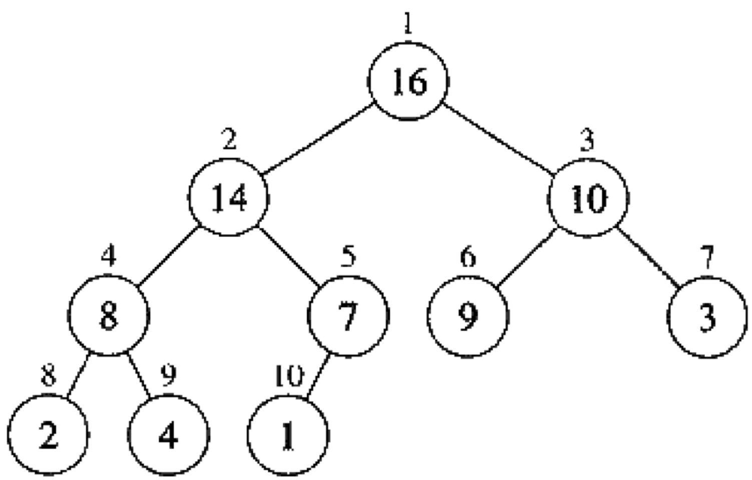
CS583 Lecture 03
Sorting&Order Statistics

Jyh-Ming Lien

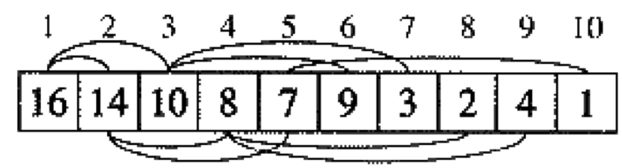
some materials here are based on Prof. Shehu, and Prof. Wang's past lecture notes

Heap Sort

- What is a heap? What is NOT a heap?



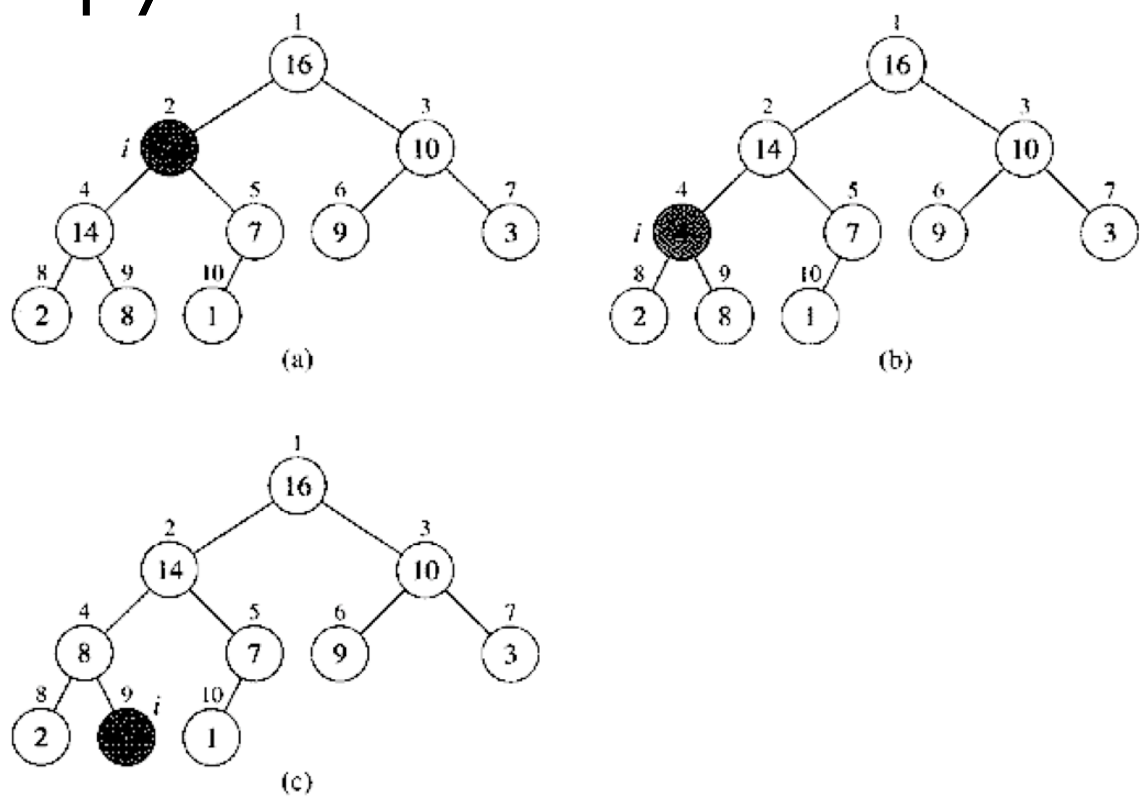
(a)



