

Reading: GT 1.4, pp. 31–33 [less than 20 min]

This lecture is devoted to examples deepening definitions shown previously. The PREFIX-AVERAGES problem is not important at all. It is important to understand complexity analysis in it, and how nested loops contribute to quadratic complexity.

Ingredients: big-Oh “calculus”, prefix averages, solution 1, solution 2.

Big-Oh “calculus”

A major advantage of Big-Oh is that it classifies algorithms into relatively few groups based on their performance:

$$O(1) \subseteq O(\lg n) \subseteq O(n) \subseteq O(n \log n) \subseteq O(n^2) \subseteq O(n^3) \subseteq O(2^n) . \quad (1)$$

The big-Oh notation behaves remarkably reasonably. Example:

$$(N + O(1)) \cdot (N + O(\log N) + O(1)) \quad (2)$$

is the same as

$$N^2 + O(N) + O(N \log N) + O(\log N) + O(N) + O(1) \quad (3)$$

which in turn is the same as

$$N^2 + O(N \log N) \quad (4)$$

and nearly the same as

$$O(N^2) \quad (5)$$

Prefix Averages

Problem definition:

input: an array X with n numbers**output:** an array A such that $A[i]$ is the average of $X[0], \dots, X[i]$ for $i = 0 \dots n-1$:

$$A[i] = \frac{1}{i+1} \sum_{j=0}^i X[j] \quad (6)$$

Applications in economics and statistics. For example each entry in X gives annual return of a mutual fund. Then A contains average returns for growing prefixes of years in the period.

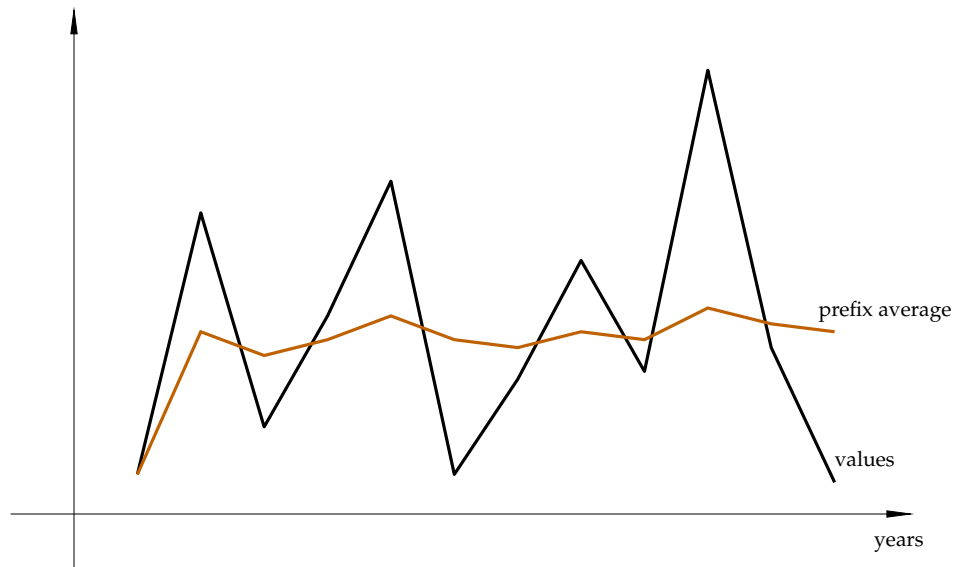


Figure 1: An illustration of the prefix average function and how it is useful for smoothing a quickly changing sequence of values (See GT Fig 1.13 p 31)

Solution 1

PREFIX-AVERAGES-1($X : \text{arr}$)	(worst case cost, $n = X.size$)	
1 \triangleright new array $A[0..X.size-1]$	c_1n	$O(n)$
2 for $i \leftarrow 0$ to $X.size-1$	$c_2(n+1)$	
3 do $a \leftarrow 0$	c_3n	$O(n)$ in total
4 for $j \leftarrow 0$ to i	$c_4(i+1)$	
5 do $a \leftarrow a + X[j]$	$c_5(i+1)$	$O(n^2)$ in total
6 $A[i] \leftarrow \frac{a}{i+1}$	c_6n	$O(n)$ in total
7 return A	c_7	$O(1)$

An asymptotic analysis of the running time in detail:

$$\begin{aligned}
T_n &= c_1n + c_2(n+1) + c_3n + \sum_{i=0}^{n-1} c_4(i+1) + \sum_{i=0}^{n-1} c_5(i+1) + c_6n + c_7 = \\
&= c_1n + c_2n + c_2 + c_3n + c_4 \sum_{i=1}^n i + c_5 \sum_{i=1}^n i + c_6n + c_7 = 1 \\
&= \frac{n(n+1)}{2}(c_4 + c_5) + n(c_1 + c_2 + c_3 + c_6) + c_7 = [\mathbf{let } c' = 0.5(c_4 + c_5)] \\
&= n^2c' + nc' + n(c_1 + c_2 + c_3 + c_6) + c_7 = [\mathbf{let } c'' = (c_1 + c_2 + c_3 + c_6 + c')] \\
&= n^2c' + nc'' + c_7 \quad (7)
\end{aligned}$$

By Theorem 1.7.5 we get that T_n is $O(n^2)$.

Rarely we are so detailed in the analysis. Instead we look at sizes of loops and assign complexities to them directly if possible obtaining:

$$O(n) + O(n^2) + O(1) = O(n^2) \quad (8)$$

Conclusion: PREFIX-AVERAGES-1 runs in a quadratic time.

Solution 2

PREFIX-AVERAGES-2(X : arr)

```

1  ▷ new array  $A[0..X.size-1]$ 
2   $s \leftarrow 0$ 
3  for  $i \leftarrow 0$  to  $X.size-1$ 
4      do  $s \leftarrow s + X[i]$ 
5           $A[i] \leftarrow \frac{s}{i+1}$ 
6  return  $A$ 

```

Running time: $O(n) + O(1) + O(n) + O(1) = O(n)$

Conclusion: AVERAGE-PREFIX-2 runs in linear time.