## Question 1: Empirical Analysis of Sorting Algorithms

We compare the worst-case running time (reverse-sorted input) of Insertion Sort ($\mathcal{O}(n^2)$) and Merge Sort ($\mathcal{O}(n \log n)$).

Table 1: Worst-Case Execution Times

Input Size (n)	Insertion Sort (s)	Merge Sort (s)
1000	0.0043	0.0011
2000	0.0136	0.0014
4000	0.0347	0.0020
8000	0.1020	0.0034
10000	0.1585	0.0042
15000	0.3512	0.0062
18000	0.4958	0.0077
20000	0.6255	0.0085

Analysis: Merge Sort consistently outperforms Insertion Sort for $n \geq 1000$. The quadratic growth of Insertion Sort is evident as doubling the input size (e.g., $10k \rightarrow 20k$) results in an approximate $4\times$ increase in time ($0.15s \rightarrow 0.62s$). In contrast, Merge Sort exhibits near-linear growth consistent with $n \log n$. The crossover point occurs at very small n (likely $n < 1000$ given the overhead of recursion versus simple loops), but for the tested range, Merge Sort is strictly superior.
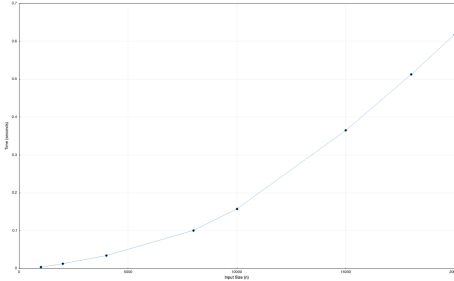
(a) Insertion Sort Growth
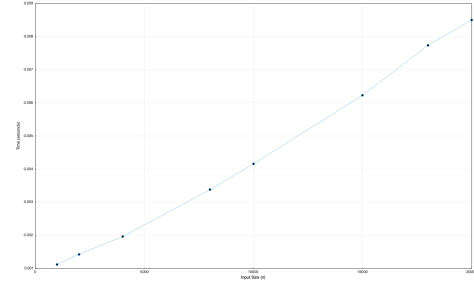
(b) Merge Sort Growth

Figure 1: Empirical Runtime Analysis

Question 2: Quicksort Stability and Input Sensitivity

We analyze Quicksort on sorted inputs versus shuffled inputs ($n = 10,000$).

- **Worst-Case (Sorted Input):** 0.205513 seconds

- **Average-Case (Shuffled Input):** 0.000922 seconds

Analysis: The standard Quicksort implementation (using the last element as the pivot) degrades to $\mathcal{O}(n^2)$ when the input is already sorted, as the partition partitions the array into size $n - 1$ and 0. This explains the high runtime of $\approx 0.2s$. Shuffling the array prior to sorting restores

the probabilistic guarantee of balanced partitions, reducing the complexity to expected $\mathcal{O}(n \log n)$ and the runtime to $\approx 0.0009s$.
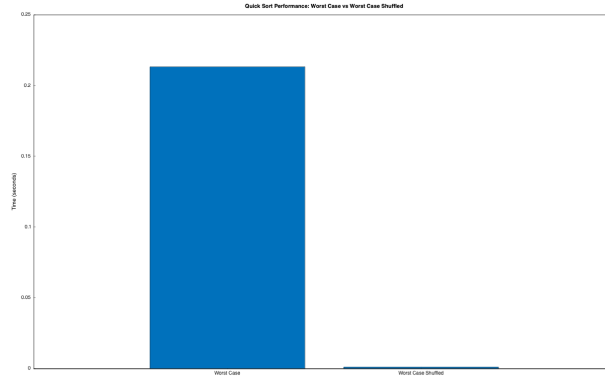

Figure 2: Impact of Input Distribution on Quicksort Performance

Question 3: Asymptotic Analysis

3.1. Prove $n + 3 \in \Omega(n)$

Definition: $f(n) \in \Omega(g(n))$ if $\exists c, n_0 > 0$ such that $0 \leq c \cdot g(n) \leq f(n)$ for all $n \geq n_0$.
   Let $f(n) = n + 3$ and $g(n) = n$. We require $n + 3 \geq c \cdot n$. Choosing $c = 1$ and $n_0 = 1$:

$$n + 3 \geq 1 \cdot n \implies 3 \geq 0 \quad \text{(True for all } n \geq 1).$$