(b)

Heap Sort

- Heapfy



Heap Sort

- build a heap: A={4,1,3,2,16,9,10,14,8,7}

Heap Sort

- Extract max
  - swap the first and the last elements
  - heapfy
- Insert a new value
  - append the new value
  - heapfy
- update the value of a node
  - change the value
  - heapfy

Heap sort

- heap sort
  - build a heap
  - extract max *n-1* times
- Sort A=[1,2,3,4,7,8,9,10,14,16]
- time complexity?

Quick sort

- divide-and-conquer (similar to merge sort)
  - more sophisticated split
  - very simple merging

```
QUICKSORT(A, p, r)
if p < r
    then q ← PARTITION(A, p, r)
         QUICKSORT(A, p, q − 1)
         QUICKSORT(A, q + 1, r)
```

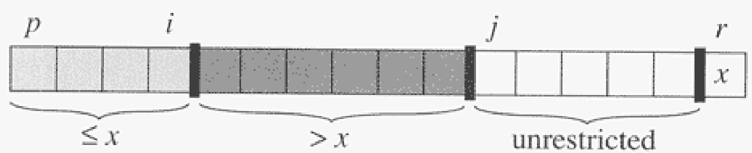
Quick sort

- partition
  - move everything smaller to the left
  - more everything larger to the right
- example: sort A=[1,2,3,4,7,8,9,10,14,16]

