On the difference between logarithmic runtime and linear runtime

Maxwell Aladago
January 23, 2019

Often, the difference between the binary search algorithm and the sequential search algorithm can appear as only a difference of names; a simple matter of replacing the word 'binary' with 'sequential'. If you are more vigilant, you may observe that binary search runs in $O(log_2 n)$, and sequential search $O(n)$ worst case time for an input size n. Even then, you may still not appreciate how big a *difference* that is. I hope to bring out this difference between the two algorithms, and generally the difference between logarithmic and linear runtimes through a few simple comparisons.

Imagine you have a computer A which can only execute 1 instruction per second. This is a terribly slow computer, slower than any viable computer ever invented! But that is all you have, for now. You need to find -5 from a list of 10^{20} integers. Your life depends on it. All the integers are sorted in non-decreasing order. -5 is not in the list but you do not know. So you begin your search.

If you decide to use sequential search. It is implemented so well that it has no overhead at all. It takes just a single instruction to determine whether an arbitrary integer is your search term or not. Unfortunately, the sequential search is implemented naively without using the advantage of the sorting to terminate the search prematurely if we encounter integers greater than the search key. I.e, $T_{seq}(n) = n$. For this scenario, you will need 10^{20} seconds to discover that -5 is not in the list. This is equivalent to 3.162 trillion years, a figure far far greater than the estimated age of our universe[1]. Your search will be hopeless. You'll be in trouble.

But if you decide to use binary search. It's implemented poorly such that its runtime is $T_{bin}(n) = 100log_2 n$. You still have machine A and the same set of integers. For this second scenario, it will take you only $T_{bin}(10^{20}) = 100 \times log_2 10^{20} = 100 \times 20 \times 3.32 \approx 6644$ seconds. This is equivalent to **1.85 hours**, just 21 minutes more than the duration of a single lecture in Ashesi. If the total number of your integers were 10^{82}, the estimated number of atoms in the known universe, even then, you'll still need only 7.56 hours to figure out whether an arbitrary integer is in the list or not.

Assume by magic you've suddenly acquired the world's fastest computer, Summit[2] which can execute 200,000 trillion calculations per second at peak speed. Let's call this computer B. Using similar scenarios as before, sequential search will take $\frac{10^{20}}{2 \times 10^{17}} = 5 \times 10^2$ seconds, about 8.33 minutes. On computer B, binary search will take $\frac{6644}{2 \times 10^{17}} = 3.32 \times 10^{-l4}$, about **0.00003 nanoseconds**.

Note: This write-up is simply to illustrate how far apart the two algorithms are as the input size gets larger. Typical input sizes are not as large as the example used in this write-up.

[1]The universe is estimated to be about 13.772 billion years

[2]This is as of November 2018