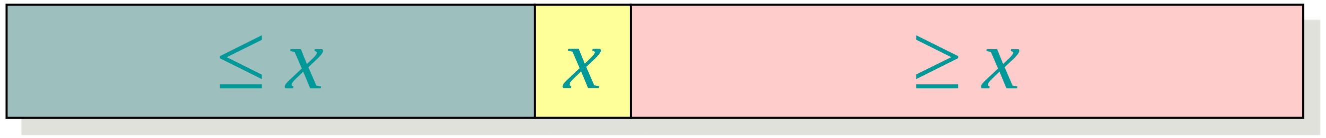
Lecture 4: QuickSort

DIVIDE AND CONQUER

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.

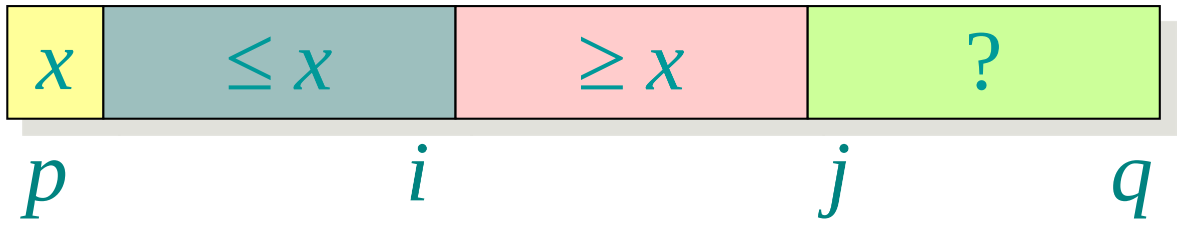
Key: *Linear-time partitioning subroutine.*

PARTITIONING SUBROUTINE

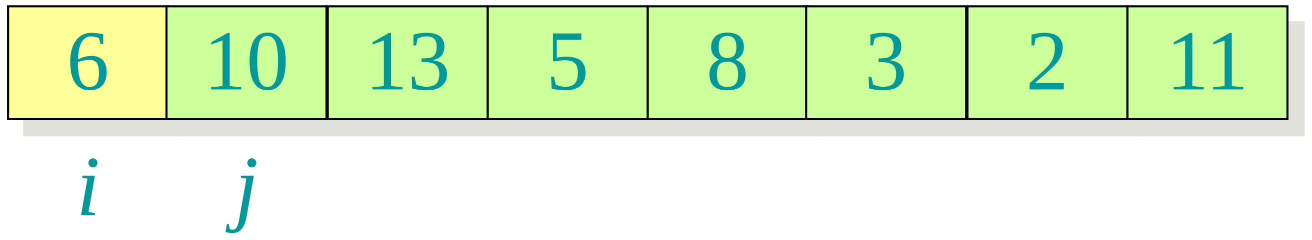
```
PARTITION($A, p, q$)  ⊳ $A[p\,.\,.\,q]$
    $x \leftarrow A[p]$          ⊳ pivot $= A[p]$
    $i \leftarrow p$
    for $j \leftarrow p + 1$ to $q$
        do if $A[j] \leq x$
               then $i \leftarrow i + 1$
                    exchange $A[i] \leftrightarrow A[j]$
    exchange $A[p] \leftrightarrow A[i]$
    return $i$
```

Running time $= O(n)$ for n elements.

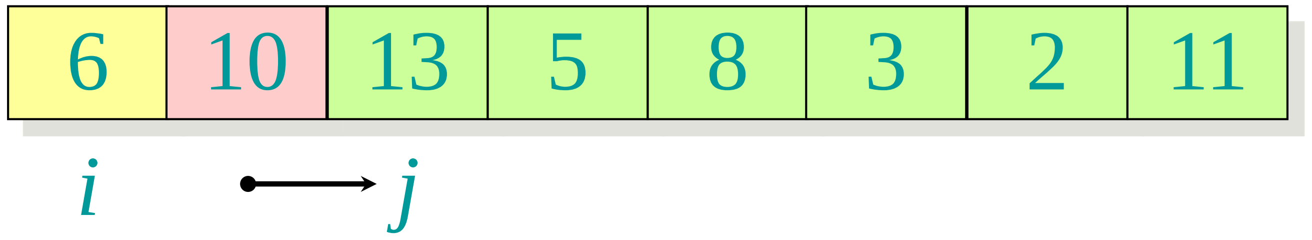
Invariant:



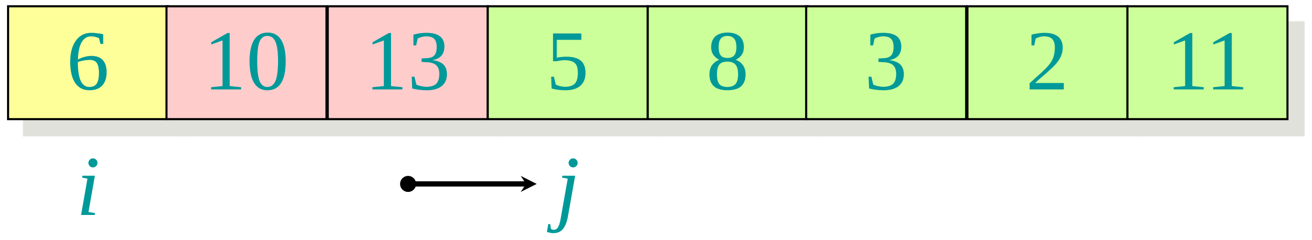
EXAMPLE OF PARTITIONING



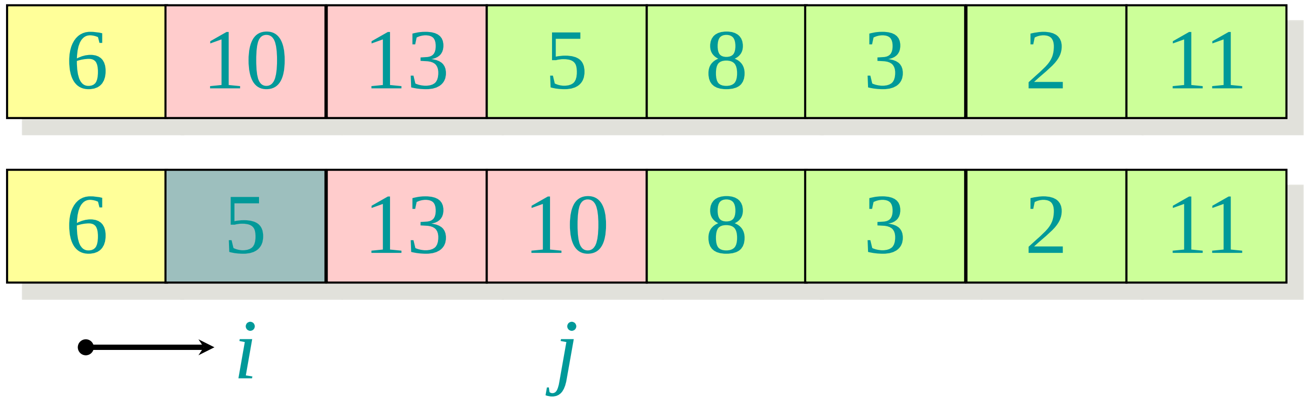
EXAMPLE OF PARTITIONING



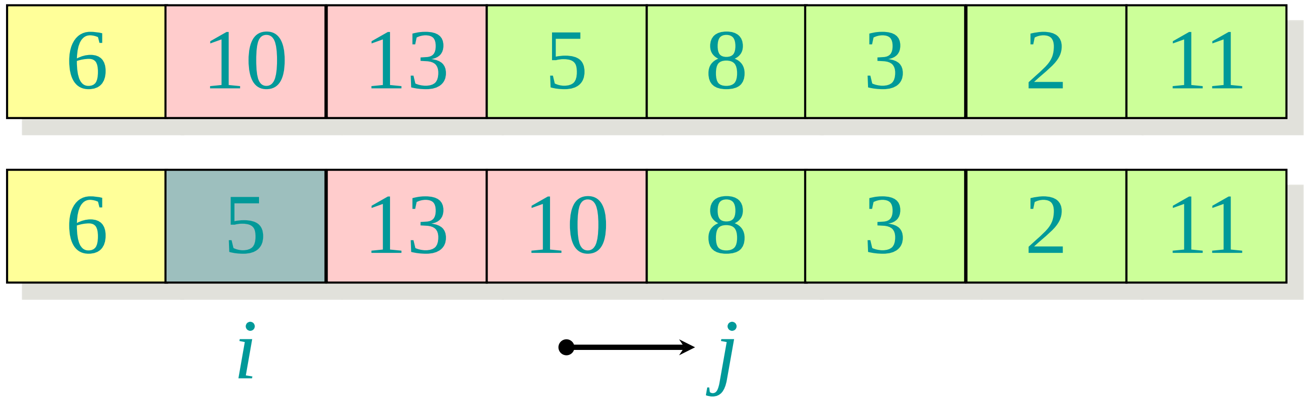
EXAMPLE OF PARTITIONING



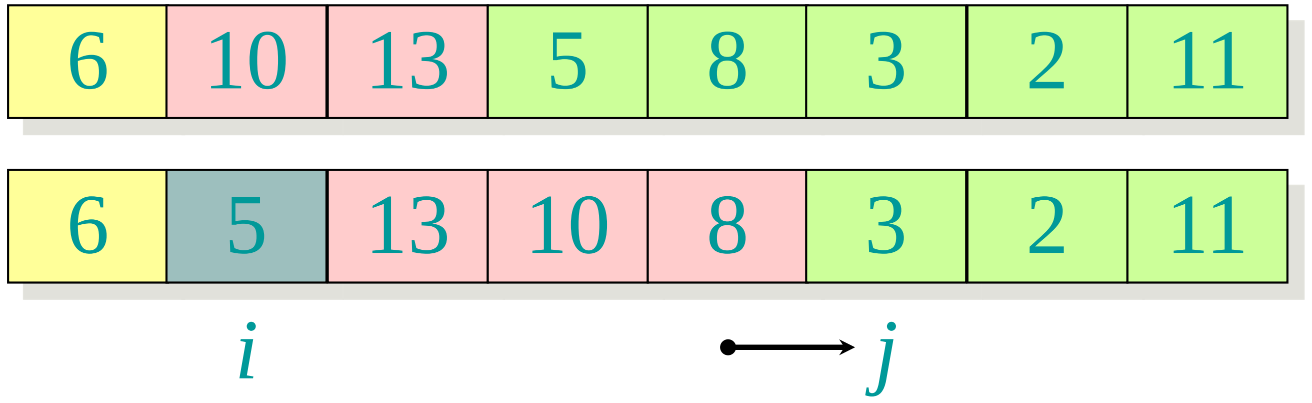
EXAMPLE OF PARTITIONING



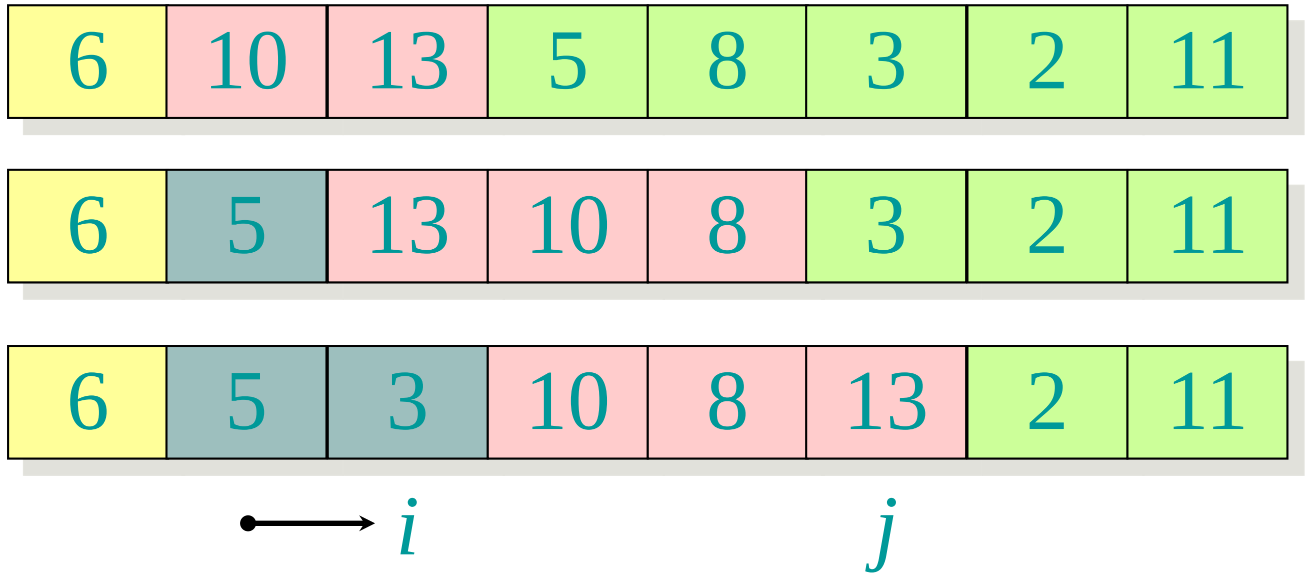
EXAMPLE OF PARTITIONING



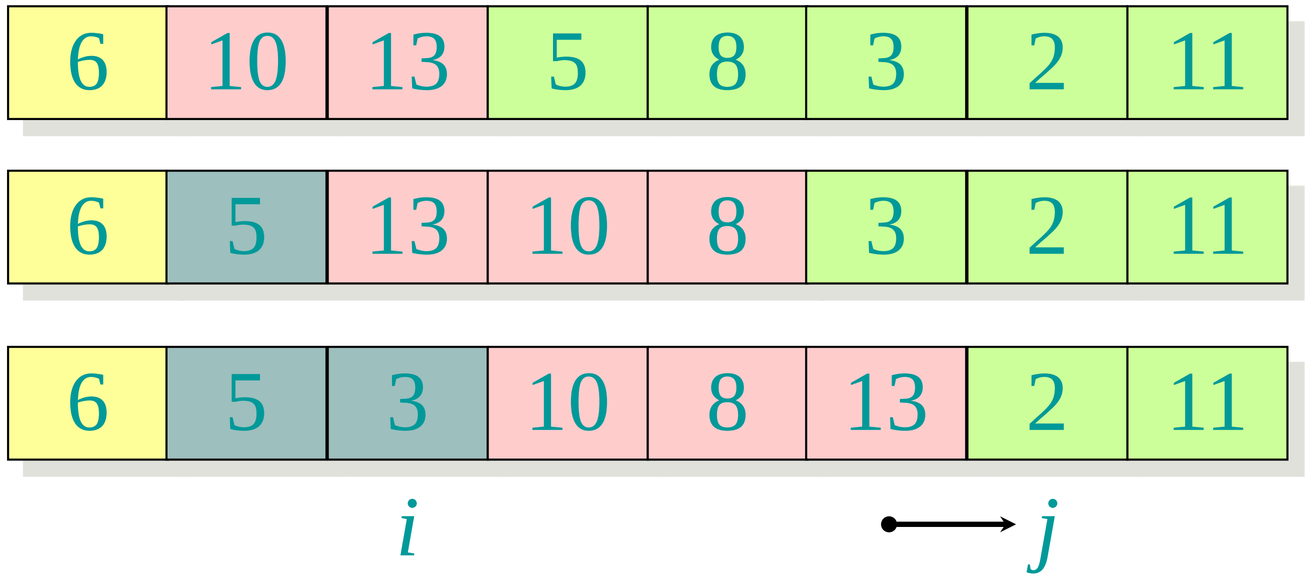
EXAMPLE OF PARTITIONING