Quick sort

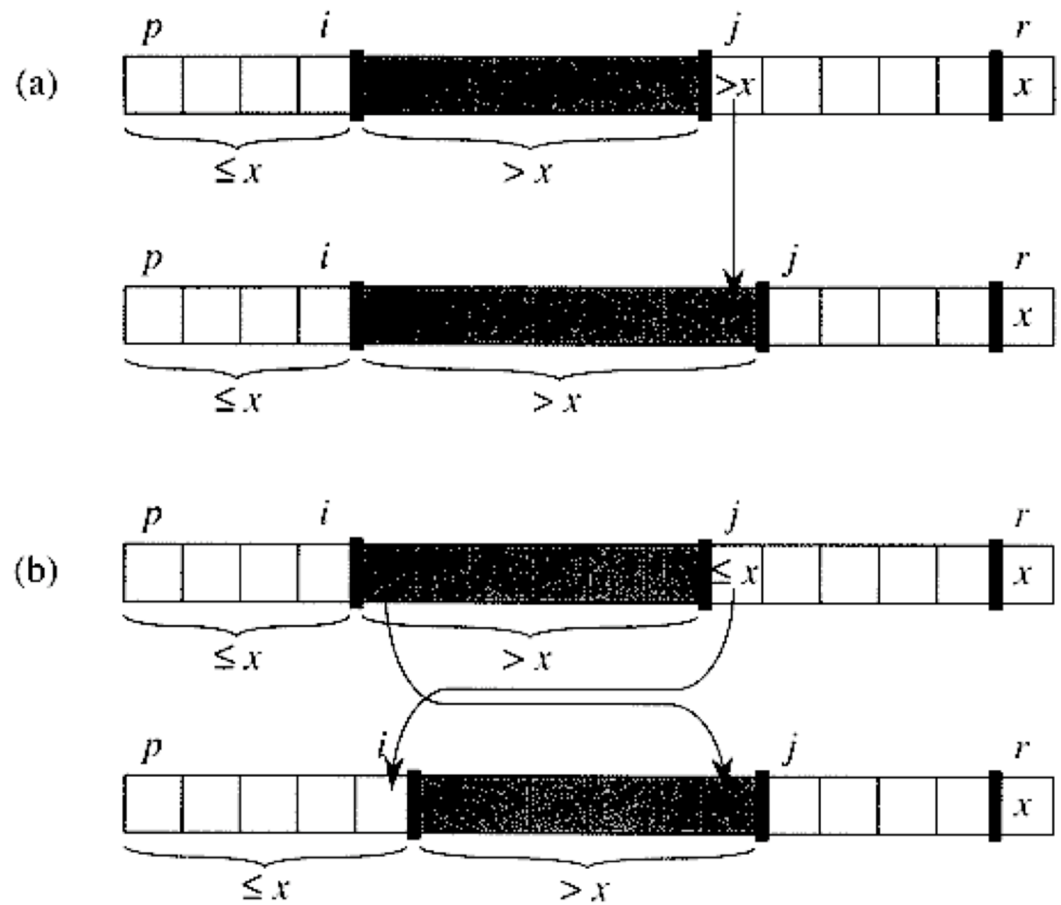
- in-place partition

```
PARTITION(A, p, r)
x ← A[r]
i ← p − 1
for j ← p to r − 1
     do if A[j] ≤ x
           then i ← i + 1
                exchange A[i] ↔ A[j]
exchange A[i + 1] ↔ A[r]
return i + 1
```



Quick sort

- in-place partition



Quick sort

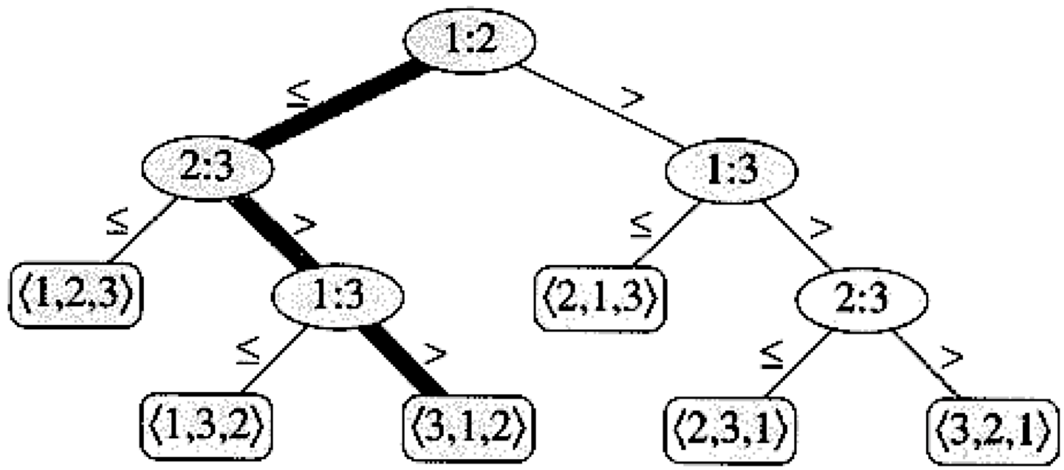
- Time complexity
  - best case
  - worst case

Quick sort

- randomization
  - pick the pivot at random
- what is the consequence of this?

