Objectives:

- learn one version of Quicksort
- learn careful average-case analysis
- learn how to deal with "histories" in recurrences

Number of comparisons for sorting algorithms

Insertion Sort:	$\Theta(n^2)$	worst case
	$O(kn)$	if $\leq k$ items out of order
Mergesort:	$\Theta(n \lg n)$	worst case
Heapsort:	$\Theta(n \lg n)$	worst case
Quicksort:	$\Theta(n^2)$	worst case
	$\Theta(n \lg n)$	average case
Lower Bound:	$\Omega(n \lg n)$	worst case and average case

Four ways to apply recursion to sorting

algorithm	**decomposition**	**recombination**
Insertion sort	all-but-last/last	insert
Mergesort	split in half	merge
Selection sort	largest/all-but-largest	concatenate
Quicksort	$\approx$ smaller-half/larger-half	concatenate

The first two do the sorting during recombination, the others during decomposition.

NOTE: Heapsort is a variant of selection sort — the heap makes the selection of the largest item more efficient.

Quicksort

Quicksort outline

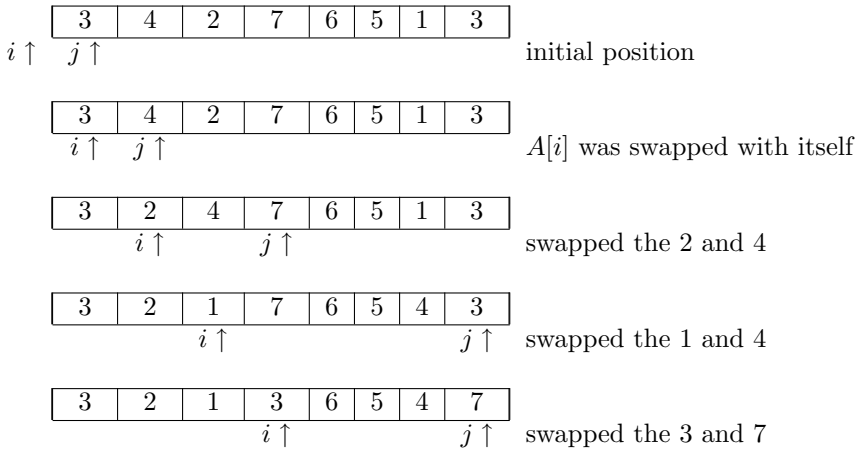
Quicksort $A[p \ldots r]$
  if $p < r$ **then**
    choose a partition element $x \in A[p \ldots r]$
    partition $A[p \ldots r]$ so that
      $A[i] \leq x$ for all $p \leq i \leq q-1$
      $A[q] = x$
      $A[i] \geq x$ for all $q+1 \leq i \leq r$
    recursively Quicksort $A[p \ldots q-1]$ and $A[q+1 \ldots r]$

The partition procedure

function PARTITION(A, p, r) **is**
▷ returns final index of "pivot"
  $i \leftarrow p-1$; $j \leftarrow p$
  $x \leftarrow A[r]$;
  ▷ INV: $A[k] \leq x$ for $p \leq k \leq i$ AND $A[k] > x$ for $i+1 \leq k \leq j-1$
  while $j < r$ **do**
    if $A[j] \leq x$ **then**
      $i \leftarrow i+1$
      exchange $A[i] \rightleftharpoons A[j]$
    $j \leftarrow j+1$
  exchange $A[i+1] \rightleftharpoons A[r]$
  return $i+1$
end PARTITION

How many comparisons does this take? Look at an example:

$x = 3$



Assumption: x, the pivot, is equally likely to end up in any one of the n possible array positions.

The assumption holds if all elements are distinct and all permutations are equally likely, or if we use "randomized" Quicksort, in which the pivot is chosen at random instead of being the last element of the (sub)array.

The new edition of the book takes a much slicker approach that involves looking at the expected number of comparisons involving a particular element throughout the whole sorting process.

event x is i-th smallest	**cost** (comparisons) $T(n) = n + \cdots$	**probability**
$i = 1$	$T(0) + T(n-1)$	$1/n$
$i = 2$	$T(1) + T(n-2)$	$1/n$
$\cdots$		
general i	$T(i-1) + T(n-i)$	$1/n$
$\cdots$		
$i = n$	$T(n-1) + T(0)$	$1/n$

We can assume that $T(0) = T(1) = 0$. The recurrence is as follows:

$$\begin{aligned} T(n) &= n - 1 + \frac{1}{n} \cdot [T(0) + T(n-1)] \\ &\quad + \frac{1}{n} \cdot [T(1) + T(n-2)] \\ &\quad \cdots \\ &\quad + \frac{1}{n} \cdot [T(n-1) + T(0)] \\ &= n - 1 + \frac{2}{n} \cdot \sum_{i=0}^{n-1} T(i) \end{aligned}$$

We can guess that this is $\mathrm{O}(n \lg n)$ and prove it by induction (similarly we can prove $\Omega(n \lg n)$), or we can use some algebraic tricks on the recurrence itself.

Guess and prove by induction

We can show by induction that $T(n) \le an \lg n$ for some constant $a > 0$. This is certainly the case for $n = 0$ and $n = 1$. We assume it to be true for all $k < n$ and show that it holds for n as well (with a specific a to be chosen later).

$$\begin{aligned} T(n) &= n - 1 + \frac{2}{n} \cdot \sum_{i=1}^{n-1} T(i) \\ &\le n - 1 + \frac{2}{n} \cdot \sum_{i=1}^{n-1} ai \lg i && \text{by induction hypothesis} \\ &= n - 1 + \frac{2a}{n} \cdot \sum_{i=1}^{n-1} i \lg i \\ &\le n - 1 + \frac{2a}{n} \left(\frac{1}{2} n^2 \lg n - \frac{1}{8} n^2 \right) && \text{by an argument shown below} \\ &= n - 1 + an \lg n - \frac{a}{4} n \\ &\le an \lg n && \text{as long as } a \ge 4c' \end{aligned}$$

The important thing here is that we end up with the same constant a that we started with. Consider, for example, the following flawed proof that $T(n)$ is $\mathrm{O}(n)$.

