Quick Sort - Analysis

Acknowledgements: Dr. Saleena N, CSED, NITC.

Quick Sort - Overview

- Partition into two subarrays with respect to a pivot
- Recursively Quick Sort the partitions
- Base Case of recursion – array of size 1

Quick Sort – Running Time

- **Recursive**
  - **Time for Partitioning**
  - **Time to recursively sort the partitions**

Divide and Conquer – Recurrence

$$T(n) = \Theta(1) \quad \textbf{if } n <= c$$
$$= a\,T(n/b) + D(n) + C(n) \quad \textbf{otherwise}$$

- Number of subproblems – ***a ?***
- Size of the subproblems - value of ***b?***
- ***D(n)*** – time to divide the problem into subproblems ?
- ***C(n)*** – time to combine the solutions ?

Quick Sort - Divide and Conquer

- Number of subproblems : ***a* =2**
- Size of subproblems : value of ***b* = ?**
- ***D(n)*** **:** time for partitioning
- ***C(n)*** **:** No work to combine, subarrays are already sorted
- ***f(n)=D(n)*** = Time for partitioning

Quick Sort - Partitioning

- Array $A[p..r]$
- Input Size, $n = r - p + 1$
- What is the running time of PARTITION()?
- Argue that the running time of PARTITION() is $\Theta(n)$

Quick Sort – Partitioning on sorted input

Quicksort on an already sorted array

Input: 1 2 3 4 5 6 7 8

- Partitions ?
- Observations regarding partition size after each step?

Quick Sort – Partitioning on sorted input

- Subproblems :
  1. Size n-1
  2. Size 0
- An unbalanced partitioning

Quick Sort – Unbalanced Partitioning

- Subproblems : Size n-1, Size 0

$$
\begin{aligned}
T(n) &= T(n-1) + T(0) + \Theta(n) \\
&= T(n-1) + \Theta(n)
\end{aligned}
$$

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

Quick Sort – Worst Case

- Unbalanced Partitioning - Subproblems : Size n-1, Size 0

$$T(n) = T(n\text{-}1) + T(0) + \Theta(n)$$
$$= T(n\text{-}1) + \Theta(n)$$

- Worst Case Running Time is $\Theta(n^2)$

Quick Sort – Best Case

- Balanced Partitioning – Most even possible split
- Subproblems : each of size no more than *n/2*

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

Case 2 of Master Theorem: $T(n) = \Theta(n \lg n)$

Best Case Running Time is $\Theta(n \lg n)$

Quick Sort – Running Time

- Depends on how balanced is the partitioning at every level of recursion
- Balanced partition - Asymptotically faster

- Best Case Running Time: $\Theta(n \lg n)$
- Worst Case Running Time: $\Theta(n^2)$

Quick Sort – Running Time

- Best Case Running Time: $\Theta(n \lg n)$
- Worst Case Running Time: $\Theta(n^2)$
  - Already sorted array (best case ($\Theta(n)$) for insertion sort)

Even with a worst case running time of $\Theta(n^2)$, Quicksort is often the best practical choice. Why?

Quick Sort – Average Case

- Average Case Running Time: *O(n lg n)*
  - *Closer to the best case*
  - *Quicksort is often the best practical choice because its average behaviour is good*

Quick Sort – Balanced Partitioning

Suppose a 9-to-1 proportional split at all levels (highly unlikely)

$T(n) = T\,(9n\,/\,10) + T(\,n\,/10) + cn$

- Draw the recursion tree
- Cost at each level?

