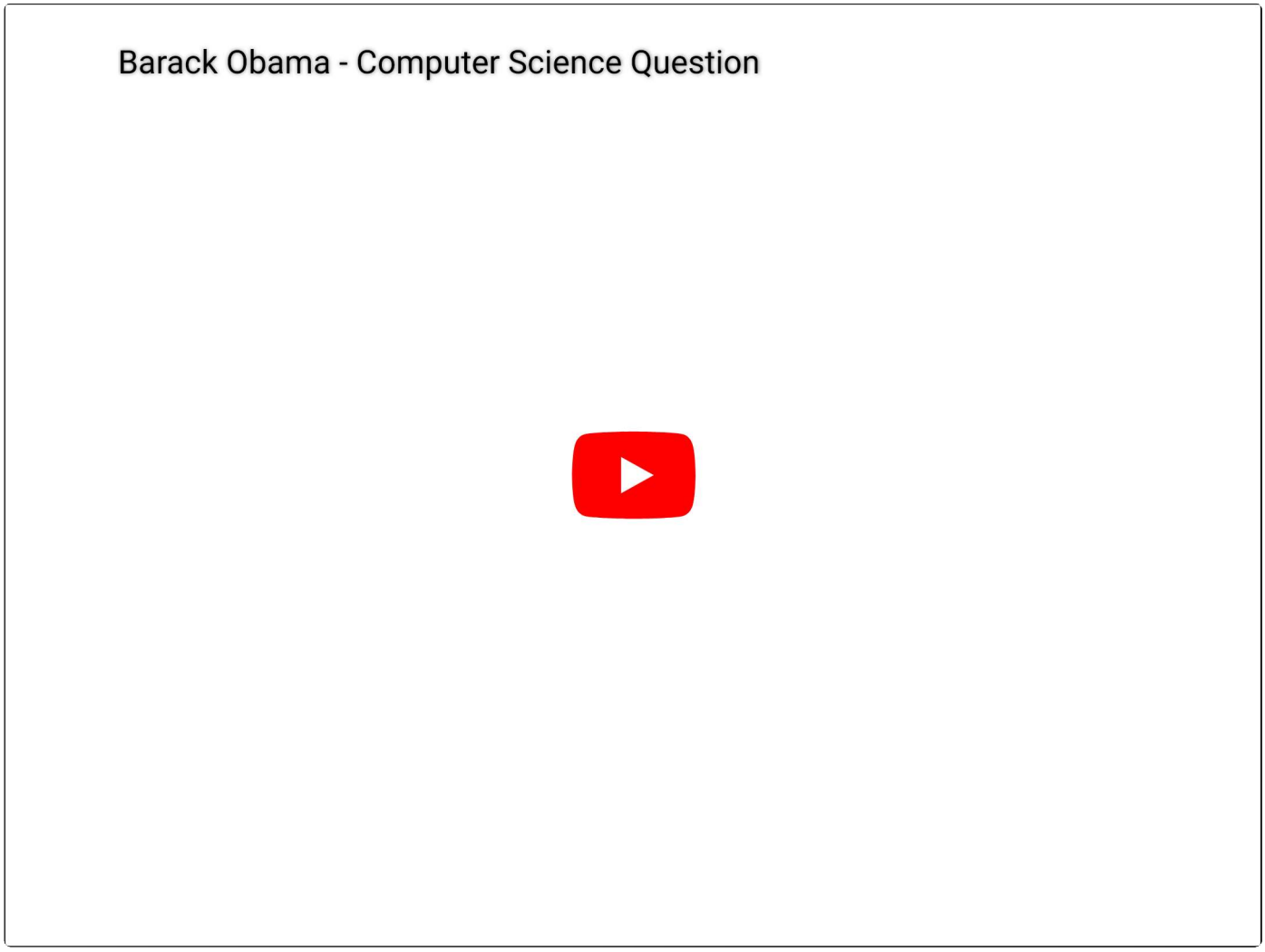
36.3 Radix Sorting Integers

ft. Obama



Obama knows Bubble sort! This is amazing!

BTW: A 32-bit integer is a normal integer in Java and other programming languages that spans 4 billion values. This is something we learned in Hashing chapter 20 and is a 61C concept!

Summary of Video above

Barack Obama gets a Google Interview question: What is the most efficient way to sort a million 32-bit integers? Obama says that he would recommend not using Bubble sort! That's right folks! Obama knows his 61B 😄.

What's the answer?

