Design and Analysis of Algorithms
Lecture-5:
Quicksort & Linear Time Sorting

Prof. Eugene Chang

Overview

- Quicksort
  - Concept
  - Time Complexity Analysis
- More about sorting
  - Theoretical lower-bound
  - Linear-time sorting algorithms
  - Stability of sorting

- Part of the slides are based on material from Prof. Jianhua Ruan, The University of Texas at San Antonio

Quick sort

- Another divide and conquer sorting algorithm – like merge sort
- Anyone remember the basic idea?
- The worst-case and average-case running time?
- Learn some new algorithm analysis tricks

Quicksort

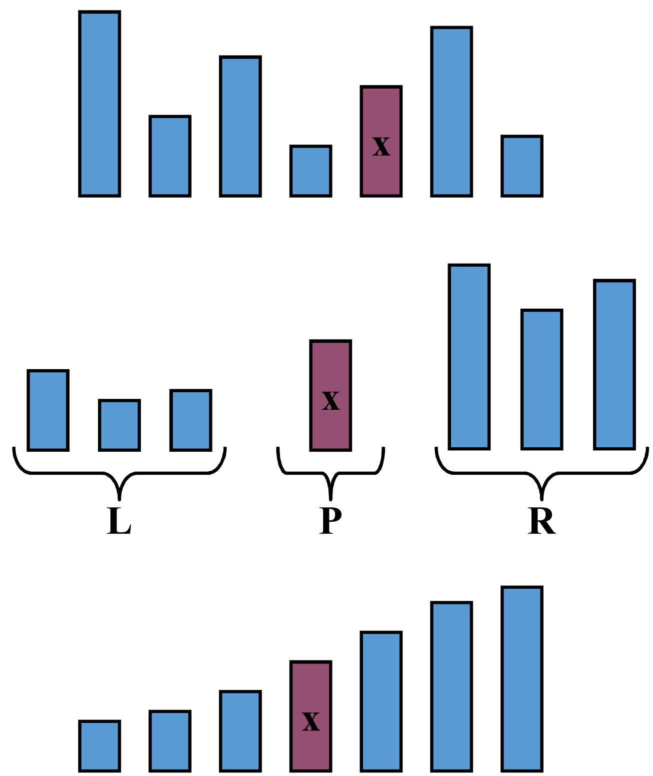
- Main idea:
  - Find a Pivot element
  - Split array into elements less than pivot, equal to pivot, and greater than pivot, called partitioning
  - Recursively sort the pieces

Divide and Conquer

1. Pick a pivot element

2. Put everything <pivot on the left and everything > pivot on right.

3. Recursively Sort the left and right



Quick sort

Quicksort an n-element array:

1. ***Divide:*** Partition the array into two subarrays around a ***pivot*** x such that elements in lower subarray $\leq x \leq$ elements in upper subarray.

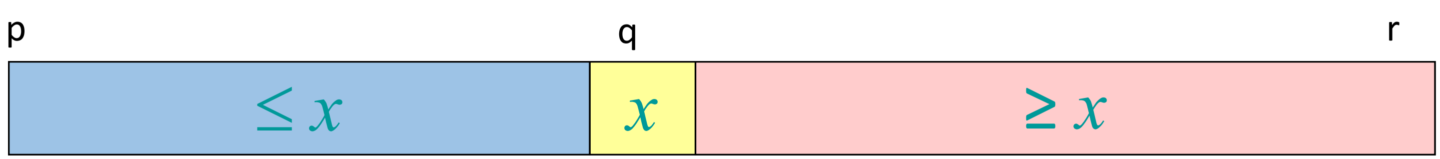
$\leq x$	x	$\geq x$

2. ***Conquer:*** Recursively sort the two subarrays.
3. ***Combine:*** Trivial.

Key: *Linear-time partitioning subroutine.*

Partition

- All the action takes place in the **`partition()`** function
  - Rearranges the subarray in place
  - End result: two subarrays
    - All values in first subarray $\leq$ all values in second
  - Returns the index of the "pivot" element separating the two subarrays



Pseudocode for quicksort

QUICKSORT(A, p, r)
  if $p < r$
    then $q \leftarrow$ PARTITION(A, p, r)
      QUICKSORT$(A, p, q-1)$
      QUICKSORT$(A, q+1, r)$

Initial call: QUICKSORT$(A, 1, n)$

Idea of partition

- If we are allowed to use a second array, it would be easy

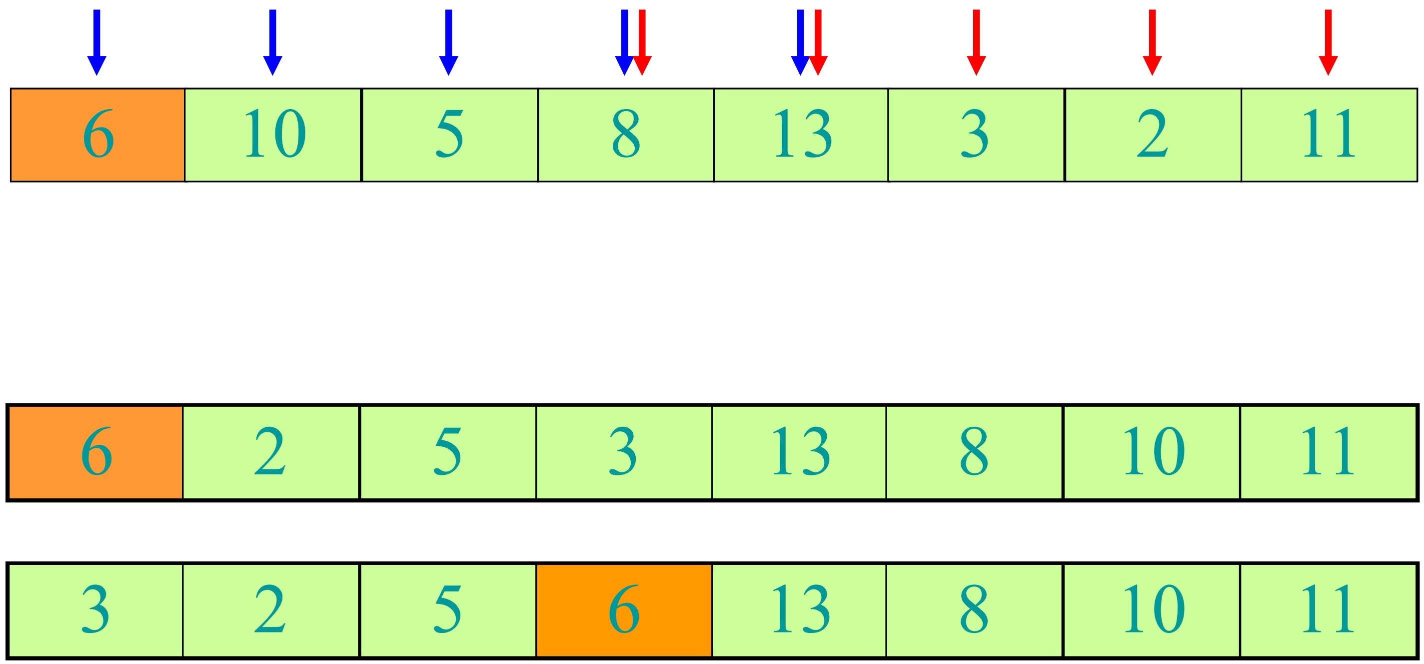
6	10	5	8	13	3	2	11

6	5	3	2	11	13	8	10

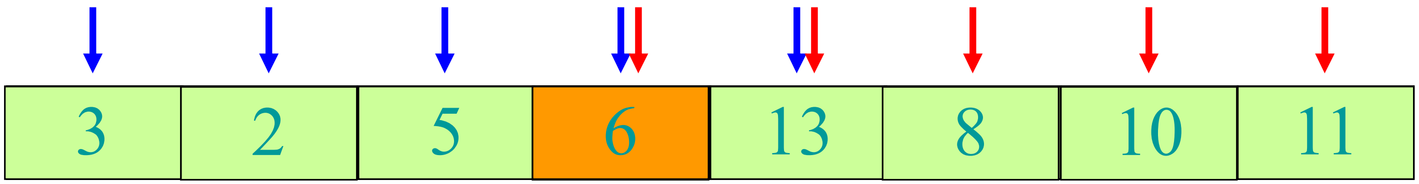
2	5	3	6	11	13	8	10

Another idea

- Keep two iterators: one from head, one from tail



In-place Partition



Partition In Words

- Partition(A, p, r):
  - Select an element to act as the "pivot" (*which?*)
  - Grow two regions, A[p..i] and A[j..r]
    - All elements in A[p..i] <= pivot
    - All elements in A[j..r] >= pivot
  - Increment i until A[i] > pivot
  - Decrement j until A[j] < pivot
  - Swap A[i] and A[j]
  - Repeat until i >= j
  - Swap A[j] and A[p]
  - Return j

Note: different from book's **`partition()`**, **`which`** uses two iterators that both move forward.

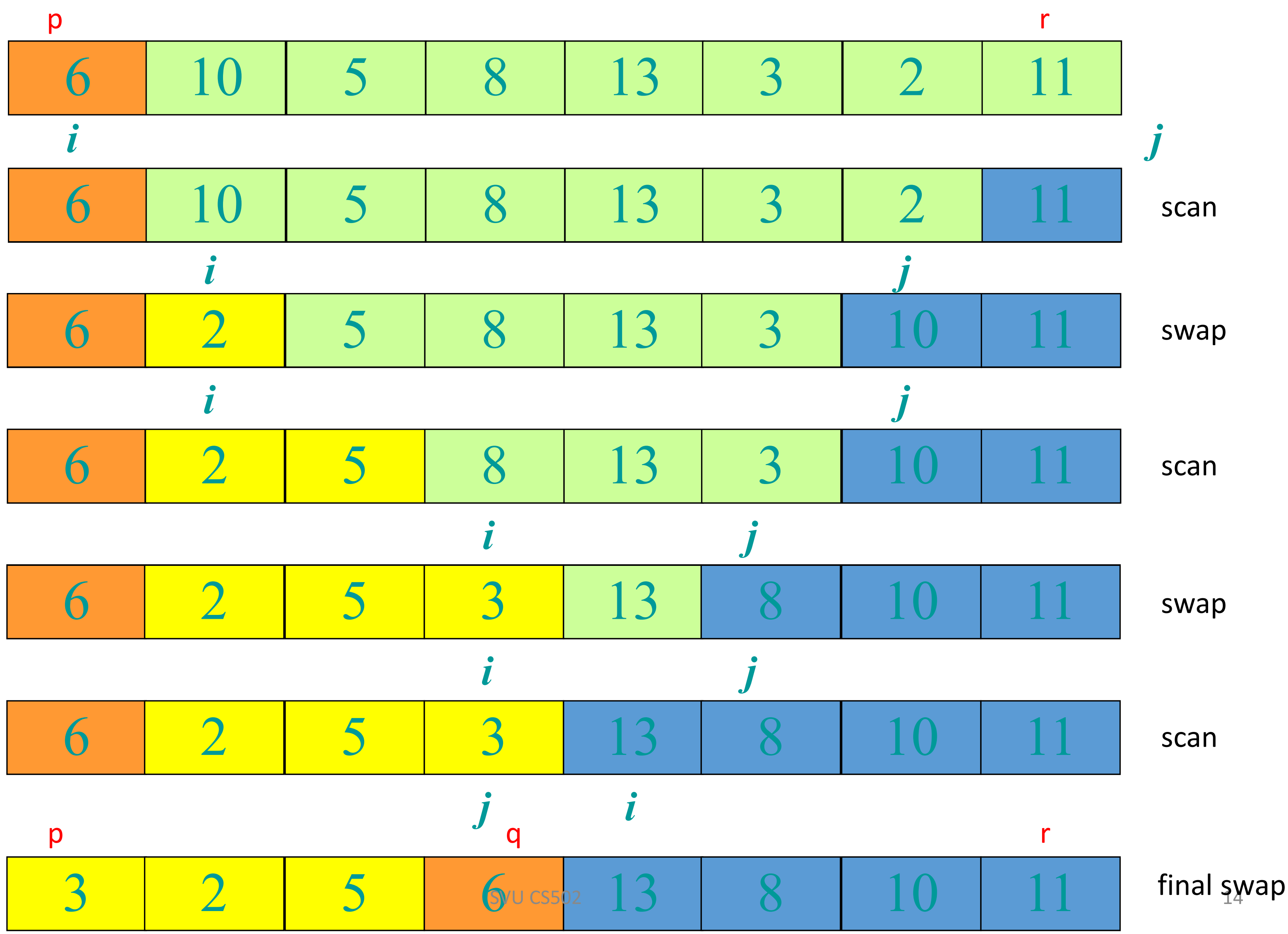
Partition Code

```
Partition(A, p, r)
    x = A[p];        // pivot is the first element
    i = p;
    j = r + 1;
    while (TRUE) {
        repeat
            i++;
        until A[i] > x or i >= j;
        repeat
            j--;
        until A[j] < x or j < i;
        if (i < j)
            Swap (A[i], A[j]);
        else
            break;
    }
    swap (A[p], A[j]);
    return j;
```