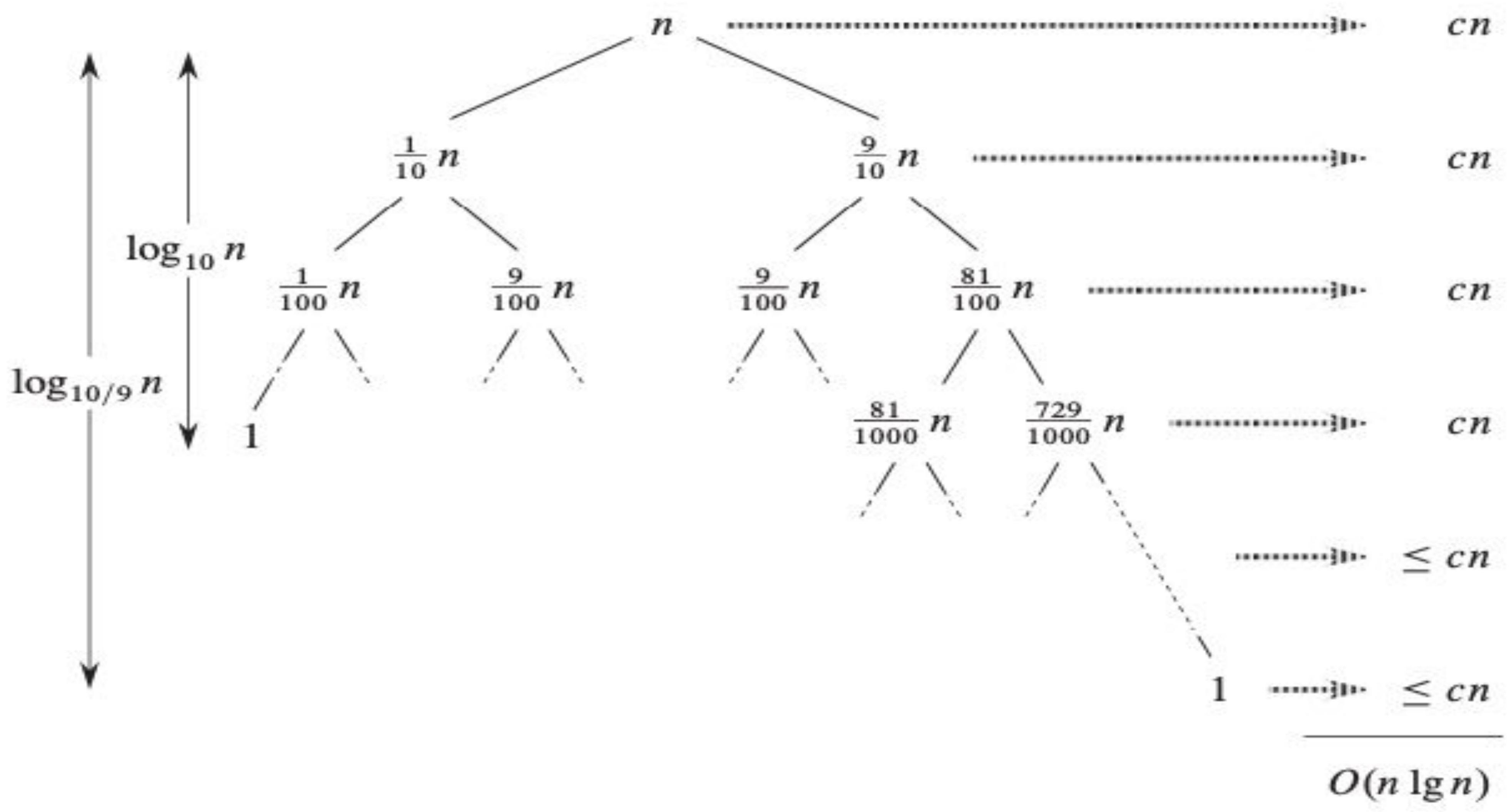


Figure 7.4 A recursion tree for QUICKSORT in which PARTITION always produces a 9-to-1 split, yielding a running time of $O(n \lg n)$. Nodes show subproblem sizes, with per-level costs on the right. The per-level costs include the constant c implicit in the $\Theta(n)$ term.

Quick Sort – Balanced Partitioning

$T(n) = T(9n / 10) + T(n /10) + cn$

Till depth $log_{10} n$, every level has cost cn

$$log_{10} n = \Theta(lg\, n)$$

Note: Changing the base of a logarithm from one constant to another changes the value of the logarithm by a constant factor.

