Abstract. We present a simplified derivation of the fact that the complexity-theoretic lower bound of comparison-based sorting algorithms, both for the worst-case and for the average-case time measure, is $\Omega(n \log n)$.

The standard proofs typically are directly presented over decision trees. The proof for the average-case however relies on differential calculus, which presents a main hurdle in undergraduate presentations. Here we present a simplified derivation of this result based on the well-known Kraft inequality for binary trees. This inequality enables one to derive the worst-case lower bound via a very simple argument. It also yields the average-case lower bound, via a similar argument as for the worst-case and also involves an elegant and simple inequality obtained by the Danish mathematician Jensen. The Jensen inequality essentially implies that, for convex functions, the function value of a mean of numbers is bounded by the mean of the function values of these numbers.

This approach removes the need to present the results based on differential calculus and makes the material easily accessible for undergraduate Computer Science students.

ACM CCS Categories and Subject Descriptors: F.2: Analysis of Algorithms and Problem Complexity

Key words: Comparison-Based Algorithms, Sorting, Lower Bounds, Complexity
1. Introduction

In many contexts and applications, static timing analysis provides information on the performance of algorithms and of software in general. Worst-case time analysis and in second instance average-case time analysis, form two main timing approaches used in practice. Lower bounds on worst-case time and on average-case time are useful indicators of the limits to which algorithms can be improved. A main class of algorithms for which such lower bounds have been determined are the so-called comparison-based sorting algorithms.

Lower bounds for the worst-case and for the average-case time of comparison based sorting algorithms were established based on decision trees [Aho et al. 1987][Cormen et al. 1997]. The proof of the average-case lower bound relies on differential calculus. In particular it involves the determination of a minimum for a two-valued function [Aho et al. 1987][Cormen et al. 1997]. Its reliance on multivariable calculus makes it unsuitable for teaching at an undergraduate Computer Science level, since this topic no longer forms a standard part of most Computer Science undergraduate degrees. This paper presents a simplified proof for the lower bound on the worst-case and average-case timing analysis of comparison-based sorting algorithms, where the proof of the latter no longer requires more advanced calculus.

In order to make the paper self-contained, we will recall some basic definitions. Moreover, for the sake of completeness, we will include proofs for some well-known classical results since the current presentation is also intended to serve as useful reference material for research students and for undergraduate Computer Science courses. As usual we will carry out worst-case and average-case time analysis for inputs of a fixed size $n$. In doing so, we tacitly assume that there are only finitely many inputs of a given size. Of course in practice there will be infinitely many such inputs. Due to the fact that, for the purpose of time analysis, one identifies inputs which are equivalent up to the relative order between elements, one can safely assume that there are finitely many inputs of a fixed size.

We will only consider algorithms which take lists as inputs in the following. We recall that there are exactly $n!$ lists of size $n$, after identification up to relative order between elements and assuming without loss of generality that lists have pairwise distinct elements [Aho et al. 1987].

**Definition 1.** The worst-case running time $T^W_A(n)$ of an algorithm $A$ on all inputs of size $n$ is the greatest (worst) running time which the algorithm takes on any input of size $n$.

**Definition 2.** The average-case running time $T_A(n)$ of an algorithm $A$ on all inputs of size $n$ is defined as:

$$T_A(n) = \frac{T_A(L_1) + T_A(L_2) + \ldots + T_A(L_{n!})}{n!},$$

(1.1)

where $L_1, \ldots, L_{n!}$ are the $n!$ possible input lists of size $n$ for this algorithm.
Definition 3. A comparison-based algorithm is an algorithm in which every action is based on a prior comparison between elements.

The study of such algorithms is important for several factors:
- This class includes a large number of data organization algorithms including searching and sorting algorithms.
- Their asymptotic time (worst-case and average-case) is guaranteed to satisfy a lower bound, namely $\Omega(n \log_2 n)$.
- Their running time is generally directly linked to the number of comparisons carried out during their execution. This enables one to approximate their running time via a static determination of the number of comparisons carried out during their execution.