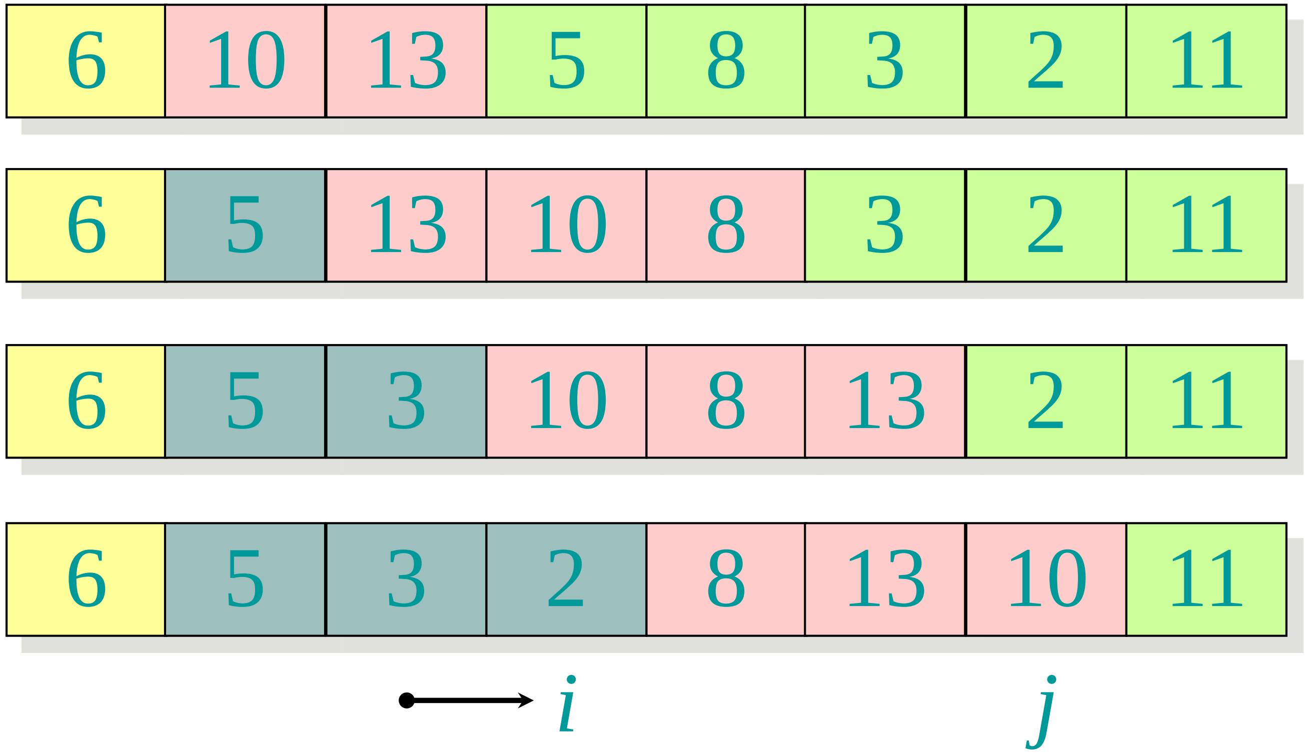
EXAMPLE OF PARTITIONING



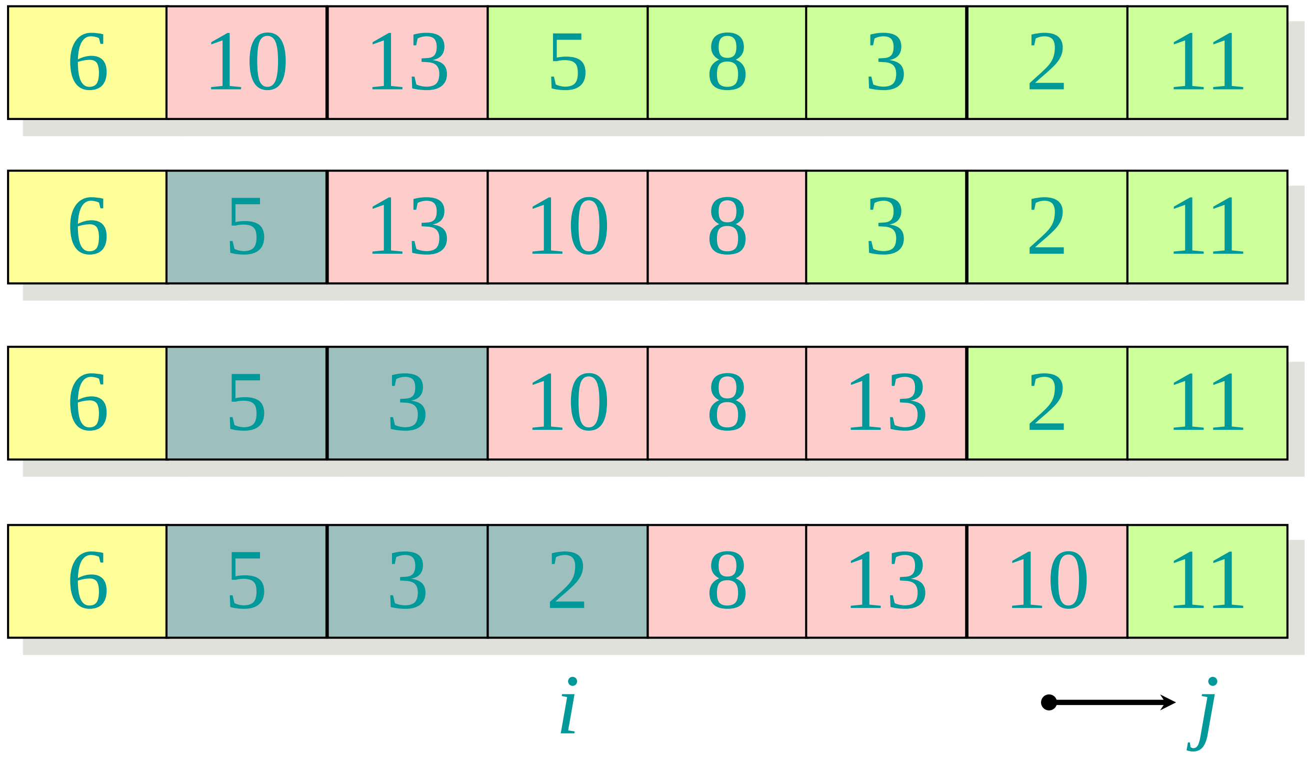
EXAMPLE OF PARTITIONING



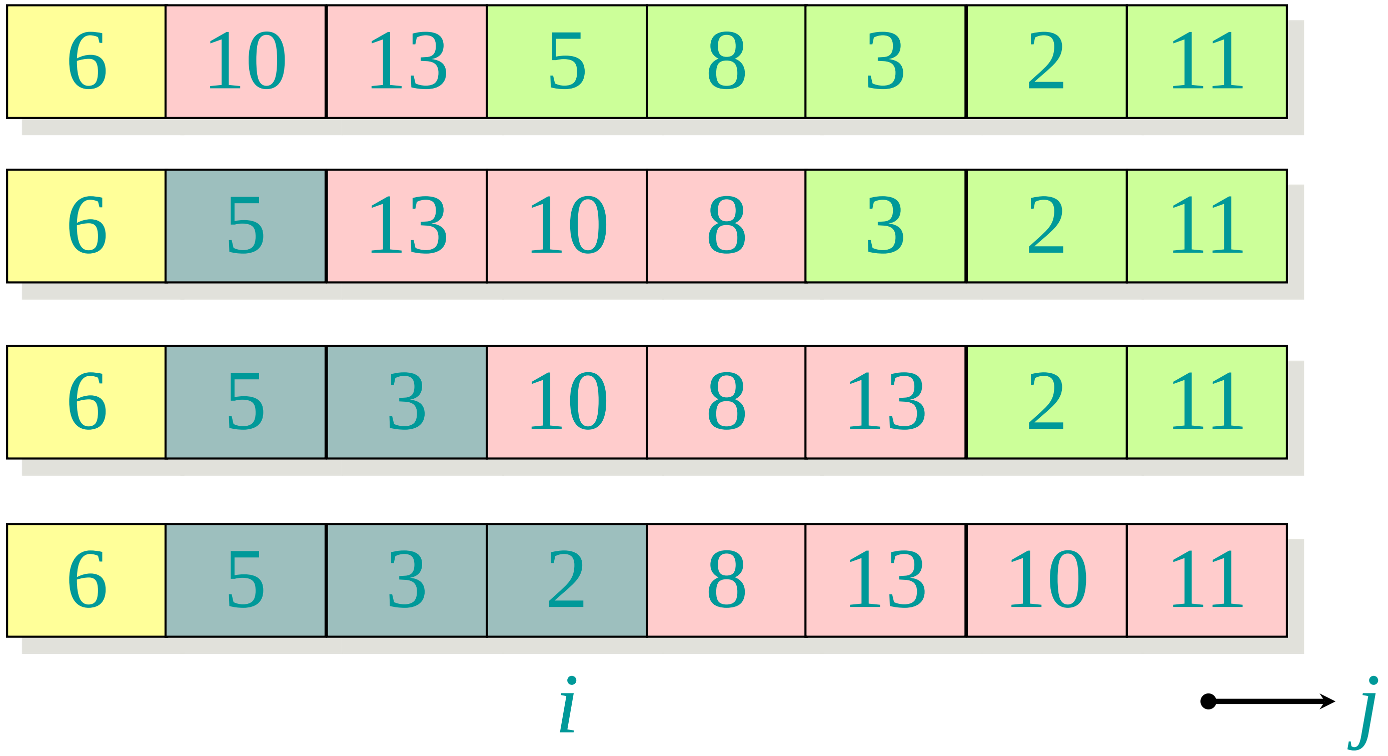
EXAMPLE OF PARTITIONING



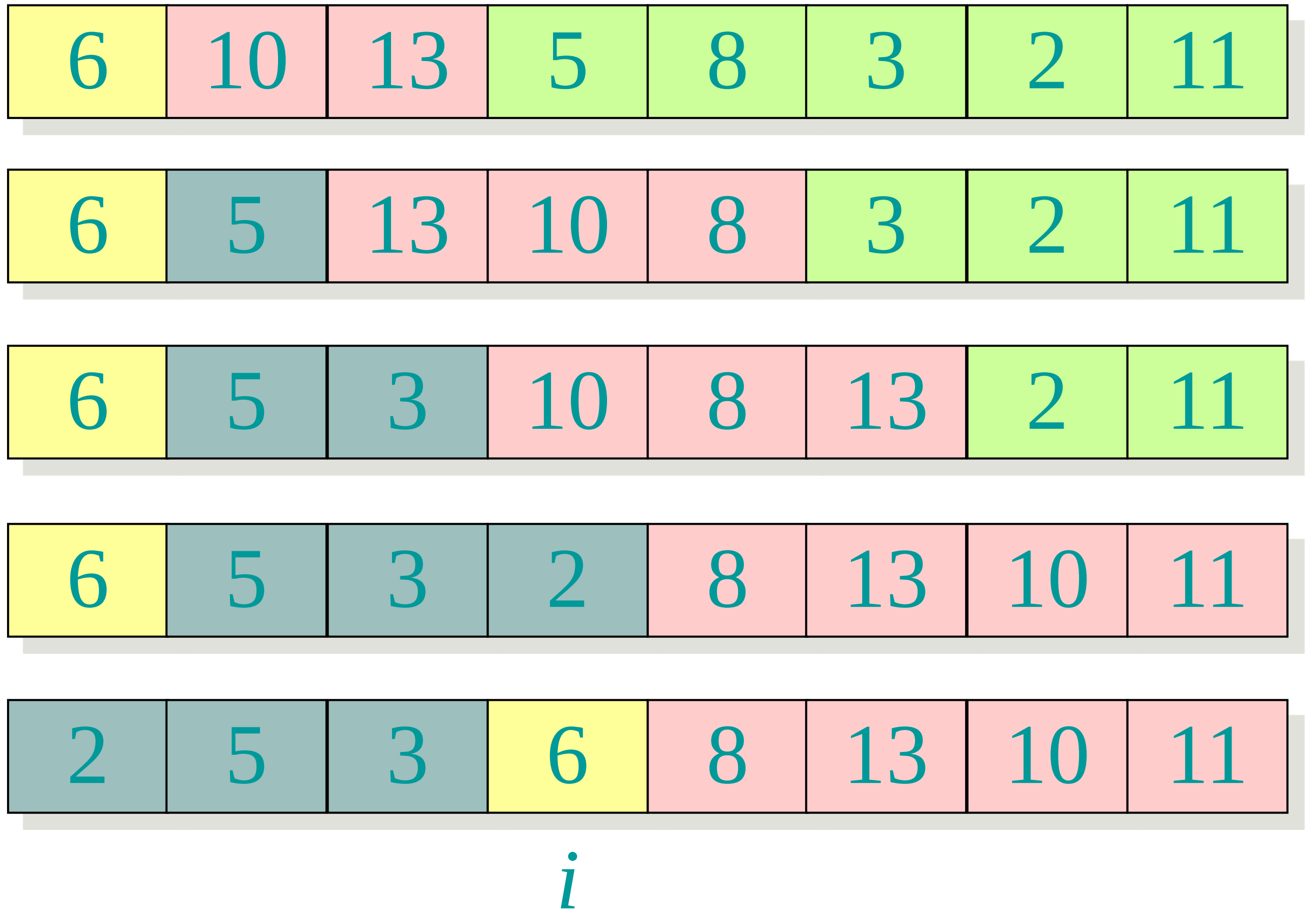
EXAMPLE OF PARTITIONING



EXAMPLE OF PARTITIONING



EXAMPLE OF PARTITIONING



PSEUDOCODE FOR QUICKSORT

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