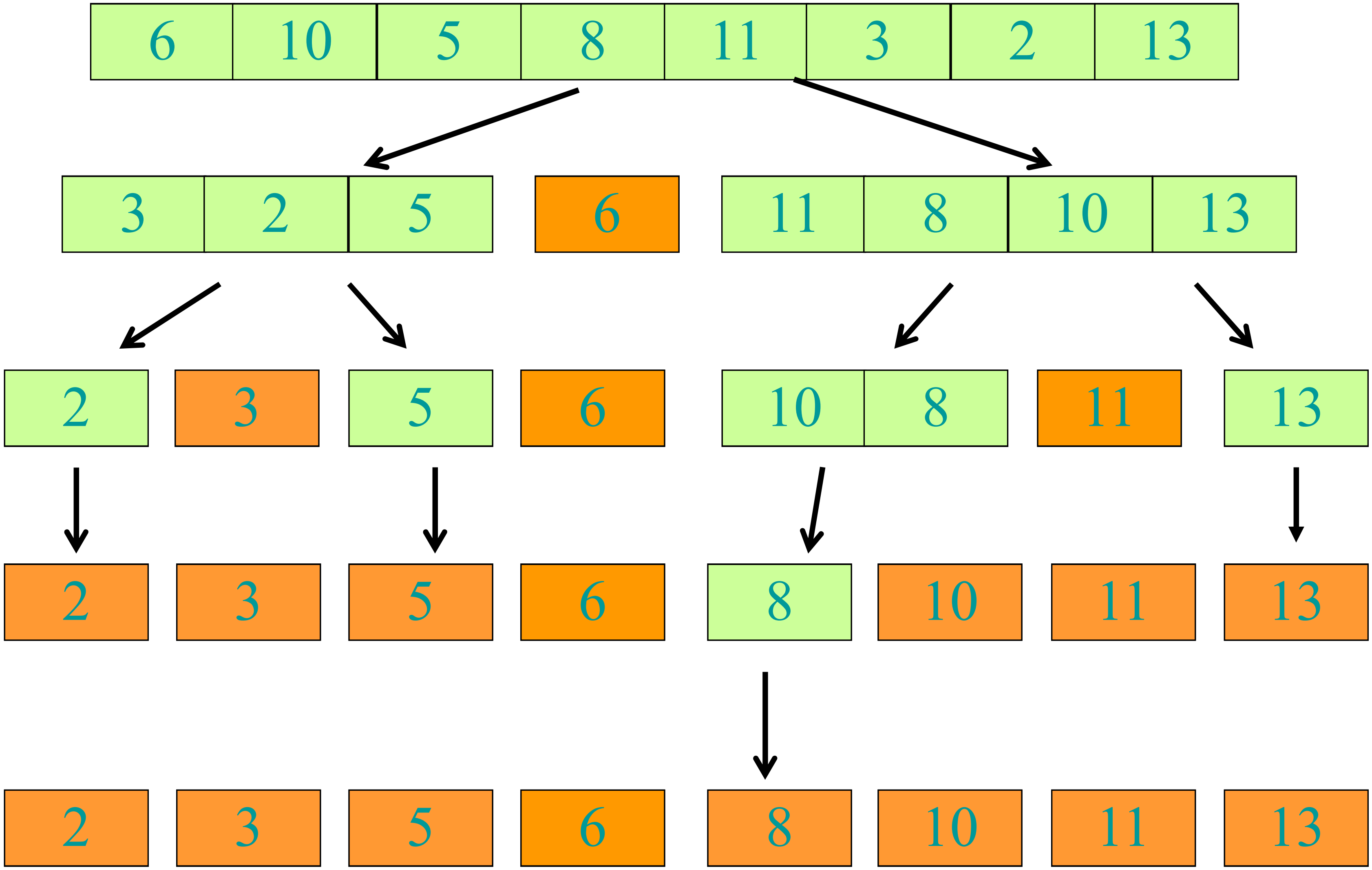
What is the running time of **`partition()`**?

`partition()` *runs in* $\Theta(n)$ *time* x = 6

Partition example



Quick sort
example



Analysis of quicksort

- Assume all input elements are distinct.
- In practice, there are better partitioning algorithms for when duplicate input elements may exist.
- Let $T(n) =$ worst-case running time on an array of n elements.

Worst-case of quicksort

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

- Input sorted or reverse sorted.
- Partition around min or max element.
- One side of partition always has no elements.

$$\begin{aligned} T(n) &= T(0) + T(n-1) + \Theta(n) \\ &= \Theta(1) + T(n-1) + \Theta(n) \\ &= T(n-1) + \Theta(n) \\ &= \Theta(n^2) \qquad \textbf{\textit{(arithmetic series)}} \end{aligned}$$

