Lower Bounds and Impossibility Results

Ian Chen

July 5, 2026

1 Introduction

The biggest question in computer science is whether P = NP. We know that P $\leq$ NP, i.e. NP is stronger, but the difficult question is to prove the impossibility of a polynomial time algorithm for some NP-Hard problem.

But P vs NP is not what I want to focus on. There are several famous lower-bounds and impossibility results that I have seen as exercises in undergraduate algorithms. Here is an organized compilation of some informative ones.

2 Adversaries

Given an algorithm $\mathcal{A}$, an adversary $\mathcal{B}$ would construct an instance to a problem that $\mathcal{A}$ does poorly on. Analyzing these adversarial instances is one way of proving the existence of difficult instances, lower-bounding the *worst-case performance* of $\mathcal{A}$. This section contains illustrative two exercises from the Fall 2025 instance of UIUC's CS 473, and the Fall 2024 offering of Princeton's COS 521 respectively.

2.1 Majority Trees

A majority tree is a perfect ternary tree of depth n, where nodes are labeled with 0 or 1. Each internal node is assigned the label of the majority of its children. For any deterministic algorithm that takes as input the leaves and outputs the label of the root, we will show that it must examine every leaf, i.e. lower-bounding its runtime to $\Omega(3^n)$.

Adversaries are more powerful than algorithms: if $\mathcal{A}$ is a Turing machine, then $\mathcal{B}$ is a universal Turing machine — it can simulate $\mathcal{A}$ and use that to construct bad instances. For any $\mathcal{A}$, let x_{k+1} be the $(k+1)$-th leaf that it looks at, which is a function (as $\mathcal{A}$ is deterministic) of $(x_1, \ldots, x_k)$ the previous leaves it looked at, as well as $(y_1, \ldots, y_k)$ their respective values[1]. Let $\mathcal{B}_i, i \in \{0, 1\}$ be adversaries, setting $y_k = f_i(x_k \mid x_1, \ldots, x_{k-1}, y_1, \ldots, y_{k-1})$, where

$$f_i(x \mid (x_1, y_1), \ldots, (x_{k-1}, y_{k-1})) = \begin{cases} i & \text{if } x \text{ is the root} \\ 0 & \text{elif no siblings of } x \text{ has value 0} \\ 1 & \text{elif no siblings of } x \text{ has value 1} \\ f_i(\textit{parent}(x)) & \text{otherwise} \end{cases}$$

Now, these two instances are both valid majority trees. Moreover, they are *identical* until x_{3^n} is chosen, and thus, any $\mathcal{A}$ that makes a decision solely based on $(x_1, y_1), \ldots, (x_{3^n-j}, y_{3^n-j})$ for $j > 0$ must be wrong on at least one of the two instances (as they have different values for the root). Thus, if $\mathcal{A}$ is a correct algorithm, it must examine all 3^n leaves, which shows our lower bound on $\Omega(3^n)$ runtime.

Note that there is a simple Las-Vegas algorithm that beats this in expectation: pick the leaves in random order. If the two chosen leaves are the same value (i.e. with probably $1/3$), then we no longer have to examine the last leaf, and

[1] We are *not* assuming anything about $\mathcal{A}$ besides that it is a deterministic function of the input; in particular, we don't assume that it computes the value of any interior node. This is perhaps why adversary arguments are so general.

this holds for any (potentially adversarial) instance. Indeed, this achieves an expected runtime of $(8/3)^n$.

2.2 Sampling

A natural subroutine in many randomized algorithms is to compute a value over a sample as a heuristic. For example, for median-finding using quick-select, using a random uniformly pivot takes $\lesssim 3.4n$ expected comparisons, picking a random pivot according to a subsampled array brings the factor down to $\lesssim 1.5n$ in expectation. Formally,

1. Sample $r = (cn)^{2/3}$ elements from the array
2. Set $a \leftarrow (r/2) - c\sqrt{r}$ and $b \leftarrow (r/2) + c\sqrt{r}$ ranks from sampled array
3. Partition into L, M, R by $x \le a$, $a < x \le b$, and $b < x$
4. Re-sample if $|L| > n/2$ or $|R| > n/2$ or $|M| > 4cn/\sqrt{r}$, else recurse on M

which takes $1.5n + o(n)$ comparisons (e.g. bounding the re-sampling probability using Chebyschev).

