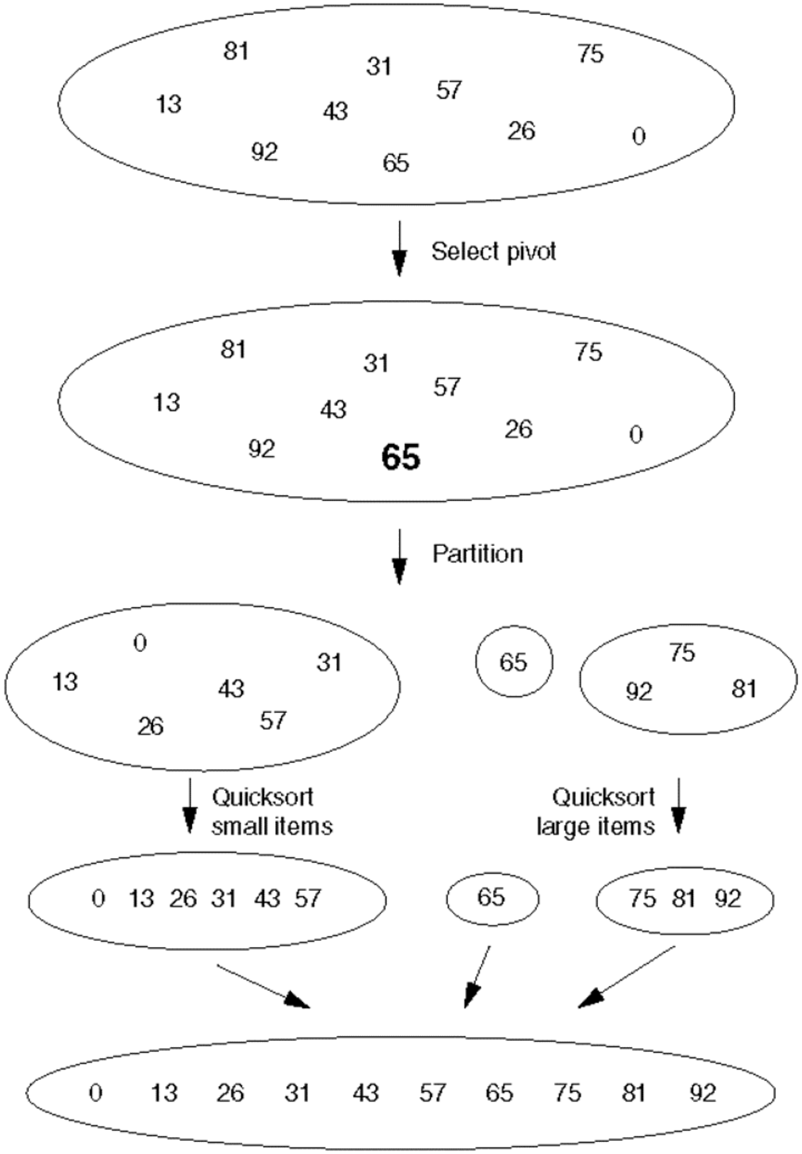
Figure 8.10 Quicksort



Partition

Figure A If 6 is used as pivot, the end result after partitioning is as shown in the Figure B.

2	1	4	5	0	3	9	8	7	6

Figure B Result after Partitioning

2	1	4	5	0	3	6	8	7	9

```
algorithm QuickSort(array a, integer p, integer r)
{   if (p < r) then
        q = Partition(a, p, r)
        QuickSort(a, p, q-1)
        QuickSort(a, q+1, r)
}
```

Sort array a from locations p through r

```
algorithm Partition(array A, integer p, integer r)
{ x = a[r]
  i = p-1
  for j = p to r-1 do
      if a[j] <= x) then
          i++
          exchange(a[i], a[j])
exchange(a[i+1], a[r])
return i+1
}
```

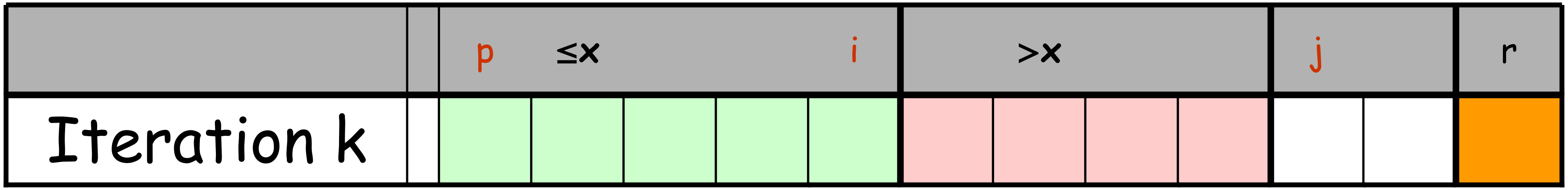
Partition array a from locations p through r using any pivot.
Return location of pivot