Worst-case recursion tree

$$T(n) = T(0) + T(n-1) + n$$

Worst-case recursion tree

$$T(n) = T(0) + T(n–1) + n$$

$$T(n)$$

Worst-case recursion tree

$$T(n) = T(0) + T(n-1) + n$$

$$n$$

$$T(0) \qquad T(n-1)$$

Worst-case recursion tree

$$T(n) = T(0) + T(n-1) + n$$

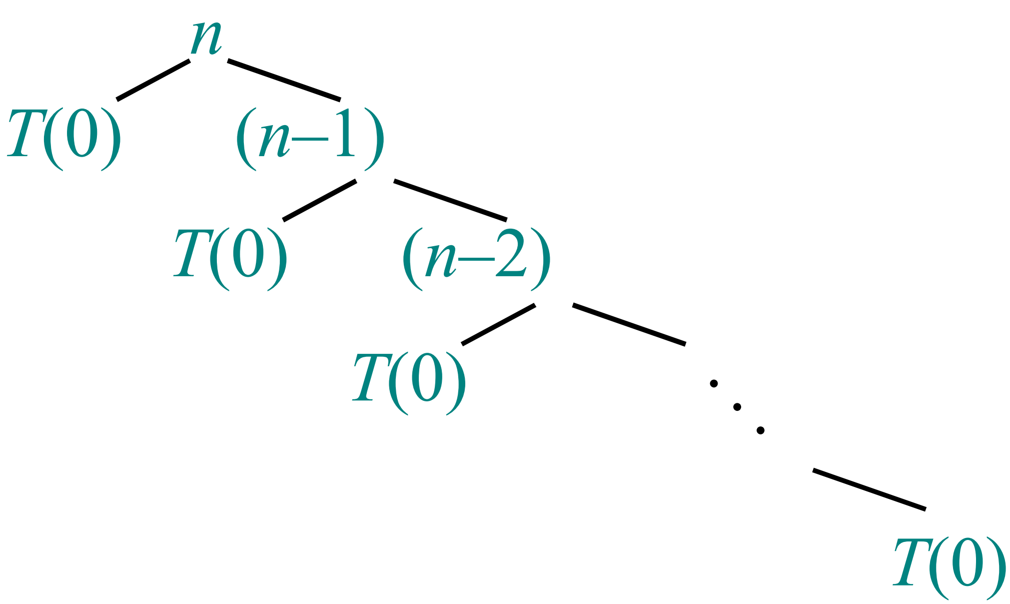
$$n$$

$$T(0) \qquad (n-1)$$

$$T(0) \qquad T(n-2)$$

Worst-case recursion tree

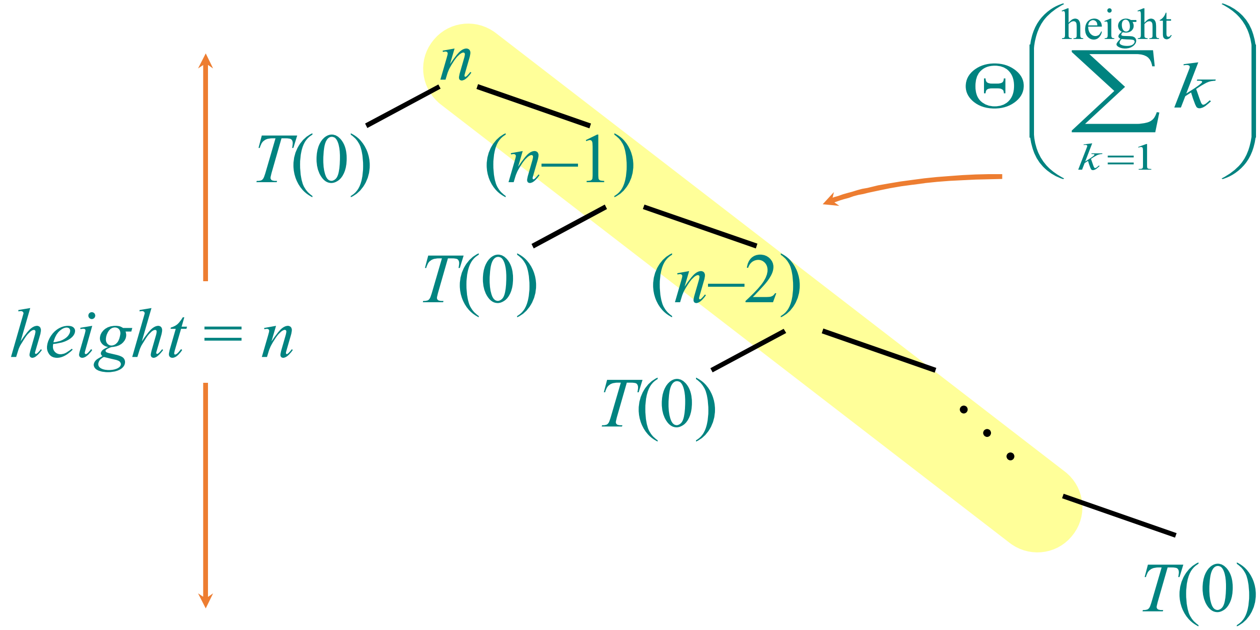
$$T(n) = T(0) + T(n-1) + n$$



Worst-case recursion tree

$$T(n) = T(0) + T(n-1) + n$$

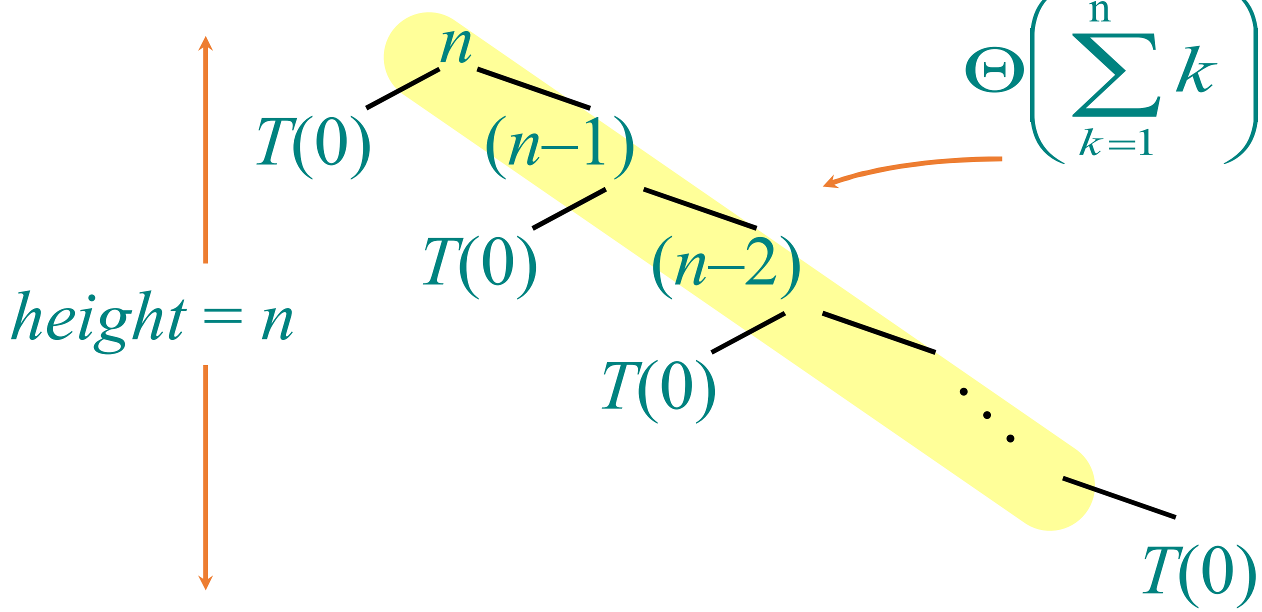
$$\Theta\left(\sum_{k=1}^{\text{height}} k\right) = \Theta\left(n^2\right)$$



Worst-case recursion tree

$$T(n) = T(0) + T(n-1) + n$$

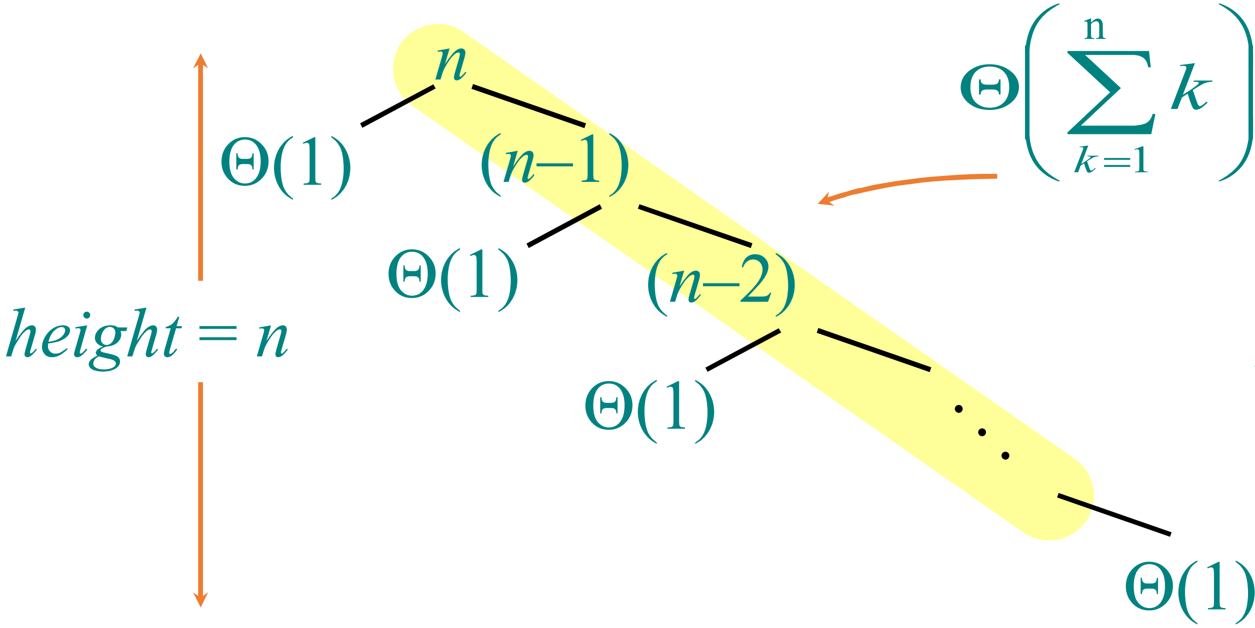
$$\Theta\left(\sum_{k=1}^{n} k\right) = \Theta\left(n^2\right)$$



Worst-case recursion tree

$$T(n) = T(0) + T(n-1) + n$$

$$\Theta\left(\sum_{k=1}^{n} k\right) = \Theta\left(n^2\right)$$



$$\begin{aligned} T(n) &= \Theta(n) + \Theta(n^2) \\ &= \Theta(n^2) \end{aligned}$$

Best-case analysis

(For intuition only!)

If we're lucky, PARTITION splits the array evenly:

$$\begin{aligned} T(n) &= 2T(n/2) + \Theta(n) \\ &= \Theta(n \log n) \end{aligned}$$

(same as merge sort)

What if the split is always $\frac{1}{10} : \frac{9}{10}$?

$$T(n) = T\left(\tfrac{1}{10}n\right) + T\left(\tfrac{9}{10}n\right) + \Theta(n)$$

What is the solution to this recurrence?

Analysis of “almost-best” case

$$T(n)$$

Analysis of “almost-best” case

$$n$$

$$T\left(\tfrac{1}{10}n\right) \qquad\qquad T\left(\tfrac{9}{10}n\right)$$

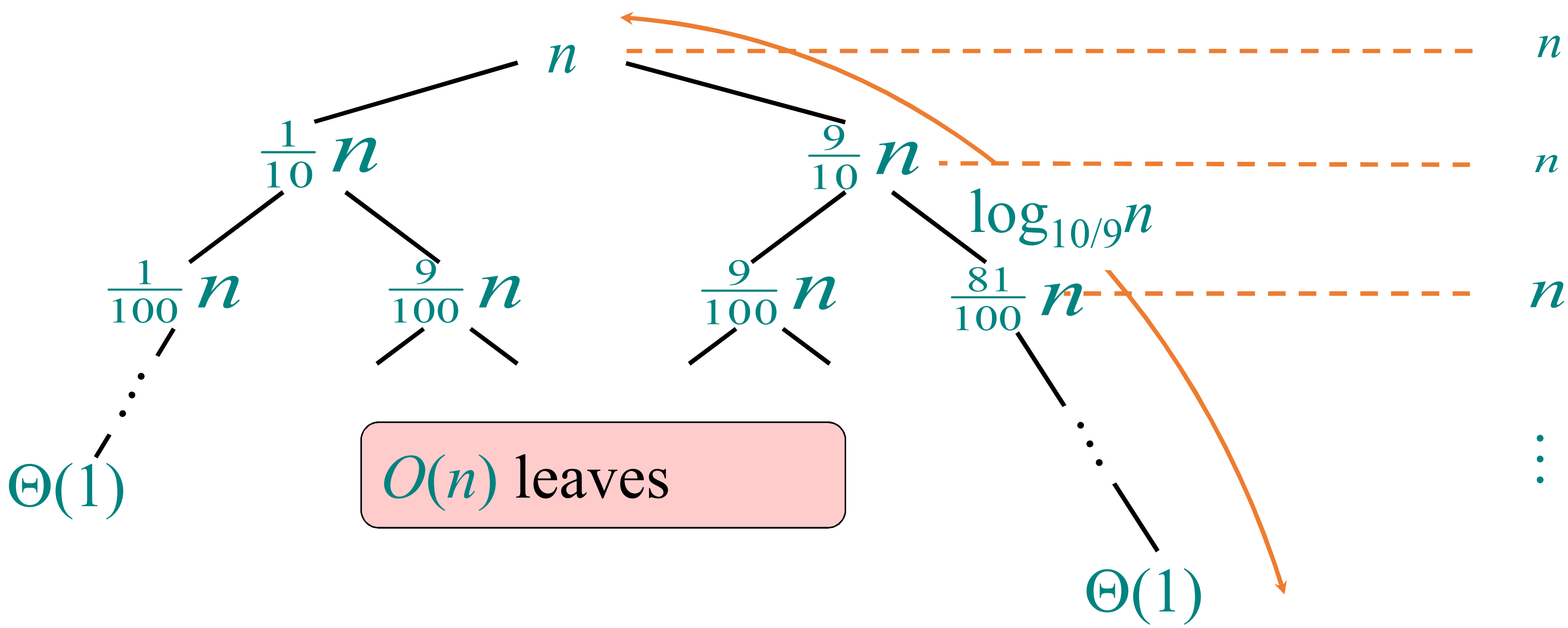
Analysis of “almost-best” case

$$n$$

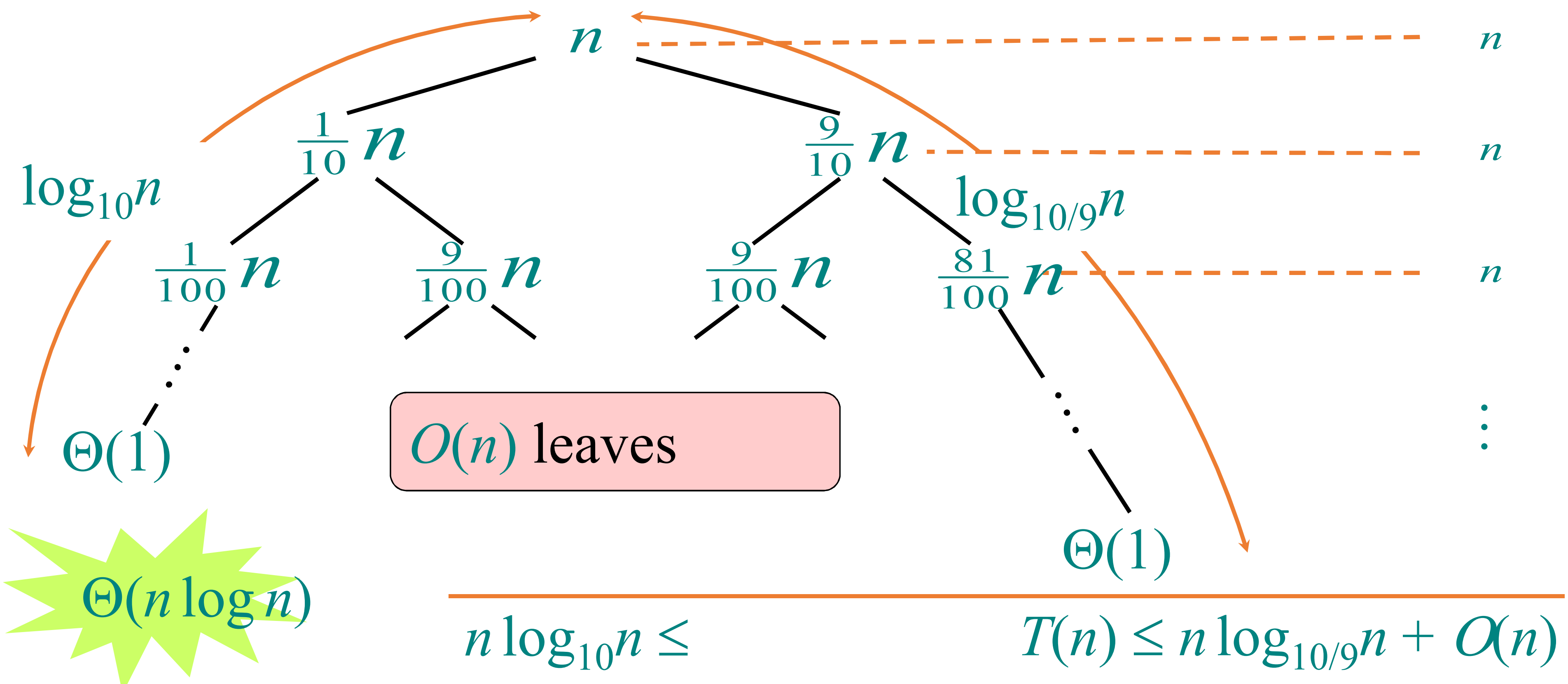
$$\frac{1}{10}n \qquad\qquad \frac{9}{10}n$$

