

## Exercise 02

Sunday, October 29, 2023 8:22 PM

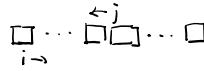
### Problem 1 to hand in: Running Time

The following pseudocode describes the `bubble_sort` algorithm. This is another way to solve the SORTING computation problem: Given a finite sequence  $A$  of pairwise distinct integers (and its length), return  $A$  sorted ascendingly.

`bubble_sort`( $A[0..n-1], n$ ):

```

1 for  $j \leftarrow n-1$  down to 1 do
2   for  $i \leftarrow 0$  to  $j-1$  do
3     if  $A[i] > A[i+1]$  then
4       key =  $A[i]$ 
5        $A[i] = A[i+1]$ 
6        $A[i+1] = \text{key}$ 
7 return  $A$ 
```



- How does `bubble_sort` work in comparison to `insertion_sort`? Describe it intuitively in one or two sentences.
- Analyse the asymptotic worst-case running time of `bubble_sort`:
  - For each line of code, write down the number of running steps (dependent on the input size) in the worst case.
  - Sum up the total number  $T(n)$  of steps the algorithm needs in the worst case for an input of size  $n$ .
  - Provide a function  $f(n)$  as a representative upper bound in  $\mathcal{O}$ -notation.
  - Proof formally that  $T(n) \in \mathcal{O}(f(n))$  holds.
- What is the (asymptotic) average running time? Briefly argue why.

a) Intuitively, the bubble\_sort algorithm has a lot in common comparing to the insertion\_sort. They both have two for-loops and the inner loop shows dependency on the first loop. On the other hand, bubble\_sort algorithm starts the loop with a whole length of  $A$ , pushing current largest number to the end.

b) Worst case: Totally inversed sequence

`bubble_sort`( $A[0..n-1], n$ ):

```

1 for  $j \leftarrow n-1$  down to 1 do
2   for  $i \leftarrow 0$  to  $j-1$  do
3     if  $A[i] > A[i+1]$  then
4       key =  $A[i]$ 
5        $A[i] = A[i+1]$ 
6        $A[i+1] = \text{key}$ 
7 return  $A$ 
```

$$\begin{array}{l|l}
 \overbrace{1 \dots n-1}^{n-1} & n-1+1 = n \\
 0 \sim j-1 & \sum_{j=1}^{n-1} j + n-1 = \frac{(n-1)(n-1)}{2} + n-1 = \frac{1}{2}(n^2-n) + n-1 = \frac{1}{2}n^2 + \frac{1}{2}n-1 \\
 0 \sim j-1 & \left. \begin{array}{l} 0 \sim j-1 \\ 0 \sim j-1 \end{array} \right\} 4 \times \sum_{j=1}^{n-1} j = 2n^2 - 2n \\
 0 \sim j-1 & \\
 & 1
 \end{array}$$

$$\begin{aligned}
 T(n) &= n + \frac{1}{2}n^2 + \frac{1}{2}n - 1 + 2n^2 - 2n - 1 \\
 &= \frac{5}{2}n^2 - \frac{1}{2}n \Rightarrow T(n) \text{ can be bounded by } f(n) = n^2
 \end{aligned}$$

• It is enough to prove that

$$\exists c > 0, n_0 > 0, \text{ s.t. } \forall n \geq n_0: \frac{5}{2}n^2 - \frac{1}{2}n \leq cn^2$$

Let  $c = \frac{7}{2}$ , we construct a function

$$\begin{aligned} J(n) &= f(n) - T(n) \\ &= \frac{7}{2}n^2 - \frac{5}{2}n^2 + \frac{1}{2}n \\ &= n^2 + \frac{1}{2}n = n(n + \frac{1}{2}) \end{aligned}$$

Let  $n_0 = 10$

$\forall n \geq 10$ , we have  $n \geq 0$  and  $n + \frac{1}{2} \geq 0$

$$\Rightarrow J(n) \geq 0 \Rightarrow f(n) \geq T(n)$$

Therefore  $\exists c > 0, n_0 > 0$ , s.t.  $\forall n \geq n_0 \quad T(n) \leq cn^2 \quad \square$

c) The average running time will still be in  $O(n^2)$

The if-condition (IF  $A[i] > A[i+1]$ ) in the inner loop has to be executed  
no matter how many inversions are in the sequence (independent on the input)

Its number of steps is  $\sum_{j=1}^{n-1} j = \frac{(n-1+1) \cdot (n-1)}{2} = \frac{1}{2}n(n-1) \in O(n^2)$