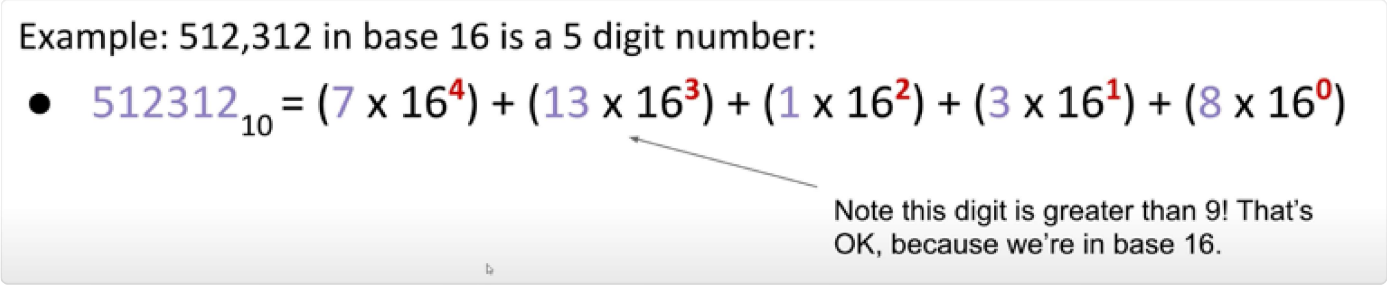
The answer to this question actually is Radix sort because we know that in a very large N limit, Radix sort is simply fastest as it is linear with runtime Θ(WN) where W is the number of digits and N is the number of integers we are sorting. This is much faster than any Comparison Sort we learned about since the fastest runtimes seem to be Θ(N*log(N)).

But how do we do it?

We don't have a charAt() for every integer. And if we converted all integers to strings, that's a very time expensive operation as well. So how would you LSD radix sort an array of integers? Instead of using charAt, maybe write a helper method like getDthDigit(int N, int d). Example: getDthDigit(15009, 2) = 5.

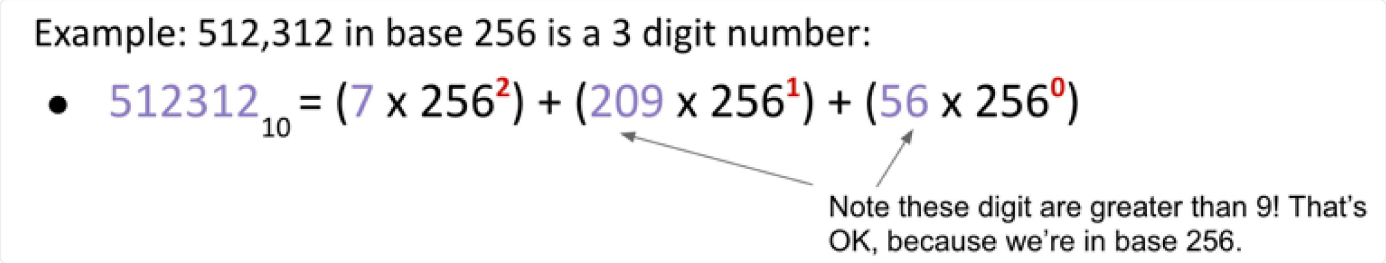
LSD Radix Sort on Integers

Note that we don't have to work with base 10. What we can instead do is increase this base to have less digits in our total number. This is really getting into 61C territory as we haven't really discussed binary representation of integers yet, but essentially what we want to do is convert to hexadecimal, a base 16 number that is represented with the following digits: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F}. Note that A is 10 and F is 15. 0 - 15 is 16 possible values for a digit! To convert a number from decimal (base 10) to hexadecimal (base 16), we do the following:



Decimal to Hexadecimal

Notice that our original number had 6 digits in base 10 and our resulting number has 5 digits! Notice we don't have to go to just base 16. We can go even bigger to base 256 to only have 3 digits! There are small yet amazing optimizations!



Decimal to Ducentohexaquinquagesimal (Base 256)

Please do not worry about memorizing what "ducentohexaquinquagesimal" is. The only important thing to learn here is that this conversion to higher bases may result in fewer digits for our LSD and MSD Radix sorts to have faster traversals.

However, there is a tradeoff between W, the size of each integer, and the size of the array we need to maintain the counts. The most oprtimal is actually base 256!

Previous
36.2 The Just-In-Time Compiler

Next
36.4 Summary

Last updated 1 year ago