CENG 218
Design and Analysis of Algorithms

Izmir Institute of Technology

Lecture 6: Quicksort

Slides were mostly prepared using the material provided by Prof. Charles E. Leiserson and Prof. Erik Demaine from MIT

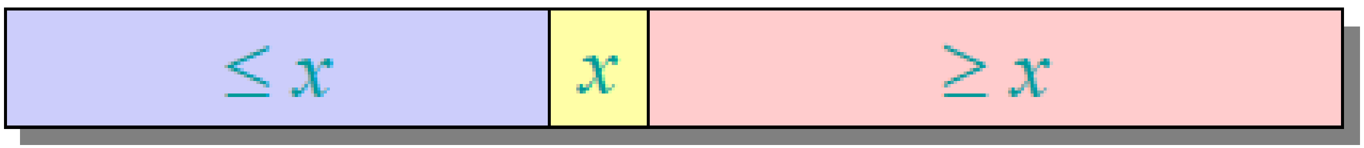
Quicksort

- Proposed by C.A.R. Hoare in 1962.
- Divide-and-conquer algorithm.
- Sorts “in place”
    - like insertion sort, but not like merge sort
    - requires less storage
- Very practical (with tuning).

Divide-and-conquer for Quicksort

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.



2. Conquer: Recursively sort the two subarrays.

3. Combine: Trivial (no extra work needed).

Key: Linear-time ($\Theta(n)$) partitioning subroutine

Partitioning subroutine

PARTITION(A, p, q) $\rhd A[p \ldots q]$

$x \leftarrow A[p]$ {we choose $A[p]$ as the pivot}

$i \leftarrow p$

for $j \leftarrow p+1$ to q

if $A[j] \leq x$ **then**

begin

$i \leftarrow i+1$

exchange $A[i] \leftrightarrow A[j]$

end

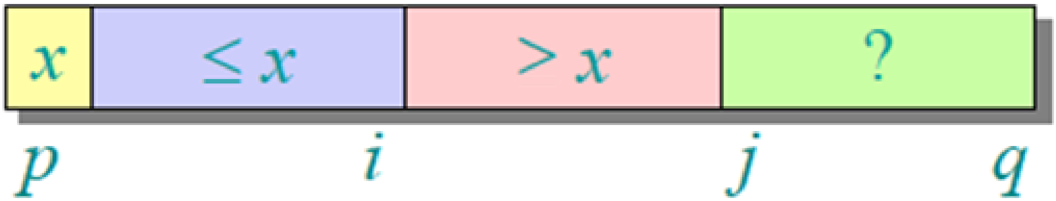
exchange $A[p] \leftrightarrow A[i]$

return i

Running time:

$\Theta(n)$

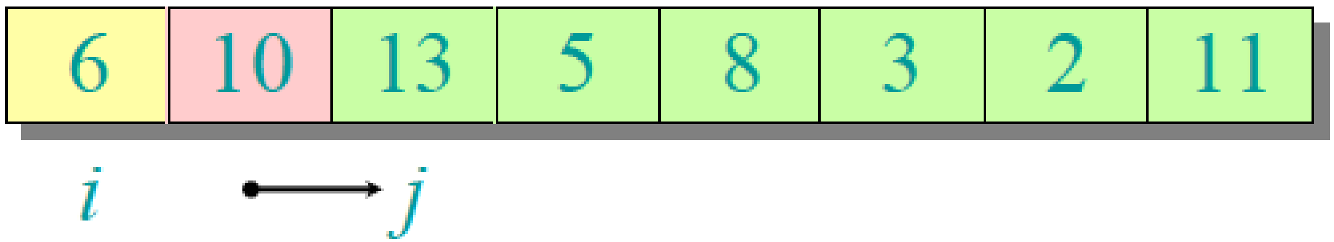
with n elements



Example of partitioning

6	10	13	5	8	3	2	11
i	j						

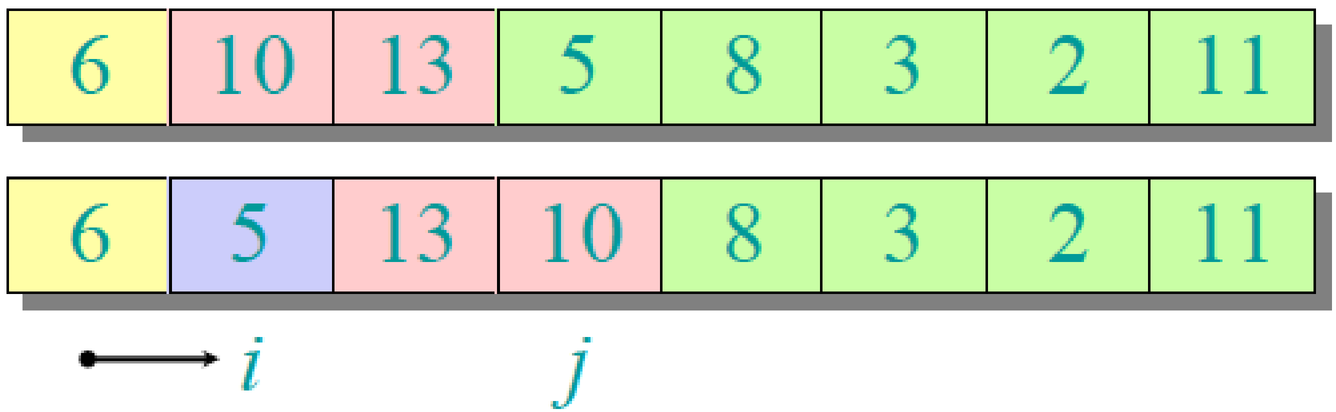
Example of partitioning



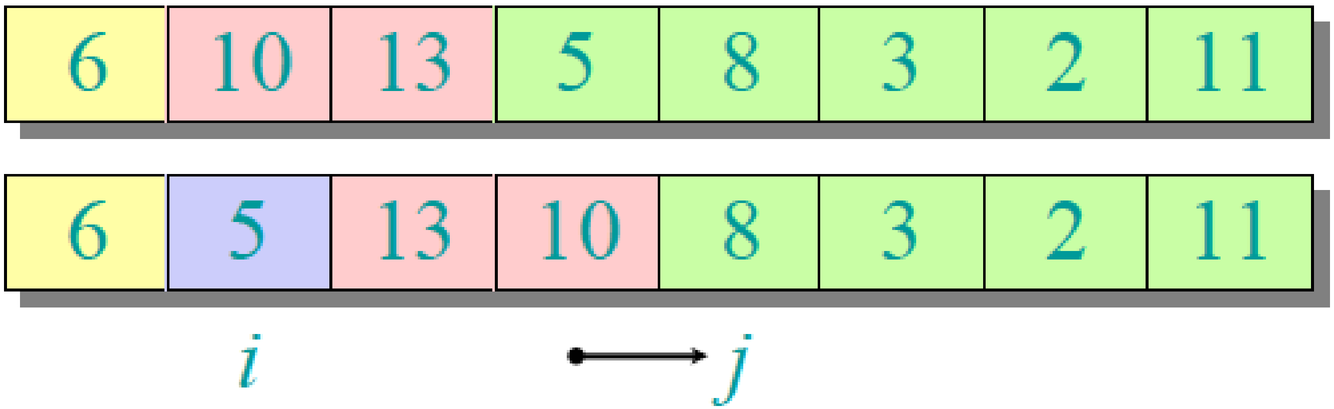
Example of partitioning

6	10	13	5	8	3	2	11
i			j				