Random sampling may be good at approximating ranks, but actually gives no approximation guarantees on the values. Consider an algorithm that samples each element with probability p, where $p = o(1/n)$, and returns an estimate based on the sample. We will prove that any such algorithm must be off by a factor of 2 with probability $> 1/3$ (for sufficiently large n, for some instance).

Consider an array with only two values x and y, say $x = 1$ and $y = 10$. The adversary constructs two arrays of size $2n$: X with $(n+1)$ xs and $(n-1)$ ys, and Y with $(n-1)$ xs and $(n+1)$ ys. If $\mathbf{Pr}(\mathcal{A}(X) \text{ bad approx}) \ge 1/2$, then X is our bad instance. Otherwise, Y must be our bad instance, because with probability $\ge (1 - \frac{1}{2})(1 - p^2) > 1/3$, the samples from X and Y are identical, and so it returns an estimate close to the median of X, bad for Y.

3 Information Theoretic Bounds

Let $X = \{x_1, \ldots, x_n\}$ be a set of n symbols, encoded and decoded to be sent across some (noiseless) channel. Shannon's source coding theorem says that $\mathbf{E}(|\text{Enc}(X)|) \ge H(X) = \sum_{i=1}^n p(x_i) \log_2 \frac{1}{p(x_i)}$; the entropy is a lower bound on the expected bits sent across channel. This is widely useful across algorithms, for example, to show that in the comparison model, it takes $\Omega(n \log n)$ comparisons to sort an array. Indeed, the result of these comparisons leads to an encoding/decoding algorithm, where setting X to be all $n!$ permutations of the array with uniform distribution, then the entropy is $H(X) = \log_2(n!) = \Omega(n \log n)$. Thus, at least one permutation takes at least $\log_2(n!)$ comparisons[2].

Proving this bound is a straightforward packing argument. Clearly, any candidate algorithm encodes any symbol using $\le B = H(X)/p_{\min}$ bits. Now, to decode x_i, which was encoded into b_i bits, is akin to taking b_i steps down a perfect binary tree of depth B (moving left/right based on 0/1); all leaves in that subtree get the same symbol x_i.

$$\underbrace{2^B}_{\text{number of leaves}} \ge \sum_{i=1}^n \underbrace{2^{B-b_i}}_{\text{leaves with } x_i \text{ symbol}} \implies 1 \ge \sum_{i=1}^n 2^{-b_i}$$

Relaxing with Lagrange multipliers, $b_i^* \propto \log_2 \frac{1}{p_i}$, and we recover $H(X)$ lower bound.

$$\min \sum_{i=1}^n p_i b_i = f(\boldsymbol{b}) \qquad \text{s.t.} \qquad 1 - \sum_{i=1}^n 2^{-b_i} = g(\boldsymbol{b}) \ge 0$$

$$p_i = \frac{\partial f(\boldsymbol{b})}{\partial b_i} = \lambda \frac{\partial g(\boldsymbol{b})}{\partial b_i} = -\log(2) 2^{-b_i}$$

[2]When we leave the comparison model, we can still make an analogy to encoding/decoding based on decision trees. For example, hashing-based comparison algorithms break this bound, by having each query give more than 2 outcomes!

4 No Free Lunch

4.1 Supervised Learning

The fundamental theorem of learning theory states the equivalence, for $0-1$ hypothesis classes, of

1. PAC learnability
2. Agnostic PAC learnability
3. Finite VC-dimension

where the two non-trivial arguments are for finite VC-dimension implies agnostic PAC learnability (by ϵ-nets), and PAC learnability implies finite VC-dimension. This second result, where infinite VC-dimension hypothesis classes cannot be PAC-learned, is our No-Free lunch theorem that we shall prove.

