DESIGN AND ANALYSIS OF ALGORITHMS

CS 4120/5120

SORTING - QUICKSORT

AGENDA

- The definition of the sorting problem
- Quicksort

THE SORTING PROBLEM

- A general description of the sorting problem
  - Input: A sequence of n numbers $< a_1, a_2, \ldots, a_n >$
  - Output: A permutation (reordering) $< a'_1, a'_2, \ldots, a'_n >$ of the input sequence such that $a'_1 \leq a'_2 \leq \cdots \leq a'_n$.

- The general sorting problem is a $\Omega(n \lg n)$ problem.
  - However, there are algorithms that sort special input data in linear time.

SORTING ALGORITHMS

- Naïve method: bubble sort $O(n^2)$
- Insertion sort
  - Best case ($O(n)$), worst case ($O(n^2)$)
- Mergesort ($O(n \lg n)$)
- Quicksort
- Heapsort

QUICKSORT & RUNNING TIME

- Also a **divide-and-conquer** approach.
- Running time on array $A[p\,..\,r]$

  $T(r-p+1)$

  $=$

QUICKSORT (A, p, r)	Cost	Time
if $p < r$	$\Theta(1)$	1
$\quad \boldsymbol{q}$ = **PARTITION** (A, p, r)	$\boldsymbol{\Theta(r-p)}$	1
$\quad$ QUICKSORT $(A, p, q-1)$	$T(\boldsymbol{q}-p)$	1
$\quad$ QUICKSORT $(A, q+1, r)$	$T(r-\boldsymbol{q})$	1

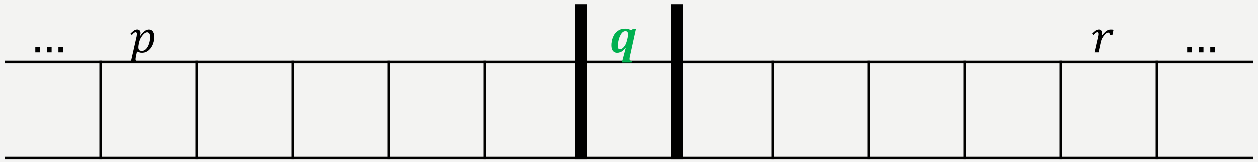
- Running time on array $A[1\,..\,n]$, plug in parameters $(A, 1, n)$.

  $T(n)$

  $=$

QUICKSORT
BEST-CASE RUNNING TIME

- **Balanced** partition



  – Recurse on roughly __________ of $A[p..r]$ ________ time(s).

  – $T(r-p+1) = \Theta(1) + \mathbf{2}T\left(\frac{\boldsymbol{r-p}}{\mathbf{2}}\right) + \Theta(r-p) = \mathbf{2}T\left(\frac{\boldsymbol{r-p}}{\mathbf{2}}\right) + \Theta(r-p)$ (simplified)

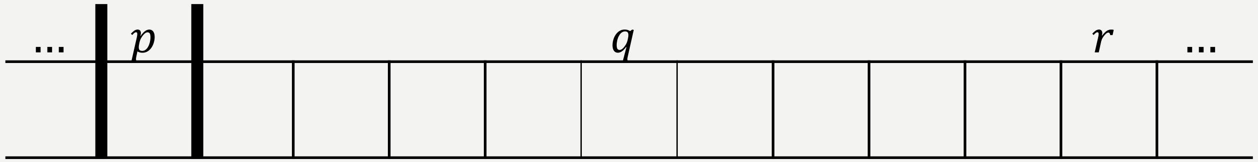
QUICKSORT
BEST-CASE ON $A[1..n]$

- The running time $T(r-p+1) =$ $\mathbf{2}T(\boldsymbol{n}/\mathbf{2}) + \Theta(n)$ $= T(n) = \Theta(n \lg n)$
  - Recursion tree

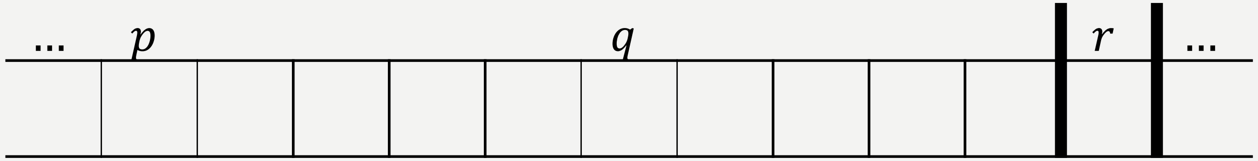
Level	Tree	Node	Per-level Cost
0	cn	1	$1 \times cn$
1	cn/2 cn/2	2	$2 \times cn/2$
2	cn/4 cn/4 cn/4 cn/4	4	$2^2 \times cn/2^2$
⋮	⋮ ⋮ ⋮ ⋮ ⋮ ⋮		⋮
lg n	T(1) … T(1)	n	$\Theta(n)$

QUICKSORT
WORST-CASE RUNNING TIME

- **Unbalanced** partition
  - Either



  - Or



    - Recurse on __________ of $A[p..r]$ ________ time(s).
  - $T(r-p+1) = \Theta(1) + T(\boldsymbol{r}-\boldsymbol{p}) + \Theta(r-p) = T(\boldsymbol{r}-\boldsymbol{p}) + \Theta(r-p)$ (simplified)

QUICKSORT
WORST-CASE ON $A[1..n]$

- The running time $T(r-p+1) = \quad T(\boldsymbol{n}-\mathbf{1}) + \Theta(n) \; = T(n) \; = O(n^2)$
  - Recursion tree

