Randomized Algorithms:

Quicksort

- C.A. Hoare (1962) proposed it
- sorting algorithm based on divide & conquer (like mergesort)
- sorts in place
  - uses O(1) storage beyond array (like insertion sort, unlike mergesort)
- practical

- several versions:
  - basic - good in average case
  - randomized - good on all inputs, in expectation
  - deterministic - good in worst case
Main Idea

(1) Divide
- pick a "pivot element" \( x \), how? \( \text{Randomly?} \)
- partition array into 2 subarrays
  \[ L = \text{elements} < x \]
  \[ G = \text{elements} > x \]

Example:

\[
\begin{array}{cccc}
6 & 10 & 13 & 5 & 8 \\
\end{array}
\]
pivot = 8
\[
\begin{array}{c}
L = \begin{array}{c}
6 \\
5 \\
6 \\
\end{array} \\
G = \begin{array}{c}
10 \\
13 \\
\end{array}
\end{array}
\]

(2) Conquer
- recursively sort \( L, G \)

(3) Combine
- nothing
**Basic Partition** \( (A, p, r) \)

\[ X \leftarrow A[p] \]
\[ i \leftarrow p \]

for \( j \leftarrow p+1 \) to \( r \)

\[ \text{if } A[j] \leq X \]
\[ \text{then } i \leftarrow i+1 \]

exchange \( A[i] \leftarrow A[j] \)

exchange \( A[p] \leftarrow A[i] \)

**Invariant:** (until final exchange)

```
...... x \leq x \geq x ?
\hline
P i j r
```

\( L \) (so far) \( G \) (so far)

**Runtime:** \( O(r-p) \)
Example:

pivot $x = 6$

1. $P \geq 6$
   
   \[
   \begin{array}{cccccccc}
   6 & 10 & 13 & 5 & 8 & 3 & 2 & 11 \\
   \end{array}
   \]

2. $i \leq 6$
   
   \[
   \begin{array}{cccccccc}
   6 & 5 & 13 & 10 & 8 & 3 & 2 & 11 \\
   \end{array}
   \]

3. $j \geq 6$
   
   \[
   \begin{array}{cccccccc}
   6 & 5 & 3 & 10 & 8 & 13 & 2 & 11 \\
   \end{array}
   \]

4. $i \leq 6$
   
   \[
   \begin{array}{cccccccc}
   6 & 5 & 3 & 2 & 4 & 13 & 10 & 11 \\
   \end{array}
   \]

5. final swap:
   
   \[
   \begin{array}{cccccccc}
   2 & 5 & 3 & 6 & 8 & 13 & 10 & 11 \\
   \end{array}
   \]
Quicksort \((A, p, r)\)

\[\begin{align*}
\text{if } p &\leq r \quad \text{then return} \\
\text{Else } q &\leftarrow \text{Partition} \,(A, p, r) \\
\text{QuickSort} \,(A, p, q-1) \\
\text{QuickSort} \,(A, q+1, r)
\end{align*}\]

\[\Rightarrow \Theta(r-p) \text{ time}\]
\[\Rightarrow T(q-p)\]
\[\Rightarrow T(r-q)\]

Initial Call: \text{QuickSort} \,(A, 1, n)\]

Analysis Assumption: all elements distinct

(Otherwise, use better Partition algorithm)

Worst-case analysis:\n
- worst case = input sorted or reverse sorted
  \[\Rightarrow \text{partition around } x = \text{min or max}\]
  \[\Rightarrow \text{get } 0: n-1 \text{ split}\]
  so \[T(n) = T(0) + T(n-1) + \Theta(n)\]
  \[= T(n-1) + \Theta(n)\]
  \[= \Theta(n^2) \text{ arithmetic series}\]

Using recursion tree get:

\[\Theta(n)\]

\[\Theta(n)\]

\[\Theta(n)\]

\[\Theta(n)\]

\[\sum \Theta(n^2)\]
Why is Quicksort good?

**Lucky case:** partition array evenly

\[ T(n) = 2T\left(\frac{n}{2}\right) + \Theta(n) \]
\[ = \Theta(n \log n) \]

**Pretty lucky case:** partition into \( \frac{1}{10} : \frac{9}{10} \) split

\[ T(n) = T\left(\frac{1}{10}n\right) + T\left(\frac{9}{10}n\right) + \Theta(n) \]
\[ = \Theta(n \log n) \]

\[ T(n) = \Theta(n \log n) \]

**Alternate Lucky/Unlucky**

\[ L(n) = 2L\left(\frac{n}{2}\right) + \Theta(n) \]

\[ W(n) = L(n-1) + \Theta(n) \]

\[ L(n) = 2 \left( L\left(\frac{n}{2}-1\right) + \Theta\left(\frac{n}{2}\right) \right) + \Theta(n) \]
\[ = 2 L\left(\frac{n}{2}-1\right) + \Theta(n) \]
\[ = \Theta(n \log n) \]
How can we make a partition usually lucky?
(how likely can it be to pick a bad element?)

1. true "on average"
is: if input uniform.

2. pick random pivot

3. pick median pivot

Randomized Quicksort

- partition around random element in $A[p,r]
- choose $i$ uniformly from \{p, p+1, ..., r\}
- exchange $A[i] \leftrightarrow A[p]$
- rest as before (calling Rand-Quicksort recursively)

- runtime:
  - independent of input (no "bad" inputs)
  - depends on random choices (can have "really unlucky" choices)
Main Issue: How Many Comparisons?

Rename elements: (Recall distinctness assumption)

\[ z_1 < z_2 < \ldots < z_n \]
(elements from list, written in sorted order)

Define subsets:

\[ z_{ij} = \{ z_i, z_{i+1}, \ldots, z_j \} \]
(elements between \( z_i \) and \( z_j \))

When are \( z_i \) and \( z_j \) compared?

- Not compared if pivot chosen in \( z_{ij} \setminus \{ z_i, z_j \} \)

In example:

- \( z = 8, 10, 11, 13 \) not compared to \( z = 2, 3, 5 \)
  - because 6 chosen as pivot
- but 6 compared to all of \( z = 8, 10, 11, 13 \) and \( z = 2, 3, 5 \)

- Compared at most once and only if
  \( z_i \) or \( z_j \) chosen as pivot before
  any other element in \( z_{ij} \)

\[ \Pr[z_i \neq z_j \text{ compared}] = \Pr[z_i \text{ or } z_j \text{ chosen as pivot before others in } z_{ij}] = \frac{1}{j-i+1} \]
(since any elt in \( z_{ij} \) equally likely to be chosen first)
Finally:

1. Define Indicator Variable
   \[ X_{ij} = 1 \] if \[ Z_i \neq Z_j \] compared otherwise

2. Let number of comparisons \[ X = \sum_{i=1}^{n} \sum_{j=1}^{n} X_{ij} \]

Then \[ E[X] = E \left[ \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} X_{ij} \right] \]

\[ = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} E[X_{ij}] \]

\[ = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j - i + 1} \]

\[ = \sum_{i=1}^{n-1} \sum_{k=1}^{n-i} \frac{2}{k+1} \]

Set \[ K = j - i \]

\[ \leq \sum_{i=1}^{n-1} \sum_{k=1}^{n} \frac{2}{k} \]

\[ \leq O(n \log n) \]