Comparison-based sorting algorithms have the useful property that the entire computation over inputs of a fixed size can be represented via a decision tree [Aho et al. 1987][Cormen et al. 1997]. Essentially, a decision tree's root represents all possible inputs of a given size. Each leaf represents the computational outcome of the execution of the algorithm on one of the inputs. Along the branch from the root to a leaf, all comparisons made during this computation are recorded. Due to the nature of comparison-based algorithms, i.e. the sole reliance on comparisons to determine the computation flow, and due to the fact that inputs are identified up to relative order between elements, distinct inputs must lead to distinct computation paths. Hence, since we only consider algorithms which take lists as inputs, decision trees in our context will have $n!$ leaves.

Moreover, each edge on a decision tree is labeled with a comparison between two elements. The path length, i.e. the number of edges on a path from the root to a leaf, is the number of comparisons carried out during the computation on the input giving rise to the output recorded at the leaf under consideration. I.e. path lengths in decision trees correspond directly to computation time, i.e. comparison time, for the algorithm. Clearly, the worst-case time of a comparison-based algorithm, for inputs of given size, is the longest path in its decision tree, while the average-case time of the algorithm is the average path-length in its decision tree.

There has been a considerable amount of work on timing analysis of comparison-based sorting algorithms [Knuth 1973]. We recall another basic definition for an asymptotic lower bound of a function.

Definition 4. A function $f(n)$ is said to be (asymptotically) bounded from below by function $g(n)$ and is denoted as $f(n) \in \Omega(g(n))$ if

$$\exists n_0, \exists c > 0, \forall n \geq n_0, f(n) \geq cg(n) \quad (1.2)$$

For the worst-case and average-case timing of any comparison-based sorting algorithm, the following holds:

Theorem 1. Any deterministic comparison-based sorting algorithm must perform $\Omega(n \log_2 n)$ comparisons for execution on inputs of size $n$ in both the worst-case and in the average-case.
The rest of the paper is organized as follows. Section 2 covers the basic definitions and mathematical identities required to derive the lower bounds. We give the proofs for worst-case in Section 3 and for average-case in Section 4. We conclude the paper with some final remarks in Section 5.

2. Preliminaries

Before proceeding to the formal proofs, we present some basic inequalities and definitions which will be used to present the proofs.

During the formal proofs, we will use two mathematical inequalities, which are presented here.

2.1 Kraft’s inequality

We recall the well-known Kraft inequality for binary trees [Cormen et al. 1997], which plays a well-known role in Computer Science, e.g. in the context of characterizing binary trees, Information Theory and prefix codes.

**Definition 5.** Given a tree \( T \) with \( N \) leaves. The path length sequence of the binary tree is the sequence \((l_1, \ldots, l_N)\) consisting of the lengths of the paths from the root to the \( N \) leaves of \( T \). The Kraft number of a binary tree \( T \) is defined to be \( K(T) = \sum_{i=1}^{N} 2^{-l_i} \).

**Proposition 1.** Consider a binary tree with path length sequence \((l_1, \ldots, l_N)\), then

\[
K(T) \leq 1.
\]

2.2 Jensen’s inequality

Jensen’s inequality regards convex real-valued functions \( f \), for which the function values over an interval \([a, b]\) are always bounded by the line segment values for the segment connecting \((a, f(a))\) and \((b, f(b))\).

**Definition 6.** A real-valued function \( f \) defined on an interval is called convex, if for any two points \( x \) and \( y \) in its domain and any \( t \) in \([0,1]\), we have

\[
(*) \quad f(tx + (1-t)y) \leq tf(x) + (1-t)f(y).
\]

If we consider the convexity inequality \((*)\) for the case where \( t = \frac{1}{2} \), we obtain that

\[
f\left(\frac{x+y}{2}\right) \leq \frac{f(x)}{2} + \frac{f(y)}{2}.
\]

In other words, “the value of the mean is bounded by the mean of the values”.

