

CS473 - Algorithms I

Lecture 6-a Analysis of Quicksort

View in slide-show mode

Analysis of Quicksort

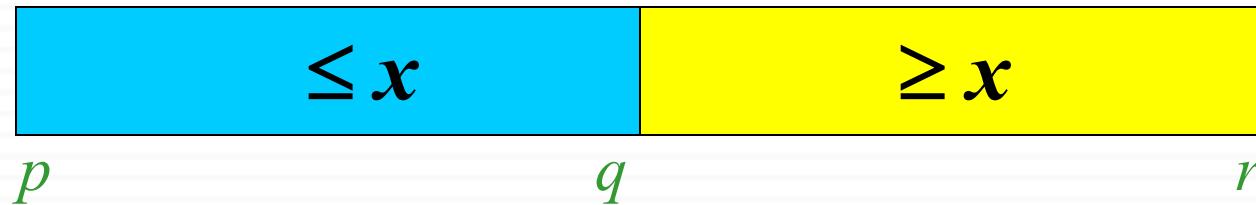
QUICKSORT (A, p, r)

if $p < r$ then

$q \leftarrow \text{H-PARTITION}(A, p, r)$

QUICKSORT(A, p, q)

QUICKSORT($A, q + 1, r$)



Assume *all elements are distinct* in the following analysis

Question

QUICKSORT(A, p, r)

if $p < r$ then

$q \leftarrow \text{H-PARTITION}(A, p, r)$

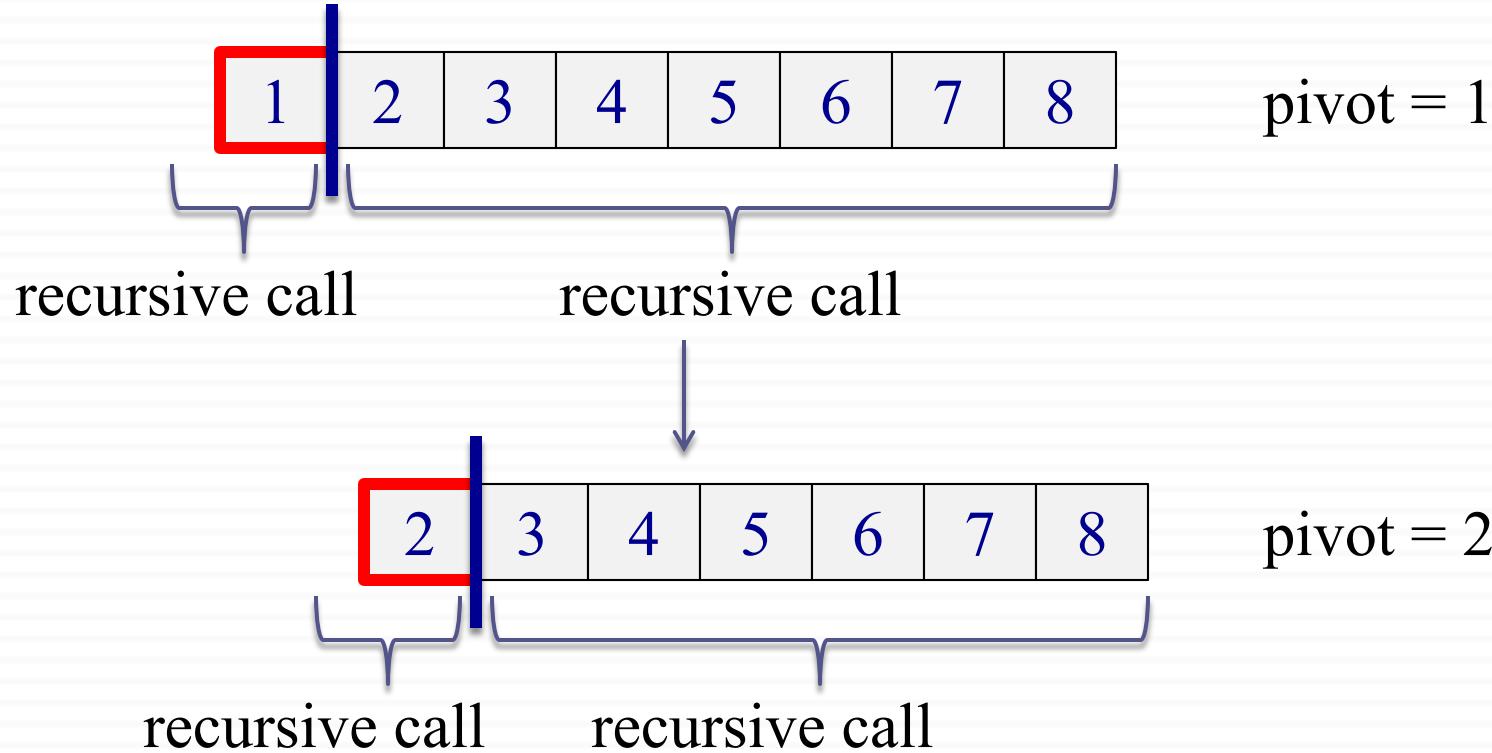
QUICKSORT(A, p, q)

QUICKSORT(A, q +1, r)

Q: Remember that **H-PARTITION** always chooses A[p] (*the first element*) as the **pivot**. What is the runtime of **QUICKSORT** on an already-sorted array?

