# Algorithms Analysis - Handout 3

Konrad Wojda, 9307820244

## Exercise 1

To visualize this algorithm I decided to write it in Python and print after each step:

### Code

```python
def counting_sort(A):
    k = max(A)
    C = [0] * (k + 1)
    B = ["_"] * len(A)

    print(f"Array A: {A}")

    for a in A:
        C[a] += 1

    print(f"Array C containing number of elements equal to i: {C}")

    for i in range(1, len(C)):
        C[i] += C[i - 1]

    print(f"Array C contains the number of elements less than or equal to i: {C}")


    for i, a in enumerate(reversed(A)):
        B[C[a] - 1] = a
        C[a] -= 1
        print(f"{i+1}.")
        print(f"B : {B}")
        print(f"C : {C}")

    return B

A = [6, 0, 2, 0, 1, 3, 4, 6, 1, 3, 2]
print(counting_sort(A))
```

### Output

```
(.venv) konradwojda@konradwojda-comp:~/studia/HYU-ALANA/03-hw$ python3 counting-sort.py
Array A: [6, 0, 2, 0, 1, 3, 4, 6, 1, 3, 2]
Array C containing number of elements equal to i: [2, 2, 2, 2, 1, 0, 2]
Array C contains the number of elements less than or equal to i: [2, 4, 6, 8, 9, 9, 11]
```

```
1.
B : ['_', '_', '_', '_', '_', 2, '_', '_', '_', '_', '_']
C : [2, 4, 5, 8, 9, 9, 11]
2.
B : ['_', '_', '_', '_', '_', 2, '_', 3, '_', '_', '_']
C : [2, 4, 5, 7, 9, 9, 11]
3.
B : ['_', '_', '_', 1, '_', 2, '_', 3, '_', '_', '_']
C : [2, 3, 5, 7, 9, 9, 11]
4.
B : ['_', '_', '_', 1, '_', 2, '_', 3, '_', '_', 6]
C : [2, 3, 5, 7, 9, 9, 10]
5.
B : ['_', '_', '_', 1, '_', 2, '_', 3, 4, '_', 6]
C : [2, 3, 5, 7, 8, 9, 10]
6.
B : ['_', '_', '_', 1, '_', 2, 3, 3, 4, '_', 6]
C : [2, 3, 5, 6, 8, 9, 10]
7.
B : ['_', '_', 1, 1, '_', 2, 3, 3, 4, '_', 6]
C : [2, 2, 5, 6, 8, 9, 10]
8.
B : ['_', 0, 1, 1, '_', 2, 3, 3, 4, '_', 6]
C : [1, 2, 5, 6, 8, 9, 10]
9.
B : ['_', 0, 1, 1, 2, 2, 3, 3, 4, '_', 6]
C : [1, 2, 4, 6, 8, 9, 10]
10.
B : [0, 0, 1, 1, 2, 2, 3, 3, 4, '_', 6]
C : [0, 2, 4, 6, 8, 9, 10]
11.
B : [0, 0, 1, 1, 2, 2, 3, 3, 4, 6, 6]
C : [0, 2, 4, 6, 8, 9, 9]
[0, 0, 1, 1, 2, 2, 3, 3, 4, 6, 6]
```

**Comment step by step**

First we create empty arrays B (where sorted array will be stored) and C (where counting will be done).

In first loop we count number of elements equal to index in array C. The array C after counting: `[2, 2, 2, 2, 1, 0, 2]`

In second loop we accumulate counts so that `C[i]` contains number of elements less or equal then `i`. Output: `[2, 4, 6, 8, 9, 9, 11]`

In third loop we need to iterate over array `A` in revered order and insert values

into empty array `B`. After inserting we will decrease counter in array `C`. Arrays after each iteration:

1. B : [_, _, _, _, _, 2, _, _, _, _, _]
   C : [2, 4, 5, 8, 9, 9, 11]
2. B : [_, _, _, _, _, 2, _, 3, _, _, _]
   C : [2, 4, 5, 7, 9, 9, 11]
3. B : [_, _, _, 1, _, 2, _, 3, _, _, _]
   C : [2, 3, 5, 7, 9, 9, 11]
4. B : [_, _, _, 1, _, 2, _, 3, _, _, 6]
   C : [2, 3, 5, 7, 9, 9, 10]
5. B : [_, _, _, 1, _, 2, _, 3, 4, _, 6]
   C : [2, 3, 5, 7, 8, 9, 10]
6. B : [_, _, _, 1, _, 2, 3, 3, 4, _, 6]
   C : [2, 3, 5, 6, 8, 9, 10]
7. B : [_, _, 1, 1, _, 2, 3, 3, 4, _, 6]
   C : [2, 2, 5, 6, 8, 9, 10]
8. B : [_, 0, 1, 1, _, 2, 3, 3, 4, _, 6]
   C : [1, 2, 5, 6, 8, 9, 10]
9. B : [_, 0, 1, 1, 2, 2, 3, 3, 4, _, 6]
   C : [1, 2, 4, 6, 8, 9, 10]
10. B : [0, 0, 1, 1, 2, 2, 3, 3, 4, _, 6]
    C : [0, 2, 4, 6, 8, 9, 10]
11. B : [0, 0, 1, 1, 2, 2, 3, 3, 4, 6, 6]
    C : [0, 2, 4, 6, 8, 9, 9]

### Final sorted array

The final sorted array is: `[0, 0, 1, 1, 2, 2, 3, 3, 4, 6, 6]`