Initial call: QUICKSORT(A, 1, n)

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

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

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

WORST-CASE RECURSION TREE

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

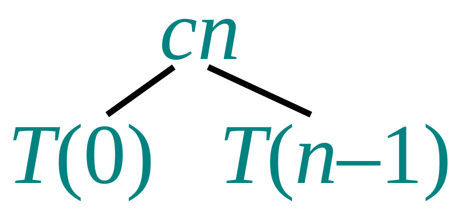
WORST-CASE RECURSION TREE

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

$$T(n)$$

WORST-CASE RECURSION TREE

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



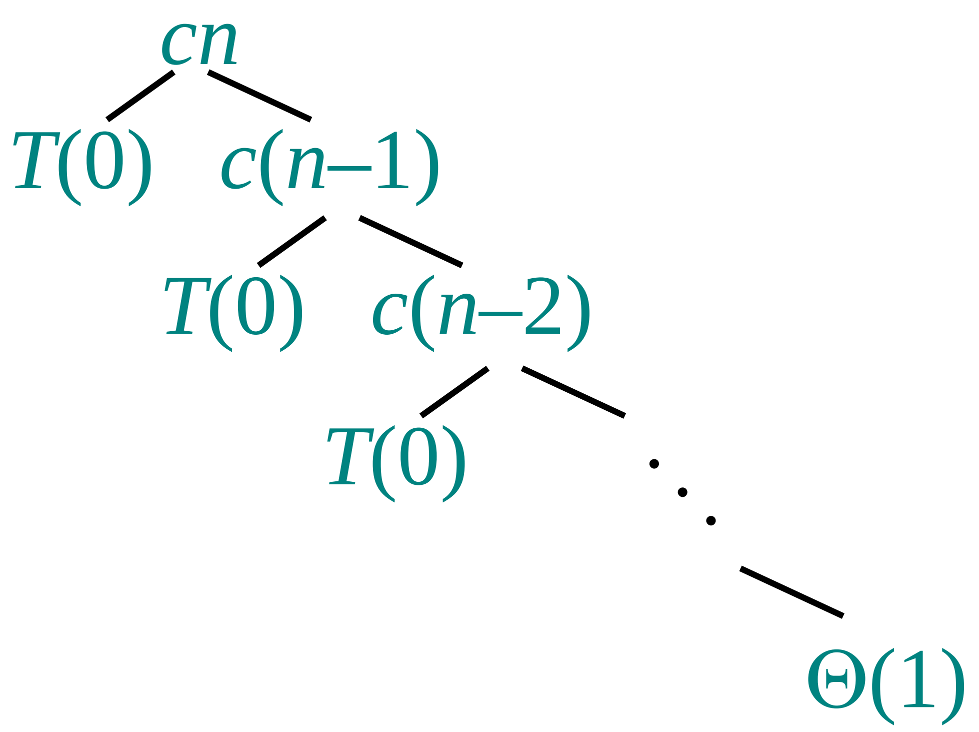
WORST-CASE RECURSION TREE

$$T(n) = T(0) + T(n\text{–}1) + cn$$

$$\begin{tikzcd}
 & cn \arrow[dl, dash] \arrow[dr, dash] & & \\
T(0) & & c(n\text{–}1) \arrow[dl, dash] \arrow[dr, dash] & \\
 & T(0) & & T(n\text{–}2)
\end{tikzcd}$$

