# Announcement

## Midterm Exam 2

- Thursday, March 24 in class
- Covered material: Lecture 10 → the class on Tuesday March 22
- Do not forget to prepare your cheat sheet (a single-side letter-size paper)

# Announcement

Programming Assignment #1 has been posted in Blackboard and course website.

# Quicksort

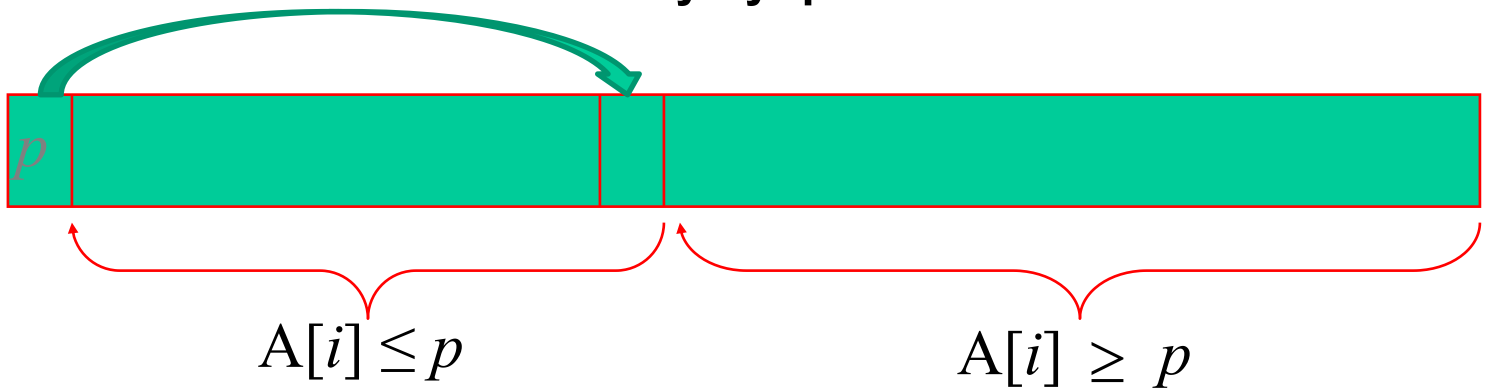
Select a *pivot* (partitioning element)

Rearrange the list so that all the elements in the positions before the pivot are smaller than or equal to the pivot and those after the pivot are larger than or equal to the pivot

Exchange the pivot with the last element in the first (i.e., ≤) sublist– the pivot is now in its final position

Partition into two sublists.

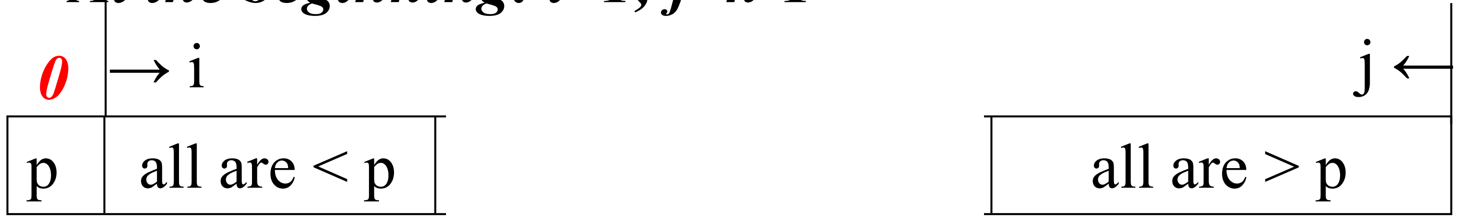
Sort the two sublists individually by quicksort



# Illustrations

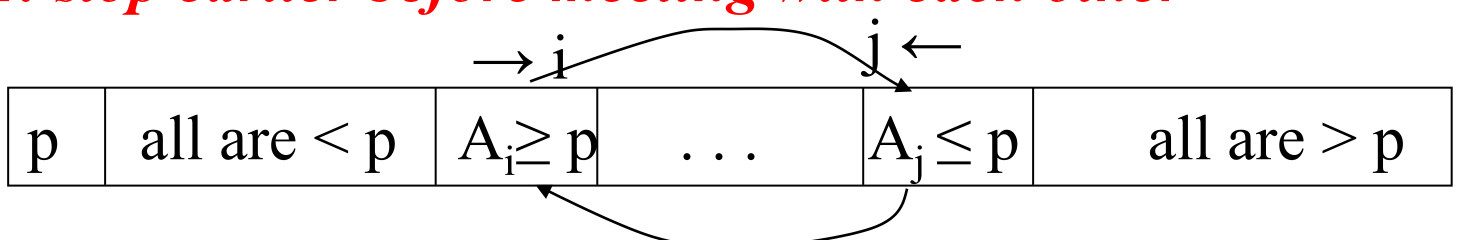
**Search from left to right and right to left simultaneously**