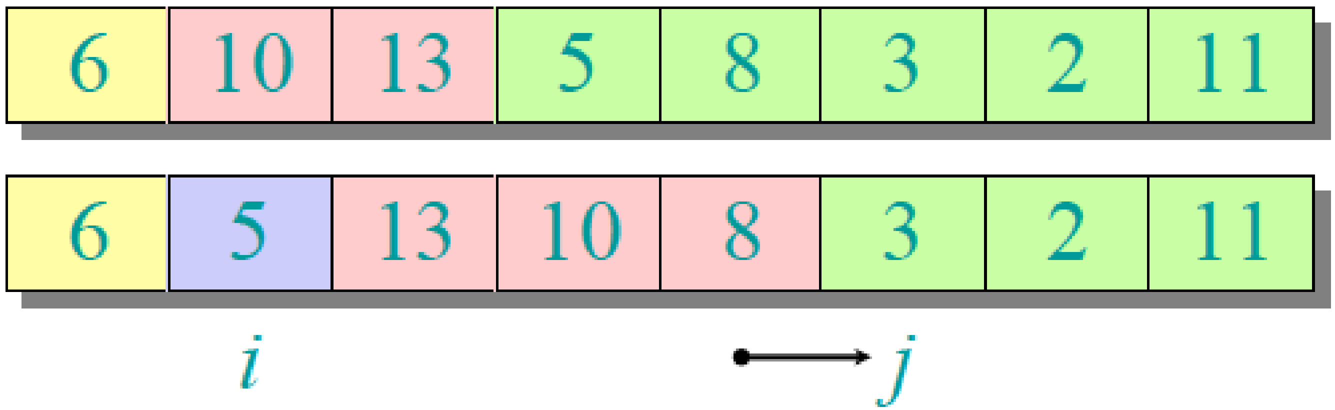
Example of partitioning



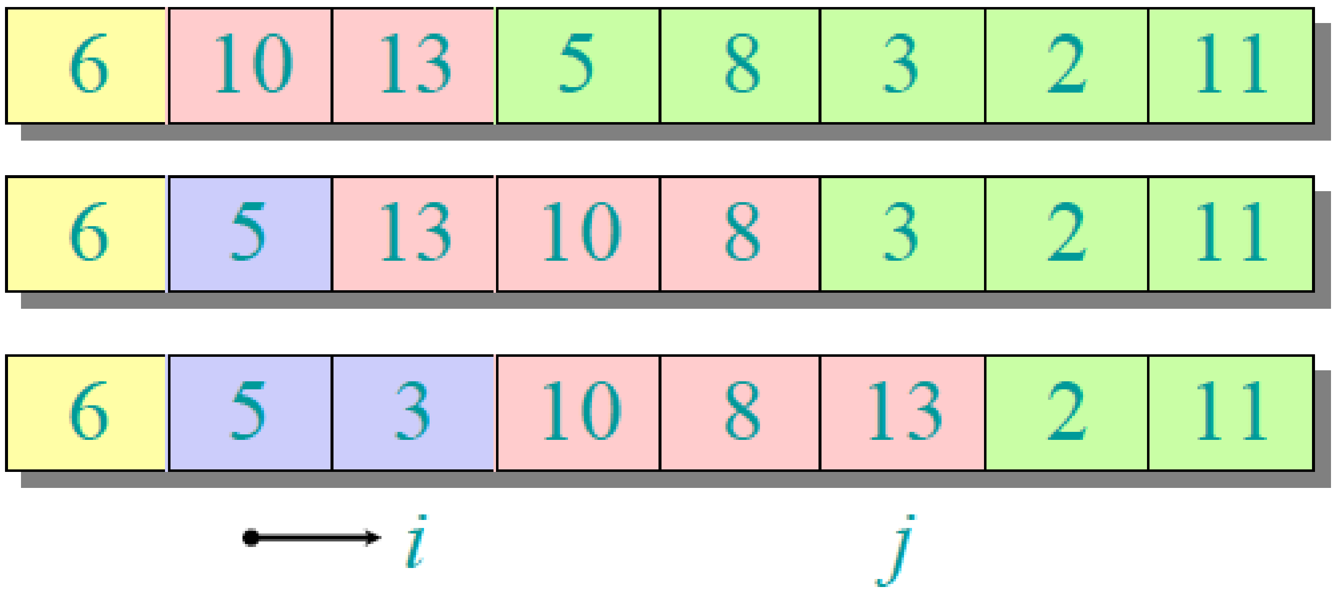
Example of partitioning



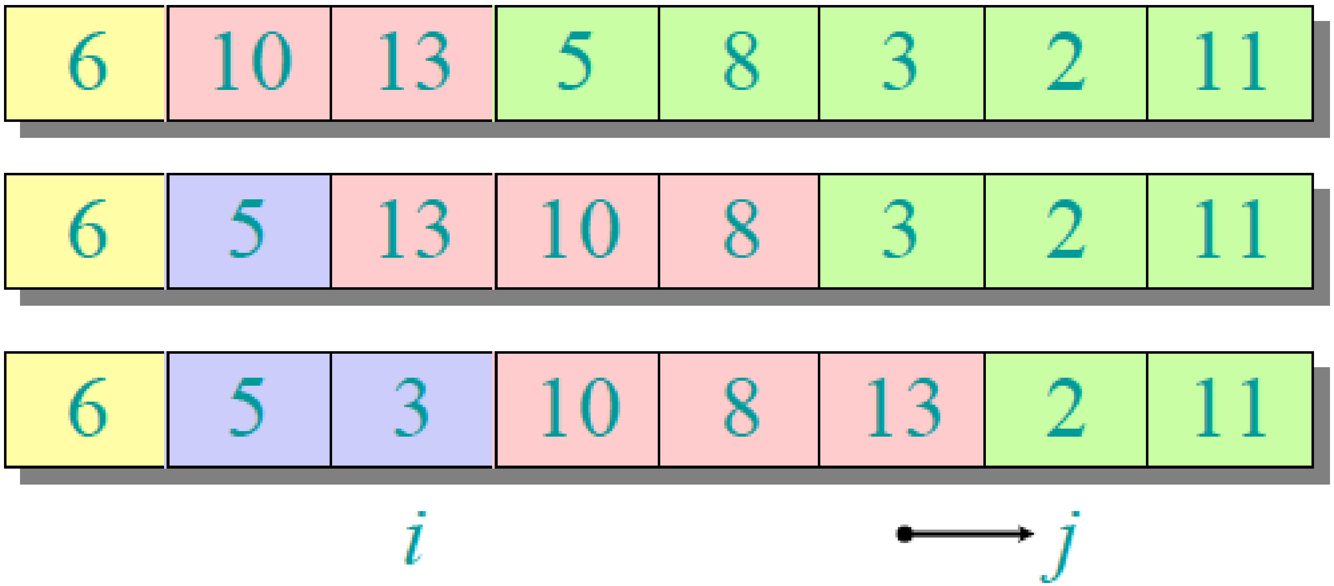
Example of partitioning



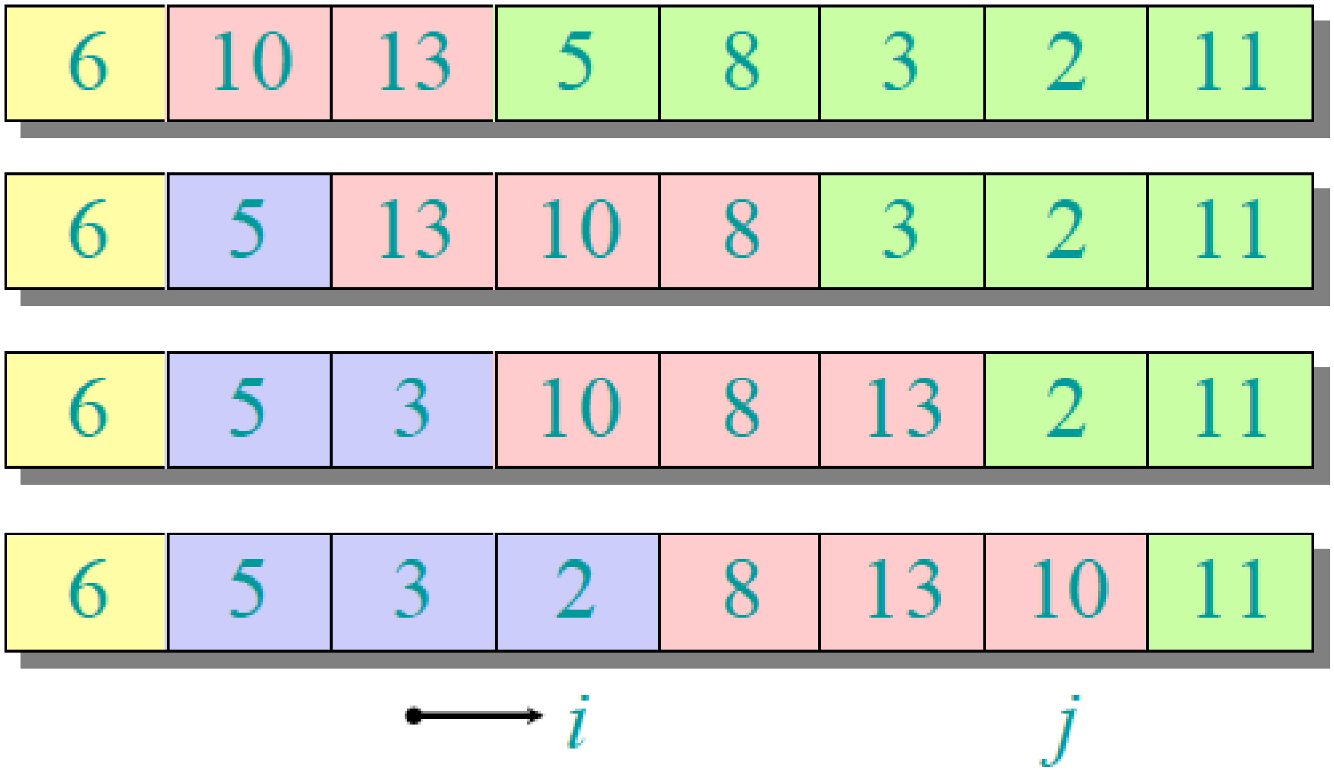
Example of partitioning