Lower bound

- Lower bound on comparison-based sorting



- There are n! possible permutations

Linear time sorting

- counting sort

```
COUNTING-SORT(A, B, k)
for i ← 0 to k
      do C[i] ← 0
for j ← 1 to length[A]
      do C[A[j]] ← C[A[j]] + 1
▷ C[i] now contains the number of elements equal to i.
for i ← 1 to k
      do C[i] ← C[i] + C[i − 1]
▷ C[i] now contains the number of elements less than or equal to i.
for j ← length[A] downto 1
      do B[C[A[j]]] ← A[j]
         C[A[j]] ← C[A[j]] − 1
```

- time complexity

Counting sort

- step one
  - A=[2,5,6,0,2,3,0,3]
  - C=
- step two
  - C=
- step three
  - C=
  - B=

Radix sort

- sort bit by bit (or digit by digit)

```
RADIX-SORT(A, d)
1  for i ← 1 to d
2      do use a stable sort to sort array A on digit i
```

- example: 329, 457, 657, 839, 436, 720, 355

- time complexity=

Bucket sort

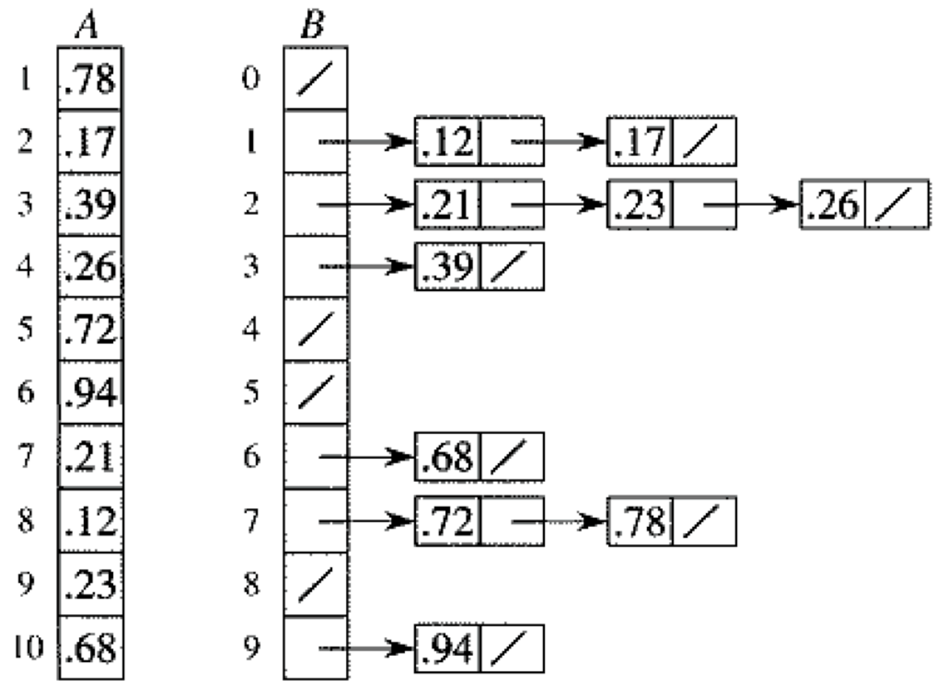
- assume the values in A are between 0 and 1

```
BUCKET-SORT(A)
n ← length[A]
for i ← 1 to n
      do insert A[i] into list B[⌊nA[i]⌋]
for i ← 0 to n − 1
      do sort list B[i] with insertion sort
concatenate the lists B[0], B[1], . . . , B[n − 1] together in order
```

- time complexity
  - expected complexity for each bucket is O(2-1/n)

Bucket sort

- example



Order Statistics

- find max, find min
  - find min
  - find max
- find k-th smallest
- example: 5, 9, 13, 1, 11, 9 ,12, 30, 29, find the 4th smallest value

Order Statistics

- random select

```
RANDOMIZED-SELECT(A, p, r, i)
if p = r
    then return A[p]
q ← RANDOMIZED-PARTITION(A, p, r)
k ← q − p + 1
if i = k            ▷ the pivot value is the answer
    then return A[q]
elseif i < k
    then return RANDOMIZED-SELECT(A, p, q − 1, i)
else return RANDOMIZED-SELECT(A, q + 1, r, i − k)
```

- What is the time complexity?