***At the beginning: i=1; j=n-1***




***Stop searching while the conditions violate the requirements***

***Case 1: stop earlier before meeting with each other***

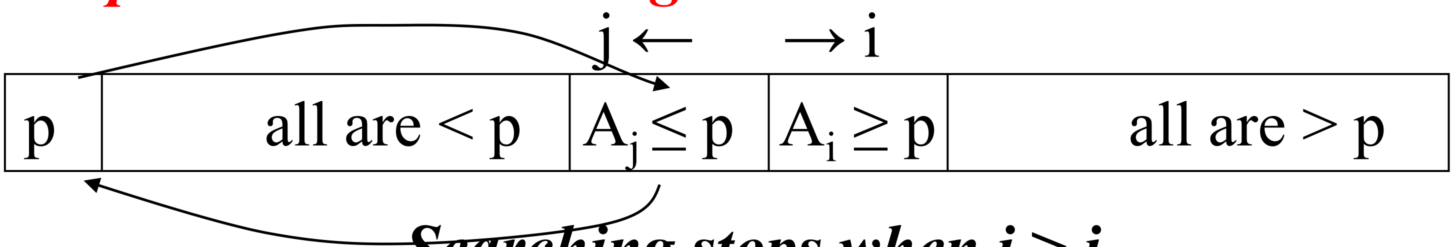



***i < j, Swap A[i] and A[j]***

***After swapping, keep searching***

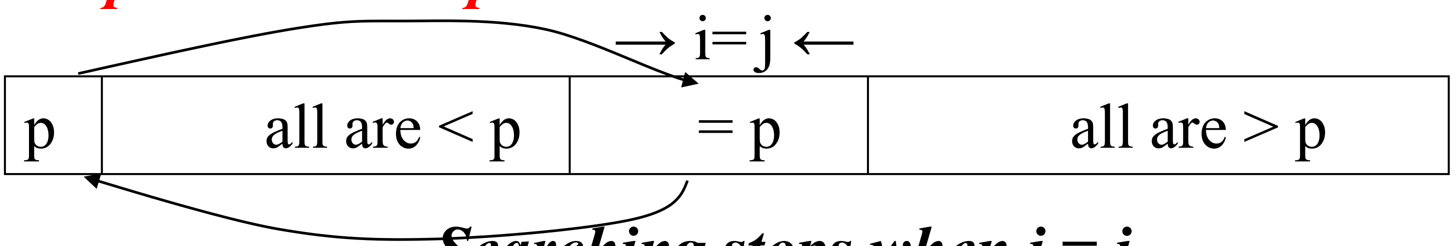
# Illustrations

*Case 2: stop when two searching directions cross*




***Searching stops when i > j***

*Case 3: stop at the same position*




***Searching stops when i = j***

***For both two cases, the pivot position = j***

***Swap** $A[p]$ **and** $A[j]$*

# QuickSort Algorithm

**ALGORITHM** $QuickSort(A[l..r])$

**if** $l < r$

$s \leftarrow Partition(A[l..r])$ // $s$ **is a split position**

$QuickSort\ A[l..s-1]$

$QuickSort\ A[s+1..r]$

# The partition algorithm

**Algorithm** *Partition*$(A[l..r])$
//Partitions a subarray by using its first element as a pivot
//Input: A subarray $A[l..r]$ of $A[0..n-1]$, defined by its left and right
// indices $l$ and $r$ $(l < r)$
//Output: A partition of $A[l..r]$, with the split position returned as
// this function's value
$p \leftarrow A[l]$ ← The leftmost element in the subarray is chosen as the pivot
$i \leftarrow l$; $j \leftarrow r + 1$
**repeat**
    **repeat** $i \leftarrow i + 1$ **until** $A[i] \geq p$ or $i = r$
    **repeat** $j \leftarrow j - 1$ **until** $A[j] \leq p$ or $j = l$
    swap$(A[i], A[j])$
**until** $i \geq j$
swap$(A[i], A[j])$ //undo last swap when $i \geq j$
swap$(A[l], A[j])$
**return** $j$

Do not need frequent memory access

# Quicksort Example

**5 3 1 9 8 2 4 7**

Initialization:
i=1 and j=7

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| | i | | | | | | j |
| 5 | 3 | 1 | 9 | 8 | 2 | 4 | 7 |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |

From left to right, compare:
5 and 3,
5 and 1,
5 and 9

From right to left, compare:
5 and 7,
5 and 4

5 comparisons

# Quicksort Example

**5 3 1 9 8 2 4 7**

First stop

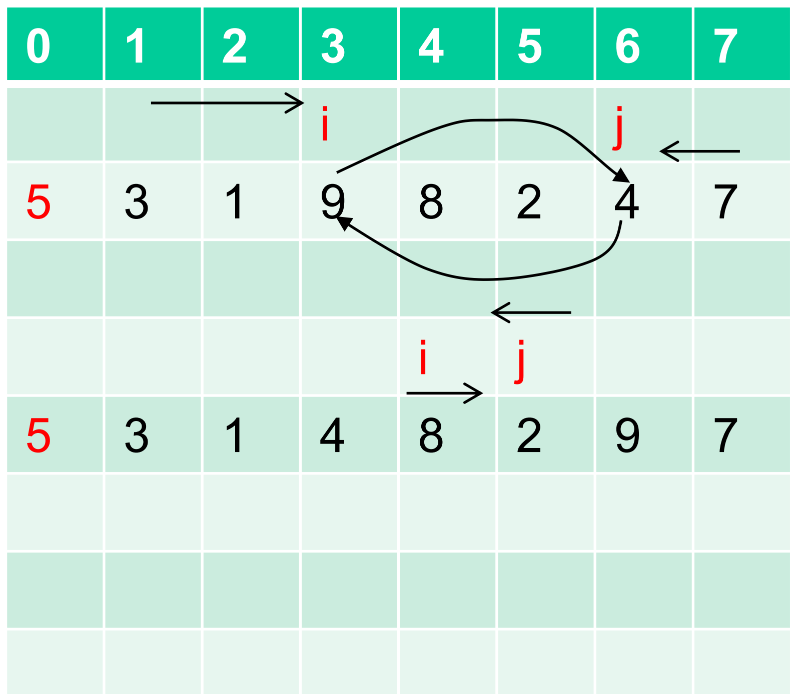
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| | | | → i | | | j ← | |
| 5 | 3 | 1 | 9 | 8 | 2 | 4 | 7 |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |

# Quicksort Example

**5 3 1 9 8 2 4 7**

Swap 4 and 9

**Keep working:**
From left to right,
compare:
5 and 8




From right to left,
compare:
5 and 2

2 comparisons

# Quicksort Example

**5 3 1 9 8 2 4 7**

**Second stop --**
**Swap 8 and 2**

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| | | | | i | j | | |
| 5 | 3 | 1 | 9 | 8 | 2 | 4 | 7 |
| | | | | | | | |
| | | | | j | i | | |
| 5 | 3 | 1 | 4 | 2 | 8 | 9 | 7 |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |

**Keep working:**
From left to right, compare:
5 and 8

From right to left, compare:
5 and 2

2 comparisons

# Quicksort Example

5 3 1 9 8 2 4 7

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| | | | | j | i | | |
| 5 | 3 | 1 | 4 | 2 | 8 | 9 | 7 |
| | | | | | | | |

