## Lecture 4: February 15, 2001

### Today:

- Quicksort
  (or *Randomized* algorithm for sorting)



## Quicksort

Yet another sorting algorithm, but:

- Sorts "in place" (doesn't require additional array)
  - like insertion sort
  - unlike merge sort
- Very practical (with tuning)



## Quicksort — Divide-and-conquer algorithm

1. **Divide:** Partition array into 2 subarrays such that elements in lower part $\leq$ elements in higher part.
2. **Conquer:** Recursively sort 2 subarrays.
3. **Combine:** Trivial (because in place).



### Key:

Linear-time ($\Theta(n)$) partitioning procedure

$$\begin{array}{|c|c|c|} \hline \quad \leq x \quad & x & \qquad \geq x \qquad \\ \hline \end{array}$$

## Partition procedure

Partition$(A, p, r)$
(Partition $A[p..r]$ around random
element $x = A[k]$)
$k \leftarrow \text{Random}(p \ldots r)$
$x \leftarrow A[k]$
$i \leftarrow p - 1$
$j \leftarrow r + 1$

**while** TRUE **do**
    **repeat** $j \leftarrow j - 1$ **until** $A[j] \le x$
    **repeat** $i \leftarrow i + 1$ **until** $A[i] \ge x$
    **if** $i < j$ **then** exchange $A[i] \leftrightarrow A[j]$
    **else** quit (and return $j$)



## Correctness proof idea

Loop invariant:

| $\le x$ | ? | $\ge x$ |
|---|---|---|

$p$ $i$ $j$ $r$

Time $= \Theta(n)$ for $n$-element subarray



## Quicksort — Recursive algorithm

Quicksort$(A, p, r)$
**if** $p < r$
**then** $q \leftarrow \text{Partition}(A, p, r)$ $\triangleright$ around $A[r$
Quicksort$(A, p, q - 1)$
Quicksort$(A, q + 1, r)$

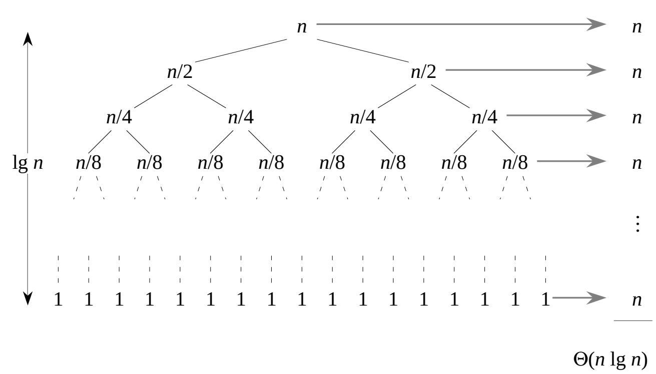
Initial call: Quicksort$(A, 1, length[A])$



## Analysis of Quicksort (best case)

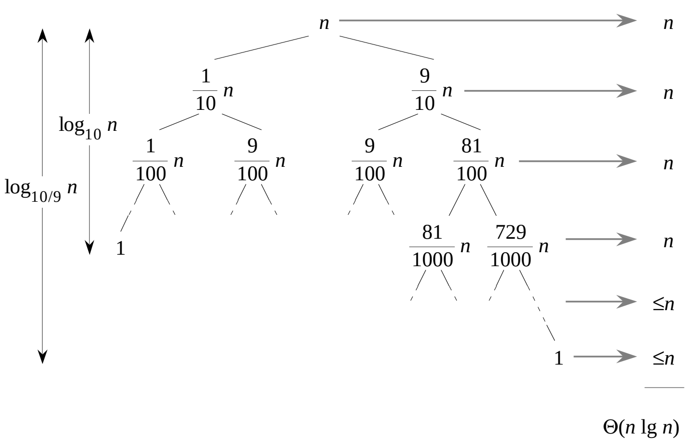
If we're lucky, Partition always splits array evenly

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



Suppose the split is $\frac{1}{10} : \frac{9}{10}$

$$\begin{aligned} T(n) &= T(n/10) + T(9n/10) + \Theta(n) \\ &= \Theta(n \lg n) \quad \text{Still lucky!} \end{aligned}$$






## Analysis of Quicksort (worst case)

How might we be unlucky?

