra nodes go from left to right at the lowest level
- Where the value at each node is ≥ the values at its children (if any)
- This is called the ***heap property*** for max-heaps

Example

Storing Heaps

- As arrays!
- Root of tree is:
- Parent of A[i] is:
- Left child of A[i] is:
- Right child of A[i] is:

Example

- n = 13

92 85 73 81 44 59 64 13 23 36 32 18 54

Functions on Heaps

- MAX-HEAPIFY
- BUILD-MAX-HEAP
- HEAPSORT
- MAX-HEA-INSERT
- HEAP-EXTRACT-MAX
- HEAP-INCREASE-KEY
- HEAP-MAXIMUM

MAX-HEAPIFY

$\text{MAX-HEAPIFY}(A, i, n)$
$l \leftarrow \text{LEFT}(i)$
$r \leftarrow \text{RIGHT}(i)$
if $l \leq n$ and $A[l] > A[i]$
  then $largest \leftarrow l$
  else $largest \leftarrow i$
if $r \leq n$ and $A[r] > A[largest]$
  then $largest \leftarrow r$
if $largest \neq i$
  then exchange $A[i] \leftrightarrow A[largest]$
    $\text{MAX-HEAPIFY}(A, largest, n)$

Example

- 15 6 4 8 5 3 1 2 7   i = 2

BUILD-MAX-HEAP

$\text{BUILD-MAX-HEAP}(A, n)$
for $i \leftarrow \lfloor n/2 \rfloor$ **downto** 1
  do $\text{MAX-HEAPIFY}(A, i, n)$

Example

- 4 3 7 13 1 20 12 16 2 18

HEAPSORT

HEAPSORT(A, n)
BUILD-MAX-HEAP(A, n)
for $i \leftarrow n$ **downto** 2
    do exchange $A[1] \leftrightarrow A[i]$
        MAX-HEAPIFY$(A, 1, i - 1)$

Example

- 20 18 12 16 3 7 4 13 2 1

Priority Queues

- Priority Queues are an example of an application of heaps.
- A priority queue is a data structure for maintaining a set of elements, each with an associated key.

16

Priority Queues

- Max-priority queue supports dynamic set operations:
  - INSERT(S, x): inserts element x into set S.
  - MAXIMUM(S): returns element of S with largest key.
  - EXTRACT-MAX(S): removes and returns element S with largest key.
  - INCREASE-KEY(S, x, k): increases value of element x's key to k. Assume k >= x's current key value.

17

HEAP-MAXIMUM(A)

HEAP-MAXIMUM(A)
return $A[1]$

Time: $\Theta(1)$.

18

HEAP-EXTRACT-MAX

- Given the array A:
  - Make sure heap is not empty.
  - Make a copy of the maximum element.
  - Make the last node in the tree the new root.
  - Re-heapify the heap, with one fewer node.
  - Return the copy of the maximum element.

```
HEAP-EXTRACT-MAX(A, n)
if n < 1
   then error "heap underflow"
max ← A[1]
A[1] ← A[n]
MAX-HEAPIFY(A, 1, n − 1)    ▷ remakes heap
return max
```

19

Example

- 15 6 4 8 5 3 1 2 7

20

HEAP-INCREASE-KEY

- Given set S, element x, and new key value k:
  - Make sure >= x's current key.
  - Update x's key value to k.
  - Traverse the tree upward comparing x to its parent and swapping keys if necessary, until x's key is smaller than its parent's key.

```
HEAP-INCREASE-KEY(A, i, key)
if key < A[i]
   then error "new key is smaller than current key"
A[i] ← key
while i > 1 and A[PARENT(i)] < A[i]
     do exchange A[i] ↔ A[PARENT(i)]
        i ← PARENT(i)
```

21

Example

- Increase key of node 6 in previous example to 20

MAX-HEAP-INSERT

- Given a key k to insert into the heap:
  - Insert a new node in the very last position in the tree with the key -infinity.
  - Increase the -infinity key to k using the HEAP-INCREASE-KEY procedure.

MAX-HEAP-INSERT(A, key, n)
$A[n+1] \leftarrow -\infty$
HEAP-INCREASE-KEY$(A, n+1, key)$

Example

- Insert 12 into the above heap.