i >= j ⇨ ***Pivot position s=4***

9 comparisons

| l=0, r=7 |
|---|
| S=4 |

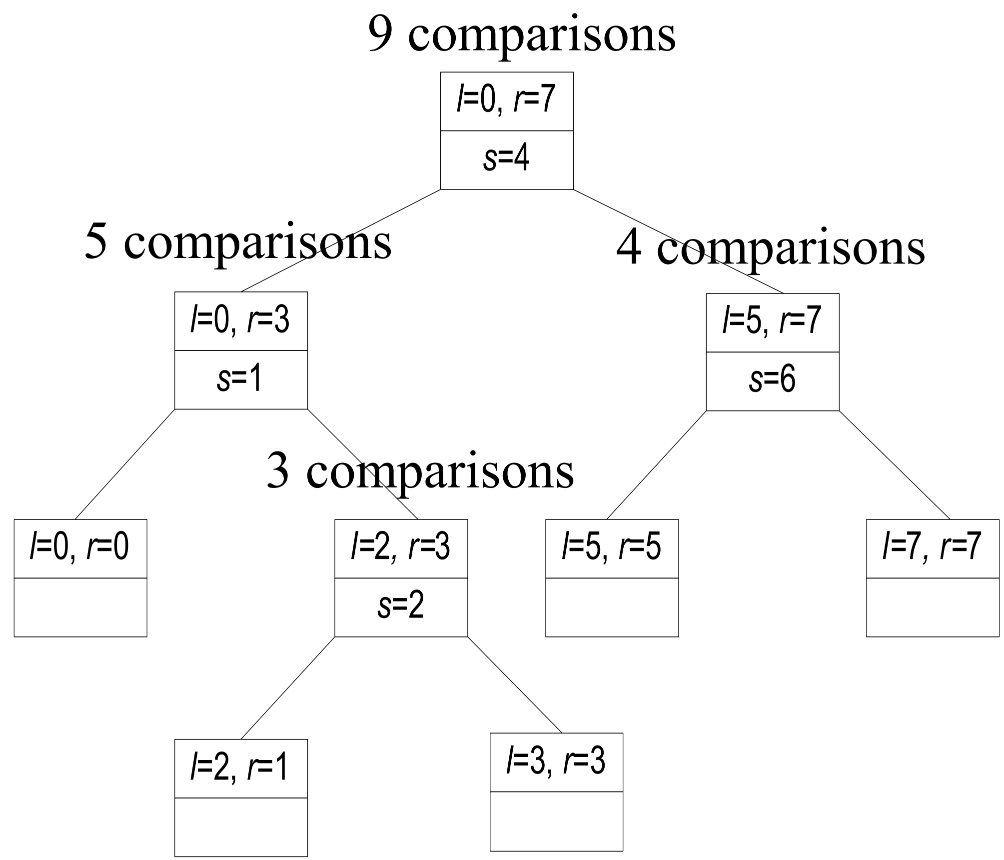
# Quicksort Example

**5 3 1 9 8 2 4 7**

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| | → | → | i | | | j | ← |
| 5 | 3 | 1 | 9 | 8 | 2 | 4 | 7 |
| | | → | → | i | j | ← | |
| 5 | 3 | 1 | 4 | 8 | 2 | 9 | 7 |
| | | | | j | i | | |
| 5 | 3 | 1 | 4 | 2 | 8 | 9 | 7 |
| 2 | 3 | 1 | 4 | 5 | 8 | 9 | 7 |

***Pivot position s=4***

**Perform quicksort on these two new arrays separately**




9+5+4+3=21 comparisons

# More Examples of Quicksort

23, 53, 2, 78, 12, 54, 1, 8

12, 31, 11, 55, 12, 79, 81, 2

# Efficiency of Quicksort

***Basic operation:*** key comparison

***Best case***: split in the middle — $\Theta(n \log n)$

$$C_{best} = 2C_{best}(\lfloor n/2 \rfloor) + f(n) \qquad \textbf{\textit{for}} \quad n > 1, C_{best}(1) = 0$$

$$f(n) = \begin{cases} n+1 & i \neq j \\ n & i = j \end{cases}$$ ⇨ So you don't need to count

***Master Theorem: a=2,b=2,k=1***

$$C_{best} \in \Theta(n \log n)$$

# Efficiency of Quicksort

**Worst case**: sorted array! — $\Theta(n^2)$

$$C_{worst}(n) = C_{worst}(n-1) + n + 1$$

$j \leftarrow \quad \rightarrow i$

| | | | | |
|---|---|---|---|---|
| A[0] | A[1] | … | A[n-2] | A[n-1] |

**Average case**: random arrays — $\Theta(n \log n)$

***Assumption: the partition can happen in any position*** $0 \le p \le n-1$ ***with an equal probability***

$$C_{avg}(0) = 0, C_{avg}(1) = 0$$

$$C_{avg}(n) = \sum_{p=0}^{n-1} \left\{ \underbrace{\frac{1}{n}}_{\text{probability}} * \left[ (n+1) + \overbrace{C_{avg}(p)}^{\text{First subarray}} + \underbrace{C_{avg}(n-1-p)}_{\text{second subarray}} \right] \right\} \approx 2n \ln n$$

## Improvements of Quicksort

- **Better pivot selection: median-of-three partitioning avoids worst case in sorted files**

- **Quicksort is effective for large array**
  - Switch to insertion sort on small subarrays

**Possible issue: Not stable!**

- Stability: the relative order or records with equal search keys is not changed during sorting

# Mergesort vs. Quicksort

| | Mergesort | Quicksort |
|---|---|---|
| Basic operation | key comparison | key comparison |
| Best case | O(nlogn) | O(nlogn) |
| Average case | O(nlogn) | O(nlogn) |
| Worst case | O(nlogn) | $O(n^2)$ |
| Stable | yes | no |

ALGORITHM *Merge*($B[0..p-1], C[0..q-1], A[0..p+q-1]$)
…
if $B[i] \le C[j]$
$A[k] \leftarrow B[i]; i \leftarrow i+1$
else $A[k] \leftarrow C[j]; j \leftarrow j+1$
…

**Algorithm Partition**

repeat
repeat $i \leftarrow i+1$ until $A[i] \ge p$
repeat $j \leftarrow j-1$ until $A[j] \le p$
swap($A[i], A[j]$)
until $i \ge j$

***Inner loop procedure***