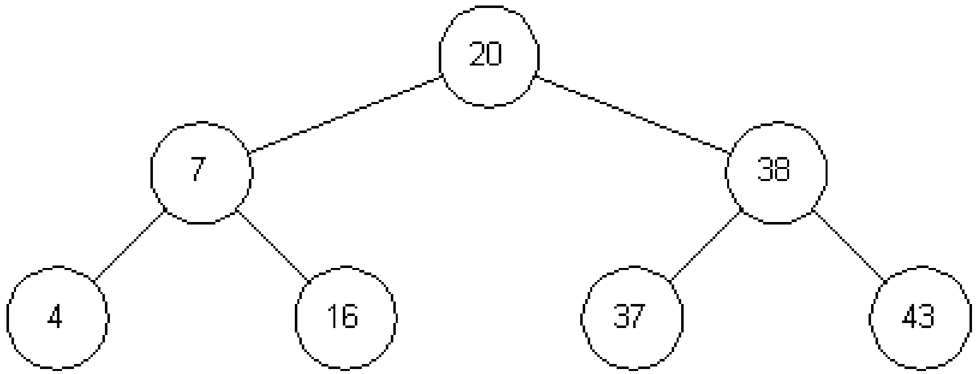
Storing binary trees as arrays



20	7	38	4	16	37	43

Heaps (Max-Heap)

43	16	38	4	7	37	20

43	16	38	4	7	37	20	2	3	6	1	30

HEAP represents a binary tree stored as an array such that:

- Tree is filled on all levels except last
- Last level is filled from left to right
- Left & right child of i are in locations 2i and 2i+1
- HEAP PROPERTY:
  Parent value is at least as large as child's value

HeapSort

- First convert array into a heap (BUILD-MAX-HEAP, p133)
- Then convert heap into sorted array (HEAPSORT, p136)

Max-Heapify(array a, integer i)

```
l = left(i)
r = right(i)
if ((l <= size(a)) & (a[l] > a[i])) then
        largest = l
else largest = i
if ((r <= size(a)) & (a[r] > a[largest])) then
        largest = r
if largest ≠ i then
        swap(a[i], a[largest])
        Max-Heapify(a, largest)
```

O(log(size of subtree))

O(height of node in location i)

??

p130

Build-Max-Heap(array a)

size[a] = length[a];

for i = ⌊ length[a]/2 ⌋ downto 1 do

    Max-Heapify(a,i)

HeapSort(array a)

```
Build-Max-Heap(a);                          ??
for i = length(a) downto 2 do
        swap(a[1], a[i]);
        size[a] --;                          O(n log n)
        Max-Heapify(a,1);      O(log n)
```

Total: O(n log n)

HeapSort Analysis

For the HeapSort analysis, we need to compute:

$$\sum_{h=0}^{\lfloor \log n \rfloor} \frac{h}{2^h}$$

We know from the formula for geometric series that

$$\sum_{k=0}^{\infty} x^k = \frac{1}{1-x}$$

Differentiating both sides, we get

$$\sum_{k=0}^{\infty} kx^{k-1} = \frac{1}{(1-x)^2}$$

Multiplying both sides by x we get

$$\sum_{k=0}^{\infty} kx^k = \frac{x}{(1-x)^2}$$

Now replace $x = 1/2$ to show that

$$\sum_{h=0}^{\lfloor \log n \rfloor} \frac{h}{2^h} \leq \frac{1}{2}$$

Sorting Algorithms

- Number of Comparisons
- Number of Data Movements
- Additional Space Requirements

Sorting Algorithms

- Selection Sort
- Insertion Sort
- Bubble Sort
- Shaker Sort

- Merge Sort
- Heap Sort
- Quick Sort

- Bucket & Radix Sort
- Counting Sort

Animation Demos

http://www-cse.uta.edu/~holder/courses/cse2320/lectures/applets/sort1/heapsort.html

http://cg.scs.carleton.ca/~morin/misc/sortalg/

Bucket Sort

- N values in the range [a..a+m-1]
- For e.g., sort a list of 50 scores in the range [0..9].
- Algorithm
  - Make m buckets [a..a+m-1]
  - As you read elements throw into appropriate bucket
  - Output contents of buckets [0..m] in that order
- Time O(N+m)

Stable Sort

- A sort is stable if equal elements appear in the same order in both the input and the output.
- Which sorts are stable? Homework!

Radix Sort

3 5 9	3 5 9	3 3 6	3 3 6
3 5 7	3 5 7	3 5 9	3 5 1
3 5 1	3 5 1	3 5 7	3 5 5
7 3 9	3 3 6	3 5 1	3 5 7
3 3 6	3 5 5	3 5 5	3 5 9
7 2 0	7 3 9	7 2 0	7 2 0
3 5 5	7 2 0	7 3 9	8 3 9

Algorithm

for i = 1 **to** d **do**

  sort array A on digit i using any sorting algorithm

Time Complexity: $O((N+m) + (N+m^2) + \ldots + (N+m^d))$

Space Complexity: $O(m^d)$

Radix Sort

3 2 9		7 2 **0**		7 **2** 0		**3** 2 9
4 5 7		3 5 **5**		3 **2** 9		**3** 5 5
6 5 7		4 3 **6**		4 **3** 6		**4** 3 6
8 3 9	→	4 5 **7**	→	8 **3** 9	→	**4** 5 7
4 3 6		6 5 **7**		3 **5** 5		**6** 5 7
7 2 0		3 2 **9**		4 **5** 7		**7** 2 0
3 5 5		8 3 **9**		6 **5** 7		**8** 3 9

Algorithm

for i = 1 **to** d **do**

    sort array A on digit i using a stable sort algorithm

Time Complexity: O((n+m)d)

Counting Sort

Initial Array

1	2	3	4	5	6	7	8
2	5	3	0	2	3	0	3

Counts

0	1	2	3	4	5
2	0	2	3	0	1

Cumulative Counts

0	1	2	3	4	5
2	2	4	7	7	8