- ✗ a) $\Theta(n)$
- ✓ c) $\Theta(n^2)$
- ✗ b) $\Theta(n\log n)$
- ✗ d) cannot provide a tight bound

Example: An Already Sorted Array



Partitioning always leads to 2 parts of size 1 and $n-1$

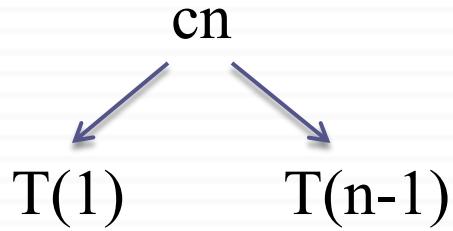
Worst Case Analysis of Quicksort

- Worst case is when the PARTITION algorithm always returns imbalanced partitions (*of size 1 and n-1*) in every recursive call
 - This happens when the pivot is selected to be either the **min** or **max** element.
 - This happens for H-PARTITION when the input array is already sorted or reverse sorted

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

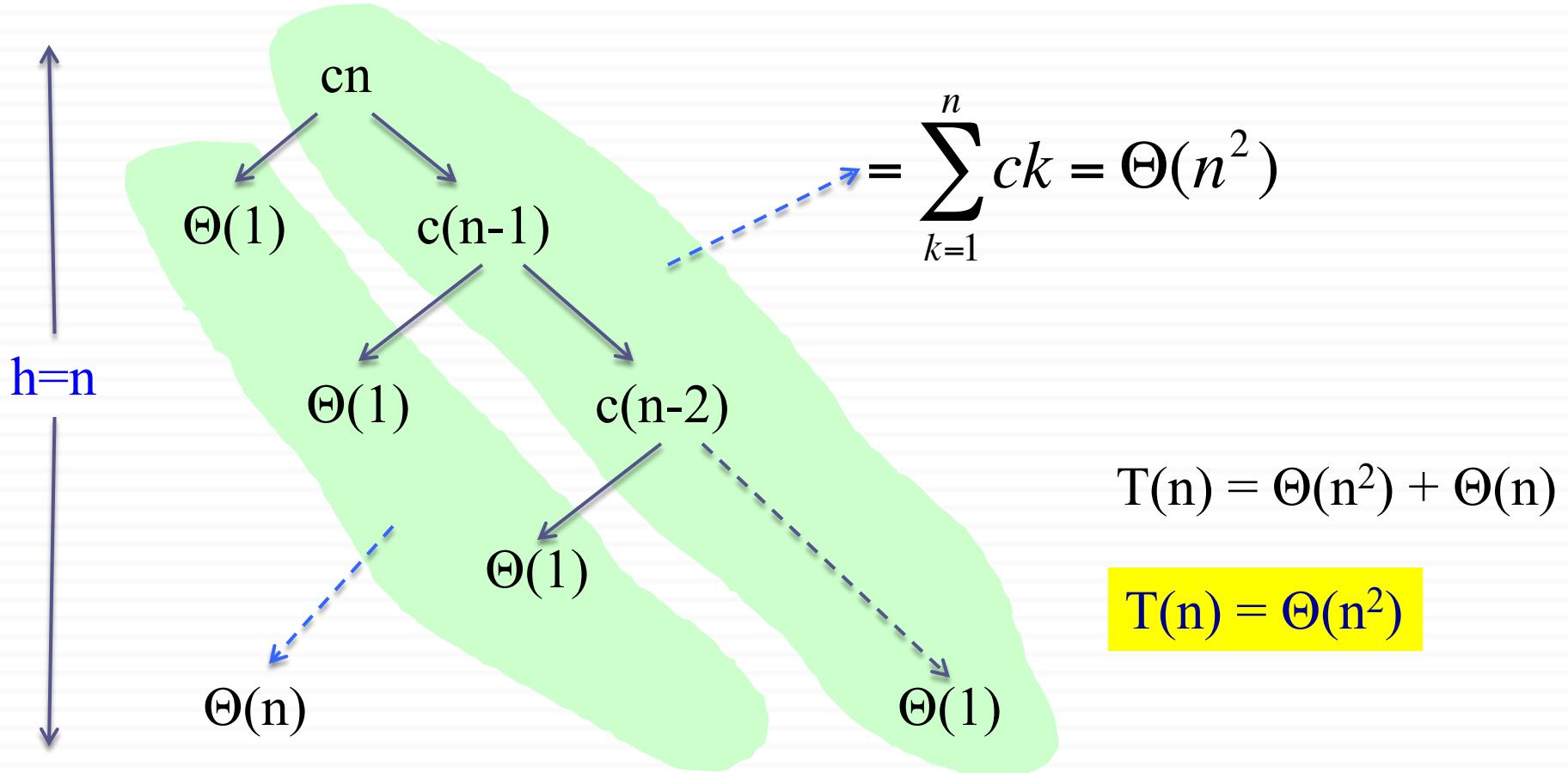
Worst Case Recursion Tree

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



Worst Case Recursion Tree

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



Best Case Analysis (for intuition only)

- If we're *extremely lucky*, H-PARTITION splits the array *evenly* at *every* recursive call

