Comparison-based sorting

Problem: Given items $a_1, \ldots, a_n$, sort the $a_i$’s in non-decreasing order

Assume only that items belong to some totally ordered universe

Measure cost by # of comparisons

Heapsort: an $O(n \log n)$ sorting algorithm

Build a heap — cost $(n)$

Perform $n$ Delete Min’s

- each Delete Min has cost $O(\log n)$
- total cost: $O(n \log n)$
Mergesort: a recursive $O(n \log n)$ algorithm

Basic step: merge two sorted lists

\[
\begin{array}{cccc}
1 & 3 & 6 & 9 \\
2 & 4 & 6 \\
\hline
3 & 6 & 9 \\
2 & 4 & 6 \\
\hline
3 & 6 & 9 \\
2 & 4 & 6 \\
1 \\
\hline
6 & 9 \\
6 \\
\hline
1 & 2 \\
1 & 2 \\
1 & 2 \\
6 & 9 \\
6 \\
\hline
1 & 2 & 3 & 4 \\
1 & 2 & 3 & 4 \\
1 & 2 & 3 & 4 \\
1 & 2 & 3 & 4 \\
1 & 2 & 3 & 4 \\
1 & 2 & 3 & 4 \\
1 & 2 & 3 & 4 & 6 & 9
\end{array}
\]

If lengths are $n_1$ and $n_2$, cost of merge is $n_1 + n_2 - 1$
MergeSort(⟨a₁, . . . , aₙ⟩):

if \( n = 1 \) then
    return ⟨a₁⟩
else
    \( L_1 \leftarrow \text{MergeSort}(⟨a₁, . . . , a_{\lfloor n/2 \rfloor}⟩) \)
    \( L_2 \leftarrow \text{MergeSort}(⟨a_{\lfloor n/2 \rfloor}+1, . . . , aₙ⟩) \)
    return Merge(\( L_1, L_2 \))

Recursion tree — example \( n = 6 \):

- Each node has an associated size and cost
- Goal: add up all costs
Recursion tree analysis for mergesort

- If the size of a node is $m$, its cost is $m - 1$
- If the size of a node is $m > 1$, the size of its children are $\lfloor m/2 \rfloor$ and $\lceil m/2 \rceil$, which sum to $m$
- Let
  - $h :=$ height of recursion tree
  - $s_j :=$ sum of sizes at level $j = 0 \ldots h$
  - $w_j :=$ sum of costs at level $j = 0 \ldots h$
  - $w := \sum_{i=0}^{h} w_j =$ total cost

- $s_j \leq n$ for $j = 0 \ldots h$
- $w_j \leq s_j$ for $j = 0 \ldots h - 1$ and $w_h = 0$
• $w \leq \sum_{i=0}^{h-1} s_j \leq nh$

• Useful fact: if $m \leq 2^k$, where $k \geq 1$, then
  $\lfloor m/2 \rfloor \leq 2^{k-1}$ and $\lceil m/2 \rceil \leq 2^{k-1}$

• Let $k = \lceil \log_2 n \rceil$, so $n \leq 2^k$

• For $j = 0 \ldots h$, size of each node at level $j$ is
  $\leq 2^{k-j}$

• $h \leq \lceil \log_2 n \rceil$, and $w \leq n\lceil \log_2 n \rceil$
Quick Sort: a randomized sorting algorithm

**QSort(L):**

if $|L| \leq 1$ then
    return $L$
else
    choose $p$ from $L$ at random
/* $p$ is the “pivot” */
partition $L$ into 3 sublists: $L_{<p}, L_{=p}, L_{>p}$
return $QSort(L_{<p}) \parallel L_{=p} \parallel QSort(L_{>p})$

Correctness: clear
The “Quick” in “Quick Sort” comes from a very tight inner loop

An idea from Bentley & McIlroy (1993)

Two inner loops:
- moving $b$: scan over $<$, swap $=$, halt on $>$
- moving $c$: scan over $>$, swap $=$, halt on $<$

Swap elements $b$ and $c$, $b++$, $c--$

Repeat until $b$ crosses $c$

When finished, the $=$’s are swapped to the middle
This code uses $n - 1$ comparisons to partition.