Thus, constants $c = 1, n_0 = 1$ satisfy the condition. □

3.2. Prove $n + 3 \in \mathcal{O}(n^2)$

Definition: $f(n) \in \mathcal{O}(g(n))$ if $\exists c, n_0 > 0$ such that $0 \leq f(n) \leq c \cdot g(n)$ for all $n \geq n_0$.
   We require $n + 3 \leq c \cdot n^2$. Dividing by n^2: $\frac{1}{n} + \frac{3}{n^2} \leq c$. The LHS is a decreasing function for $n \geq 1$. At $n = 1$, LHS $= 4$. At $n = 3$, LHS $= 1/3 + 1/3 \approx 0.66 \leq 1$. Choosing $c = 1$ and $n_0 = 3$:

$$n + 3 \leq n \cdot n + 3 \leq n^2 \quad \text{False for general } c.$$

Let's strictly evaluate $n+3 \leq 1 \cdot n^2$ for $n \geq 3$: $3^2 = 9, 3+3 = 6$. $6 \leq 9$. Since n^2 grows quadratically and n linearly, this holds. Thus, $c = 1, n_0 = 3$ satisfy the condition. □

3.3. Prove/Disprove $n + 3 \in \Theta(n^2)$

Definition: $f(n) \in \Theta(g(n))$ if $f(n) \in \mathcal{O}(g(n))$ AND $f(n) \in \Omega(g(n))$.
   From 3.2, we know $n + 3 \in \mathcal{O}(n^2)$. We check if $n + 3 \in \Omega(n^2)$. This requires $n + 3 \geq c \cdot n^2$ for large n.

$$\lim_{n \to \infty} \frac{n + 3}{n^2} = 0.$$

Since the limit is 0, $n + 3$ is strictly asymptotically smaller than n^2. No constant $c > 0$ exists to satisfy the lower bound for large n. Therefore, $n + 3 \notin \Omega(n^2)$, implying $n + 3 \notin \Theta(n^2)$. □

3.4. Prove/Disprove $2^{n+1} \in \mathcal{O}(n+1)$

We compare exponential growth 2^{n+1} to linear growth $n+1$.

$$\lim_{n\to\infty} \frac{2^{n+1}}{n+1} = \lim_{n\to\infty} \frac{2 \cdot 2^n}{n+1} = \infty \quad \text{(by L'Hopital's rule)}.$$

Since the ratio diverges, 2^{n+1} is not upper-bounded by $c(n+1)$. False.   □

3.5. Prove $2^{n+1} \in \Theta(2^n)$

We require $c_1 2^n \leq 2^{n+1} \leq c_2 2^n$. Rewrite $f(n)$: $2^{n+1} = 2 \cdot 2^n$. This is exactly $2 \cdot g(n)$. By choosing $c_1 = 2$ and $c_2 = 2$ (or $c_1 = 1, c_2 = 3$), the condition holds for all $n \geq 1$. Thus, $2^{n+1} \in \Theta(2^n)$.   □

Question 4: The Master Theorem

The Master Theorem applies to recurrences of the form $T(n) = aT(n/b) + f(n)$. Here, $a = 8, b = 2$. The critical exponent is $\log_b a = \log_2 8 = 3$.

4.1. $f(n) = n$

$f(n) = n^1 = \mathcal{O}(n^{\log_b a - \epsilon})$ with $\epsilon = 2$. Case 1 applies: $T(n) = \Theta(n^{\log_b a}) = \Theta(n^3)$.

4.2. $f(n) = n^2$

$f(n) = n^2 = \mathcal{O}(n^{3-\epsilon})$ with $\epsilon = 1$. Case 1 applies: $T(n) = \Theta(n^3)$.

4.3. $f(n) = n^3$

$f(n) = \Theta(n^{\log_b a})$. Case 2 applies: $T(n) = \Theta(n^{\log_b a} \log n) = \Theta(n^3 \log n)$.