Page 146, CLRS

QuickSort

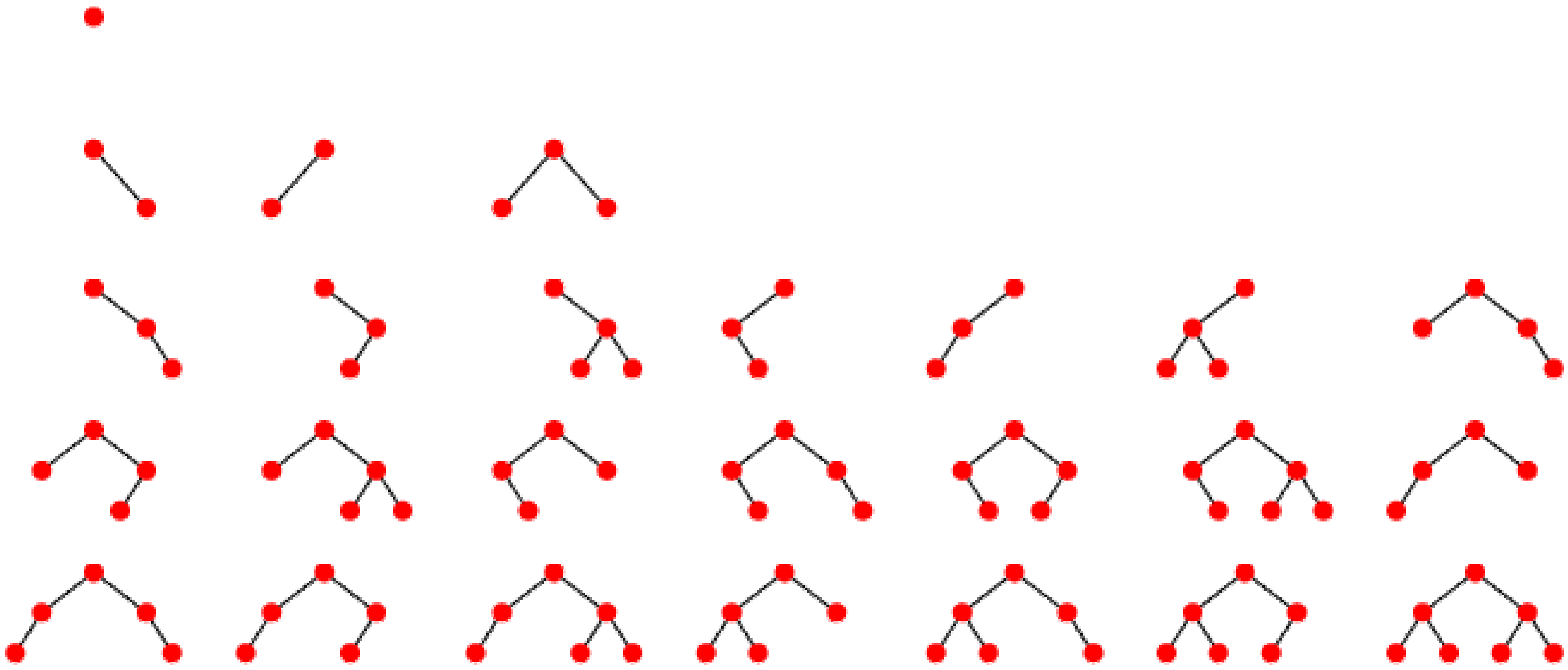
Array indices		0	1	2	3	4	5	6	7
p = 0; r = 7; i = -1; j = 0		2	8	7	1	3	5	6	4
i = 0; a[0] ↔ a[0]; j = 1		2	8	7	1	3	5	6	4
i = 0; j = 2		2	8	7	1	3	5	6	4
i = 0; j = 3		2	8	7	1	3	5	6	4
i = 1; a[1] ↔ a[3]; j = 4		2	1	7	8	3	5	6	4
i = 2; a[2] ↔ a[4]; j = 5		2	1	3	8	7	5	6	4
i = 2; j = 6		2	1	3	8	7	5	6	4
i = 2; j = 7 > 6		2	1	3	8	7	5	6	4
a[3] ↔ a[7]		2	1	3	4	7	5	6	8

QuickSort



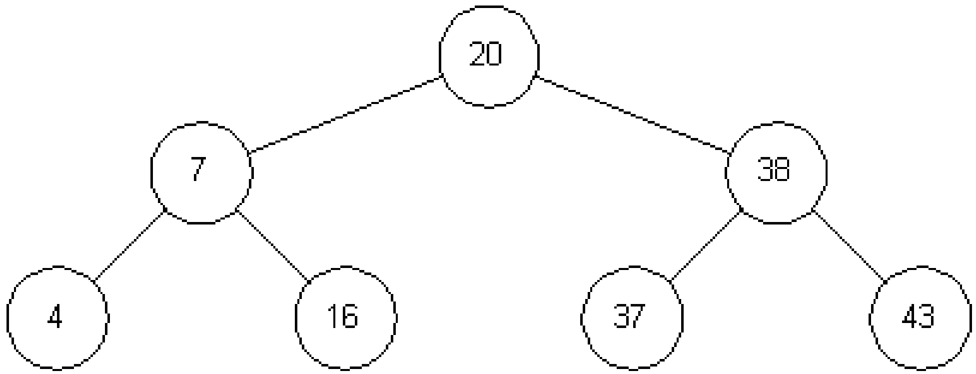
Invariant:

Binary Trees



http://mathworld.wolfram.com/BinaryTree.html

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)

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 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+k)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