Using a ternary partitioning scheme is necessary to guarantee good performance when there are a large number of duplicates.

### Recursion Analysis

Consider the recursion tree.

If a node has size $m$, the cost of that node is $\leq m - 1$, and the sizes of its children sum to $\leq m - 1$.

At level $j$, the sizes at level $j$ sum to at most $n - j$.

Height of recursion tree is $\leq n - 1$.

Total cost is $\leq n^2$. 
Define $W := \text{total cost}$

This is a random variable

**Theorem.** $E[W] = O(n \log n)$

Intuition: with every partition, we expect to get an even split, and so it should behave roughly like Merge Sort

Details — more complicated!

Several different proofs . . .

Here is a “slick” one, due to Alan Siegel
\( N_i := \) size of node \( i \)

\( \mathcal{L}_j := \) set of indices at level \( j \)

\[ \mathcal{L}_j = \{2^j, \ldots, 2^{j+1} - 1\} \]

\( S_j := \sum_{i \in \mathcal{L}_j} N_i \), \( T_j := \sum_{i \in \mathcal{L}_j} N_i^2 \)

The \( N_i \)'s, \( S_j \)'s, and \( T_j \)'s are \textit{random variables}

Let \( k \geq 0 \) be a parameter

**Key observation:** \( W \leq nk + T_k \)

- Cost at each level \( j = 0, \ldots, k - 1 \) is \( \leq S_j \leq n \)
- For each \( i \in \mathcal{L}_k \), the total cost of the tree rooted at \( i \) is \( \leq N_i^2 \) (worst-case analysis)
\[ W \leq nk + T_k \implies \mathbb{E}[W] \leq nk + \mathbb{E}[T_k] \]

**Strategy:** show that we can choose \( k \) such that

1. \( k = O(\log n) \), and
2. \( \mathbb{E}[T_k] = O(n) \)

**Claim 1.** \( \mathbb{E}[T_j] \leq (2/3)^j n^2 \) for \( j = 0, 1, \ldots \)

Assume Claim 1 for the moment

Set \( k := \lceil \log_{3/2} n \rceil \)

Then \( (2/3)^k \leq 1/n \), and so \( \mathbb{E}[T_k] \leq n \)

That proves the theorem, assuming Claim 1

**Claim 2.** \( \mathbb{E}[T_{j+1}] \leq (2/3)\mathbb{E}[T_j] \) for \( j = 0, 1, \ldots \)

Claim 1 follows immediately from Claim 2, and the fact that \( \mathbb{E}[T_0] = n^2 \)
Let’s first prove that $E[T_1] \leq (2/3)n^2$

$T_1 = N_2^2 + N_3^2$

Imagine the items are in an array $A[1..n]$ in sorted order

Let $R$ be the index of the pivot in $A$

$R$ is uniformly distributed over $\{1, \ldots, n\}$

$N_2 \leq R - 1$ and $N_3 \leq n - R$

$$E[(R-1)^2] = \sum_{i=1}^{n} (i-1)^2/n = \frac{1}{n} \sum_{i=0}^{n-1} i^2$$

$$\leq \frac{1}{n} \int_{0}^{n} x^2 \, dx = \frac{1}{n} \cdot \frac{n^3}{3} = \frac{n^2}{3}$$
The distribution of $n - R$ is the same as that of $R - 1$