$log_b\, a = log_c\, a / \quad log_c\, b$

Quick Sort – Balanced Partitioning

- $T(n) = T(9n / 10) + T(n /10) + cn$
- Till depth $log_{10} n$, every level has cost cn. Below that at every level, total cost $<= cn$
- Recursion terminates at depth $log_{10/9} n \quad = \Theta(lg\ n)$
- Total cost: $O(n\ lg\ n)$

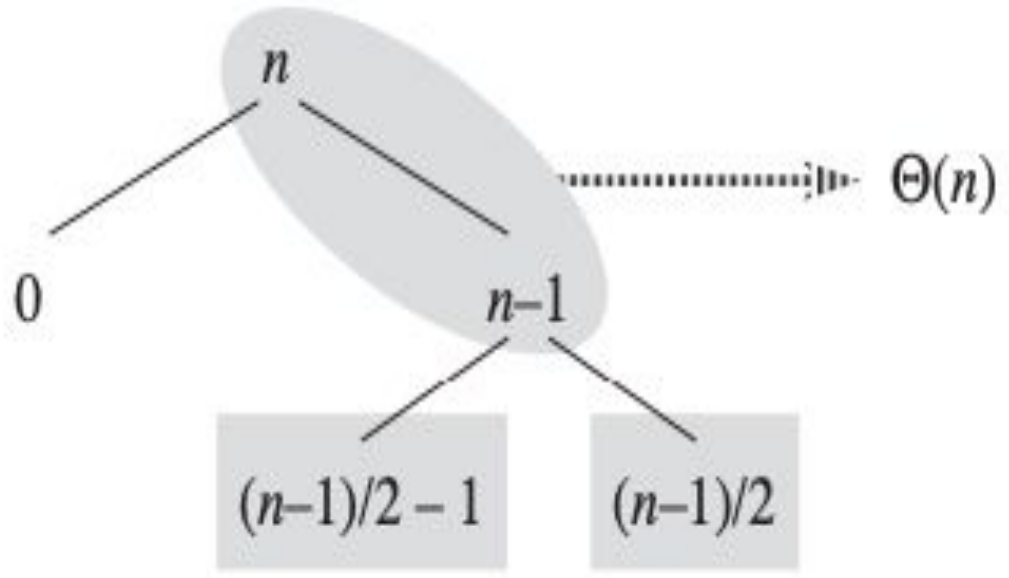
Quick Sort – Balanced Partitioning

- Any split of constant proportionality
  - Recursion tree depth *Θ(lg n)*
  - Cost at each level *O(n)*
  - Running time *O(n lg n)*
- Highly unlikely to get same proportional split at each level

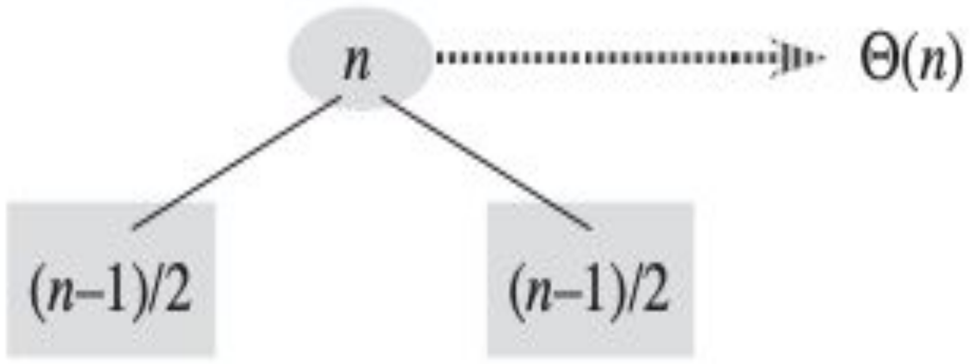
Quick Sort – Average Case Running Time

- Assume all permutations of the input numbers are equally likely
- In the average case,
  - PARTITION produces a mix of good and bad splits
  - good and bad splits distributed randomly throughout the tree
  - Suppose good/bad splits at alternate levels
    - $n \rightarrow 0, n-1 \rightarrow ((n-1)/2) - 1, (n-1)/2$
    - running time is *O(n lg n)* but with a slightly larger constant than best case

Good and Bad splits



(a)



(b)

Quick Sort – Randomized Version

- Choose pivot randomly
- Expected running time: $\Theta(n \lg n)$
- Regarded as sorting algorithm of choice for large enough inputs

Reference

T H Cormen, C E Leiserson, R L Rivest, C Stein *Introduction to Algorithms*, 3rd ed., PHI, 2010