What do we know about Quicksort?

a brief summary of facts learned since the 1960s

Robert Sedgewick
Princeton University
What do we know about Quicksort?

C.A.R. Hoare invented it in 1959 and wrote a definitive paper introducing it in 1962.

"There's one good thesis left in Quicksort" [Knuth, 1973]

"I hope this is the last paper we see about Quicksort" [Anonymous, 1975]

This talk: Some important facts learned since the 1960s

• Students should read Hoare's paper (still).
• Randomization is critical (and must be done carefully).
• Attention must be paid to equal keys.
• Quicksort is optimal.
• Substantial improvements are possible for a common key type.
• The limiting distribution is not normal
• Multiway partitioning is effective on modern machines.
• Quicksort remains the method of choice.

next talk
Students should read Hoare's paper (still)

Why? It is a quintessential example of algorithm science.

Algorithm science

The application of the scientific method to the design and analysis of algorithms

- Create a mathematical model.
- Develop hypotheses about real-world performance.
- Run experiments to validate the hypotheses.
- Iterate on the basis of facts learned.

The theory of algorithms is typically not algorithm science

TofA: "Running time is $O(N \log N)$"  ❌

AS: "Running time is $\sim cN \ln N$ for some constant $c$"  ✓
Students should read Hoare's paper (still)

Why? It is a quintessential example of **algorithm science**.

**Theorem.** Hoare (1961): Quicksort running time for \( N \) random inputs is

\[
\sim cN \ln N
\]

where \( c \) is a *machine-dependent constant*.

**Hypothesis.** Doubling \( N \) will increase the running time by a factor of about \( 2 + (2 \ln 2)/\ln N \).

Proof:

\[
\frac{2cN \ln(2N)}{cN \ln N} = 2 + \frac{2 \ln 2}{\ln N}
\]

<table>
<thead>
<tr>
<th>( N )</th>
<th>( T_N )</th>
<th>( T_N/T_{N/2} )</th>
<th>( 2 + (2 \ln 2)/\ln N )</th>
</tr>
</thead>
<tbody>
<tr>
<td>500</td>
<td>81</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1000</td>
<td>188</td>
<td>2.32</td>
<td>2.12</td>
</tr>
<tr>
<td>2000</td>
<td>407</td>
<td>2.16</td>
<td>2.18</td>
</tr>
</tbody>
</table>

**Validation.**

Results can be (and have been) validated in countless real-world applications ever since.
Randomization is critical (and must be done carefully).

Analysis assumes

- Partitioning element is chosen at random.
- Keys are distinct.
- Subfiles after partitioning are randomly ordered.

Early (problematic) examples

- Too-clever partitioning on the median of three.
- Partition on the value of a random element.

Developer: Can't randomize—too difficult to test and maintain.

Response: Randomly permute input before the sort.

Bigger problem: Keys are not necessarily distinct in real applications.
Attention must be paid to equal keys

Initial goal: avoid quadratic performance (happens in typical implementations if all keys are equal)

Two easy fixes (RS, 1977)
- Stop pointers on equal keys (trivial to implement).
- Three-way partitioning (worth the expense?).

Elegant fix (Bentley-McIlroy, 1993)
- Swap equal keys to ends, then swap into place.
- Uses just N-1 (three-way) compares.
- Only one "extra" exchange per equal key.
- Still in use today.

in response to a user complaint

```c
void quicksort(Item a[], int l, int r)
{ int i = l-1, j = r, p = l-1, q = r; Item v = a[r];
  if (r <= l) return;
  for (;;) {
    while (a[++i] < v)
      while (v < a[--j]) if (j == l) break;
    if (i >= j) break;
    exch(a[i], a[j]);
    if (a[i] == v) { p++; exch(a[p], a[i]); }
    if (v == a[j]) { q--; exch(a[j], a[q]); }
  }
  exch(a[i], a[r]); j = i-1; i = i+1;
  for (k = l; k < p; k++, j--) exch(a[k], a[j]);
  for (k = r-1; k > q; k--, i++) exch(a[i], a[k]);
  quicksort(a, l, j);
  quicksort(a, i, r);
}
```
Quicksort is optimal

Starting point: \( N \) keys having \( n \) different values with multiplicities \( x_1, x_2, \ldots, x_n \).

Assume keys are randomly ordered and that three-way partitioning is used.

Let \( C \) be the average number of compares.

Lower bound (information theory): \[ C > NH - N \quad \text{where} \quad H = - \sum_{1 \leq i \leq n} p_i \lg p_i \quad \text{with} \quad p_i = x_i/N \]

Exact value (Sedgewick, 1975): \[ C = N - n + 2QN \quad \text{where} \quad Q = \sum_{1 \leq k < j \leq n} \frac{p_k p_j}{p_k + \ldots + p_j} \]

**Theorem.** [Bentley-Sedgewick] *Average number of compares is within a constant factor of optimal.*

Proof: \( Q < H \ln 2 \)

Substantial improvements are possible for a common key type.

Suppose that keys are strings (sequences of characters)

3-way string Quicksort (Bentley-Sedgewick, 1997)
- 3-way partition on first character of each key.
- Recursively sort 3 subfiles.
- Use substring excluding first char in the middle.
- Sorts random strings with $2N \ln N$ character compares
- Optimal

Ternary search trees
- Apply same idea to binary search trees.
- Simple algorithm, writes itself
- Faster than hashing.
The limiting distribution is not normal (and is a challenge to characterize)

What is the \textit{limiting distribution} of the number of compares?

If it exists, it is not normal (Henniquin, 1987).

It exists (Regnier, 1989).

Is a unique fixed point of an explicit distributional identity (Rosler, 1991).

Has extremely small tails (several authors, 1991-2015)

Explicit tail bounds (Janson, 2015):

\[
\exp\left(-e^{\gamma x + \ln \ln x + O(1)}\right) \leq \Pr(Z \leq -x) \leq \exp\left(-e^{\gamma x + O(1)}\right) \quad \gamma = \frac{1}{2} - \frac{1}{\ln 2}
\]

\[
\exp\left(-x \ln x - x \ln \ln x + O(x)\right) \leq \Pr(Z \geq x) \leq \exp\left(-x \ln x + O(x)\right)
\]
What do we know about Quicksort?

C.A.R. Hoare invented it in 1959 and wrote a definitive paper introducing it in 1962.

Some important facts learned since the 1960s

- Students should read Hoare's paper (still).
- Randomization is critical (and must be done carefully).
- Attention must be paid to equal keys.
- Quicksort is optimal.
- Substantial improvements are possible for a common key type.
- The limiting distribution is not normal.
- Multiway partitioning is effective on modern machines.
- Quicksort remains the method of choice.

next talk