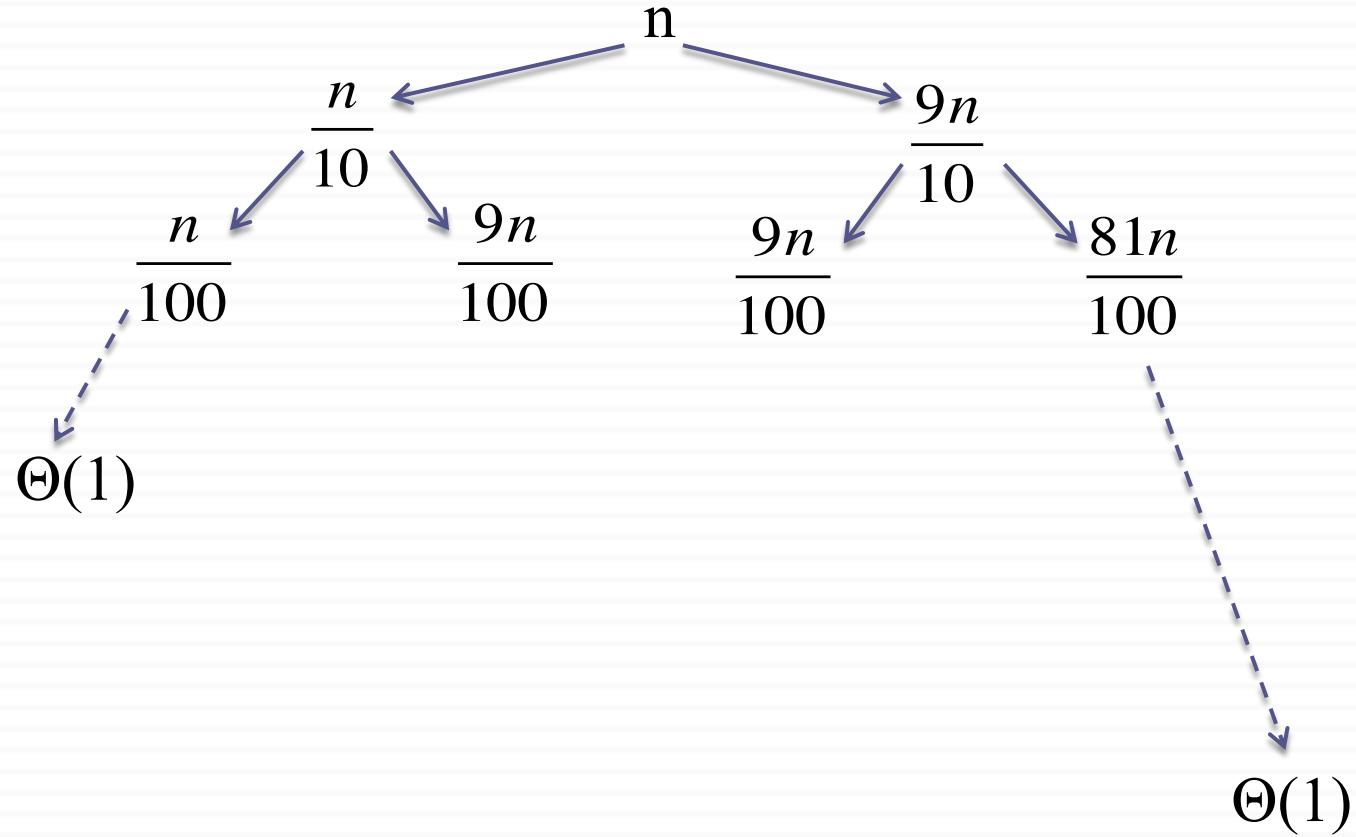
$$\begin{aligned} T(n) &= 2 T(n/2) + \Theta(n) \\ &= \Theta(n \lg n) \end{aligned}$$

→ same as merge sort

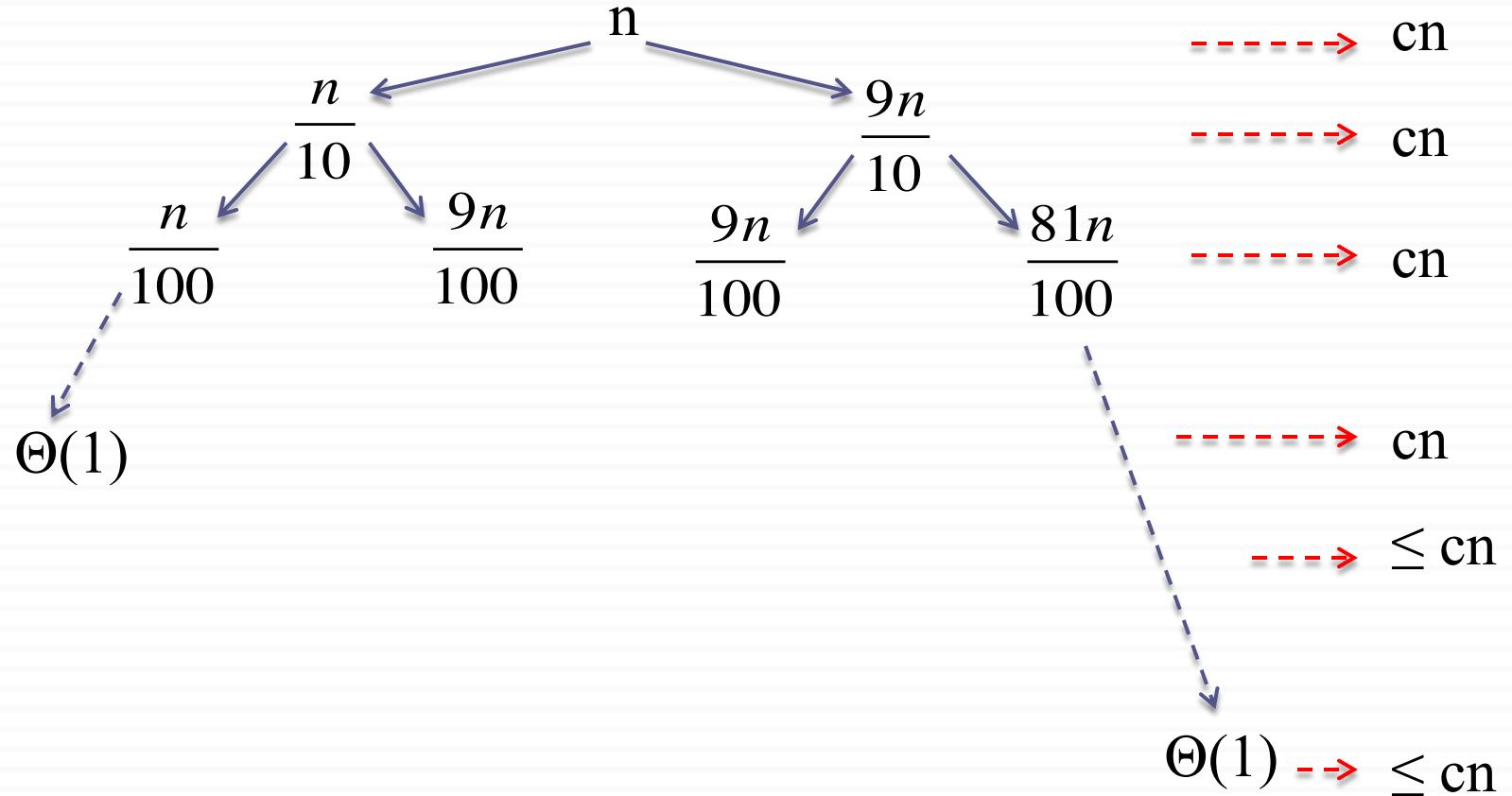
- Instead of splitting 0.5:0.5, what if every split is 0.1:0.9?