- one side of partition has 1 element

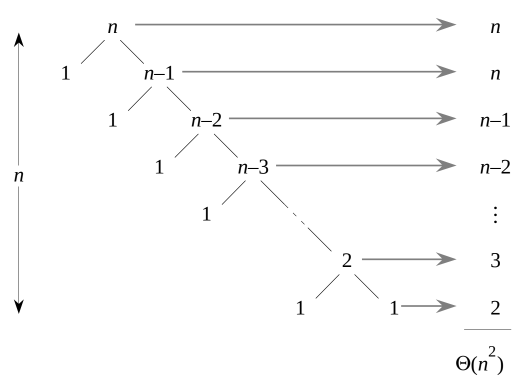
$$\begin{aligned} T(n) &= T(1) + T(n-1) + \Theta(n) \\ &= T(n-1) + \Theta(n) \qquad \text{because } T(1) = \Theta(1) \\ &= \sum_{k=1}^{n} \Theta(k) = \Theta(\sum_{k=1}^{n} k) \\ &= \Theta(n^2) \quad \text{[arithmetic series]} \end{aligned}$$



## Recursion Tree

- Left nodes ($\Theta(1)$) add up to $\Theta(n)$
- Root and right nodes add up to $\Theta\left(\Sigma_{k=1}^{n} k\right) = \Theta(n^2)$
- Total: $\Theta(n) + \Theta(n^2) = \Theta(n^2)$





When does worst case occur?

- Our "random" elements $x$ are $1, 2, 3, 4 \ldots n$
- Does not look very random !



## Analysis of QUICKSORT

- Randomized algorithm - running time varies even for the same input. Therefore, we can say the running time is "this much" on the *average*, or it is "this much" with *certain probability*.
- Assume random number generates independent choices throughout algorithm.
- For technical reasons, assume all input elements are distinct. Picking index of element is equivalent to picking element.



## Key observations

- the running time bounded by $O(n)$ times the depth of the recursion tree (as seen on earlier pictures)
- "nice splits" happen with "nice probability"
- if we have large enough number of trials, and each trial has "nice probability" of success, we have many successes with high probability



## Lucky partitions

Let $a_i$ be the $i$-th smallest element in $A[\cdot]$ (i.e., with the rank $i$).

Let $D_i$ denote the depth of $a_i$ in the recursion tree. The total tree depth is $D = \max_i D_i$.

We say a split (say, of $A[p \ldots r]$) is "lucky" for $a_i$, if the part where $a_i$ ends up in is of size $\leq 3/4(r-p+1)$, i.e., if we reduce the array size by a factor of $3/4$.

After $\log_{4/3} n$ lucky splits, $a_i$ is a leaf !

## "Lucky" lemma

**Lemma:** At any time, $a_i$ is lucky with probability at least $1/4$ (could be even higher).

**Proof:** If $i \le n/2$, choosing $x$ to be any element with rank in $\{n/2 \ldots 3n/4\}$ creates a lucky split. If $i \ge n/2$, situation is symmetric.



## The analysis

Let $l = C \log_{4/3} n$, where $C$ is a "large" constant (set later).

What is the probability that $D_i > l$ ? It is at most the probability that in $l$ trials, each having success probability $\ge 1/4$, we were *not* successful $\ge l - t = l - \log_{4/3} n$ times (since $\log_{4/3} n$ lucky splits are enough to make $a_i$ a leaf).

The latter probability is at most

$$\binom{l}{l-t}(1-1/4)^{l-t}$$

(will elaborate on that later)



## The analysis ctd.

$$\begin{aligned}
\binom{l}{l-t}(1-1/4)^{l-t} &= \binom{l}{t}(3/4)^{l-t} \\
&\le \left(\frac{el}{t}\right)^t (3/4)^{l-t} \\
&= \left(\frac{eC \log_{4/3} n}{\log_{4/3} n}\right)^t (3/4)^{(C-1)\log_{4/3} n} \\
&= \frac{(eC)^{\log_{4/3} n}}{n^{C-1}} \\
&= n^{\log_{4/3} eC - (C-1)}
\end{aligned}$$

When $C$ is large enough (e.g., 100), the probability is smaller than $1/n^2$.



## Finishing the analysis

We proved that $\Pr[D_i > l] \le 1/n^2$.

Therefore, the probability that *there exists* $i$ such that $D_i > l$ is at most

$$\Pr[D_1 > l] + \Pr[D_2 > l] \ldots \Pr[D_n > l] \le n \cdot 1/n^2 = 1/n$$

Therefore probability that $\max_i D_i \le l$ is at least $1 - 1/n$.

## Appendix

Why is the probability that "in $l$ trials, each having success probability $\geq 1/4$, we were *not* successful $\geq l-t$ times" at most

$$\binom{l}{l-t}(3/4)^{l-t}$$

?

- if we were not successful $\geq l-t$ times, we were not successful during some set of $l-t$ trials
- the probability we were not successful during a *fixed* set of exactly $l-t$ trials is at most $(3/4)^{l-t}$
- there are $\binom{l}{l-t}$ subsets of trials of size $l-t$