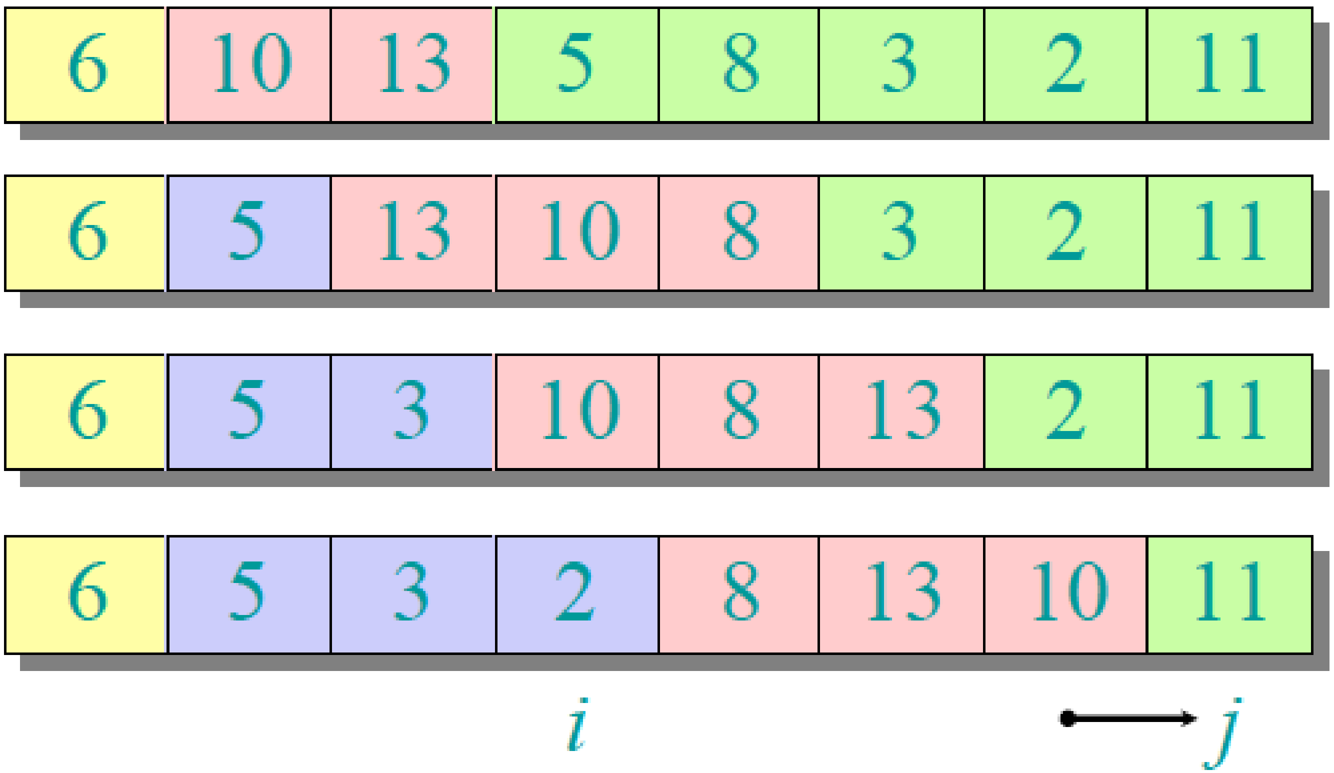
Example of partitioning



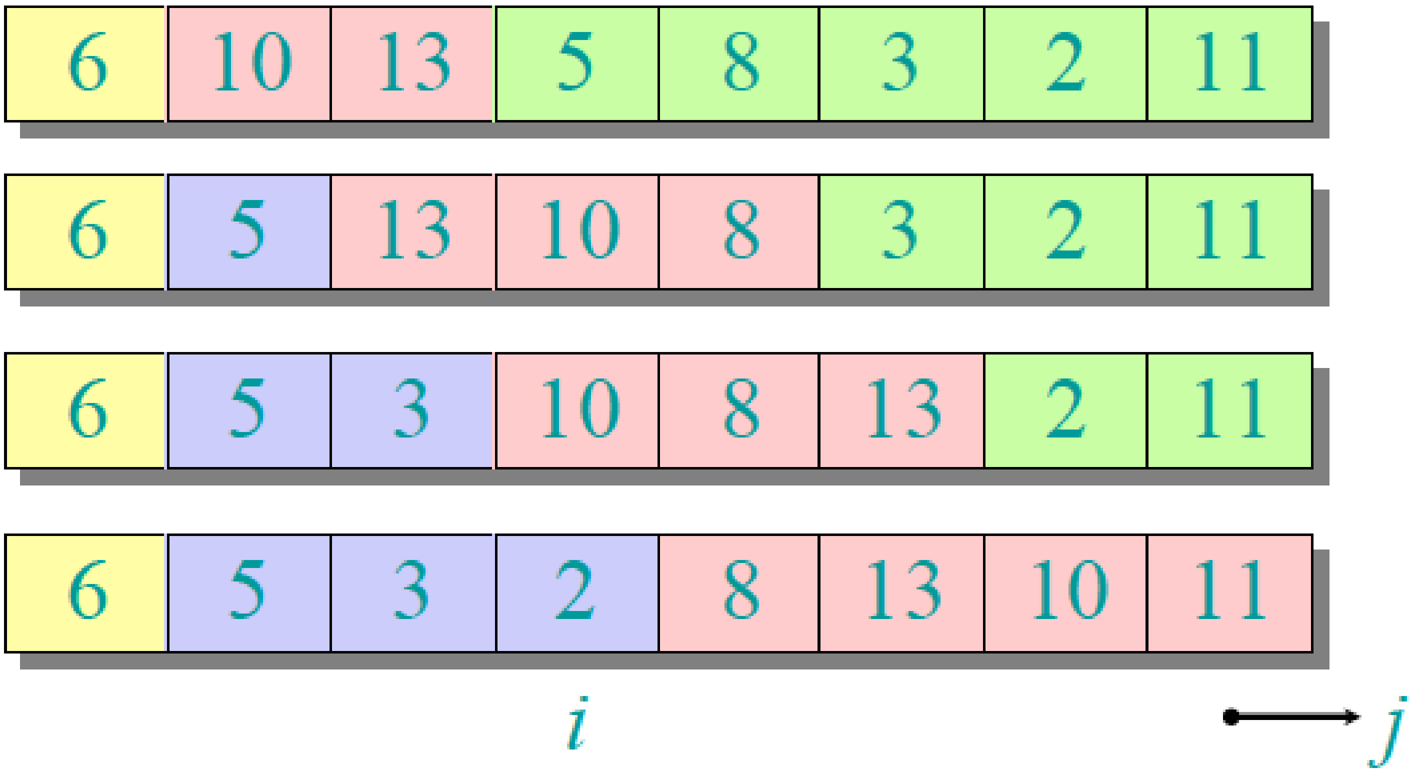
Example of partitioning



Example of partitioning



Example of partitioning



Example of partitioning

6	10	13	5	8	3	2	11
6	5	13	10	8	3	2	11
6	5	3	10	8	13	2	11
6	5	3	2	8	13	10	11
2	5	3	6	8	13	10	11
			i				

Pseudocode for Quicksort

QUICKSORT (A, p, q)

if $p < q$ **then**

begin

$r \leftarrow$ PARTITION(A, p, q)

QUICKSORT (A, p, r-1)

QUICKSORT (A, r+1, q)

end

Initial call: QUICKSORT (A, 1, n)

Analysis of Quicksort

Let $T(n)$ = worst-case running time on an array of n elements.