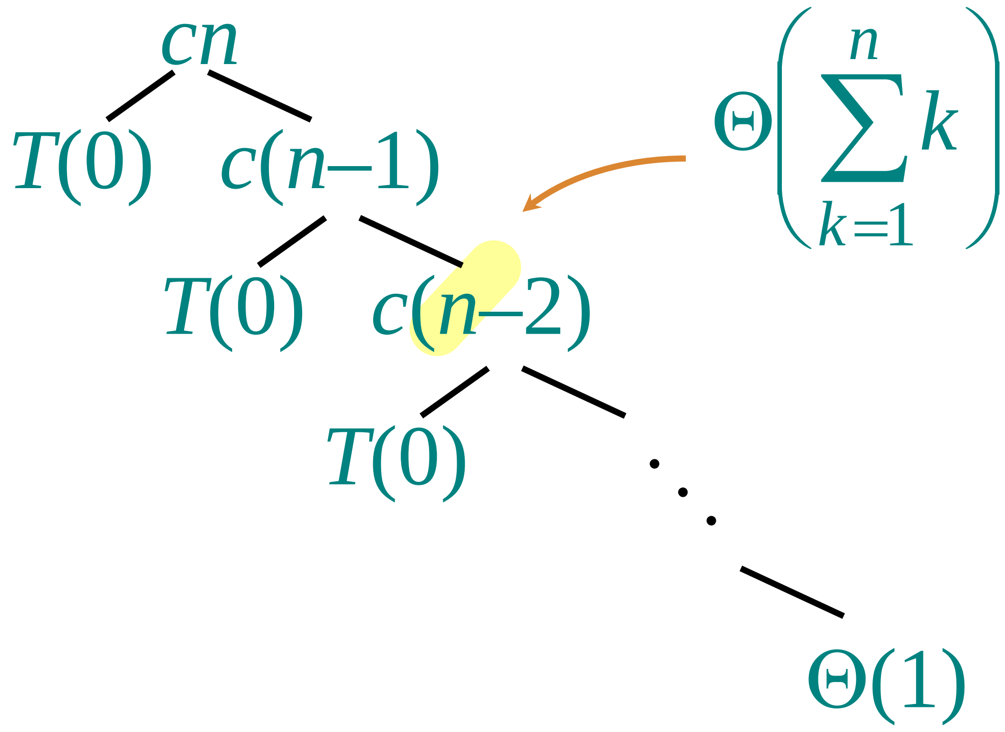
WORST-CASE RECURSION TREE

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



WORST-CASE RECURSION TREE

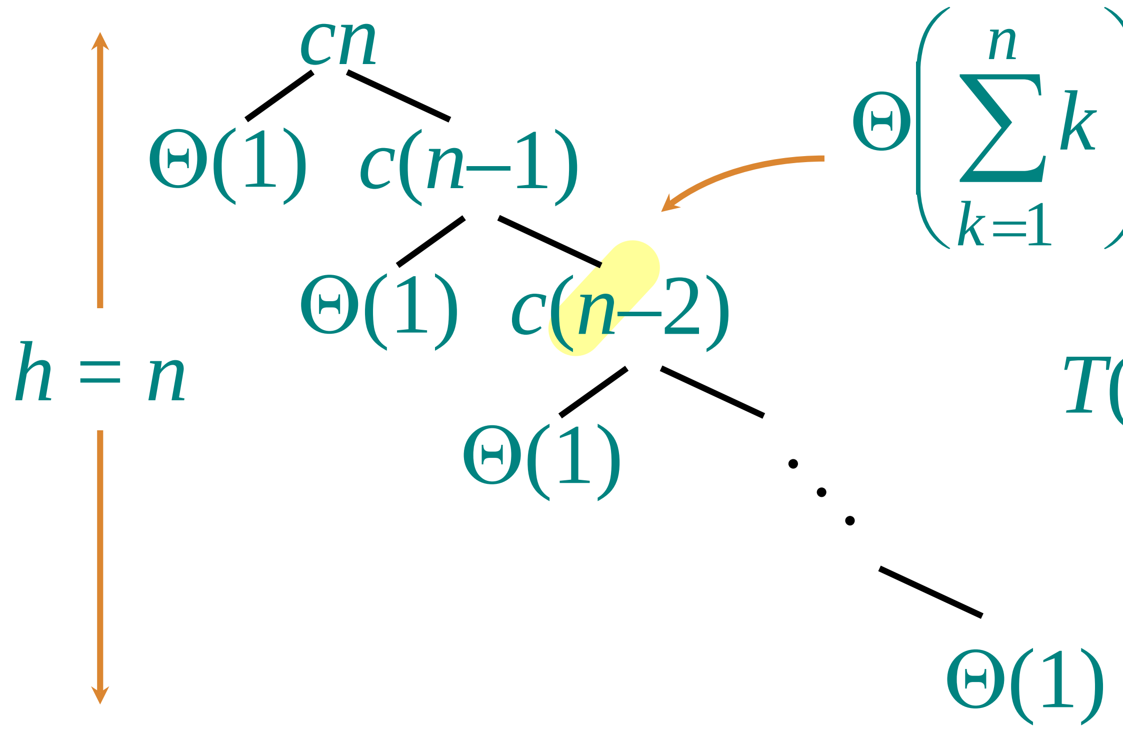
$$T(n) = T(0) + T(n-1) + cn$$



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

WORST-CASE RECURSION TREE

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



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

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

BEST-CASE ANALYSIS

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

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

(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

$$T(n)$$

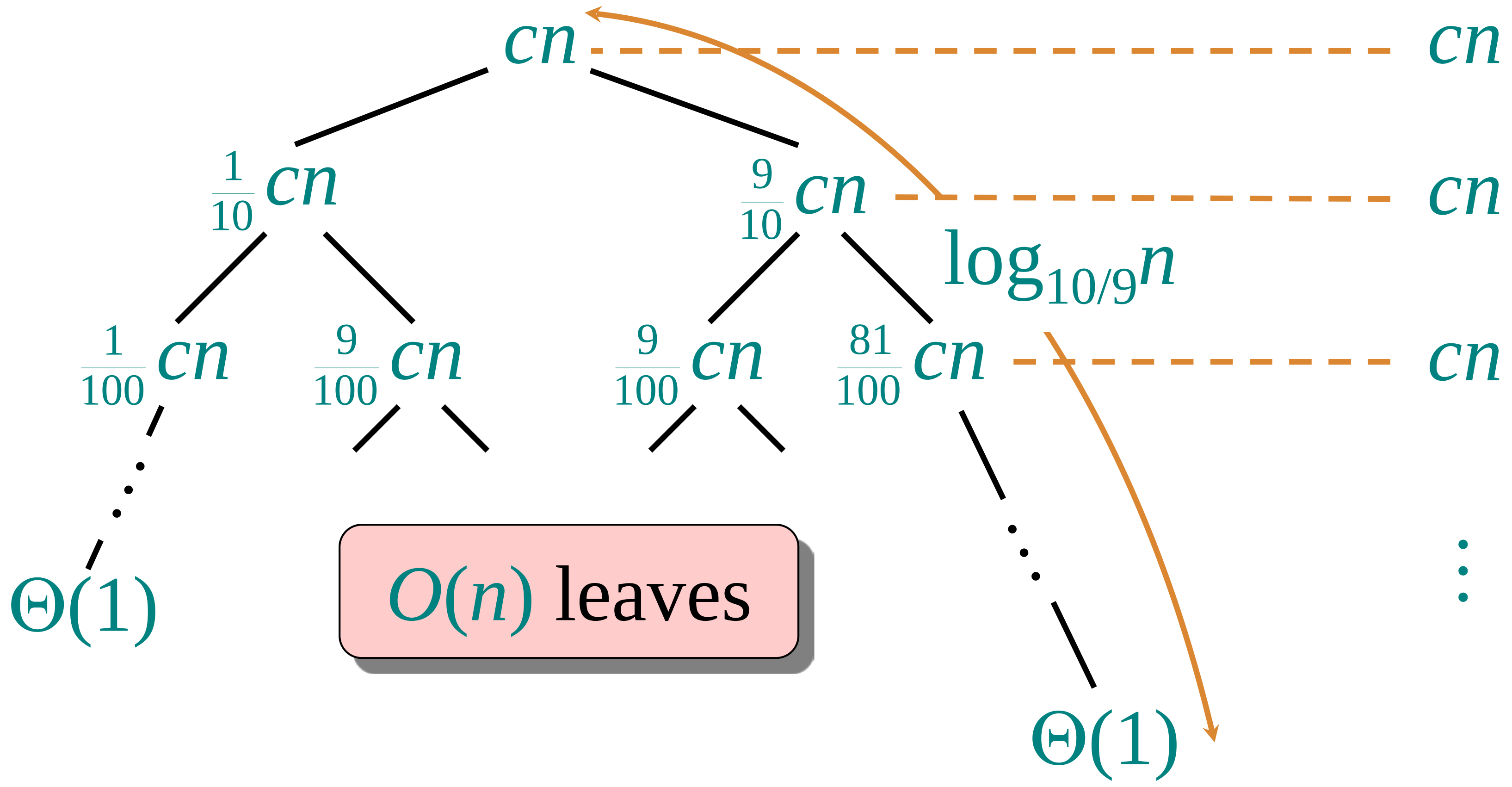
ANALYSIS OF "ALMOST-BEST" CASE

$$\begin{tikzcd}
 & cn \arrow[dl, no head] \arrow[dr, no head] & \\
T\left(\frac{1}{10}n\right) & & T\left(\frac{9}{10}n\right)