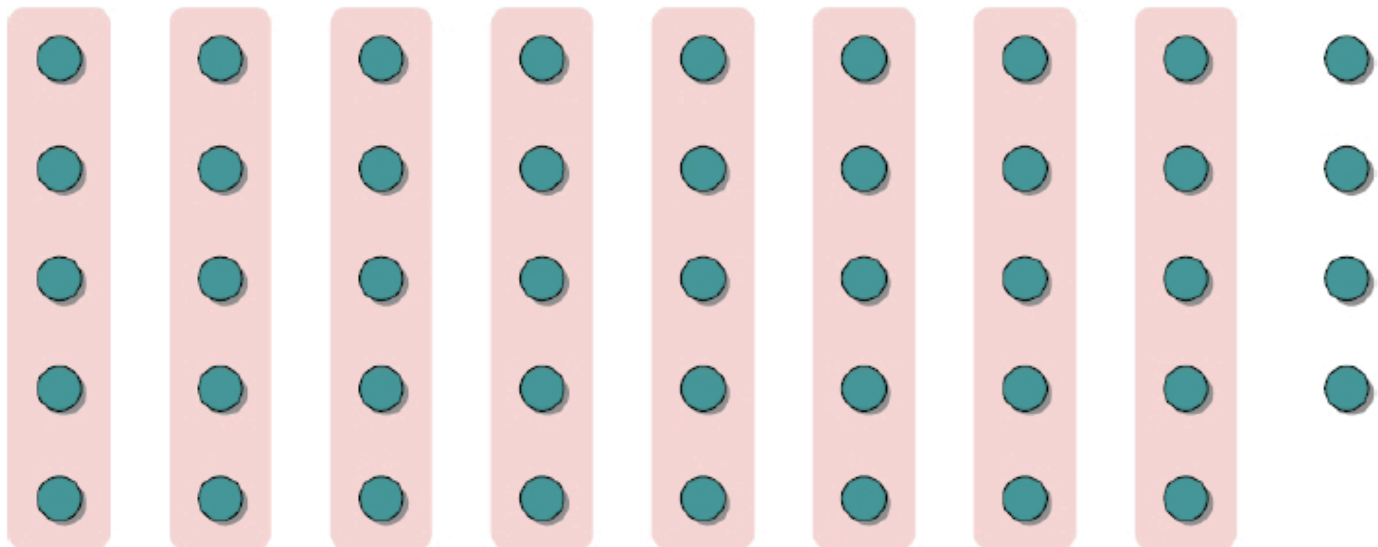
Order Statistics

- deterministic select

Divide S into $\lfloor \frac{|S|}{5} \rfloor$ sequences of size 5
$L \leftarrow \{$ leftover elements$\}$ if any
Sort each 5-sequence
$M \leftarrow$ the medians of the 5-sequences
$m \leftarrow$ SELECT $(\lceil \frac{|M|}{2} \rceil, M)$
$S_1 \leftarrow$ the elements in S that are $< m$
$S_2 \leftarrow$ the elements in S that are $= m$
$S_3 \leftarrow$ the elements in S that are $> m$
if $|S_1| \geq i$ **then** return SELECT (i, S_1)
    elseif $|S_1| + |S_2| \geq i$ **then** return m
    else return SELECT $(i - |S_1| - |S_2|, S_3)$

Order Statistics

- deterministic select



- What is the time complexity?

Order Statistics

Claim:

- At least one-quarter of the elements are $\leq m$
- At least one-quarter of the elements are $\geq m$

Thus,

- at most three-quarters of the elements are $> m$ (i.e. $|S_3| < 3n/4$), and
- at most three-quarters of the elements are $< m$ (i.e. $|S_1| < 3n/4$).

So

$$T(n) \leq \begin{cases} cn & n \leq 49 \\ T(\frac{n}{5}) + T(\frac{3n}{4}) + cn & n \geq 50 \end{cases}$$