$$T\left(\frac{1}{100}n\right)\; T\left(\frac{9}{100}n\right) \qquad T\left(\frac{9}{100}n\right)\; T\left(\frac{81}{100}n\right)$$

Analysis of “almost-best” case



Analysis of “almost-best” case



Quicksort Runtimes

- Best-case runtime $T_{best}(n) \in \Theta(n \log n)$
- Worst-case runtime $T_{worst}(n) \in \Theta(n^2)$

- Worse than mergesort? Why is it called quicksort then?
- Its average runtime $T_{avg}(n) \in \Theta(n \log n)$
- Better even, the expected runtime of **randomized quicksort** is $\Theta(n \log n)$

Randomized quicksort

- Randomly choose an element as pivot
  - Every time need to do a partition, throw a die to decide which element to use as the pivot
  - Each element has 1/n probability to be selected

```
Rand-Partition(A, p, r)
    d = random();    // a random number between 0 and 1
    index = p + floor((r-p+1) * d);  // p<=index<=r
    swap(A[p], A[index]);
    Partition(A, p, r);  // now do partition using A[p] as pivot
```

Running time of randomized quicksort

$$T(n) = \begin{cases} T(0) + T(n-1) + dn & \text{if } 0 : n-1 \text{ split,} \\ T(1) + T(n-2) + dn & \text{if } 1 : n-2 \text{ split,} \\ \quad\vdots & \\ T(n-1) + T(0) + dn & \text{if } n-1 : 0 \text{ split,} \end{cases}$$

- The expected running time is an average of all cases

Expectation

$$\overline{T}(n) = \frac{1}{n}\sum_{k=0}^{n-1}\left(\overline{T}(k) + \overline{T}(n-k-1)\right) + n$$

$$\overline{T}(n) = \frac{1}{n}\sum_{k=0}^{n-1}\left(\overline{T}(k) + \overline{T}(n-k-1)\right) + n$$

$$= \frac{2}{n}\sum_{k=0}^{n-1}\overline{T}(k) + n$$

Solving recurrence

1. Recursion tree (iteration) method
   - Good for guessing an answer
2. Substitution method
   - Generic method, rigid, but may be hard
3. Master method
   - Easy to learn, useful in limited cases only
   - Some tricks may help in other cases

Substitution method

The most general method to solve a recurrence (prove O and Ω separately):

1. ***Guess*** the form of the solution:
   (e.g. using recursion trees, or expansion)
2. ***Verify*** by induction (inductive step).

Expected running time of Quicksort

$$\overline{T}(n) = \frac{2}{n}\sum_{k=0}^{n-1}\overline{T}(k) + n$$

- Guess $\overline{T}(n) = O(n \log n)$
- We need to show that $\overline{T}(n) \le cn \log n$ for some c and sufficiently large n
- Use $T(n)$ instead of $\overline{T}(n)$ for convenience

- Fact: $$T(n) = \frac{2}{n}\sum_{k=0}^{n-1} T(k) + n$$
- Need to show: *T(n) ≤ c n log (n)*
- Assume: *T(k) ≤ ck log (k) for 0 ≤ k ≤ n-1*
- Proof:

$$\begin{aligned}
T(n) &= \frac{2}{n}\sum_{k=0}^{n-1} T(k) + n \\
&\le \frac{2c}{n}\sum_{k=0}^{n-1} k \log k + n \\
&\le \frac{2c}{n}\left(\frac{n^2}{2}\log n - \frac{n^2}{8}\right) + n
\end{aligned}$$

using the fact that $\sum_{k=0}^{n-1} k \log k \le \frac{n^2}{2}\log n - \frac{n^2}{8}$

$$\begin{aligned}
&\le cn \log n - \frac{cn}{4} + n \\
&\le cn \log n
\end{aligned}$$

if c ≥ 4. Therefore, by defintion, T(n) = Θ (nlogn)

Tightly Bounding
The Key Summation

$$\sum_{k=0}^{n-1} k \lg k = \sum_{k=1}^{n-1} k \lg k$$

$$\lim_{x \to 0} x \lg x = 0$$

$$= \sum_{k=1}^{\lceil n/2 \rceil - 1} k \lg k + \sum_{k=\lceil n/2 \rceil}^{n-1} k \lg k$$

Split the summation for a tighter bound

$$\leq \sum_{k=1}^{\lceil n/2 \rceil - 1} k \lg k + \sum_{k=\lceil n/2 \rceil}^{n-1} k \lg n$$

***The* $\lg k$ *in the second term is bounded by* $\lg n$**

$$= \sum_{k=1}^{\lceil n/2 \rceil - 1} k \lg k + \lg n \sum_{k=\lceil n/2 \rceil}^{n-1} k$$

Move the* $\lg n$ *outside the summation

Tightly Bounding
The Key Summation

$$\sum_{k=1}^{n-1} k \lg k \le \sum_{k=1}^{\lceil n/2 \rceil - 1} k \lg k + \lg n \sum_{k=\lceil n/2 \rceil}^{n-1} k$$

The summation bound so far

$$\le \sum_{k=1}^{\lceil n/2 \rceil - 1} k \lg (n/2) + \lg n \sum_{k=\lceil n/2 \rceil}^{n-1} k$$

The* lg *k in the first term is bounded by* lg *n/2

$$= \sum_{k=1}^{\lceil n/2 \rceil - 1} k (\lg n - 1) + \lg n \sum_{k=\lceil n/2 \rceil}^{n-1} k$$

lg *n/2* = lg *n* - 1

$$= (\lg n - 1) \sum_{k=1}^{\lceil n/2 \rceil - 1} k + \lg n \sum_{k=\lceil n/2 \rceil}^{n-1} k$$

Move (*lg *n* - 1*) outside the summation

Tightly Bounding
The Key Summation

$$\sum_{k=1}^{n-1} k \lg k \le (\lg n - 1) \sum_{k=1}^{\lceil n/2 \rceil - 1} k + \lg n \sum_{k=\lceil n/2 \rceil}^{n-1} k$$

The summation bound so far

$$= \lg n \sum_{k=1}^{\lceil n/2 \rceil - 1} k - \sum_{k=1}^{\lceil n/2 \rceil - 1} k + \lg n \sum_{k=\lceil n/2 \rceil}^{n-1} k$$

Distribute the (lg n - 1)

$$= \lg n \sum_{k=1}^{n-1} k - \sum_{k=1}^{\lceil n/2 \rceil - 1} k$$

The summations overlap in range; combine them

$$= \lg n \left(\frac{(n-1)(n)}{2} \right) - \sum_{k=1}^{\lceil n/2 \rceil - 1} k$$

The Guassian series

Tightly Bounding
The Key Summation

$$\sum_{k=1}^{n-1} k \lg k \le \left(\frac{(n-1)(n)}{2} \right) \lg n - \sum_{k=1}^{\lceil n/2 \rceil - 1} k$$

The summation bound so far

$$\le \frac{1}{2}\left[n(n-1) \right] \lg n - \sum_{k=1}^{n/2-1} k$$

Rearrange first term, place upper bound on second

$$\le \frac{1}{2}\left[n(n-1) \right] \lg n - \frac{1}{2}\left(\frac{n}{2} \right)\left(\frac{n}{2} - 1 \right)$$

Guassian series

$$\le \frac{1}{2}\left(n^2 \lg n - n \lg n \right) - \frac{1}{8} n^2 + \frac{n}{4}$$

Multiply it all out

Tightly Bounding
The Key Summation

$$\sum_{k=1}^{n-1} k \lg k \le \frac{1}{2}\left(n^2 \lg n - n \lg n\right) - \frac{1}{8}n^2 + \frac{n}{4}$$

$$= \frac{1}{2}n^2 \lg n - \frac{1}{8}n^2 + \left(\frac{n}{4} - \frac{n}{2}\lg n\right)$$

$$\le \frac{1}{2}n^2 \lg n - \frac{1}{8}n^2 \text{ when } \quad n \ge 2$$

Done! !!

Comparison

	Time complexity	Stable?	In-place?
Merge sort			
Quick sort			
Heap sort			

Comparison

	Time complexity	Stable?	In-place?
Merge sort	Θ (n log n)	Yes	No
Quick sort	Θ(n log n) expected. Θ(n^2) worst case	No	Yes
Heap sort	Θ (n log n)	No	Yes

More about sorting

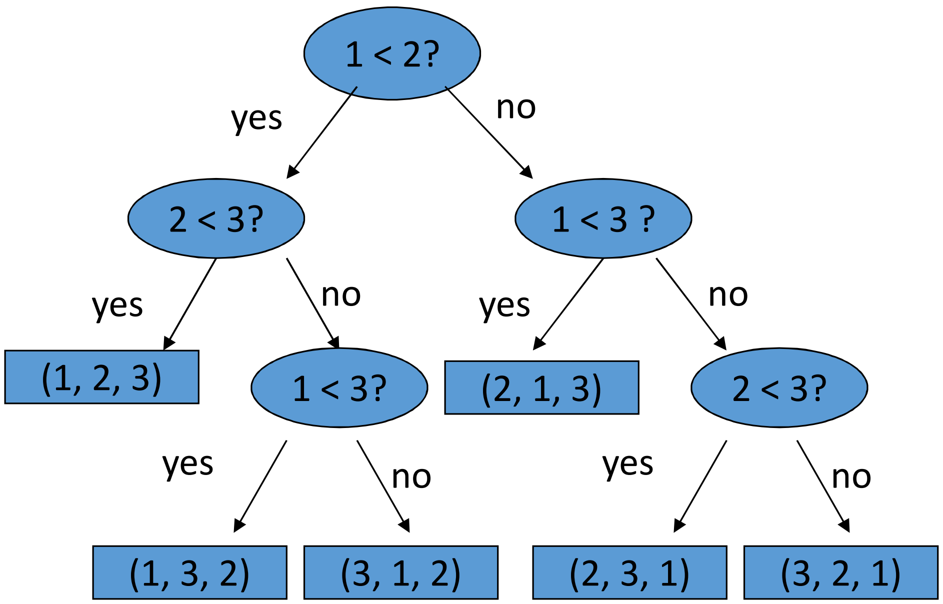
- How many sorting algorithms do you know?
- What are their time complexity?
- What’s common about them?
- Can we do better than $\Theta(n \log n)$?
- Yes and no