Thus, $E[N_2^2] \leq n^2/3$, $E[N_3^2] \leq n^2/3$, and

$$E[T_1] = E[N_2^2] + E[N_3^2] \leq (2/3)n^2$$

More generally, consider any node $i$ in the tree

**Conditioning argument:**

$$E[N_{2i}^2] = \sum_m E[N_{2i}^2 \mid N_i = m] \Pr[N_i = m]$$

$$\leq \sum_m (m^2/3) \Pr[N_i = m] = (1/3)E[N_i^2]$$

Similarly, $E[N_{2i+1}^2] \leq (1/3)E[N_i^2]$

Claim 2 is now clear (by linearity of expectation)
Selection

**General problem:** Given a list $L$ of $n$ items, and $k \in \{1, \ldots, n\}$, find $k$th smallest element in $L$

**Quick Select:** a randomized selection algorithm

$QSelect(L)$:

choose $p$ from $L$ at random
partition $L$ into 3 sublists: $L_{<p}, L_{=p}, L_{>p}$
if $|L_{<p}| \geq k$ then
    return $QSelect(L_{<p}, k)$
else if $|L_{<p}| + |L_{=p}| \geq k$ then
    return $p$
else
    return $QSelect(L_{>p}, k - |L_{<p}| - |L_{=p}|)$
Let $W := \text{cost of algorithm}$

**Theorem.** $E[W] = O(n)$

For $j = 0, 1, 2, \ldots$ let $N_j := \text{size of the subproblem at level } j \text{ (or zero if none)}$

**Claim.** $E[N_j] \leq \alpha^j n$ for some constant $\alpha < 1$, and for each $j = 0, 1, 2, \ldots$

Using the claim, we have:

- $W \leq N_0 + N_1 + \cdots$
- $E[W] \leq E[N_0] + E[N_1] + \cdots \leq n \sum_{j \geq 0} \alpha^j = (1/(1 - \alpha))n$
It suffices to show that $E[N_{j+1}] \leq \alpha E[N_j]$ for each $j = 0, 1, 2, \ldots$

Let’s deal with $j = 0$ — the rest follows by a “conditioning argument,” as in Quick Sort.

Consider level 0, and let $S$ be the size of $|L_{<\rho}|$ and $T$ the size of $|L_{>\rho}|$

We want to show $E[N_1] \leq \alpha n$

$$N_1 \leq \max(S, T) \leq \sqrt{S^2 + T^2}$$

$$E[N_1] \leq \sqrt{E[S^2 + T^2]} \quad (E[X]^2 \leq E[X^2])$$

$$\leq \sqrt{(2/3)n^2} \quad \text{(by Quick Sort analysis)}$$

$$= \sqrt{2/3 \cdot n}$$
That proves the claim with $\alpha = \sqrt{2/3} \approx 0.816$

Analysis was a bit sloppy — claim holds with $\alpha = 3/4$
Deterministic linear-time selection

Idea:

1. divide $L$ into $\approx n/5$ blocks of size 5
2. sort each block, and compute median of each block
3. let $M :=$ the list of medians (so $|M| \approx n/5$)
4. recursively find the median $p$ of $M$
5. use $p$ as the pivot, and proceed as in Quick Select
Consider a single recursive invocation

Local cost is $O(n)$

Both $|L_{<p}|$ and $|L_{>p}|$ are $\leq (7/10)n + O(1)$

Two recursive calls:

- one of size at most $n/5 + O(1)$
- one of size at most $(7/10)n + O(1)$
Sum of subproblem sizes is $\leq 0.9n + c$, for some constant $c$

Choose $n_0$ such that $0.9n + c \leq 0.91n$ for all $n \geq n_0$

Implementation: halt recursion when $n < n_0$

Let $s_j :=$ sum of sizes at level $j$, for $j = 0, 1, 2, \ldots$

We have $s_j \leq (0.91)^jn$ for $j = 0, 1, 2, \ldots$

If total cost is $w$, then for some constant $d$:

$$w \leq d \sum_{j \geq 0} s_j \leq d \sum_{j \geq 0} (0.91)^jn \leq (100/9)dn$$