$$\begin{aligned} T(n) &= T(n/10) + T(9n/10) + \Theta(n) \\ \rightarrow &\text{ solve this recurrence} \end{aligned}$$

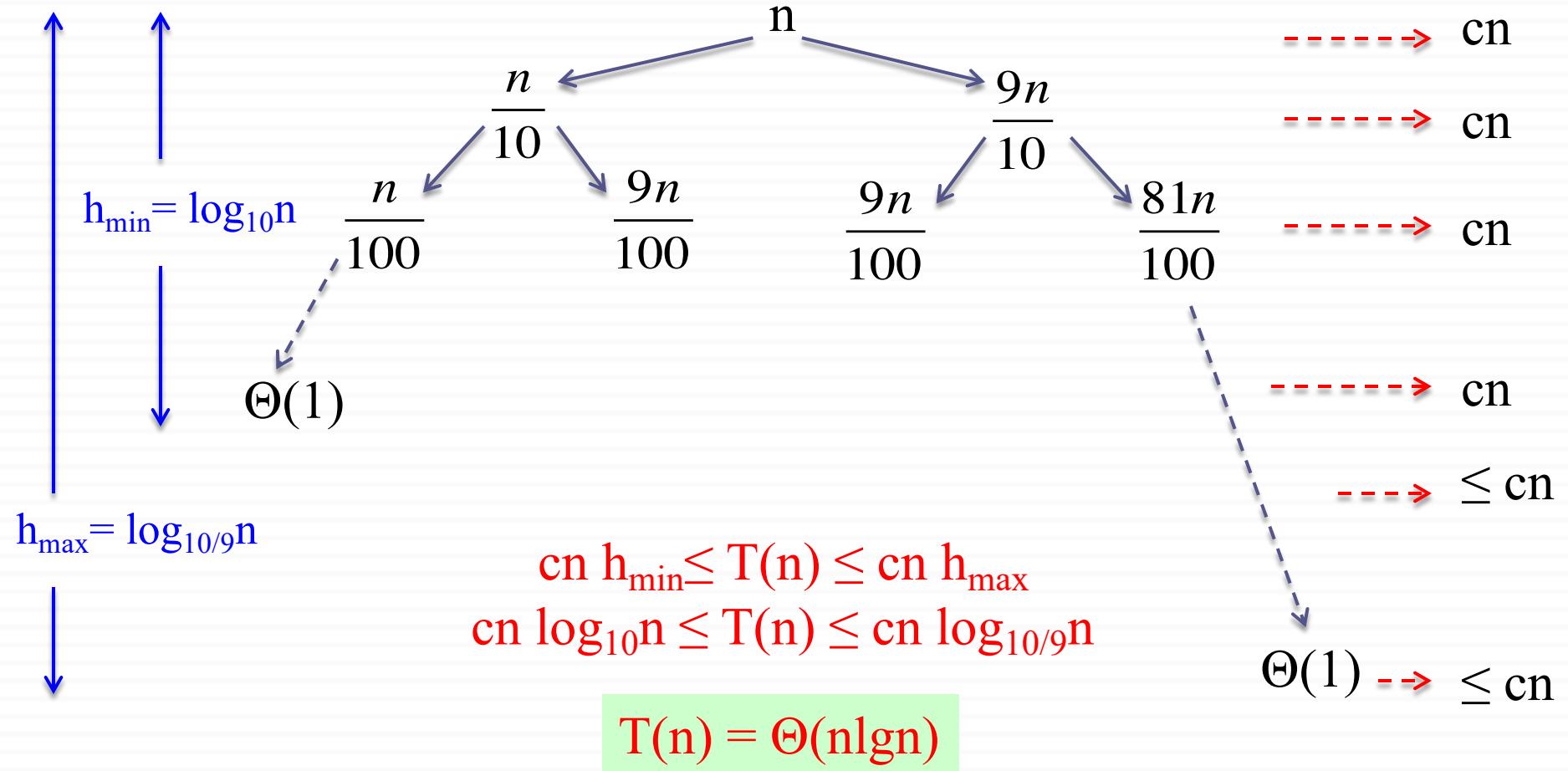
“Almost-Best” Case Analysis



“Almost-Best” Case Analysis



“Almost-Best” Case Analysis



Balanced Partitioning

- We have seen that if H-PARTITION always splits the array with 0.1-to-0.9 ratio, the runtime will be $\Theta(n \lg n)$.
- Same is true with a split ratio of 0.01-to-0.99, etc.
- Possible to show that if the split has always constant ($\Theta(1)$) proportionality, then the runtime will be $\Theta(n \lg n)$.
- In other words, for a constant α ($0 < \alpha \leq 0.5$):
 α -to-(1- α) proportional split yields $\Theta(n \lg n)$ total runtime

