Exercise 1: punti 18

Simulate the Counting Sort algorithm (stable version) on sorting the array 5 5 6 1 2 1 4 3 5.

Exercise 2: punti 7

Consider a binary tree T with n nodes. Each node u has a color $u.\mathsf{color}$ which could be either white or black.

Design and analyze an efficient algorithm to compute the number of nodes u such that the subtree rooted at u contains more white nodes than black nodes.

Exercise 3: punti 5

Given an undirected graph $G = (V, E)$, the measure $d(G)$ equals the largest smallest distance between to pairs of nodes in the graph. More precisely, let $m(u, v)$ be the length of the shortest path from u to v, $d(G) = max_{(u,v)\in V^2} m(u, v)$.

If the graph is not connected, $d(G) = +\infty$.

Design an algorithm to compute $d(G)$.

Exercise 1: punti 18

Simulate the Counting Sort algorithm (stable version) on sorting the array 5 5 6 1 2 1 4 3 5.

C

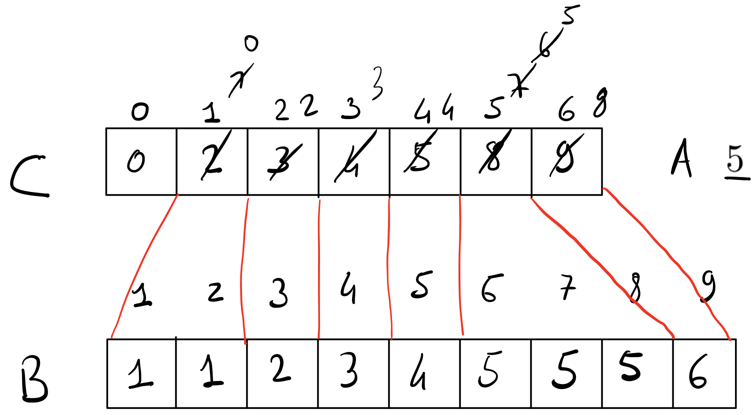
0	1	2	3	4	5	6
0	2	1	1	1	3	1

C[i] = # occs of i in A

C

0	1	2	3	4	5	6
0	2	3	4	5	8	9

C[i] = # items ≤ i



C

0	1	2	3	4	5	6
	~~1~~ 0	2	3	4	~~7~~ ~~6~~ 5	8
0	~~2~~	~~3~~	~~4~~	~~5~~	~~8~~	~~9~~

A 5 5 6 1 2 1 4 3 5.

B

1	2	3	4	5	6	7	8	9
1	1	2	3	4	5	5	5	6

Exercise 2: punti 7

Consider a binary tree T with n nodes. Each node u has a color u.color which could be either white or black.

Design and analyze an efficient algorithm to compute the number of nodes u such that the subtree rooted at u contains more white nodes than black nodes.

```
                                    // returns   # of node satisfying
                                                   the property
count (x)                                        # of white nodes
    if x == NIL :   return 0, 0, 0               # of black nodes
    rl, wl, bl = count (x.left)
    rr, wr, br = count (x.right)
    rx = rl + rr
    wx = wl + wr
    bx = bl + br
    if x.color == "white"   :  wx += 1
    else                    :  bx += 1
    if wx > bx : rx += 1
    return rx, wx, bx
```

Exercise 3: punti 5

Given an undirected graph $G = (V, E)$, the measure $d(G)$ equals the largest smallest distance between to pairs of nodes in the graph. More precisielly, let $m(u, v)$ be the length of the shortest path from u to v, $d(G) = max_{(u,v)\in V^2} m(u, v)$.

If the graph is not connected, $d(G) = +\infty$.

Design an algorithm to compute $d(G)$.

$$\max_{u \in V} \underbrace{\max_{v \in V} m(u, v)}_{}$$

```
diameter (G)
    d = $-\infty$
    for $u \in V$
        BFS (u)
        for $v \in V$
            if v.color == white
                return $+\infty$
            d = max (d, v.d)
    return d
```

complexity : $\Theta(|V| \cdot (|V| + |E|))$ time