# Tutorial (week 3)

All the answers can be given with pseudocode or with Python. Remember that most of the time, many solutions are possible.

1. Order the following list of functions by the big-Oh notation.

| | | | | |
|---|---|---|---|---|
| $6n \log n$ | $2^{100}$ | $\log \log n$ | $\log^2 n$ | $2^{logn}$ |
| $2^{2^n}$ | $\lceil \sqrt{n} \rceil$ | $n^{0.01}$ | $1/n$ | $4n^{3/2}$ |
| $3n^{0.5}$ | $5n$ | $\lfloor 2n \log^2 n \rfloor$ | $2^n$ | $n \log_4 n$ |
| $4^n$ | $n^3$ | $n^2 \log n$ | $4^{\log n}$ | $\sqrt{\log n}$ |

2. Give a big-Oh characterization, in terms of $n$, of the running time of the `Loop1`, `Loop2`, `Loop3`, `Loop4` and `Loop5` methods shown below.

Loop 1

```
def loop1(n):
    s = 0
    for i in range(n):
        s = s + i
```

Loop 2

```
def loop2(n):
    p = 1
    for i in range(2*n):
        p = p * i
```

Loop 3

```
def loop3(n):
    p = 1
    for i in range(n**2):
        p = p * i
```

Loop 4

```
def loop4(n):
    s = 0
    for i in range(2*n):
        for j in range(i):
            s = s + i
```

Loop 5

```
def loop5(n):
    s = 0
    for i in range(n**2):
        for j in range(i):
            s = s + i
```

3. Show that $(n+1)^5$ is $O(n^5)$.
4. Show that $2^{n+1}$ is $O(2^n)$.
5. Describe a recursive algorithm for finding both the minimum and the maximum elements in an array A of $n$ elements. Your method should return a pair (a, b), where a is the minimum element and b is the maximum. What is the running time of your method ?
6. Consider the following recurrence equation, defining a function $T(n)$:

$$T(n) = \begin{cases} 1 & \text{if } n = 1 \\ T(n-1) + n & \text{otherwise} \end{cases}$$

   Show, by induction, that $T(n) = n(n+1)/2$.
7. An array A contains $n-1$ unique integers in the range $[0, n-1]$; that is, there is one number from this range that is not in A. Design an $O(n)$-time algorithm for finding that number. You are allowed to use only $O(log(n))$ additional space besides the array A itself.
8. Suppose that each row of an $n \times n$ array A consists of 1's and 0's such that, in any row of A, all the 1's come before any 0's in that row. Assuming A is already in memory, describe a method running in $O(n)$ time (not $O(n^2)$ time) for finding the row of A that contains the most 1's.
9. Give a recursive algorithm to compute the product of two positive integers $m$ and $n$ using only addition.
10. Given an array, A, describe an efficient algorithm for reversing A. For example, if A = [3, 4, 1, 5], then its reversal is A = [5, 1, 4, 3]. You can only use $O(1)$ memory in addition to that used by A itself. What is the running time of your algorithm?
11. Given an array, A, of $n$ integers, find the longest subarray of A such that all the numbers in that subarray are in sorted order. What is the running time of your method?
12. Given an array, A, of $n$ positive integers, each of which appears in A exactly twice, except for one integer, $x$, describe an $O(n)$-time method for finding $x$ using only a single variable besides A and the iterating variable.

## Hints

- **Question 10:** reverse the array by using index pointers that start at the two ends.
- **Question 12:** consider using the XOR function.