Q. What is the worst-case for Quicksort?

A. Repeatedly partitioning around min or max element. One side of partition always has no elements. $T(n) = T(0) + T(n–1) + \Theta(n)$

Q. When does this happen?

A. Input is already sorted (or reverse sorted).

Worst-case recursion tree

$$T(n) = \Theta(1) + T(n-1) + cn$$

$T(n)$

Worst-case recursion tree

$$T(n) = \Theta(1) + T(n-1) + cn$$

$$\begin{array}{ccc} & cn & \\ \swarrow & & \searrow \\ \Theta(1) & & T(n-1) \end{array}$$

Worst-case recursion tree

$$T(n) = \Theta(1) + T(n-1) + cn$$

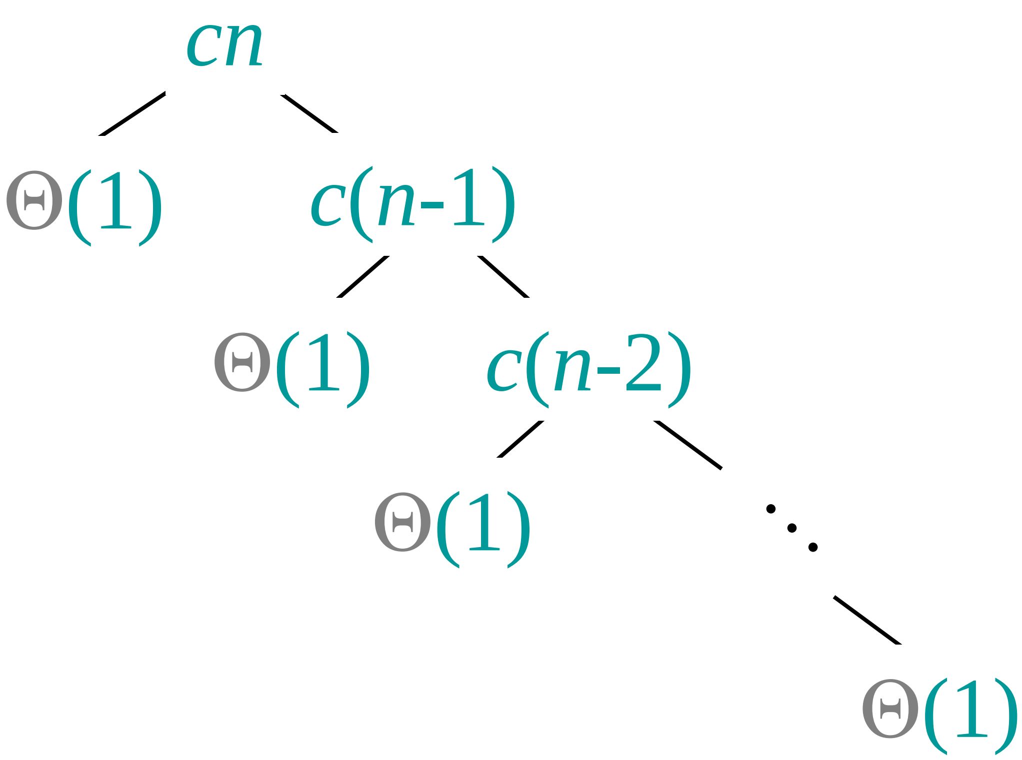
$$cn$$

$$\Theta(1) \qquad c(n-1)$$

$$\Theta(1) \qquad T(n-2)$$

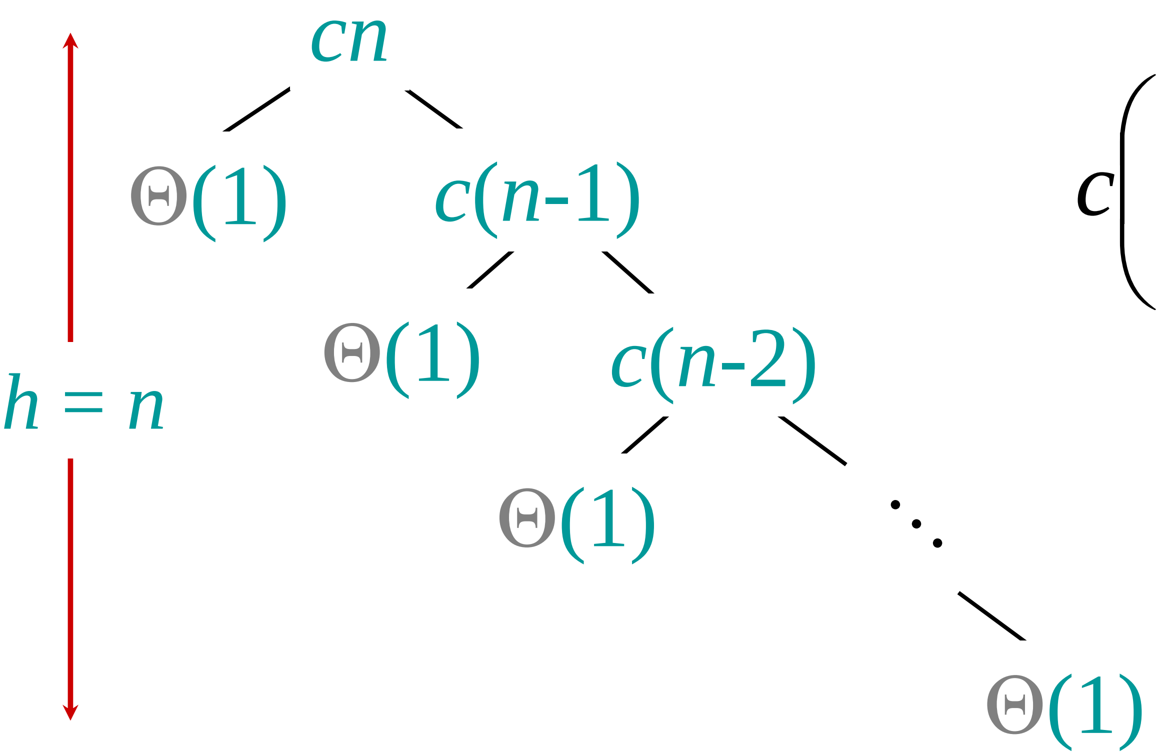
Worst-case recursion tree

$$T(n) = \Theta(1) + T(n-1) + cn$$



Worst-case recursion tree

$$T(n) = \Theta(1) + T(n-1) + cn$$



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

Best-case analysis

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

$T(n) = 2T(n/2) + \Theta(n)$

$= \Theta(n \lg n)$ (..same as merge sort)

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

$T(n) = T(n/10) + T(9n/10) + \Theta(n)$

What is the solution to this recurrence?

Analysis of 'balanced' partitioning

$T(n)$

Analysis of 'balanced' partitioning

$$
\begin{array}{ccc}
 & cn & \\
\swarrow & & \searrow \\
T(n/10) & & T(9n/10)
\end{array}
$$

