Case study: SelectionSort() Run Time Analysis

Yao Chen

Show that in both best and worst cases, the order of growth of Selection Sort is quadratic using the derivation of T(n) and explain in your own words why it's quadratic.

Detailed derivation of $T(n)$.

SelctionSort	cost	time
`n=A.length`		
`for j = 1 to n-1`	c_1	n
`   smallest = j`	c_2	$n-1$
`   for i = j+1 to n`	c_3	$\sum_1^{n-1}\sum_{j+1}^{n} 1 = \frac{n^2+n}{2}$
`      if A[i] < A[smallest]`	c_4	$\sum_1^{n-1}\sum_{j+1}^{n-1} t_j = t_j\frac{n^2+n}{2}$
`         smallest = i`	c_5	$\sum_1^{n-1}\sum_{j+1}^{n-1} t_j = t_j\frac{n^2+n}{2}$
`   exchange A[j] with A[smallest]`	c_6	$n-1$

$$T(n) = c_1 n + c_2(n-1) + c_3\frac{n^2+n}{2} + c_4 t_j\frac{n^2+n}{2} + c_5 t_j\frac{n^2+n}{2} + c_6(n-1)$$

$$= \left(\frac{c_3}{2} + \frac{c_4 t_j}{2} + \frac{c_5 t_j}{2}\right)n^2 + \left(c_1 + c_2 + \frac{c_3}{2} + \frac{c_4 t_j}{2} + \frac{c_5 t_j}{2} + c_6\right)n + (-c_2 - c_6)$$

- Best case: the array is already sorted

$$t_j = 0, T(n) = \left(\frac{c_3}{2}\right)n^2 + \left(c_1 + c_2 + \frac{c_3}{2} + c_6\right)n + (-c_2 - c_6)$$

- Worst case: the array is sorted in reverse order.

$$t_j = 1, T(n) = \left(\frac{c_3}{2} + \frac{c_4}{2} + \frac{c_5}{2}\right)n^2 + \left(c_1 + c_2 + \frac{c_3}{2} + \frac{c_4}{2} + \frac{c_5}{2} + c_6\right)n + (-c_2 - c_6)$$

And therefore, $T(n) = an^2 + bn + c$, quadratic function of n, in both best and worst cases.

Explanation why the order of growth is quadratic in both the best and worst cases.

Selection Sort

1. Get the length `n` of the array `A`
2. Iterate through each element in `A` using index `j`.
3. For each `j`, find the smallest element in the sub-array starting from `j+1` to `n` and remember its index.
4. Swap the element at j with the smallest found in previous step.

The quadratic growth mainly comes from **the nested loop structure**. For each element `j`, we're potentially comparing it to `n-j` other elements. As `n` grows, the number of comparisons grows quadratically.

In best case, $t_j = 0$, which means the array is already sorted and no exchanges are needed, the index of smallest element in the nested loop structure remains to be `j`, but we still do the comparisons. The order of growth remains quadratic because of the nature of the nested loops.

In worst case, $t_j = 1$, which means we are doing the maximum number of exchanges, we need to replace the index of the smallest element in the sub-array for each `j`. But again, the dominating factors here is the comparisons due to the nested loop structure.

This confirms that for both cases, the order of growth for Selection Sort is quadratic.