As before, assume that $T(k) \leq ak$ for all $k < n$ (the basis is trivial).

$$\begin{aligned}
T(n) &= n - 1 + \frac{2}{n} \cdot \sum_{i=1}^{n-1} T(i) \\
&\leq n - 1 + \frac{2}{n} \cdot \sum_{i=1}^{n-1} ai && \text{by induction hypothesis} \\
&= n - 1 + \frac{2a}{n} \cdot \sum_{i=1}^{n-1} i \\
&= n - 1 + \frac{2a}{n} \left(\frac{n^2}{2} - \frac{n}{2} \right) \\
&= n - 1 + an - a \\
&\leq (c' + a)n && \text{hey, it's just a constant times } n\text{, so O}(n)\text{, right?}
\end{aligned}$$

Wrong!

Bound on $\sum i \lg i$

$$\begin{aligned}
\sum_{i=1}^{n-1} i \lg i &= \sum_{i=1}^{\lceil n/2 \rceil - 1} i \lg i + \sum_{i=\lceil n/2 \rceil}^{n-1} i \lg i \\
&\leq (\lg n - 1) \sum_{i=1}^{\lceil n/2 \rceil - 1} i + \lg n \sum_{i=\lceil n/2 \rceil}^{n-1} i \\
&= \lg n \sum_{i=1}^{n-1} i - \sum_{i=1}^{\lceil n/2 \rceil - 1} i \\
&\leq \frac{1}{2} n(n-1) \lg n - \frac{1}{2} \left(\frac{n}{2} - 1 \right) \frac{n}{2} \\
&\leq \frac{1}{2} n^2 \lg n - \frac{1}{8} n^2
\end{aligned}$$

Algebraic tricks for history elimination

$$\begin{aligned}
T(n) &= n - 1 + \frac{2}{n} \cdot \sum_{i=2}^{n-1} T(i) \\
T(n-1) &= n - 2 + \frac{2}{n-1} \cdot \sum_{i=2}^{n-2} T(i) \\
nT(n) - (n-1)T(n-1) &= 2T(n-1) + n(n-1) - (n-1)(n-2) \\
nT(n) &= (n+1)T(n-1) + 2(n-1) \\
\frac{T(n)}{n+1} &= \frac{T(n-1)}{n} + \frac{2(n-1)}{n(n+1)}
\end{aligned}$$

Now let $B(n)$ denote $T(n)/(n+1)$, which means that $B(n-1)$ is $T(n-1)/n$. This gives us a much simpler linear recurrence, one with no "history".

$$B(n) \quad = \quad B(n-1) + \frac{2(n-1)}{n(n+1)}$$

Solving this recurrence is straightforward.

$$\begin{aligned} B(n) &= B(n-1) + \frac{2(n-1)}{n(n+1)} \\ B(1) &= 0 \\ B(n) &= \sum_{i=2}^{n} \frac{2(i-1)}{i(i+1)} \\ &\leq \sum_{i=2}^{n} \frac{2}{i+1} \quad \text{throw away the } -1 \\ &= 2\sum_{i=3}^{n+1} \frac{1}{i} \;\leq\; 2(\ln(n+1) - \ln 2) \end{aligned}$$

So $B(n) \in O(\lg n)$ implies $T(n) \in O(n \lg n)$.

A similar argument can be used to show that $B(n) \in \Omega(\lg n)$ and therefore $T(n) \in \Omega(n \lg n)$.

Here's a recap of the solution method.

1. $T(n) = n-1+\frac{2}{n} \cdot \sum_{i=2}^{n-1} T(i)$ (needed to get rid of summation "history"; so subtracted $T(n-1)$ from $T(n)$).
2. Let $B(n) = T(n)/(n+1)$ (that gave a recurrence with no history)
3. Solved $B(n)$ recurrence to get $B(n) \in O(\lg n)$.
4. Substituted back to get $T(n) = B(n)(n+1)$
   which is in $\mathrm{O}(n \lg n)$

A Completely Different Approach

Another way to look at the behavior of Quicksort is to consider cost from the point of view of each possible pair of keys. In this approach, the expected cost is the sum over all pairs of

probability that there is a comparison involving this pair $\times$ cost of 1 comparison

What is the probability that a comparison occurs between x_i and x_j, the i-th and j-th key in the final sorted order (assume $i < j$)?

Consider the set of keys $\{x_i, x_{i+1}, \ldots, x_{j-1}, x_j\}$. As soon as any one of these keys is chosen as pivot, x_i and x_j will no longer be in the same partition and can no longer be compared. The last time all keys of the set occur in the same partition, there is a probability of $2/(j-i+1)$ that x_i or x_j is the pivot and therefore x_i and x_j will be compared. In all other cases, they will never be compared.

So the expected number of comparisons is

$$\sum_{\text{all pairs } (i,j)} 2/(j-i+1) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} 2/(j-i+1)$$

$$\text{letting } k = j-i \qquad = \sum_{i=1}^{n-1} \sum_{k=1}^{n-i} 2/(k+1) \leq \sum_{i=1}^{n-1} \sum_{k=1}^{n} 2/k = 2(n-1)H_n \in O(n \lg n)$$