Balanced Partitioning

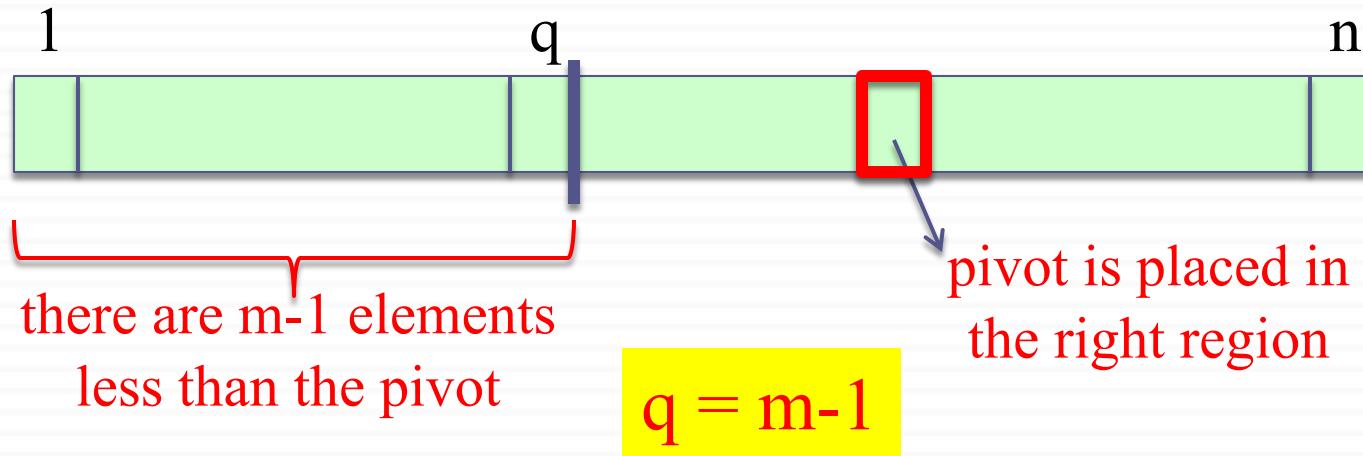
- In the rest of the analysis, assume that *all input permutations* are **equally likely**.
 - ▣ This is only to gain some intuition
 - ▣ We cannot make this assumption for average case analysis
 - ▣ We will revisit this assumption later
- Also, assume that **all input elements are distinct**.

- What is the probability that **H-PARTITION** returns a split that is more balanced than **0.1-to-0.9**?

Balanced Partitioning

Reminder: *H-PARTITION* will place the pivot in the right partition unless the pivot is the smallest element in the arrays.

Question: If the pivot selected is the m^{th} smallest value ($1 < m \leq n$) in the input array, what is the size of the left region after partitioning?



Balanced Partitioning

Question: What is the probability that the **pivot** selected is the m^{th} smallest value in the array of size **n**?

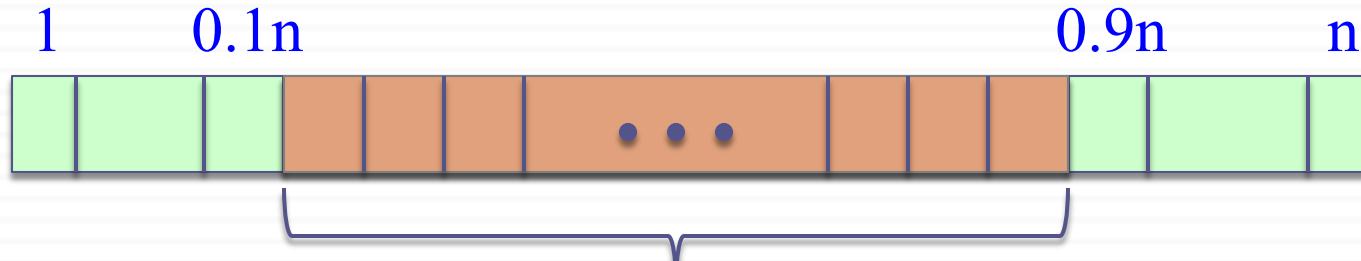
$1/n$ (since all input permutations are equally likely)

Question: What is the probability that the left partition returned by **H-PARTITION** has size **m**, where $1 < m < n$?

$1/n$ (due to the answers to the previous 2 questions)

Balanced Partitioning

Question: What is the probability that **H-PARTITION** returns a split that is more balanced than **0.1-to-0.9**?



The partition boundary will be in this region
for a more balanced split than 0.1-to-0.9

$$\text{Probability} = \sum_{q=0.1n+1}^{0.9n-1} \frac{1}{n} = \frac{1}{n}(0.9n - 1 - 0.1n - 1 + 1) = 0.8 - \frac{1}{n}$$

