Need for Speed

This chapter discusses analogies from real life to draw a parallel between algorithmic complexity and comparisons we make in day to day situations.

The Race Track

Let's say you are a participant in your school's annual sports competition, which includes the 100-meter dash. You are a fast runner but not the fastest. You know Milkha is the fastest runner in the school and will probably finish first on the day of the race. The night before the competition you explain your predicament to your mother. She innocently asks, *"But how fast is Milkha?"* She hasn't seen you or Milkha run, and has no clue about your respective sprinting abilities. You reply back *"Faster than me."*. She quizzes back *"How much faster than you?"*. You quip back *"Faster than me, but definitely slower than Usain Bolt,"* someone your mother has seen run. By providing these comparisons to your mother, you are essentially setting a lower and upper bound on the running abilities of Milkha. Milkha will definitely run definitely faster than you, but will surely be slower than Usain Bolt.

You can also compare yourself directly with Usain Bolt and say you run slower than him, but then you are not ***bounding*** your skill tightly enough; you might as well say you run slower than the speed of light. That statement, though factually correct, isn't informative enough. The closer a reference point you can share with respect to your speed, the better your friends will be able to judge your chances of winning on the competition day. However, if you tell them you are even slower than Sogelau Tuvalu then surely no one will be betting money on you.

Note how you can provide both an upper bound and a lower bound. You run no faster than Usain Bolt and you run no slower than Sogelau. The tighter these bounds are, the better. If either of these bounds becomes loose, the less informative it is.

When algorithms or data-structures are analyzed for their time or space complexity, we are - in a sense - making statements similar to the sprinting competition scenario. An algorithm's performance or space requirement is bound by an upper and lower threshold. This threshold is defined in terms of mathematical functions that operate on the input size to the algorithm. Since all the algorithms that solve a given problem are analyzed in terms of the input size to them, it gives us the ability to uniformly compare them against each other and reason about the best solution.

Why not use a stop-watch?

We, as humans, are biased to measure performance using our stop-watches. A novice reader may suggest timing different algorithm runs just like cars on a racing track to determine the more performant one. However, in the case of algorithms, there are too many variations that can affect an algorithm's performance: the hardware used for the test, the characteristics of the input fed to the algorithms, the operating system or other programs concurrently running on the system, etc. All these factors can affect an algorithm's performance.

Here's a simplistic scenario: Suppose you are tasked with determining the performance of an array vs a linked-list for search/retrieval operation. Let's say you are looking for what is placed at the first index of both the data-structures. The time to retrieve the value at the first spot will come out to be the same on your stop-watch for both the data-structures, all else being equal. However, if you were looking for what was placed at the tail-end of each data-structure, the array data-structure would win hands down! In the case of linked-list you'll be walking down to the end of the chain to retrieve the value, whereas for the array it'll be a simple memory address retrieval operation; consequently, the choice of the data-structure is impacting the performance of our retrieval operation. If you based your stopwatch tests on retrieving an item that somehow gets placed at the beginning of each of the data-structures, the result wouldn't be informative enough.

This is further amplified in the case of sorting integers. The sorting experiment we ran in the previous section was similar to a stop-watch test, but it doesn't differentiate much between the performance of merge sort and dual-pivot quicksort algorithms. In order to truly understand the difference

dual-pivot quicksort algorithms. In order to truly understand the difference, we'll be required to undertake a mathematical analysis of both algorithms.

