13.3 Checkpoint: An Exercise

Some much needed practice.

Exercise: Apply techniques 2A and 2B to `dup2`.

- Calculate the counts of each operation for the following code with respect to N.
- Predict the ***rough*** magnitudes of each one.

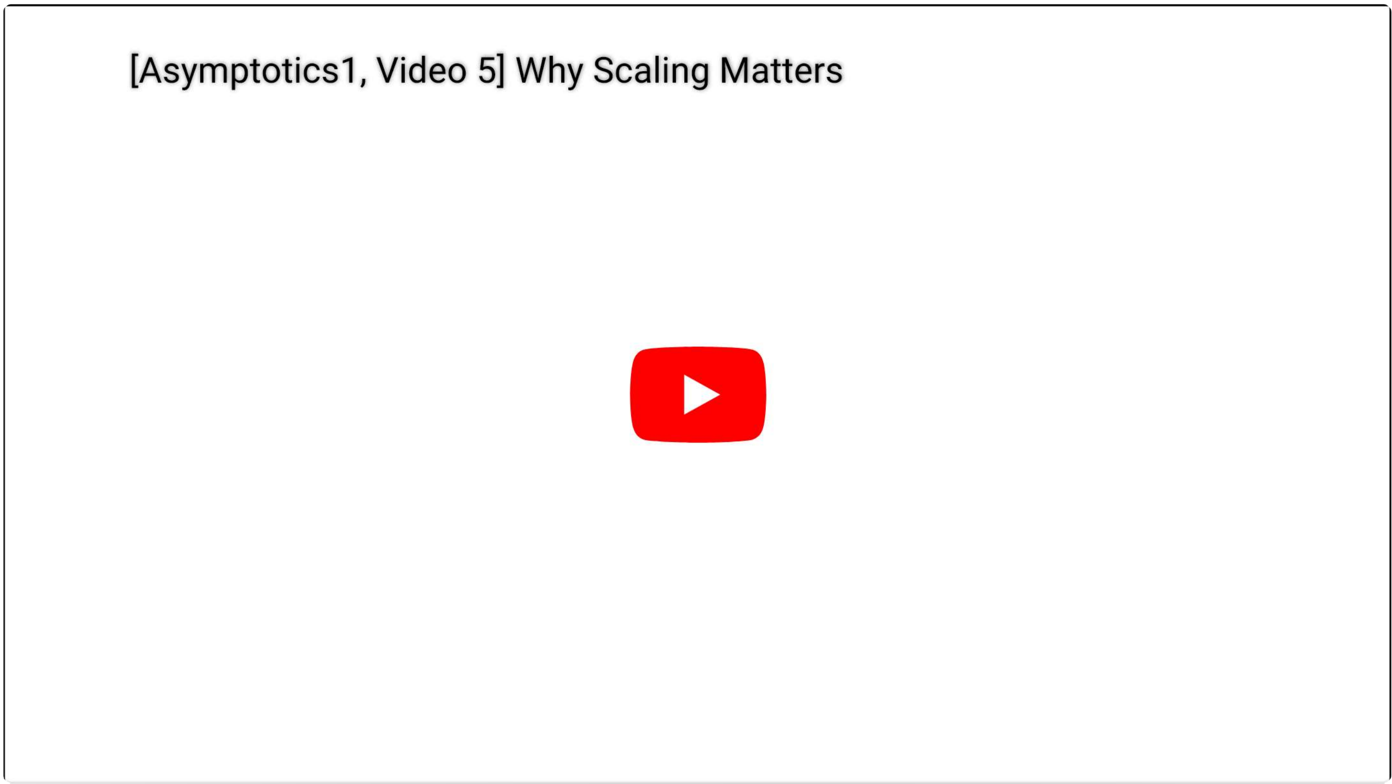
```
for (int i = 0; i < A.length - 1; i += 1){
  if (A[i] == A[i + 1]) {
    return true;
  }
}
return false;
```

Solution:

Note: It's okay if you were slightly off—as mentioned earlier, you want ***rough*** estimates.

Operation	Symbolic Count	Count (for N=10000)
i = 0	1	1
j = i+1	0 to N	0 to 10,000
<	0 to $N-1$	0 to 9,999

Operation	Symbolic Count	Count (for N=10000)
==	1 to $N-1$	1 to 9,999
array accesses	2 to $2N-2$	2 to 19998



"I have another problem for you to solve (͡° ͜ʖ ͡°)..." - Josh Hug

Let us compare the `dup1` table with the `dup2` table:

`dup1` table:

Operation	Symbolic Count	Count (for N=10000)
i = 0	1	1
j = i+1	1 to N	1 (in the best case) to 10000 (in the worst case)
<	2 to $(N^2+3N+2)/2$	2 to 50,015,001
+= 1	0 to $(N^2+N)/2$	0 to 50,005,000
==	1 to $(N^2-N)/2$	1 to 49,995,000
array accesses	2 to N^2-N	2 to 99,990,000

`dup2` table:

Operation	Symbolic Count	Count (for N=10000)
i = 0	1	1
j = i+1	0 to N	0 to 10,000
<	0 to $N-1$	0 to 9,999
==	1 to $N-1$	1 to 9,999
array accesses	2 to $2N-2$	2 to 19998

We can see that `dup2` performs significantly better than `dup1` in the worst case!

One way to rationalize this is that it takes fewer operations for `dup2` to accomplish the same goal as `dup1`.

A better realization is that the algorithm for `dup2` scales much better in the worst case (e.g. $(N^2+3N+2)/2$ vs N)

An even **better** realization is that parabolas (N^2) always grow faster than lines (N).

Previous
13.2 Runtime Characterization

Next
13.4 Asymptotic Behavior

Last updated 1 year ago