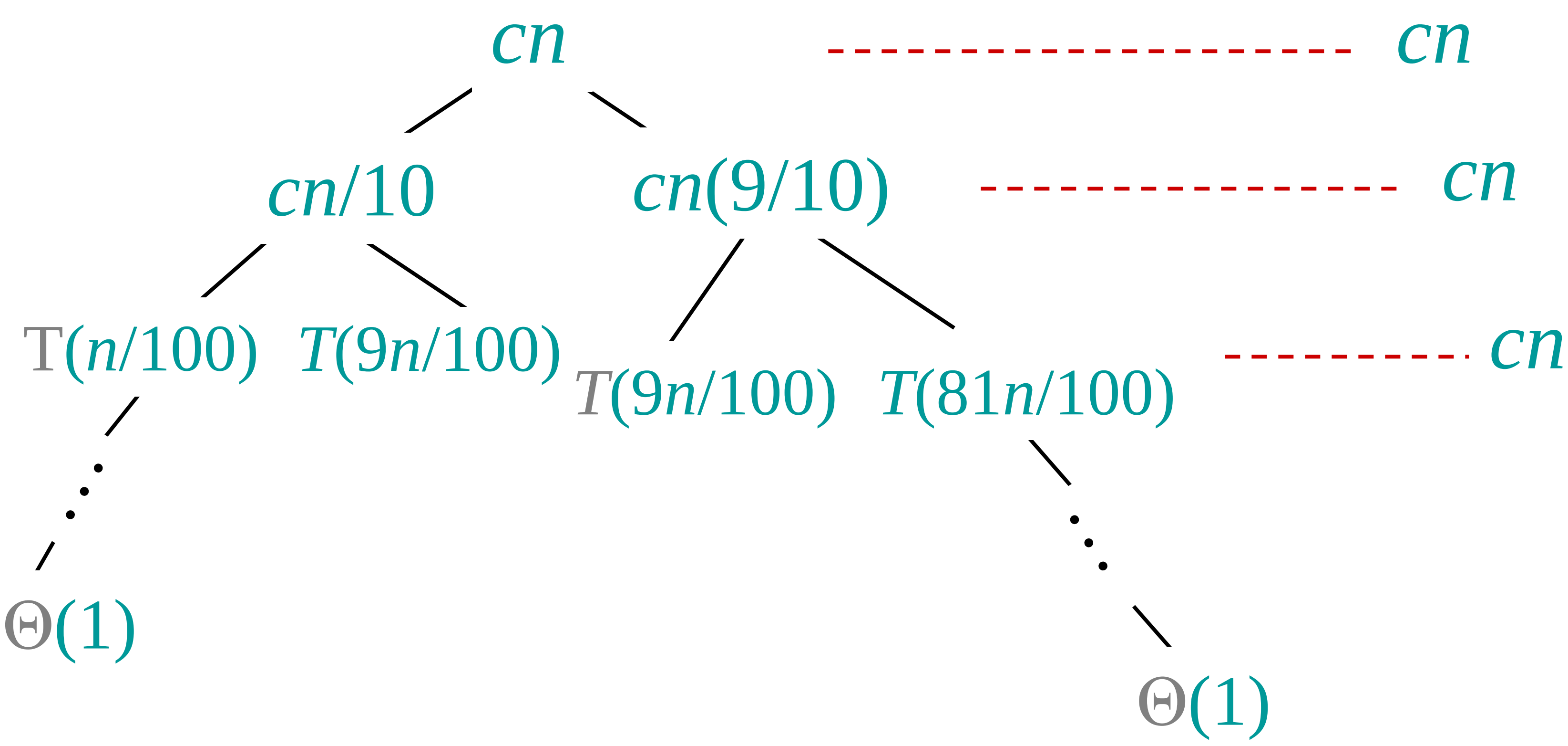
Analysis of 'balanced' partitioning

cn

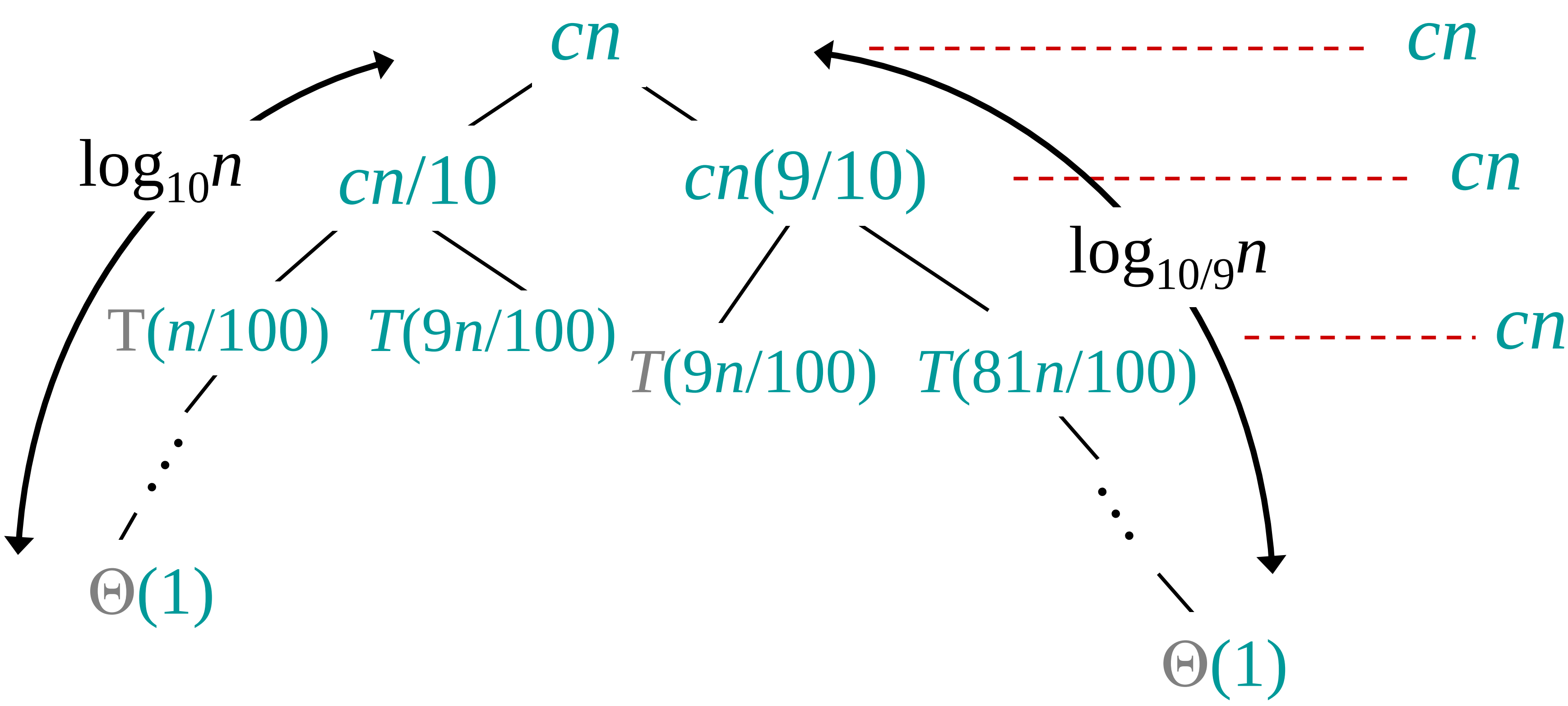
$cn/10$ $\quad$ $cn(9/10)$

$T(n/100)$ $\quad$ $T(9n/100)$ $\quad$ $T(9n/100)$ $\quad$ $T(81n/100)$

Analysis of ‘balanced’ partitioning



Analysis of ‘balanced’ partitioning



$$cn \log_{10} n \le T(n) \le cn \log_{10/9} n$$

$$T(n) \text{ is } \Theta(n \log n)$$

Average-case Quicksort

- We found out that Quicksort has a running time of $\Theta(n \log n)$
- But this is not a proof! The way to prove it is substitution method.
- Base of the logarithm depends on how balanced is the partitioning. In best case, fifty-fifty partitioning results in $\Theta(n \lg n)$.
- Asymptotically, they are all equal.
- The average-case running time of Quicksort is much closer to the best case.

Randomized Quicksort

- How can we be sure that we do not fall into the worst case?
  - ✓ Partition around a random element, or
  - ✓ Permute the ordering randomly, or
  - ✓ Select pivot as the median of randomly selected 3 elements (Median-of-3 partitioning)
- Running time is independent of input order.
- No assumptions need to be made about the input distribution.

Conclusion

- Quicksort is a great general-purpose sorting algorithm.
- Quicksort is typically over twice as fast as merge sort.

- Chapter 7 of the textbook is about Quicksort. Please try to solve exercises in 7.1 7.2 and 7.3.