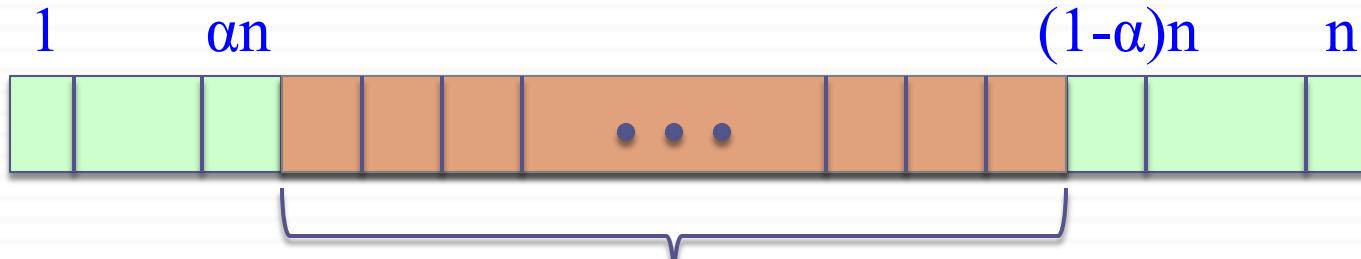
$\approx 0.8 \text{ for large } n$

Balanced Partitioning

- The probability that $H\text{-PARTITION}$ yields a split that is more balanced than 0.1-to-0.9 is 80% on a random array.
- Let $P_{\alpha >}$ be the probability that $H\text{-PARTITION}$ yields a split more balanced than $\alpha\text{-to-}(1-\alpha)$, where $0 < \alpha \leq 0.5$
- Repeat the analysis to generalize the previous result

Balanced Partitioning

Question: What is the probability that **H-PARTITION** returns a split that is more balanced than α -to- $(1-\alpha)$?



The partition boundary will be in this region
for a more balanced split than α -to- $(1-\alpha)$ n

$$\text{Probability} = \sum_{q=\alpha n+1}^{(1-\alpha)n-1} \frac{1}{n} = \frac{1}{n} ((1-\alpha)n - 1 - \alpha n - 1 + 1) = (1 - 2\alpha) - \frac{1}{n}$$

$\approx (1-2\alpha)$ for large n

Balanced Partitioning

- We found $P_{\alpha} = 1 - 2\alpha$
Examples: $P_{0.1} = 0.8$ $P_{0.01} = 0.98$
- Hence, *H-PARTITION* produces a split
 - *more balanced* than a
 - 0.1-to-0.9 split 80% of the time
 - 0.01-to-0.99 split 98% of the time
 - *less balanced* than a
 - 0.1-to-0.9 split 20% of the time
 - 0.01-to-0.99 split 2% of the time

Intuition for the Average Case

- Assumption: All permutations are equally likely
 - ▣ Only for intuition; we'll revisit this assumption later
- Unlikely: Splits always the same way at every level

- Expectation:
 - Some splits will be **reasonably balanced**
 - Some splits will be **fairly unbalanced**
- Average case: A mix of good and bad splits
 - Good* and *bad* splits distributed randomly thru the tree

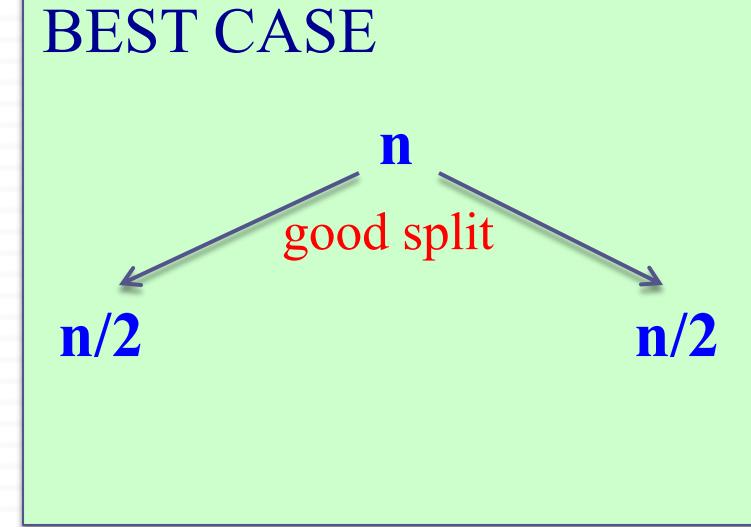
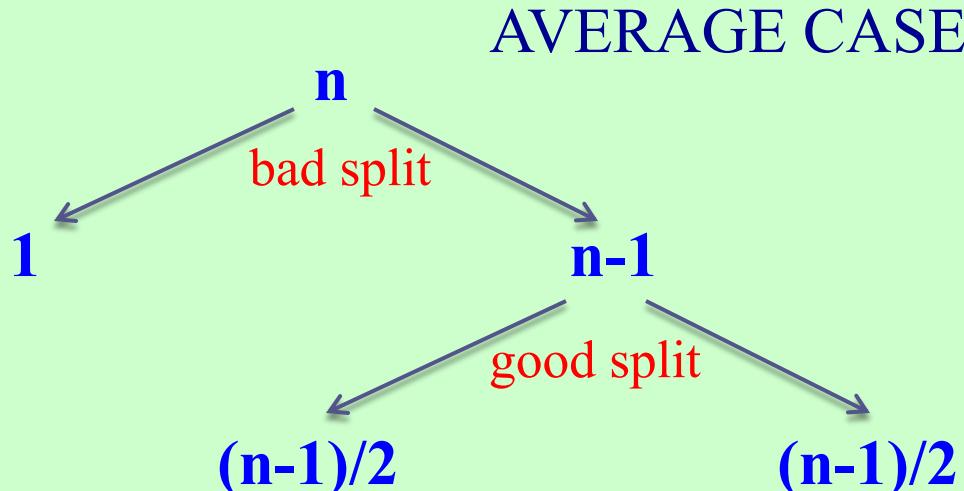
Intuition for the Average Case

- Assume for intuition: Good and bad splits occur in the alternate levels of the tree