\end{tikzcd}$$

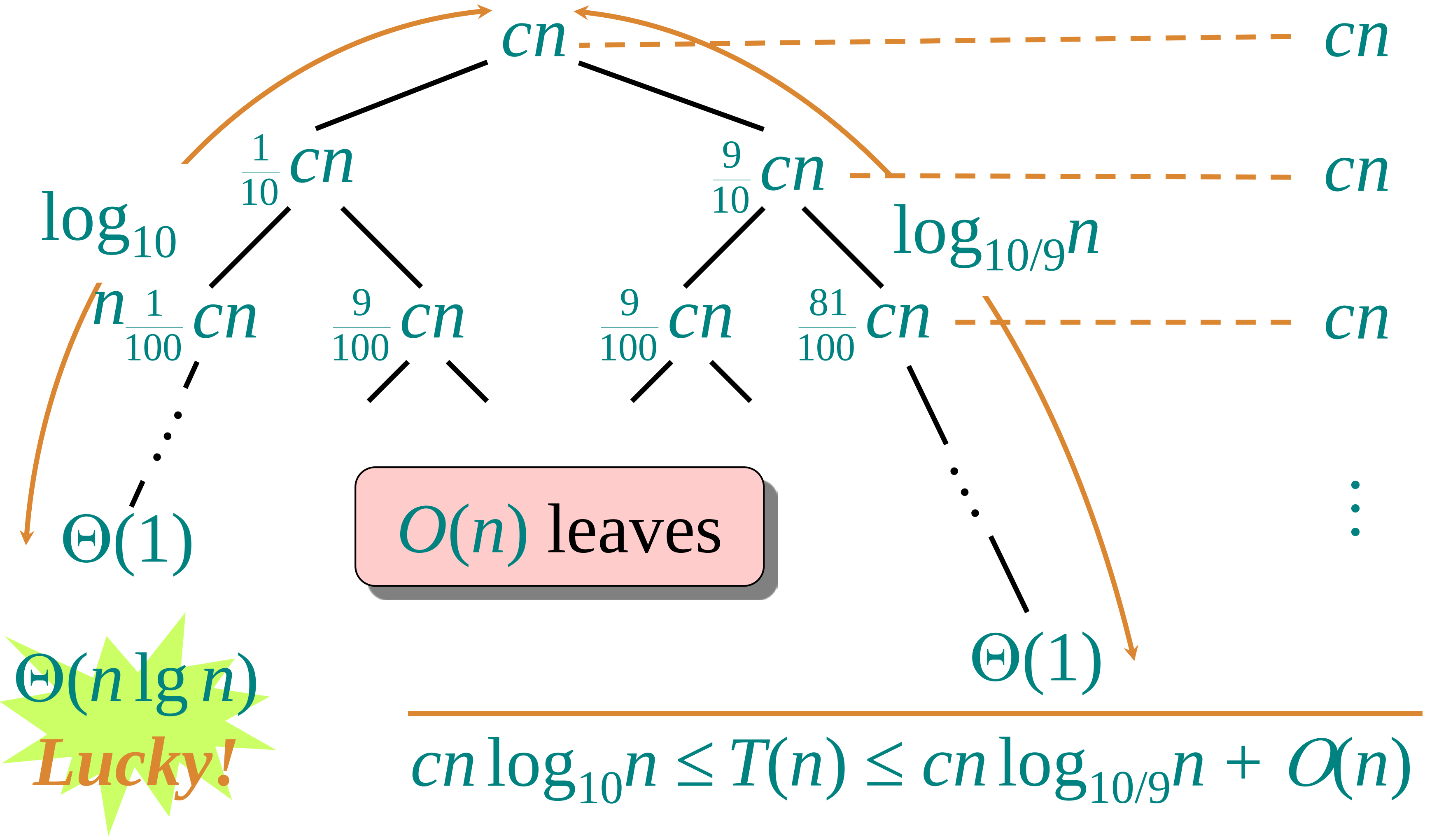
ANALYSIS OF “ALMOST-BEST” CASE

$$\begin{tikzcd}
 & & & cn \arrow[dll, dash] \arrow[drr, dash] & & & \\
 & \frac{1}{10}cn \arrow[dl, dash] \arrow[dr, dash] & & & & \frac{9}{10}cn \arrow[dl, dash] \arrow[dr, dash] & \\
T\left(\frac{1}{100}n\right) & & T\left(\frac{9}{100}n\right) & & T\left(\frac{9}{100}n\right) & & T\left(\frac{81}{100}n\right)
\end{tikzcd}$$

ANALYSIS OF "ALMOST-BEST" CASE



Analysis of "Almost-Best" Case



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

MORE INTUITION

Suppose we alternate lucky, unlucky, lucky, unlucky, lucky, ….

$$L(n) = 2U(n/2) + \Theta(n) \quad \textit{lucky}$$

$$U(n) = L(n-1) + \Theta(n) \quad \textit{unlucky}$$

Solving:

$$\begin{aligned} L(n) &= 2(L(n/2-1) + \Theta(n/2)) + \Theta(n) \\ &= 2L(n/2-1) + \Theta(n) \\ &= \Theta(n \lg n) \end{aligned}$$

Lucky!