Theoretical lower-bound

- Comparison sort: determines the relative order of two elements only by comparison
  - What else can you do …
  - Text book Ch8.1 shows that the theoretical lower-bound for any comparison-based sorting algorithm is $\Theta(n \log n)$

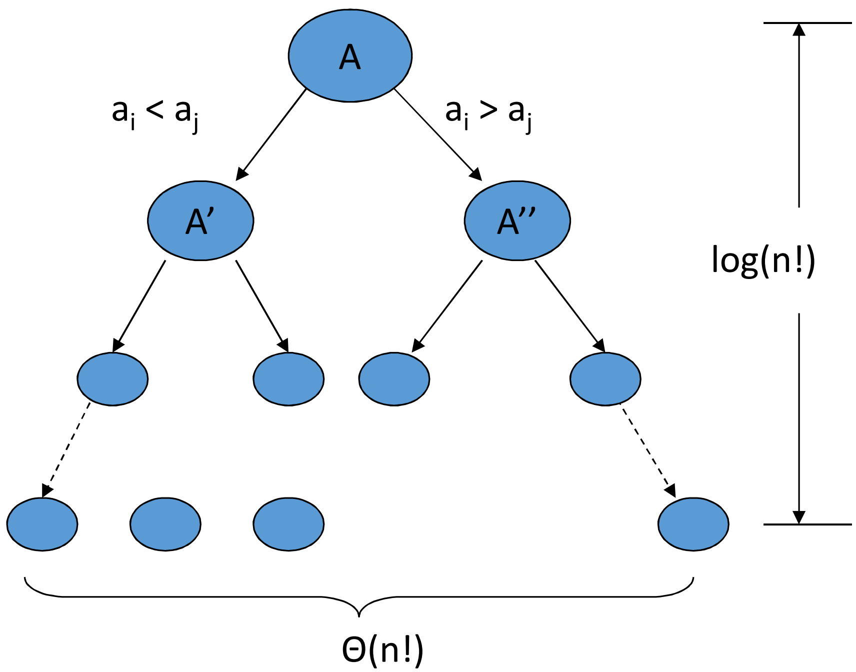
Lower-bound of comparison-based sort

- Assume an array A with 3 distinct elements, a_1, a_2, and a_3
- Use insertion sort
- # comparisons?
- A = [9 6 5]
- A = [5 6 9]
- A = …



Lower-bound of comparison-based sort

- Assume all elements are distinct, each comparison has two possible outcomes: $a_i < a_j$ or $a_i > a_j$
- Based on the outcome, change the relative order of some elements
- Output is a permutation of the input
- A correct sorting algorithm can handle any arbitrary input
- n! possible permutations
- Therefore, at least log(n!) = Θ(nlogn) comparisons in the worst case



Sorting in linear time

- Is there a problem with the theory?
- No. We are going to sort without doing comparison
- How is that possible?
- Key: knowledge about the data
  - Example: Almost sorted? All distinct? Many identical ones? Uniformly distributed?
  - The more you know about your data, the more likely you can have a better algorithm

Counting sort

- Knowledge: the numbers fall in a small range
- Example 1: sort the final exam score of a large class
  - 1000 students
  - Maximum score: 100
  - Minimum score: 0
  - Scores are integers
- Example 2: sort students according to the first letter of their last name
  - Number of students: many
  - Number of letters: 26

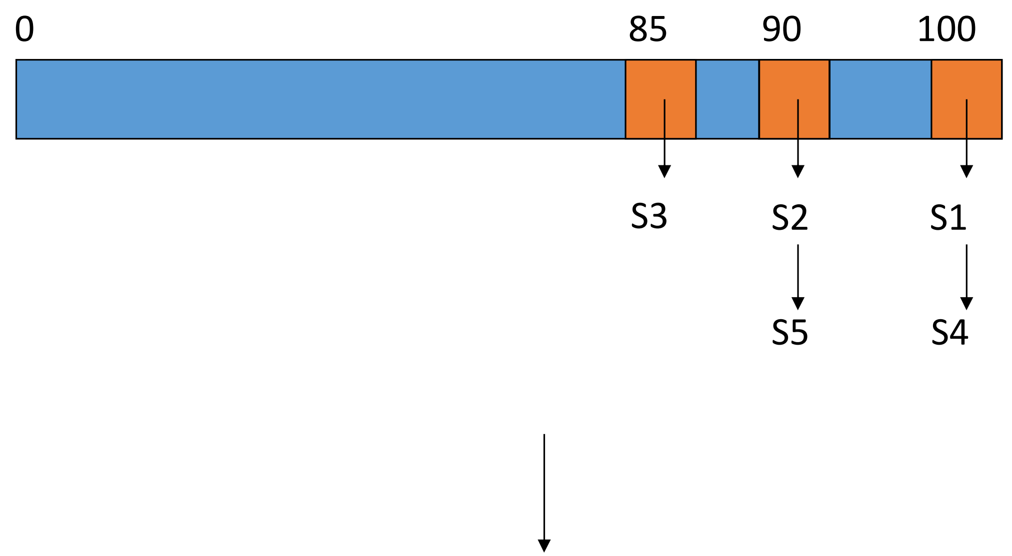
Counting sort

- ***Input***: $A[1 \,.\,.\, n]$, where $A[j] \in \{1, 2, \ldots, k\}$.
- ***Output***: $B[1 \,.\,.\, n]$, sorted.
- ***Auxiliary storage***: $C[1 \,.\,.\, k]$.

- Not an in-place sorting algorithm
- Requires Θ (n+k) additional storage besides the original array

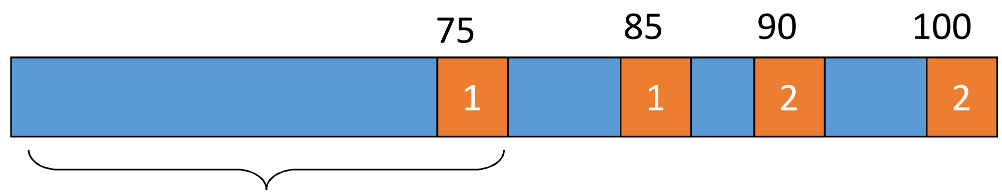
Intuition

- S1: 100
- S2: 90
- S3: 85
- S4: 100
- S5: 90
- …



… S3 … S2, S5, …, S1, S4

Intuition



50 students with score ≤ 75

What is the rank (lowest to highest) for a student with score = 75?

50

200 students with score ≤ 90

What is the rank for a student with score = 90?

200 or 199

Counting sort

1. **for** $i \leftarrow 1$ **to** k
   do $C[i] \leftarrow 0$

Initialize

2. **for** $j \leftarrow 1$ **to** n
   do $C[A[j]] \leftarrow C[A[j]] + 1$ $\triangleright$ $C[i] = |\{\text{key} = i\}|$

Count

3. **for** $i \leftarrow 2$ **to** k
   do $C[i] \leftarrow C[i] + C[i-1]$ $\triangleright$ $C[i] = |\{\text{key} \leq i\}|$

Compute running sum

4. **for** $j \leftarrow n$ **downto** 1
   do $B[C[A[j]]] \leftarrow A[j]$
   $C[A[j]] \leftarrow C[A[j]] - 1$

Re-arrange

Counting-sort example

	1	2	3	4	5
A:	4	1	3	4	3

	1	2	3	4
C:				

B:					

Loop 1: initialization

	1	2	3	4	5
A:	4	1	3	4	3
B:					

	1	2	3	4
C:	0	0	0	0

1. **for** $i \leftarrow 1$ **to** k
   do $C[i] \leftarrow 0$

Loop 2: count

A:

1	2	3	4	5
4	1	3	4	3

C:

1	2	3	4
0	0	0	1

B:

2. **for** $j \leftarrow 1$ **to** n
   do $C[A[j]] \leftarrow C[A[j]] + 1 \quad \triangleright\ C[i] = |\{\text{key} = i\}|$

Loop 2: count

	1	2	3	4	5
A:	4	1	3	4	3

	1	2	3	4
C:	1	0	0	1

B:					

2. **for** $j \leftarrow 1$ **to** n

do $C[A[j]] \leftarrow C[A[j]] + 1$ $\quad \triangleright\ C[i] = |\{\text{key} = i\}|$

Loop 2: count

A:

1	2	3	4	5
4	1	3	4	3

C:

1	2	3	4
1	0	1	1

B:

2. **for** $j \leftarrow 1$ **to** n
    do $C[A[j]] \leftarrow C[A[j]] + 1$ $\triangleright$ $C[i] = |\{\text{key} = i\}|$

Loop 2: count

	1	2	3	4	5
A:	4	1	3	4	3

	1	2	3	4
C:	1	0	1	2

B:					

2. **for** $j \leftarrow 1$ **to** n

 do $C[A[j]] \leftarrow C[A[j]] + 1$ $\quad \rhd\ C[i] = |\{\text{key} = i\}|$

Loop 2: count

	1	2	3	4	5
A:	4	1	3	4	3

	1	2	3	4
C:	1	0	2	2

B:					

2. **for** $j \leftarrow 1$ **to** n
   do $C[A[j]] \leftarrow C[A[j]] + 1$ $\quad \rhd$ $C[i] = |\{\text{key} = i\}|$

Loop 3: compute running sum

A:

1	2	3	4	5
4	1	3	4	3

C:

1	2	3	4
1	0	2	2

B:

C':

1	1	2	2

3. **for** $i \leftarrow 2$ **to** k
   do $C[i] \leftarrow C[i] + C[i-1]$ $\quad \triangleright\ C[i] = |\{\text{key} \leq i\}|$

Loop 3: compute running sum

	1	2	3	4	5
A:	4	1	3	4	3
B:					

	1	2	3	4
C:	1	0	2	2
C':	1	1	3	2

3. **for** $i \leftarrow 2$ **to** k

   do $C[i] \leftarrow C[i] + C[i-1]$ $\quad \rhd\ C[i] = |\{\text{key} \le i\}|$

Loop 3: compute running sum

	1	2	3	4	5
A:	4	1	3	4	3
B:					

	1	2	3	4
C:	1	0	2	2
C':	1	1	3	5

3. for $i \leftarrow 2$ **to** k

do $C[i] \leftarrow C[i] + C[i–1]$ ⊳ $C[i] = |\{\text{key} \le i\}|$

Loop 4: re-arrange

	1	2	3	4	5
A:	4	1	3	4	3
B:			3		

	1	2	3	4
C:	1	1	3	5
C':	1	1	3	5

4. **for** $j \leftarrow n$ **downto** 1
   do $B[C[A[j]]] \leftarrow A[j]$
   $C[A[j]] \leftarrow C[A[j]] - 1$

Loop 4: re-arrange

	1	2	3	4	5
A:	4	1	3	4	3
B:			3		

	1	2	3	4
C:	1	1	3	5
C':	1	1	2	5

4. **for** $j \leftarrow n$ **downto** 1
   do $B[C[A[j]]] \leftarrow A[j]$
   $C[A[j]] \leftarrow C[A[j]] - 1$