Good split: Best case split

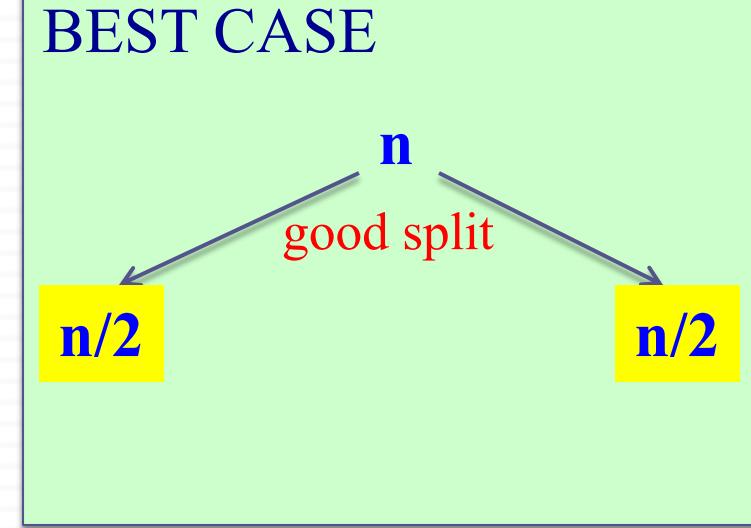
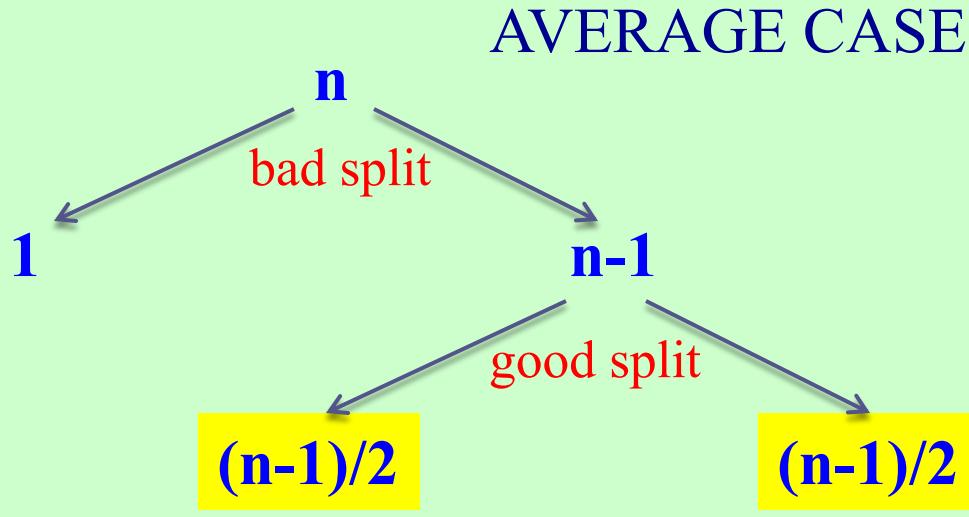
Bad split: Worst case split

Intuition for the Average Case



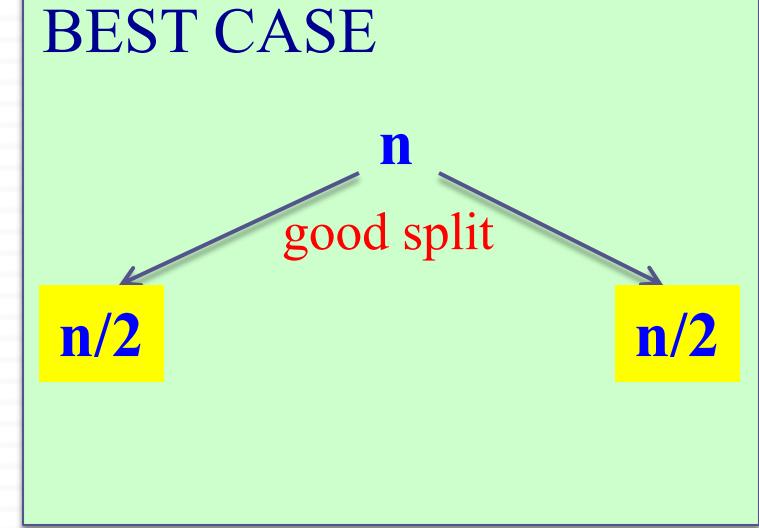
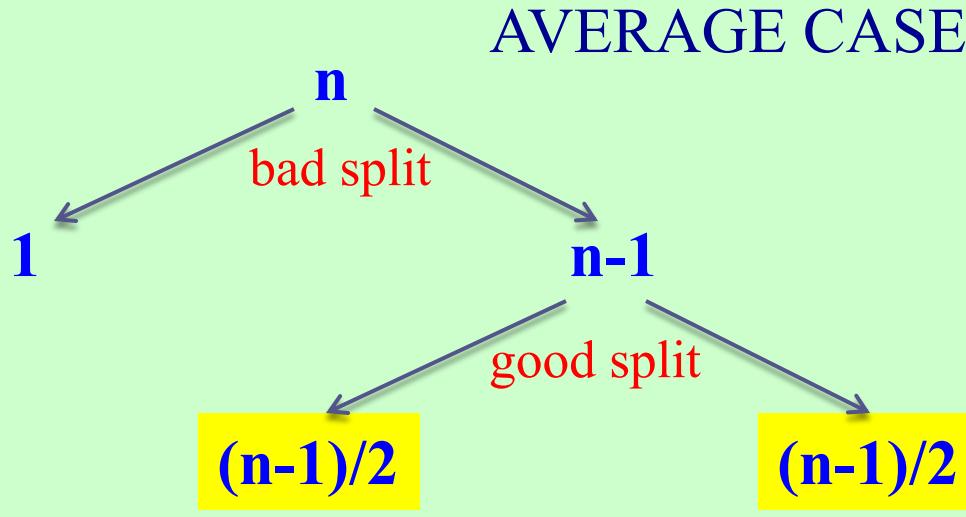
Compare 2-successive levels of avg case vs. 1 level of best case