How can we make sure we are usually lucky?

RANDOMIZED QUICKSORT

IDEA: Partition around a ***random*** element.

- Running time is independent of the input order.
- No assumptions need to be made about the input distribution.
- No specific input elicits the worst-case behavior.
- The worst case is determined only by the output of a random-number generator.

RANDOMIZED QUICKSORT ANALYSIS

Let $T(n)$ = the random variable for the running time of randomized quicksort on an input of size n, assuming random numbers are independent.

For $k = 0, 1, \ldots, n–1$, define the ***indicator random variable***

$$X_k = \begin{cases} 1 & \text{if } \textsc{Partition} \text{ generates a } k : n–k–1 \text{ split,} \\ 0 & \text{otherwise.} \end{cases}$$

$E[X_k] = \Pr\{X_k = 1\} = 1/n$, since all splits are equally likely, assuming elements are distinct.

ANALYSIS (CONTINUED)

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

$$= \sum_{k=0}^{n-1} X_k \left(T(k) + T(n-k-1) + \Theta(n)\right).$$

CALCULATING EXPECTATION

$$E[T(n)] = E\left[\sum_{k=0}^{n-1} X_k \left(T(k) + T(n-k-1) + \Theta(n)\right)\right]$$

Take expectations of both sides.

CALCULATING EXPECTATION

$$E[T(n)] = E\left[\sum_{k=0}^{n-1} X_k\big(T(k) + T(n-k-1) + \Theta(n)\big)\right]$$

$$= \sum_{k=0}^{n-1} E\big[X_k\big(T(k) + T(n-k-1) + \Theta(n)\big)\big]$$

Linearity of expectation.

CALCULATING EXPECTATION

$$E[T(n)] = E\left[\sum_{k=0}^{n-1} X_k\left(T(k) + T(n-k-1) + \Theta(n)\right)\right]$$

$$= \sum_{k=0}^{n-1} E\left[X_k\left(T(k) + T(n-k-1) + \Theta(n)\right)\right]$$

$$= \sum_{k=0}^{n-1} E[X_k] \cdot E[T(k) + T(n-k-1) + \Theta(n)]$$

Independence of X_k from other random choices.

CALCULATING EXPECTATION

$$E[T(n)] = E\left[\sum_{k=0}^{n-1} X_k\left(T(k) + T(n-k-1) + \Theta(n)\right)\right]$$

$$= \sum_{k=0}^{n-1} E\left[X_k\left(T(k) + T(n-k-1) + \Theta(n)\right)\right]$$

$$= \sum_{k=0}^{n-1} E[X_k] \cdot E[T(k) + T(n-k-1) + \Theta(n)]$$

$$= \frac{1}{n}\sum_{k=0}^{n-1} E[T(k)] + \frac{1}{n}\sum_{k=0}^{n-1} E[T(n-k-1)] + \frac{1}{n}\sum_{k=0}^{n-1} \Theta(n)$$

Linearity of expectation; $E[X_k] = 1/n$.

CALCULATING EXPECTATION

$$
\begin{aligned}
E[T(n)] &= E\left[\sum_{k=0}^{n-1} X_k\big(T(k)+T(n-k-1)+\Theta(n)\big)\right] \\
&= \sum_{k=0}^{n-1} E\left[X_k\big(T(k)+T(n-k-1)+\Theta(n)\big)\right] \\
&= \sum_{k=0}^{n-1} E[X_k]\cdot E[T(k)+T(n-k-1)+\Theta(n)] \\
&= \frac{1}{n}\sum_{k=0}^{n-1} E[T(k)] + \frac{1}{n}\sum_{k=0}^{n-1} E[T(n-k-1)] + \frac{1}{n}\sum_{k=0}^{n-1} \Theta(n) \\
&= \frac{2}{n}\sum_{k=1}^{n-1} E[T(k)] + \Theta(n)
\end{aligned}
$$

Summations have identical terms.

HAIRY RECURRENCE

$$E[T(n)] = \frac{2}{n}\sum_{k=2}^{n-1} E[T(k)] + \Theta(n)$$

(The $k = 0, 1$ terms can be absorbed in the $\Theta(n)$.)

Prove: $E[T(n)] \le a n \lg n$ for constant $a > 0$.

- Choose a large enough so that $a n \lg n$ dominates $E[T(n)]$ for sufficiently small $n \ge 2$.

Use fact: $\sum_{k=2}^{n-1} k \lg k \le \frac{1}{2} n^2 \lg n - \frac{1}{8} n^2$ (exercise).

SUBSTITUTION METHOD

$$E[T(n)] \le \frac{2}{n}\sum_{k=2}^{n-1} ak\lg k + \Theta(n)$$

Substitute inductive hypothesis.

SUBSTITUTION METHOD

$$
\begin{aligned}
E[T(n)] &\leq \frac{2}{n}\sum_{k=2}^{n-1} ak\lg k + \Theta(n) \\
&\leq \frac{2a}{n}\left(\frac{1}{2}n^2\lg n - \frac{1}{8}n^2\right) + \Theta(n)
\end{aligned}
$$

Use fact.

SUBSTITUTION METHOD

$$E[T(n)] \le \frac{2}{n}\sum_{k=2}^{n-1} ak\lg k + \Theta(n)$$

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

$$= an\lg n - \left(\frac{an}{4} - \Theta(n)\right)$$

Express as ***desired* – *residual***.

SUBSTITUTION METHOD

$$
\begin{aligned}
E[T(n)] &\le \frac{2}{n}\sum_{k=2}^{n-1} ak\lg k + \Theta(n) \\
&= \frac{2a}{n}\left(\frac{1}{2}n^2\lg n - \frac{1}{8}n^2\right) + \Theta(n) \\
&= an\lg n - \left(\frac{an}{4} - \Theta(n)\right) \\
&\le an\lg n\,,
\end{aligned}
$$

if a is chosen large enough so that $an/4$ dominates the $\Theta(n)$.

QUICKSORT IN PRACTICE

- Quicksort is a great general-purpose sorting algorithm.
- Quicksort is typically over twice as fast as merge sort.
- Quicksort can benefit substantially from ***code tuning***.
- Quicksort behaves well even with caching and virtual memory.