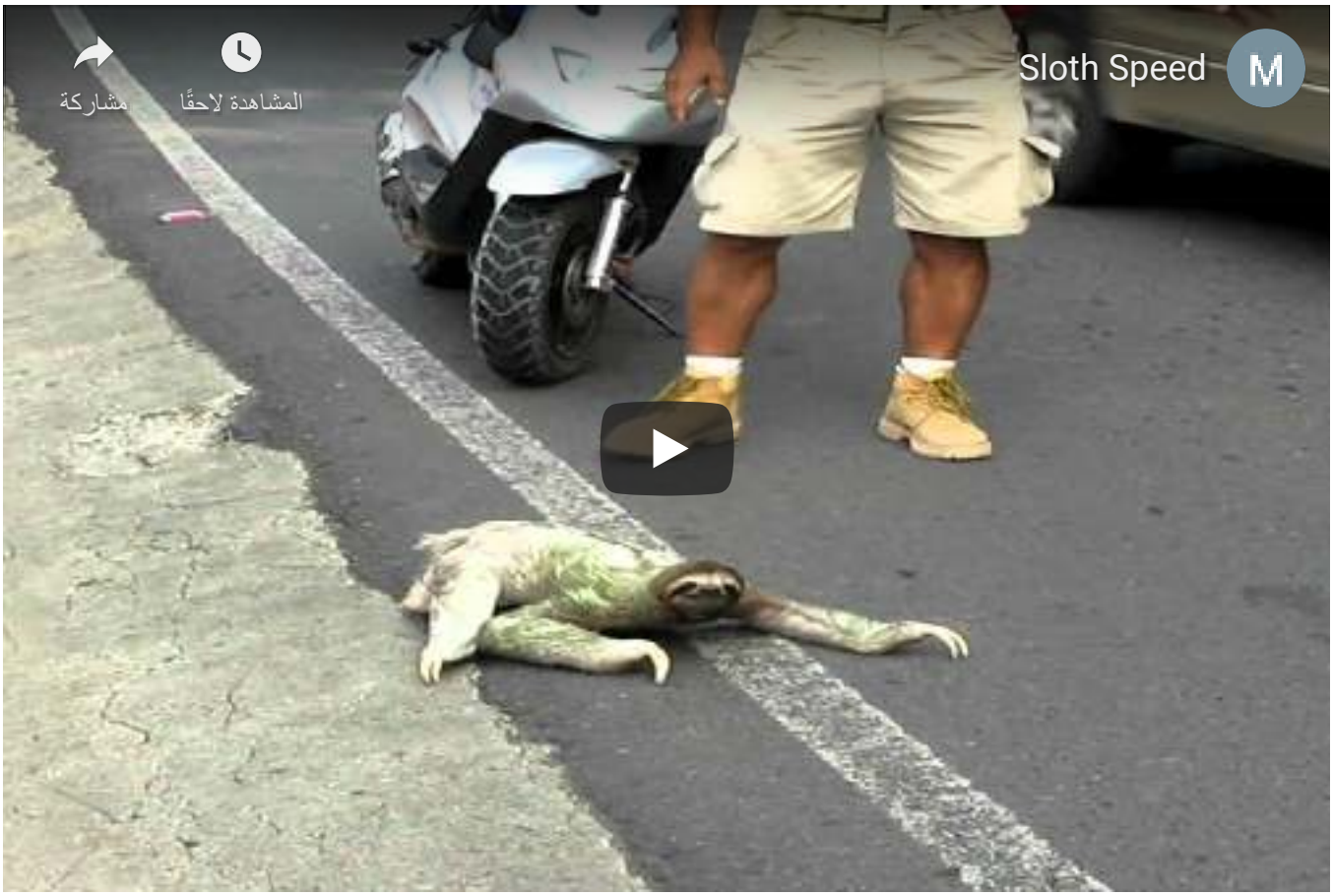
Theoretically, we can come up with inputs that'll make merge sort perform better than the quicksort algorithm but this may not be apparent from running simplistic *stop-watch* tests. Conversely, we can provide already sorted lists to different sorting algorithms and have them perform roughly the same, again clouding the true performance capabilities of each.

Uniform Comparison

We need to create a framework for analyzing algorithms and the performance of data-structures that is independent of the aforementioned variations, which can cloak the true performance of algorithms and data-structures. The basis of this framework is grounded in mathematics and tied to the input size.

Furthermore, algorithms can have their good days and bad days. On a good day, Usain Bolt broke the 100m sprint record; on a bad day, who knows he might even lose to you! But he'll definitely be faster than a sloth even on his worst day, and slower than a cheetah on his best day. Algorithms or data-structures have their good days called ***best case*** and their bad days called ***worst case***. Note the use of superlatives, as they can't really get better than their best case or worse than their worst case scenarios. Then there's the middle ground, called the **average case**, which is how an algorithm or a data-structure is expected to perform on most inputs.

In computer science and especially in interviews, we are concerned only about the worst case performance of an algorithm or a data-structure. If we have a handle on the worst case performance and consider it reasonable for the problem at hand - and unable to get worse - we can proceed in using that algorithm/data-structure. In fact, now we have a guarantee that it can't perform any worse than that. And in fact, it is likely to perform closer to its average case.


مشاركة
المشاهدة لاحقًا
Sloth Speed
M