Let $\mathcal{H}$ be a hypothesis class of infinite VC-dimension, $\mathcal{A}$ an arbitrary learner, and fix m a sample size. We construct $\mathcal{D}$ a dataset of size $2m$ that is shattered by $\mathcal{H}$, i.e. so we have $\mathcal{F}$ the 2^{2^m} different labelings. Now, there must be one labeling that has an error of at least $1/4$, since the average error on all points not in S is exactly $1/2$, and there are $m/2m$ such unseen points. Finally, with the reverse Markov inequality,

$$\Pr_S\left(\epsilon_D(\mathcal{A}(S)) \leq \frac{1}{8}\right) \leq \frac{1-1/4}{1-1/8} = 1 - \frac{1}{7}$$

so $\mathcal{D}$ cannot be PAC-learned – being approximately correct (an error of $\leq 1/8$), is not probable ($\leq 1 - 1/7$).

4.2 Unsupervised Learning

In clustering, we are given a set of points S, as well as a distance function $d : S \times S \to \mathbb{R}$, and we output $f(S, d) = \mathcal{P}$ some partition of S. In the absence of a clear evaluation for clustering, Kleinberg [3] considers the following natural axioms for clustering algorithms[3]:

1. Scale invariance: $f(S, d) = f(S, \alpha d)$
2. Richness: $\forall P, \exists d$ s.t. $f(S, d) = P$
3. Consistency: If $f(S, d) = P$, then $f(S, d') = P$, for all d' that satisfy

$$i \sim j \implies d'(i, j) \leq d(i, j), \qquad i \not\sim j \implies d'(i, j) \geq d(i, j)$$

Indeed, these axioms are debated[4], mostly due to the fact that no clustering algorithm can satisfy all three axioms.

By richness, let S be a set, and pick d_1 so that $f(S, d_1)$ is all singletons, and $f(S, d_2)$ a different partition. Then, construct $d_3 = \alpha d_2$, so that d_3 is consistent with d_1, e.g. $\alpha = \max d_1(i, j) / \min d_2(i, j)$. We have a contradiction,

$$f(S, d_3) \underset{\text{scale}}{=} f(S, d_2) \underset{\text{construction}}{\neq} f(S, d_1) \underset{\text{consistency}}{=} f(S, d_3)$$

5 Bibliographic Notes

The examples for the adversary section are from the courses references in the text. The quick-select tri-partitioning is a classical algorithm due to Floyd and Rivest [2]. Additional context is in the lower bounds chapter of Jeff Erickson's

[4]The consistency axiom is argued in Ben-David and Ackerman [1] to be overly restrictive. One idea is that f should give a score for each partition, and this score should only improve during this operation of making groups more similar. This admits other potential solutions may now be possible while being consistent. As richness invalidates k-means/k-metroids/k-medians clustering, it is also questioned.

book. Entropy is discussed in Sariel Har-Peled's Randomized Algorithms. The proof of the NFL theorem is in standard machine learning books, such as Shalev-Shwartz and Ben-David [4].

References

[1] Shai Ben-David and Margareta Ackerman. "Measures of clustering quality: A working set of axioms for clustering". In: *Advances in neural information processing systems* 21 (2008).

[2] Robert W. Floyd and Ronald L. Rivest. "Expected time bounds for selection". In: *Communications of the ACM* 18.3 (Mar. 1975), pp. 165–172. ISSN: 1557-7317. DOI: 10.1145/360680.360691. URL: http://dx.doi.org/10.1145/360680.360691.

[3] Jon Kleinberg. "An impossibility theorem for clustering". In: *Proceedings of the 16th International Conference on Neural Information Processing Systems*. NIPS'02. Cambridge, MA, USA: MIT Press, 2002, pp. 463–470.

[4] Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.