Loop 4: re-arrange

	1	2	3	4	5
A:	4	1	3	4	3
B:			3		4

	1	2	3	4
C:	1	1	2	5
C':	1	1	2	5

4. **for** $j \leftarrow n$ **downto** 1
   do $B[C[A[j]]] \leftarrow \mathrm{A}[j]$
   $C[A[j]] \leftarrow C[A[j]] - 1$

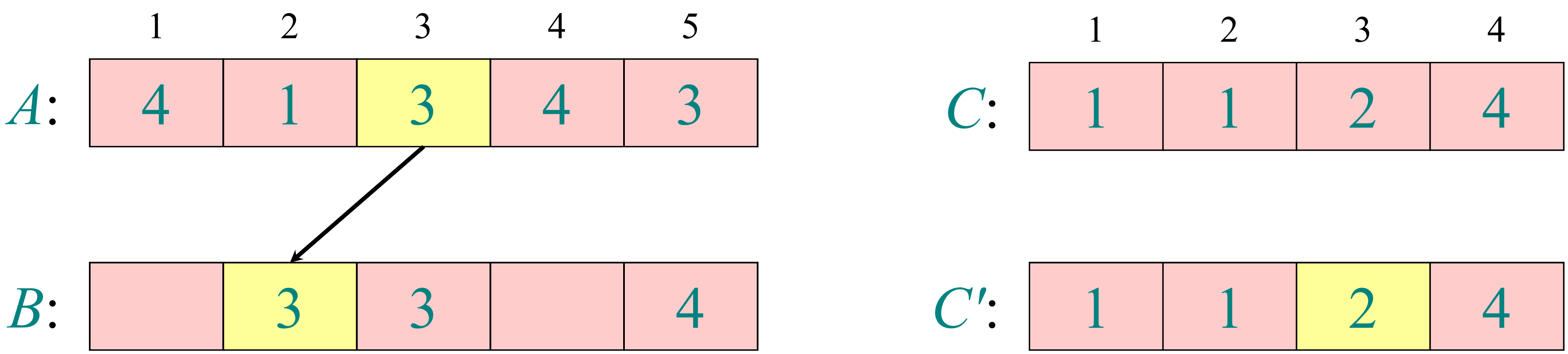
Loop 4: re-arrange

	1	2	3	4	5
A:	4	1	3	4	3
B:			3		4

	1	2	3	4
C:	1	1	2	5
C':	1	1	2	4

4. **for** $j \leftarrow n$ **downto** 1
   do $B[C[A[j]]] \leftarrow \mathrm{A}[j]$
   $C[A[j]] \leftarrow C[A[j]] - 1$

Loop 4: re-arrange

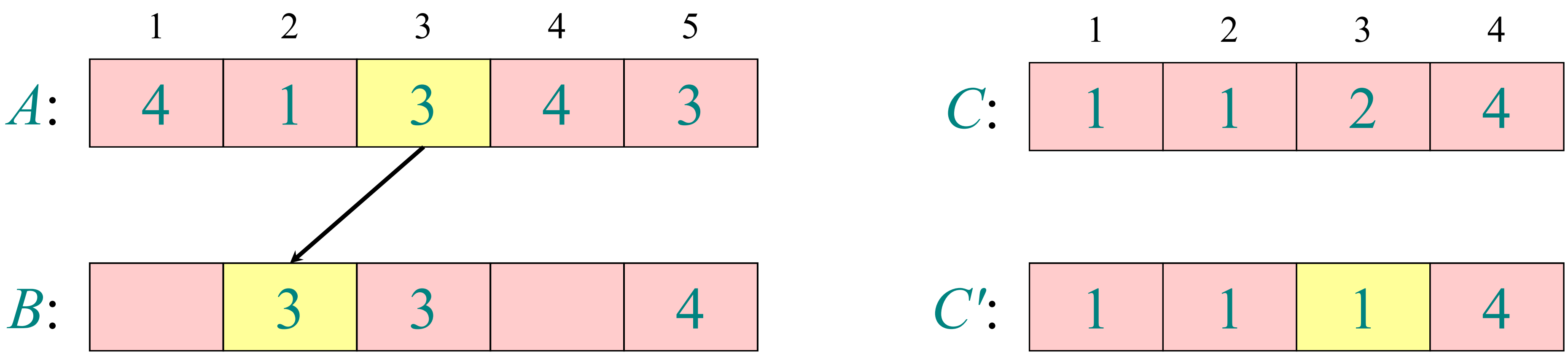


4. **for** $j \leftarrow n$ **downto** 1

do $B[C[A[j]]] \leftarrow \mathrm{A}[j]$

$C[A[j]] \leftarrow C[A[j]] - 1$

Loop 4: re-arrange

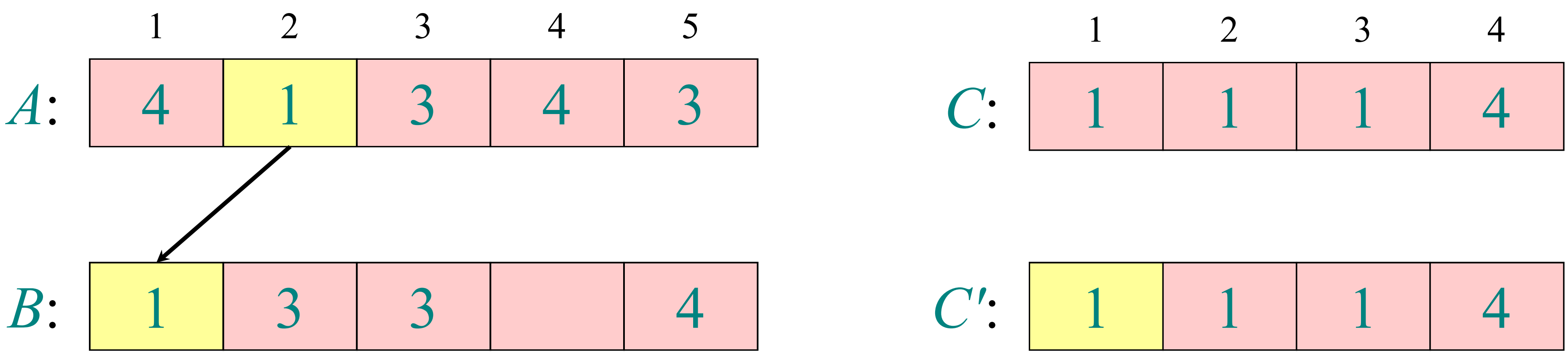


4. **for** $j \leftarrow n$ **downto** 1

do $B[C[A[j]]] \leftarrow \mathrm{A}[j]$

$C[A[j]] \leftarrow C[A[j]] - 1$

Loop 4: re-arrange

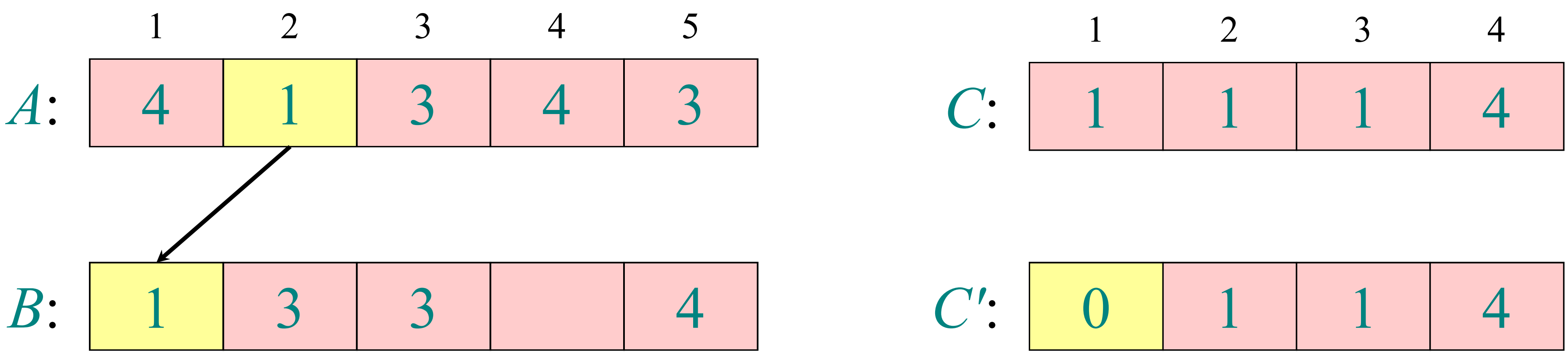


4. **for** $j \leftarrow n$ **downto** 1

 do $B[C[A[j]]] \leftarrow \mathrm{A}[j]$

 $C[A[j]] \leftarrow C[A[j]] - 1$

Loop 4: re-arrange



4. **for** $j \leftarrow n$ **downto** 1

   do $B[C[A[j]]] \leftarrow \mathrm{A}[j]$

   $C[A[j]] \leftarrow C[A[j]] - 1$

Loop 4: re-arrange

	1	2	3	4	5
A:	4	1	3	4	3
B:	1	3	3	4	4

	1	2	3	4
C:	0	1	1	4
C':	0	1	1	4

4. **for** $j \leftarrow n$ **downto** 1
   do $B[C[A[j]]] \leftarrow \mathrm{A}[j]$
   $C[A[j]] \leftarrow C[A[j]] - 1$

Loop 4: re-arrange

	1	2	3	4	5
A:	4	1	3	4	3
B:	1	3	3	4	4

	1	2	3	4
C:	0	1	1	4
C':	0	1	1	3

4. **for** $j \leftarrow n$ **downto** 1
   do $B[C[A[j]]] \leftarrow \mathrm{A}[j]$
   $C[A[j]] \leftarrow C[A[j]] - 1$

Analysis

$\Theta(k)$	**1. for** $i \leftarrow 1$ **to** k **do** $C[i] \leftarrow 0$
$\Theta(n)$	**2. for** $j \leftarrow 1$ **to** n **do** $C[A[j]] \leftarrow C[A[j]] + 1$
$\Theta(k)$	**3. for** $i \leftarrow 2$ **to** k **do** $C[i] \leftarrow C[i] + C[i-1]$
$\Theta(n)$	**4. for** $j \leftarrow n$ **downto** 1 **do** $B[C[A[j]]] \leftarrow \mathrm{A}[j]$ $C[A[j]] \leftarrow C[A[j]] - 1$
$\Theta(n + k)$	

Running time

If $k = O(n)$, then counting sort takes $\Theta(n)$ time.

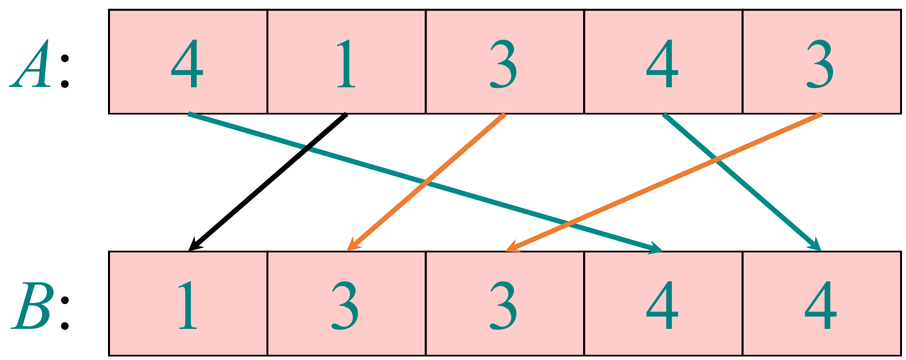
- But, theoretical lower-bound sorting takes $\Omega(n \log n)$ time!
- Problem with the theory?