Level	Tree	Node	Per-level Cost
0	cn	1	1 × cn
1	c(n-1)	1	1 × c(n-1)
2	c(n-2)	1	1 × c(n-2)
⋮	⋱		⋮
n-1	T(1)	1	Θ(1)

PROPORTIONAL SPLIT
BREAKOUT SESSION (10 minutes)

- Suppose the partitioning algorithm always produces a 9-to-1 proportional split.
  - Write the recurrence function of the quicksort algorithm that uses such a partitioning algorithm
  - Draw the recursion tree

Level	Tree	Node	Per-level Cost

  - Guess a bound

PROPORTIONAL SPLIT
BREAKOUT SESSION (10 minutes)

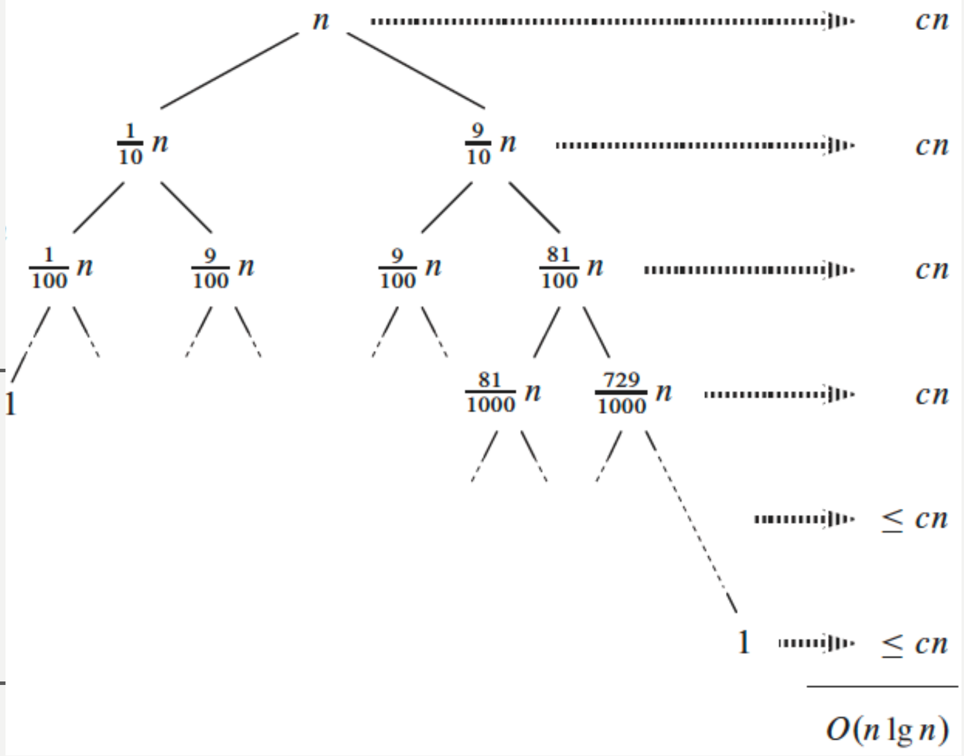
- Suppose the partitioning algorithm always produces a 9-to-1 proportional split.
  - The recurrence

    $T(n) =$
  - The recursion tree
  - Guess $T(n) =$

Level	Tree	Node	Per-level Cost

9-TO-1 PROPORTIONAL SPLIT

- The cost of each level is cn.
- The shortest path from the root to a leaf is $\log_{\frac{1}{1/10}} n = \log_{10} n$
- The longest path from the root to a leaf is $\log_{\frac{1}{9/10}} n = \log_{\frac{10}{9}} n$



PROPORTIONAL SPLIT
TIME COMPLEXITY

- Any split of constant proportionality yields a recursion tree of depth $\Theta(n)$, where the cost at each level is $O(n)$.
  - 9-to-1, 8-to-1, ...
- In other words, the running time is $O(n \log n)$ whenever the split has constant proportionality.

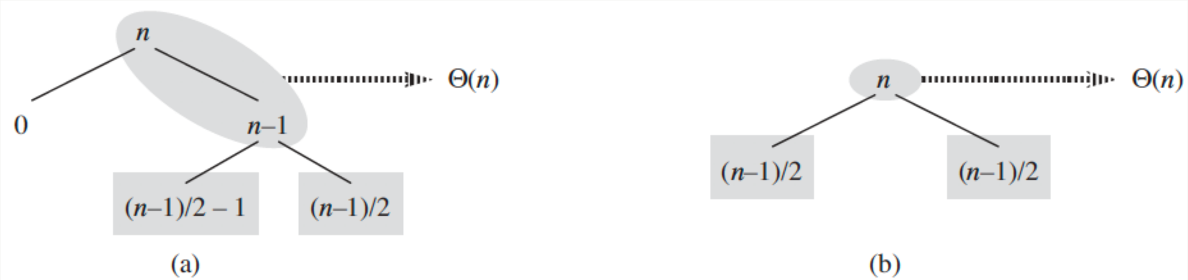
BALANCED VS. UNBALANCED PARTITION

- **Balanced** partition
  - A split of **constant** proportionality.
  - Runs asymptotically as fast as **merge sort.**
  - Note: The two subproblems are NOT necessarily half the original size.

- **Unbalanced** partition
  - A partition yielding a constant number of elements on one side of the pivot.
  - Runs asymptotically as slowly as **insertion sort.**

QUICKSORT
EXPECTED RUNNING TIME

- Use the RANDOMIZED-PARTITION algorithm.
- Average case partitioning
  - Produce a mix of "good" and "bad" splits.



- The expected running time of quicksort is $E(T(n)) = O(n \lg n)$.
  - Details of the proof can be found on page 181 of the textbook.

NEXT UP
SORTING - HEAPSORT

REFERENCE