Jensen’s inequality, named after the Danish mathematician Johan Jensen, generalizes the above statement. The inequality generalizes the above case from two values to \( n \geq 2 \) values and establishes that “the value of a convex function on an average value is bounded by the average value of the convex function”. It was proved by Jensen in 1906 [Jensen 1906].
Lemma 1. (Jensen’s inequality/finite version) If \( f \) is a convex real-valued function and \( \lambda_1, \lambda_2, \ldots, \lambda_n \) are \( n \) positive real numbers such that \( \lambda_1 + \lambda_2 + \ldots + \lambda_n = 1 \), then

\[
f(\lambda_1 x_1 + \lambda_2 x_2 + \ldots + \lambda_n x_n) \leq \lambda_1 f(x_1) + \lambda_2 f(x_2) + \ldots + \lambda_n f(x_n),
\]

for any \( x_1, x_2, \ldots, x_n \).

Note that for the case where \( \lambda_1 = \ldots = \lambda_n = \frac{1}{n} \), Jensen’s inequality simply states that a convex function value of the mean value of a selection of numbers is bounded by the mean value of the function values over these numbers. This makes Jensen’s result ideally suitable in the context of average-case time, which of course is based on the notion of a mean.

The proof is a simple induction which we include for the sake of completeness [Jensen 1906].

\[ \text{P.L.} \]

\[
\text{Consider two arbitrary positive real numbers } \lambda_1, \lambda_2, \text{ such that } \lambda_1 + \lambda_2 = 1, \text{ then convexity of } f \text{ implies } f(\lambda_1 x_1 + \lambda_2 x_2) \leq \lambda_1 f(x_1) + \lambda_2 f(x_2) \text{ for any } x_1, x_2. \text{ Hence the statement is true for } n = 2. \text{ Suppose it is true also for some } n, \text{ one needs to prove it for } n + 1. \text{ Without loss of generality, we may assume that } 0 < \lambda_i < 1 \forall \lambda_i; \text{ therefore by convexity inequality:}
\]

\[
f\left( \sum_{i=1}^{n+1} \lambda_i x_i \right) = f\left( \lambda_1 x_1 + (1 - \lambda_1) \sum_{i=2}^{n+1} \frac{\lambda_i}{1 - \lambda_1} x_i \right)
\leq \lambda_1 f(x_1) + (1 - \lambda_1) f\left( \sum_{i=2}^{n+1} \frac{\lambda_i}{1 - \lambda_1} x_i \right).
\]

Since \( \sum_{i=2}^{n+1} \frac{\lambda_i}{1 - \lambda_1} = 1 \), one can apply the induction hypotheses to the last term in the previous formula to obtain the result, namely the finite form of Jensen’s inequality.

\[ \square \]

Lemma 2. For any positive integer \( n \), the following holds:

\[
\log_2(n!) \in \Omega(n \log_2 n). \tag{2.1}
\]

We include the proof of this well-known result for the sake of completeness.

\[ \text{P.L.} \]

\[
n! = n \times (n - 1) \times (n - 2) \times \ldots \times \left\lceil \frac{n}{2} \right\rceil \times \ldots \times 2 \times 1
\Rightarrow n! \geq \left\lceil \frac{n}{2} \right\rceil \]
\[
\Rightarrow \log_2(n!) \geq \left\lceil \frac{n}{2} \right\rceil \log_2\left( \left\lceil \frac{n}{2} \right\rceil \right)
\Rightarrow \log_2(n!) \geq \left\lceil \frac{n}{2} \right\rceil \log_2(n) - \left\lceil \frac{n}{2} \right\rceil
\Rightarrow \log_2(n!) \geq \frac{1}{2} \left\lceil \frac{n}{2} \right\rceil \log_2(n), \quad \text{for } n \geq 4
\Rightarrow \log_2(n!) \geq \frac{1}{4} n \log_2(n), \quad \text{for } n \geq 4.
\]
Assuming, in Definition 4, $f(n) = \log_2(n!)$, $g(n) = n\log_2(n)$, $c = 1/4$ and $n_0 = 4$, we conclude that

$$\log_2(n!) \in \Omega(n\log_2(n)) \quad \Box$$

Finally, we recall a simple lemma, for which we leave the proof as an exercise.