Intuition for the Average Case



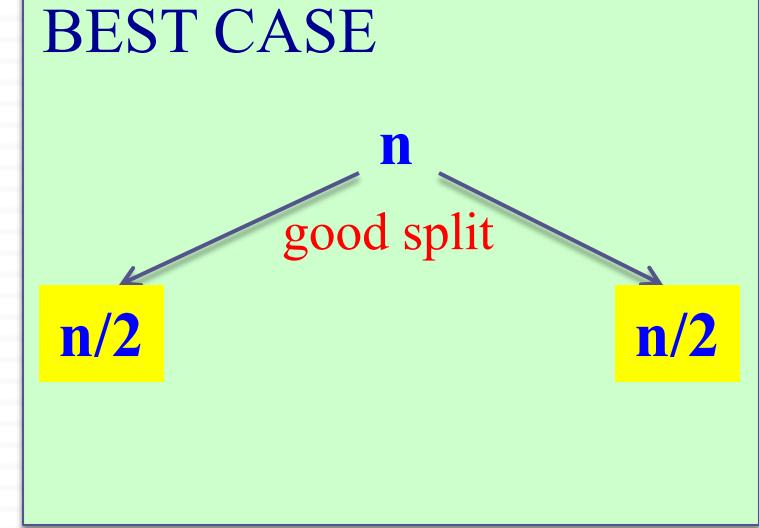
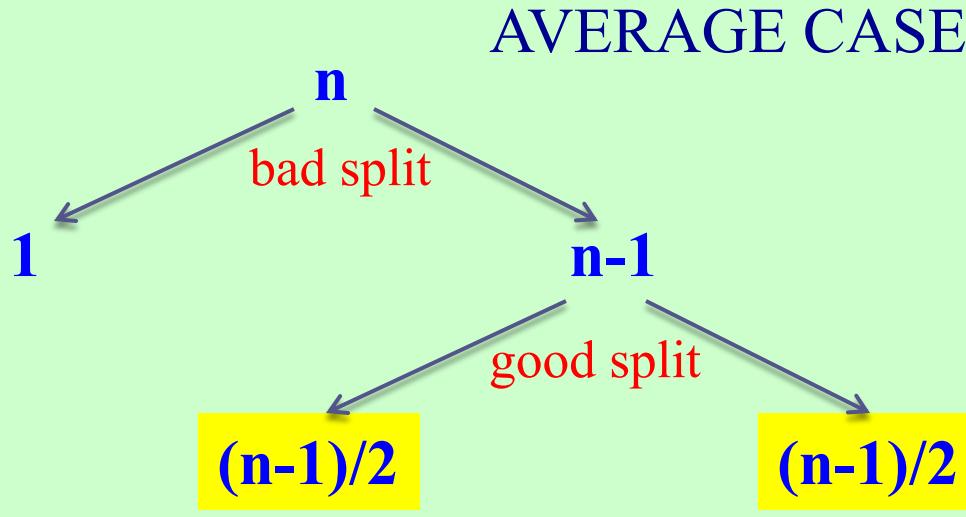
- In terms of the remaining subproblems, two levels of avg case is slightly better than the single level of the best case
- The avg case has extra divide cost of $\Theta(n)$ at alternate levels

Intuition for the Average Case



- The extra divide cost $\Theta(n)$ of bad splits absorbed into the $\Theta(n)$ of good splits.
- Running time is still $\Theta(n \lg n)$

Intuition for the Average Case



- Running time is still $\Theta(n \lg n)$
 - But, slightly larger hidden constants, because the height of the recursion tree is about twice of that of best case.

Intuition for the Average Case

- Another way of looking at it:

Suppose we alternate **lucky**, **unlucky**, **lucky**, **unlucky**, ...

We can write the recurrence as:

$$\begin{array}{ll} L(n) = 2 U(n/2) + \Theta(n) & \text{lucky split (best)} \\ U(n) = L(n-1) + \Theta(n) & \text{unlucky split (worst)} \end{array}$$

Solving:

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

How can we make sure we are usually lucky for all inputs?

Summary: Quicksort Runtime Analysis

Worst case: Unbalanced split at every recursive call

$$\begin{aligned} T(n) &= T(1) + T(n-1) + \Theta(n) \\ \rightarrow T(n) &= \Theta(n^2) \end{aligned}$$

Best case: Balanced split at every recursive call (extremely lucky)

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

Summary: Quicksort Runtime Analysis

Almost-best case: Almost-balanced split at every recursive call

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

or $T(n) = T(n/100) + T(99n/100) + \Theta(n)$

or $T(n) = T(\alpha n) + T((1-\alpha)n) + \Theta(n)$

for any constant α , $0 < \alpha \leq 0.5$

Summary: Quicksort Runtime Analysis

For a random input array, the probability of having a split

more balanced than $0.1 - \text{to}-0.9$: 80%

more balanced than $0.01 - \text{to}-0.99$: 98%

more balanced than $\alpha - \text{to}-(1-\alpha)$: $1 - 2\alpha$

for any constant α , $0 < \alpha \leq 0.5$

Summary: Quicksort Runtime Analysis

Avg case intuition: Different splits expected at different levels
→ some balanced (good), some unbalanced (bad)

Avg case intuition: Assume the good and bad splits alternate
i.e. good split → bad split → good split → ...
→ $T(n) = \Theta(n \lg n)$
(informal analysis for intuition)