Answer:

- ***Comparison sorting*** takes $\Omega(n \log n)$ time.
- Counting sort is not a ***comparison sort***.
- In fact, not a single comparison between elements occurs!

Stable sorting

Counting sort is a ***stable*** sort: it preserves the input order among equal elements.



Why this is important?
What other algorithms have this property?

Name		Address		
Last	First	Street	City	State
Bayless	Andrew	West Ave	Houston	TX
Benitez	Michael	North Ave	Los Angeles	CA
Chu	Henry	East Ave	San Diego	CA
Dangelo	David	Third St	Detroit	MI
Dawood	Hussam	Lincoln Rd	Detroit	MI
Devineni	Soujanya	Northwestern Ave	Houston	TX
Dunne	Brendan	EAST AVE.	Dallas	TX
Edwards	Brian	De Zavala Rd	San Antonio	TX
Godfrey	Daryl	MAIN ST	Austin	TX
Guerra	John	DALLAS AVE.	Austin	TX
Guevara	Clovis	University Pkwy	San Antonio	TX
Halbeisen	Gerald	FOREST CIRCLE	Los Angeles	CA
Hawkins	Richard	RIVERSIDE ST	San Antonio	TX
Hernandez	Monica	COLLEGE PKWY	San Antonio	TX
Hohmann	Shawn	COLLEGE PKWY	Cleveland	OH
Honeycutt	Richard	Southwest Ave	San Antonio	TX
Martinez	Juan	OAK CLIFF	Pheonix	AZ
Mayo	Nathan	UTSA BLVD	San Antonio	TX
Mirabal	Renato	FIRST ST	Columbus	OH
Modebe	Nnaemeka	RIVERSIDE ST	Houston	TX
Moon	Ryan	EAST AVE.	Madison	WI

Task: sort students by alphabetical order of their addresses (state, city, street).

Name		Address		
Last	First	Street	City	State
Martinez	Juan	OAK CLIFF	Pheonix	AZ
Benitez	Michael	North Ave	Los Angeles	CA
Chu	Henry	East Ave	San Diego	CA
Halbeisen	Gerald	FOREST CIRCLE	Los Angeles	CA
Dangelo	David	Third St	Detroit	MI
Dawood	Hussam	Lincoln Rd	Detroit	MI
Hohmann	Shawn	COLLEGE PKWY	Cleveland	OH
Mirabal	Renato	FIRST ST	Columbus	OH
Bayless	Andrew	West Ave	Houston	TX
Devineni	Soujanya	Northwestern Ave	Houston	TX
Dunne	Brendan	EAST AVE.	Dallas	TX
Edwards	Brian	De Zavala Rd	San Antonio	TX
Godfrey	Daryl	MAIN ST	Austin	TX
Guerra	John	DALLAS AVE.	Austin	TX
Guevara	Clovis	University Pkwy	San Antonio	TX
Hawkins	Richard	RIVERSIDE ST	San Antonio	TX
Hernandez	Monica	COLLEGE PKWY	San Antonio	TX
Honeycutt	Richard	Southwest Ave	San Antonio	TX
Mayo	Nathan	UTSA BLVD	San Antonio	TX
Modebe	Nnaemeka	RIVERSIDE ST	Houston	TX
Moon	Ryan	EAST AVE.	Madison	WI

Task: sort students by alphabetical order of their addresses (state, city, street).

Name		Address		
Last	First	Street	City	State
Godfrey	Daryl	MAIN ST	Austin	TX
Guerra	John	DALLAS AVE.	Austin	TX
Hohmann	Shawn	COLLEGE PKWY	Cleveland	OH
Mirabal	Renato	FIRST ST	Columbus	OH
Dunne	Brendan	EAST AVE.	Dallas	TX
Dangelo	David	Third St	Detroit	MI
Dawood	Hussam	Lincoln Rd	Detroit	MI
Bayless	Andrew	West Ave	Houston	TX
Devineni	Soujanya	Northwestern Ave	Houston	TX
Modebe	Nnaemeka	RIVERSIDE ST	Houston	TX
Benitez	Michael	North Ave	Los Angeles	CA
Halbeisen	Gerald	FOREST CIRCLE	Los Angeles	CA
Moon	Ryan	EAST AVE.	Madison	WI
Martinez	Juan	OAK CLIFF	Pheonix	AZ
Edwards	Brian	De Zavala Rd	San Antonio	TX
Guevara	Clovis	University Pkwy	San Antonio	TX
Hawkins	Richard	RIVERSIDE ST	San Antonio	TX
Hernandez	Monica	COLLEGE PKWY	San Antonio	TX
Honeycutt	Richard	Southwest Ave	San Antonio	TX
Mayo	Nathan	UTSA BLVD	San Antonio	TX
Chu	Henry	East Ave	San Diego	CA

Task: sort students by alphabetical order of their addresses (state, city, street).

Name		Address		
Last	First	Street	City	State
Hohmann	Shawn	COLLEGE PKWY	Cleveland	OH
Hernandez	Monica	COLLEGE PKWY	San Antonio	TX
Guerra	John	DALLAS AVE.	Austin	TX
Edwards	Brian	De Zavala Rd	San Antonio	TX
Chu	Henry	East Ave	San Diego	CA
Dunne	Brendan	EAST AVE.	Dallas	TX
Moon	Ryan	EAST AVE.	Madison	WI
Mirabal	Renato	FIRST ST	Columbus	OH
Halbeisen	Gerald	FOREST CIRCLE	Los Angeles	CA
Dawood	Hussam	Lincoln Rd	Detroit	MI
Godfrey	Daryl	MAIN ST	Austin	TX
Benitez	Michael	North Ave	Los Angeles	CA
Devineni	Soujanya	Northwestern Ave	Houston	TX
Martinez	Juan	OAK CLIFF	Pheonix	AZ
Modebe	Nnaemeka	RIVERSIDE ST	Houston	TX
Hawkins	Richard	RIVERSIDE ST	San Antonio	TX
Honeycutt	Richard	Southwest Ave	San Antonio	TX
Dangelo	David	Third St	Detroit	MI
Guevara	Clovis	University Pkwy	San Antonio	TX
Mayo	Nathan	UTSA BLVD	San Antonio	TX
Bayless	Andrew	West Ave	Houston	TX

Task: sort students by alphabetical order of their addresses (state, city, street).

Name		Address		
Last	First	Street	City	State
Bayless	Andrew	West Ave	Houston	TX
Benitez	Michael	North Ave	Los Angeles	CA
Chu	Henry	East Ave	San Diego	CA
Dangelo	David	Third St	Detroit	MI
Dawood	Hussam	Lincoln Rd	Detroit	MI
Devineni	Soujanya	Northwestern Ave	Houston	TX
Dunne	Brendan	EAST AVE.	Dallas	TX
Edwards	Brian	De Zavala Rd	San Antonio	TX
Godfrey	Daryl	MAIN ST	Austin	TX
Guerra	John	DALLAS AVE.	Austin	TX
Guevara	Clovis	University Pkwy	San Antonio	TX
Halbeisen	Gerald	FOREST CIRCLE	Los Angeles	CA
Hawkins	Richard	RIVERSIDE ST	San Antonio	TX
Hernandez	Monica	COLLEGE PKWY	San Antonio	TX
Hohmann	Shawn	COLLEGE PKWY	Cleveland	OH
Honeycutt	Richard	Southwest Ave	San Antonio	TX
Martinez	Juan	OAK CLIFF	Pheonix	AZ
Mayo	Nathan	UTSA BLVD	San Antonio	TX
Mirabal	Renato	FIRST ST	Columbus	OH
Modebe	Nnaemeka	RIVERSIDE ST	Houston	TX
Moon	Ryan	EAST AVE.	Madison	WI

Task: sort students by alphabetical order of their addresses (state, city, street).

Name		Address		
Last	First	Street	City	State
Hernandez	Monica	COLLEGE PKWY	San Antonio	TX
Hohmann	Shawn	COLLEGE PKWY	Cleveland	OH
Guerra	John	DALLAS AVE.	Austin	TX
Edwards	Brian	De Zavala Rd	San Antonio	TX
Chu	Henry	East Ave	San Diego	CA
Dunne	Brendan	EAST AVE.	Dallas	TX
Moon	Ryan	EAST AVE.	Madison	WI
Mirabal	Renato	FIRST ST	Columbus	OH
Halbeisen	Gerald	FOREST CIRCLE	Los Angeles	CA
Dawood	Hussam	Lincoln Rd	Detroit	MI
Godfrey	Daryl	MAIN ST	Austin	TX
Benitez	Michael	North Ave	Los Angeles	CA
Devineni	Soujanya	Northwestern Ave	Houston	TX
Martinez	Juan	OAK CLIFF	Pheonix	AZ
Hawkins	Richard	RIVERSIDE ST	San Antonio	TX
Modebe	Nnaemeka	RIVERSIDE ST	Houston	TX
Honeycutt	Richard	Southwest Ave	San Antonio	TX
Dangelo	David	Third St	Detroit	MI
Guevara	Clovis	University Pkwy	San Antonio	TX
Mayo	Nathan	UTSA BLVD	San Antonio	TX
Bayless	Andrew	West Ave	Houston	TX

Task: sort students by alphabetical order of their addresses (state, city, street).

Name		Address		
Last	First	Street	City	State
Guerra	John	DALLAS AVE.	Austin	TX
Godfrey	Daryl	MAIN ST	Austin	TX
Hohmann	Shawn	COLLEGE PKWY	Cleveland	OH
Mirabal	Renato	FIRST ST	Columbus	OH
Dunne	Brendan	EAST AVE.	Dallas	TX
Dawood	Hussam	Lincoln Rd	Detroit	MI
Dangelo	David	Third St	Detroit	MI
Devineni	Soujanya	Northwestern Ave	Houston	TX
Modebe	Nnaemeka	RIVERSIDE ST	Houston	TX
Bayless	Andrew	West Ave	Houston	TX
Halbeisen	Gerald	FOREST CIRCLE	Los Angeles	CA
Benitez	Michael	North Ave	Los Angeles	CA
Moon	Ryan	EAST AVE.	Madison	WI
Martinez	Juan	OAK CLIFF	Pheonix	AZ
Hernandez	Monica	COLLEGE PKWY	San Antonio	TX
Edwards	Brian	De Zavala Rd	San Antonio	TX
Hawkins	Richard	RIVERSIDE ST	San Antonio	TX
Honeycutt	Richard	Southwest Ave	San Antonio	TX
Guevara	Clovis	University Pkwy	San Antonio	TX
Mayo	Nathan	UTSA BLVD	San Antonio	TX
Chu	Henry	East Ave	San Diego	CA

Task: sort students by alphabetical order of their addresses (state, city, street).

Name		Address		
Last	First	Street	City	State
Martinez	Juan	OAK CLIFF	Pheonix	AZ
Halbeisen	Gerald	FOREST CIRCLE	Los Angeles	CA
Benitez	Michael	North Ave	Los Angeles	CA
Chu	Henry	East Ave	San Diego	CA
Dawood	Hussam	Lincoln Rd	Detroit	MI
Dangelo	David	Third St	Detroit	MI
Hohmann	Shawn	COLLEGE PKWY	Cleveland	OH
Mirabal	Renato	FIRST ST	Columbus	OH
Guerra	John	DALLAS AVE.	Austin	TX
Godfrey	Daryl	MAIN ST	Austin	TX
Dunne	Brendan	EAST AVE.	Dallas	TX
Devineni	Soujanya	Northwestern Ave	Houston	TX
Modebe	Nnaemeka	RIVERSIDE ST	Houston	TX
Bayless	Andrew	West Ave	Houston	TX
Hernandez	Monica	COLLEGE PKWY	San Antonio	TX
Edwards	Brian	De Zavala Rd	San Antonio	TX
Hawkins	Richard	RIVERSIDE ST	San Antonio	TX
Honeycutt	Richard	Southwest Ave	San Antonio	TX
Guevara	Clovis	University Pkwy	San Antonio	TX
Mayo	Nathan	UTSA BLVD	San Antonio	TX
Moon	Ryan	EAST AVE.	Madison	WI

Task: sort students by alphabetical order of their addresses (state, city, street).