**Lemma 3.** Given that

$$f(n) \geq g(n)$$

and

$$g(n) \in \Omega(h(n)),$$

the following holds:

$$f(n) \in \Omega(h(n)). \quad (2.2)$$

We continue to establish the lower-bound for worst-case and average-case comparison time based on Kraft’s inequality.

### 3. Worst-case lower bound

**Proposition 2.** The worst-case time for comparison-based sorting algorithms on lists of size $n$ is $\Omega(n\log n)$.

**Proof.** Consider a comparison-based sorting algorithm $A$ and let $DT_A(n)$ be its decision tree for inputs of size $n$. This is a binary tree and hence Kraft’s inequality applies (where $N = n!$):

$$K(DT_A(n)) = \sum_{i=1}^{n!} 2^{-l_i} \leq 1,$$

where $(l_1, \ldots, l_{n!})$ is the path-length sequence of the tree $DT_A(n)$.

Consider a path length, say $l_j$, where $j \in \{1, \ldots, n!\}$, which has maximum value among all path lengths in the set $\{l_1, \ldots, l_{n!}\}$. Clearly $T^W_A(n) = l_j$. Since $\forall i \in \{1, \ldots, n!\}. l_j \geq l_i$, we obtain that $\forall i \in \{1, \ldots, n!\}. 2^{-l_j} \leq 2^{-l_i}$ and thus

$$2^{-l_j} n! \leq \sum_{i=1}^{n!} 2^{-l_i} \leq 1.$$

Hence $2^{-l_j} \leq \frac{1}{n!}$ and thus $2^{l_j} \geq n!$. We conclude that $l_j \geq \log_2(n!)$ and hence

$$T^W_A(n) \geq \log_2(n!).$$

Further, using Lemma 2 and Lemma 3,

$$T^W_A(n) \in \Omega(n\log_2 n). \quad \Box$$
4. Average-case lower bound

**Proposition 3.** The average-case time for comparison-based sorting algorithms on lists of size \( n \) is \( \Omega(n \log n) \).

**Proof.** Note that for any comparison-based sorting algorithm \( A \) which takes input lists of size \( n \), the decision tree \( DT_A(n) \) is a binary tree with \( n! \) leaves, where say \((l_1, \ldots, l_n)\) is the path-length sequence of the tree \( DT_A(n) \).

Indeed, note that \( f(x) = 2^{-x} \) is a convex function. Hence from Jensen’s inequality, we obtain that:

\[
2^{-(\frac{l_1 + \ldots + l_n}{n!})} \leq \frac{2^{-l_1} + \ldots + 2^{-l_n}}{n!}
\]

and thus:

\[
n! 2^{-(\frac{l_1 + \ldots + l_n}{n!})} \leq 2^{-l_1} + \ldots + 2^{-l_n}.
\]

By Kraft’s inequality we get:

\[
n! 2^{-(\frac{l_1 + \ldots + l_n}{n!})} \leq 2^{-l_1} + \ldots + 2^{-l_n} \leq 1
\]

and thus

\[
2^{-(\frac{l_1 + \ldots + l_n}{n!})} \leq \frac{1}{n!},
\]

which yields

\[
\frac{l_1 + \ldots + l_n}{n!} \geq \log_2(n!).
\]

Further using Lemma 2 and Lemma 3,

\[
\overline{T}_A(n) \in \Omega(n \log_2 n). \quad \square
\]

5. Final Remarks

We presented a new derivation for the well-known result on the lower bound for the worst-case time and the average-case time of comparison-based sorting algorithms. The corresponding proofs presented for Propositions 2 and 3 are based on a simple application of Kraft’s inequality and are quite similar in nature. Moreover, the rather involved argument for the average-case lower bound, which typically combines a proof via induction over decision tree size with determination of a minimum for a two-variable real-valued function [Aho et al. 1987][Cormen et al. 1997], has been replaced via a short argument involving Jensen’s means-based inequality. Both Kraft’s inequality and Jensen’s inequality are useful to know for Computer Science students and are easy to understand based on standard high school mathematics. Hence these important lower bound results can easily be taught at an undergraduate level and standard algorithms text books can replace the multi-variable calculus approach with the simplified approach discussed above.
References