4.4. $f(n) = n^4$

$f(n) = \Omega(n^{3+\epsilon})$ with $\epsilon = 1$. Regularity condition: $af(n/b) = 8(n/2)^4 = 8(n^4/16) = \frac{1}{2}n^4 \leq cn^4$ holds for $c = 1/2 < 1$. Case 3 applies: $T(n) = \Theta(f(n)) = \Theta(n^4)$.

Question 5: Recurrence Tree and Substitution Method

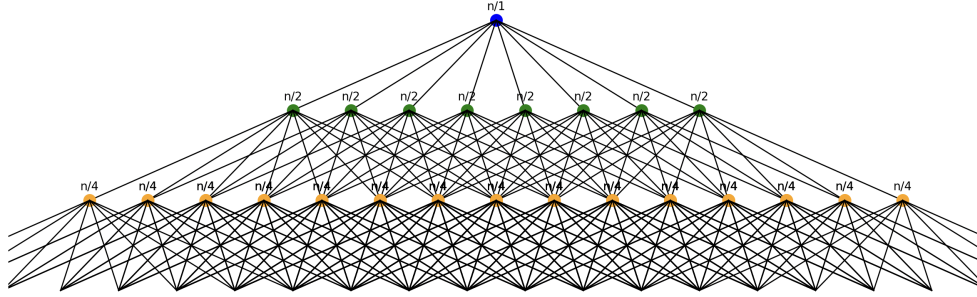Recurrence: $T(n) = 8T(n/2) + n$.

5.1 Recursion Tree Analysis

Figure 3: Recursion Tree for $T(n) = 8T(n/2) + n$

The cost at depth k is the number of nodes (8^k) times the cost per node $(\frac{n}{2^k})$.

$$\text{Cost}_k = 8^k \cdot \frac{n}{2^k} = \left(\frac{8}{2}\right)^k n = 4^k n.$$

The tree height is $L = \log_2 n$. The total cost is the sum over all levels:

$$T(n) = \sum_{k=0}^{\log_2 n} 4^k n = n \sum_{k=0}^{\log_2 n} 4^k.$$

This is a geometric series $\sum_{k=0}^{L} r^k = \frac{r^{L+1}-1}{r-1}$ with $r = 4$.

$$T(n) = n \left(\frac{4^{\log_2 n + 1} - 1}{4 - 1}\right) = \frac{n}{3}\left(4 \cdot 4^{\log_2 n} - 1\right).$$

Note that $4^{\log_2 n} = (2^2)^{\log_2 n} = (2^{\log_2 n})^2 = n^2$.

$$T(n) = \frac{n}{3}(4n^2 - 1) = \frac{4}{3}n^3 - \frac{1}{3}n.$$

Thus, the guess is $T(n) = \Theta(n^3)$.

5.2 Proof via Substitution Method

We wish to prove $T(n) \leq cn^3$ for some constant c. However, direct substitution fails for the exact form cn^3 because of the linear term. Assume a stronger hypothesis to handle lower-order terms: $T(n) \leq cn^3 - dn$ for constants $c, d > 0$.

    Inductive Step: Assume $T(k) \leq ck^3 - dk$ for $k < n$.

$$T(n) = 8T(n/2) + n$$
$$\leq 8\left(c\left(\frac{n}{2}\right)^3 - d\left(\frac{n}{2}\right)\right) + n$$
$$= 8\left(\frac{cn^3}{8} - \frac{dn}{2}\right) + n$$
$$= cn^3 - 4dn + n$$
$$= cn^3 - (4d - 1)n.$$

4

We require this to be $\leq cn^3 - dn$.

$$cn^3 - (4d - 1)n \leq cn^3 - dn$$
$$-(4d - 1)n \leq -dn$$
$$4d - 1 \geq d$$
$$3d \geq 1 \implies d \geq 1/3.$$

This holds for any $d \geq 1/3$ and any $c > 0$ sufficient to cover base cases. Therefore, $T(n) = \mathcal{O}(n^3)$.