Name		Address		
Last	First	Street	City	State
Martinez	Juan	OAK CLIFF	Pheonix	AZ
Halbeisen	Gerald	FOREST CIRCLE	Los Angeles	CA
Benitez	Michael	North Ave	Los Angeles	CA
Chu	Henry	East Ave	San Diego	CA
Dawood	Hussam	Lincoln Rd	Detroit	MI
Dangelo	David	Third St	Detroit	MI
Hohmann	Shawn	COLLEGE PKWY	Cleveland	OH
Mirabal	Renato	FIRST ST	Columbus	OH
Guerra	John	DALLAS AVE.	Austin	TX
Godfrey	Daryl	MAIN ST	Austin	TX
Dunne	Brendan	EAST AVE.	Dallas	TX
Devineni	Soujanya	Northwestern Ave	Houston	TX
Modebe	Nnaemeka	RIVERSIDE ST	Houston	TX
Bayless	Andrew	West Ave	Houston	TX
Hernandez	Monica	COLLEGE PKWY	San Antonio	TX
Edwards	Brian	De Zavala Rd	San Antonio	TX
Hawkins	Richard	RIVERSIDE ST	San Antonio	TX
Honeycutt	Richard	Southwest Ave	San Antonio	TX
Guevara	Clovis	University Pkwy	San Antonio	TX
Mayo	Nathan	UTSA BLVD	San Antonio	TX
Moon	Ryan	EAST AVE.	Madison	WI

Task: sort students by alphabetical order of their addresses (state, city, street).

Name		Address		
Last	First	Street	City	State
Bayless	Andrew	West Ave	Houston	TX
Benitez	Michael	North Ave	Los Angeles	CA
Chu	Henry	East Ave	San Diego	CA
Dangelo	David	Third St	Detroit	MI
Dawood	Hussam	Lincoln Rd	Detroit	MI
Devineni	Soujanya	Northwestern Ave	Houston	TX
Dunne	Brendan	EAST AVE.	Dallas	TX
Edwards	Brian	De Zavala Rd	San Antonio	TX
Godfrey	Daryl	MAIN ST	Austin	TX
Guerra	John	DALLAS AVE.	Austin	TX
Guevara	Clovis	University Pkwy	San Antonio	TX
Halbeisen	Gerald	FOREST CIRCLE	Los Angeles	CA
Hawkins	Richard	RIVERSIDE ST	San Antonio	TX
Hernandez	Monica	COLLEGE PKWY	San Antonio	TX
Hohmann	Shawn	COLLEGE PKWY	Cleveland	OH
Honeycutt	Richard	Southwest Ave	San Antonio	TX
Martinez	Juan	OAK CLIFF	Pheonix	AZ
Mayo	Nathan	UTSA BLVD	San Antonio	TX
Mirabal	Renato	FIRST ST	Columbus	OH
Modebe	Nnaemeka	RIVERSIDE ST	Houston	TX
Moon	Ryan	EAST AVE.	Madison	WI

Task: sort students by alphabetical order of their addresses (state, city, street).

Name		Address		
Last	First	Street	City	State
Hernandez	Monica	COLLEGE PKWY	San Antonio	TX
Hohmann	Shawn	COLLEGE PKWY	Cleveland	OH
Guerra	John	DALLAS AVE.	Austin	TX
Edwards	Brian	De Zavala Rd	San Antonio	TX
Chu	Henry	East Ave	San Diego	CA
Dunne	Brendan	EAST AVE.	Dallas	TX
Moon	Ryan	EAST AVE.	Madison	WI
Mirabal	Renato	FIRST ST	Columbus	OH
Halbeisen	Gerald	FOREST CIRCLE	Los Angeles	CA
Dawood	Hussam	Lincoln Rd	Detroit	MI
Godfrey	Daryl	MAIN ST	Austin	TX
Benitez	Michael	North Ave	Los Angeles	CA
Devineni	Soujanya	Northwestern Ave	Houston	TX
Martinez	Juan	OAK CLIFF	Pheonix	AZ
Hawkins	Richard	RIVERSIDE ST	San Antonio	TX
Modebe	Nnaemeka	RIVERSIDE ST	Houston	TX
Honeycutt	Richard	Southwest Ave	San Antonio	TX
Dangelo	David	Third St	Detroit	MI
Guevara	Clovis	University Pkwy	San Antonio	TX
Mayo	Nathan	UTSA BLVD	San Antonio	TX
Bayless	Andrew	West Ave	Houston	TX

Task: sort students by alphabetical order of their addresses (state, city, street).

Name		Address		
Last	First	Street	City	State
Guerra	John	DALLAS AVE.	Austin	TX
Godfrey	Daryl	MAIN ST	Austin	TX
Hohmann	Shawn	COLLEGE PKWY	Cleveland	OH
Mirabal	Renato	FIRST ST	Columbus	OH
Dunne	Brendan	EAST AVE.	Dallas	TX
Dawood	Hussam	Lincoln Rd	Detroit	MI
Dangelo	David	Third St	Detroit	MI
Devineni	Soujanya	Northwestern Ave	Houston	TX
Modebe	Nnaemeka	RIVERSIDE ST	Houston	TX
Bayless	Andrew	West Ave	Houston	TX
Halbeisen	Gerald	FOREST CIRCLE	Los Angeles	CA
Benitez	Michael	North Ave	Los Angeles	CA
Moon	Ryan	EAST AVE.	Madison	WI
Martinez	Juan	OAK CLIFF	Pheonix	AZ
Hernandez	Monica	COLLEGE PKWY	San Antonio	TX
Edwards	Brian	De Zavala Rd	San Antonio	TX
Hawkins	Richard	RIVERSIDE ST	San Antonio	TX
Honeycutt	Richard	Southwest Ave	San Antonio	TX
Guevara	Clovis	University Pkwy	San Antonio	TX
Mayo	Nathan	UTSA BLVD	San Antonio	TX
Chu	Henry	East Ave	San Diego	CA

Task: sort students by alphabetical order of their addresses (state, city, street).

Name		Address		
Last	First	Street	City	State
Martinez	Juan	OAK CLIFF	Pheonix	AZ
Halbeisen	Gerald	FOREST CIRCLE	Los Angeles	CA
Benitez	Michael	North Ave	Los Angeles	CA
Chu	Henry	East Ave	San Diego	CA
Dawood	Hussam	Lincoln Rd	Detroit	MI
Dangelo	David	Third St	Detroit	MI
Hohmann	Shawn	COLLEGE PKWY	Cleveland	OH
Mirabal	Renato	FIRST ST	Columbus	OH
Guerra	John	DALLAS AVE.	Austin	TX
Godfrey	Daryl	MAIN ST	Austin	TX
Dunne	Brendan	EAST AVE.	Dallas	TX
Devineni	Soujanya	Northwestern Ave	Houston	TX
Modebe	Nnaemeka	RIVERSIDE ST	Houston	TX
Bayless	Andrew	West Ave	Houston	TX
Hernandez	Monica	COLLEGE PKWY	San Antonio	TX
Edwards	Brian	De Zavala Rd	San Antonio	TX
Hawkins	Richard	RIVERSIDE ST	San Antonio	TX
Honeycutt	Richard	Southwest Ave	San Antonio	TX
Guevara	Clovis	University Pkwy	San Antonio	TX
Mayo	Nathan	UTSA BLVD	San Antonio	TX
Moon	Ryan	EAST AVE.	Madison	WI

Task: sort students by alphabetical order of their addresses (state, city, street).

Stable sort

- Most Θ(n^2) sorting algorithms are stable
  - Standard selection sort is not, but can be made so
- Most Θ(n log n) sorting algorithms are not stable
  - Except merge sort
- Generic way to make any sorting algorithm stable
  - Use two keys, the second key is the original index of the element
  - When two elements are equal, compare their second keys

  5, 6, 5, 1, 2, 3, 2, 6

  (5, 1), (6, 2), (5, 3), (1, 4), (2, 5), (3, 6), (2, 7), (6, 8)

  (5, 1) < (5, 3)

  (2, 5) < (2, 7)

How to sort very large numbers?

19809910912351818 3599
34019954038012811 5295

Those numbers are too large for the int type.
They are represented as strings.

One method: Use comparison-based sorting, but compare strings character by character.

Change if (A[i] < A[j]) to if (compare(A[i], A[j]) < 0)

```
Compare(s, t)
  for i = 1 to length(s)
    if (s[i] < t[i]) return -1;
    else if (s[i] > t[i]) return 1;
  return 0;
```

What's the cost to compare two strings, each with d characters?

$\Theta(d)$

Total cost: $\Theta(d\ n \log n)$

Radix sort

- Similar to sorting address books
- Treat each digit as a key
- Start from the least significant bit

Most significant $\downarrow$ Least significant $\downarrow$

198099109123518183599
340199540380128115295
384700101594539614696
382408360201039258538
614386507628681328936

Radix sort illustration

- Use simpler examples:

7 4 2
7 4 8
0 5 4
6 8 8
4 1 2
2 3 0
9 3 5
1 1 6
1 6 1
4 3 4
3 8 5
6 6 6
0 3 1
0 1 3
3 6 5
1 7 3
0 1 6

Radix sort illustration

- Sort the last digit:

2 3 0
1 6 1
0 3 1
7 4 2
4 1 2
0 1 3
1 7 3
0 5 4
4 3 4
9 3 5
3 8 5
3 6 5
1 1 6
6 6 6
0 1 6
7 4 8
6 8 8

Radix sort illustration

- Sort the second digit:

$\downarrow$

<u>4 1 2</u>
<u>0 1 3</u>
<u>1 1 6</u>
<u>0 1 6</u>
<u>2 3 0</u>
<u>0 3 1</u>
<u>4 3 4</u>
<u>9 3 5</u>
<u>7 4 2</u>
<u>7 4 8</u>
<u>0 5 4</u>
<u>1 6 1</u>
<u>3 6 5</u>
<u>6 6 6</u>
<u>1 7 3</u>
<u>3 8 5</u>
<u>6 8 8</u>

Radix sort illustration

- Sort the first digit:

0 1 3
0 1 6
0 3 1
0 5 4
1 1 6
1 6 1
1 7 3
2 3 0
3 6 5
3 8 5
4 1 2
4 3 4
6 6 6
6 8 8
7 4 2
7 4 8
9 3 5

Time complexity

- Sort each of the d digits by counting sort
- Total cost: $d\ (n + k)$
  - k = 10
  - Total cost: $\Theta(dn)$
- Partition the d digits into groups of 3
  - Total cost: $(n+10^3)d/3$
- We work with binaries rather than decimals
  - Partition d bits into groups of r bits
  - Total cost: $(n+2^r)d/r$
  - Choose $r = log\ n$
  - Total cost: $dn\ /\ log\ n$
  - Compare with $dn\ log\ n$
- Catch: radix sort has a larger hidden constant factor

Space complexity

- Calls counting sort
- Therefore additional storage is needed
- $\Theta(n)$