Today:
- Quicksort
  (or Randomized algorithm for sorting)

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

\[
\begin{array}{c|c|c}
\leq x & \mid x \mid & \geq x
\end{array}
\]
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** \(A[j] \leq x\)

**repeat** \(i \leftarrow i + 1\) **until** \(A[i] \geq x\)

**if** \(i < j\) **then** exchange \(A[i] \leftrightarrow A[j]\)

**else** quit (and return \(j\))

\[\begin{array}{c|c|c|c}
& \leq x & ? & \geq x \\
p & i & j & r
\end{array}\]

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

---

Quicksort — Recursive algorithm

**QuickSort** \((A, p, r)\)

**if** \(p < r\)

**then** \(q \leftarrow \text{Partition}(A, p, r)\) // around \(A[r]\)

** QuickSort** \((A, p, q - 1)\)

** QuickSort** \((A, q + 1, r)\)

Initial call: **QuickSort** \((A, 1, \text{length}[A])\)

---

Analysis of QuickSort (best case)

If we're lucky, **Partition** always splits array evenly

\[T(n) = 2T(n/2) + \Theta(n)\]

\[\Theta(n \lg n)\]
Suppose the split is \( \frac{1}{10} : \frac{9}{10} \):

\[
T(n) = T(n/10) + T(9n/10) + \Theta(n) \\
= \Theta(n \log n) \quad \text{Still lucky!}
\]

**Analysis of QUICKSORT (worst case)**

How might we be unlucky?

- one side of partition has 1 element

\[
T(n) = T(1) + T(n - 1) + \Theta(n) \\
= T(n - 1) + \Theta(n) \quad \text{because } T(1) = \Theta(1) \\
= \sum_{k=1}^{n} \Theta(k) = \Theta(\frac{n}{k-1} k) \\
= \Theta(n^2) \quad [\text{arithmetic series}]
\]

**Recursion Tree**

- Left nodes (\( \Theta(1) \)) add up to \( \Theta(n) \)
- Root and right nodes add up to \( \Theta(\sum_{k=1}^{n} k) = \Theta(n^2) \)
- 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 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 $i \leq n/2$, choosing $x$ to be any element with rank in $\{n/2 \ldots 3n/4\}$ creates a lucky split. If $i \geq n/2$, 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 $\geq 1/4$, we were not successful $\geq l - t = l - \log_{4/3} n$ times (since $\log_{4/3} n$ lucky splits are enough to make $a_i$ a leaf).

The latter probability is at most

$$
\left( \frac{l}{l - t} \right) (1 - 1/4)^{l-t}
$$

(will elaborate on that later)

---

**The analysis ctd.**

\[
\left( \frac{l}{l - t} \right) (1 - 1/4)^{l-t} = \left( \frac{l}{t} \right) (3/4)^{l-t} \\
\leq \left( \frac{e l}{t} \right) (3/4)^{l-t} \\
= \left( \frac{e C \log_{4/3} n}{\log_{4/3} n} \right)^t (3/4)^{(C-1) \log_{4/3} n} \\
= \left( \frac{e C^{\log_{4/3} n}}{n^{C-1}} \right) (3/4)^{(C-1) \log_{4/3} n} \\
= n^{\log_{4/3} e C^{-(C-1)}}
\]

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

---

**Finishing the analysis**

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

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

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

Therefore probability that $\max_i D_i \leq l$ 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 $l - t$