Exercise 1: punti 18

Show the steps performed by MergeSort to sort the array 8 6 5 2 1 9 10 7.

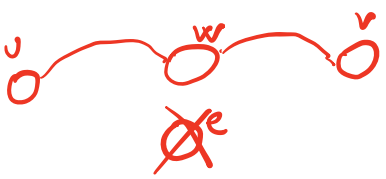
Exercise 2: punti 7

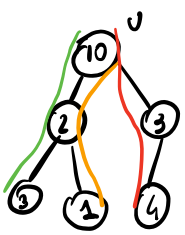
Given a tree T with n nodes. Every node u has a weight $u.w$. Let $s(u)$ be the smallest sum of weights on a path from u to a leaf (in the subtree rooted at u). In order words, consider all the paths from u to a leaf. For each of them consider the sum of the weigths on its nodes. The value $s(u)$ is the smallest of these sums.

Design an efficient algorithm to count the number of nodes u such that $s(u) > u.w$. Analyse the complexity of the proposed algorithm.

Exercise 3: punti 5

Given an undirected graph $G = (V, E)$ and four vertexes u, v, w and e, design an efficient algorithm to compute the shortest path from u to v that includes vertex w but excludes vertex e. Analyse the complexity of the proposed algorithm.



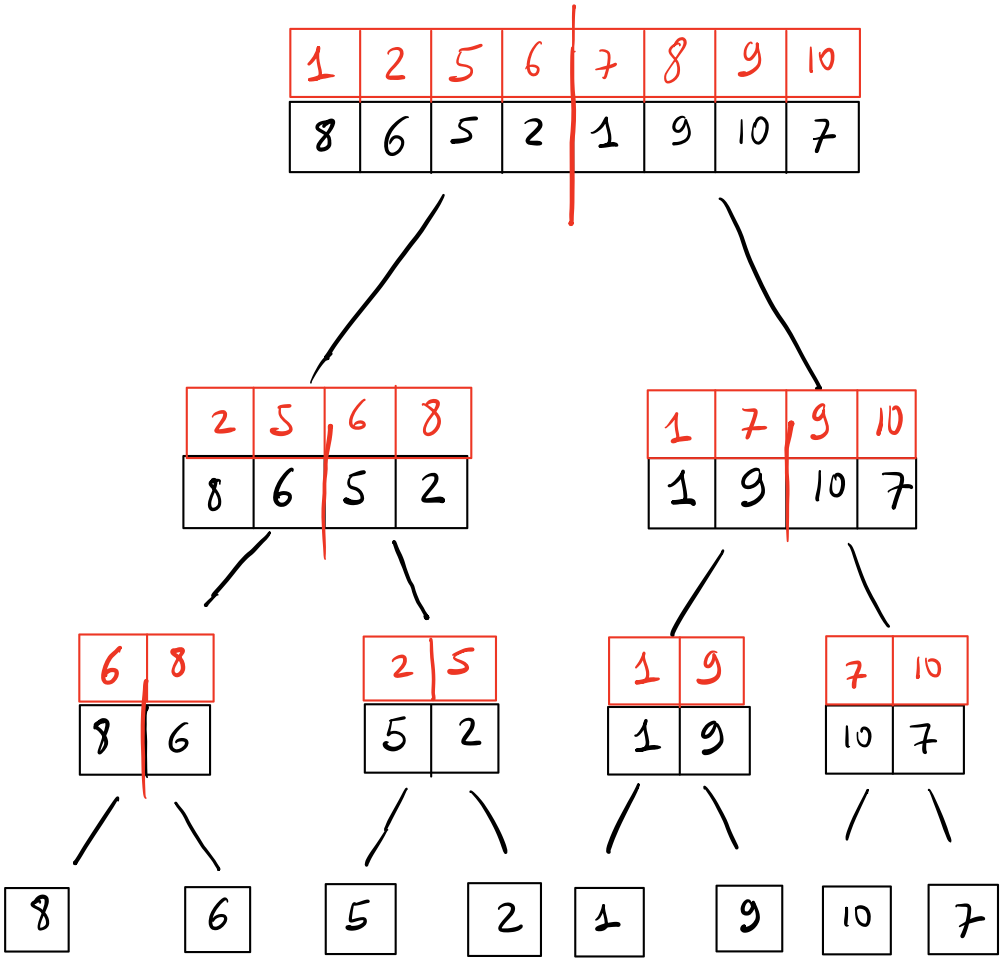


10 + 2 + 3 = 15
10 + 2 + 1 = 13
10 + 3 + 4 = 17

s(u = 10) = 13

Exercise 1: punti 18

Show the steps performed by MergeSort to sort the array 8 6 5 2 1 9 10 7.

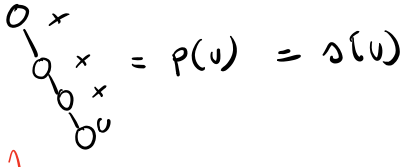


Exercise 2: punti 7

Given a tree T with n nodes. Every node u has a weight $u.w$. Let $s(u)$ be the smallest sum of weights on a path from u to a leaf (in the subtree rooted at u). In order words, consider all the paths from u to a leaf. For each of them consider the sum of the weigths on its nodes. The value $s(u)$ is the smallest of these sums.

Design an efficient algorithm to count the number of nodes u such that $s(u) > 10 \cdot u.w$. Analyse the complexity of the proposed algorithm.

```
p(u)                                  // - # nodes satisfying the property
                                      // return
                                      // - smallest sum on u-to-leaf path (i.e., s(u))
  if u == NIL  return 0, 0
  rl, sl = p(u.left)
  rr, sr = p(u.right)
  ru = rl + rr
  su = u.w + min(sl, sr)
  if su > 10 * u.w : ru += 1
  return ru, su
```




```
p(u, ppu)
  if u == NIL  return 0, 0        pu = ppu + u.w
  rl, sl = p(u.left, pu)
  rr, sr = p(u.right, pu)
  ru = rl + rr
  su = u.w + min(sl, sr)
  if su == pu : ru += 1
  return ru, su
```

Exercise 3: punti 5

Given an undirected graph $G = (V, E)$ and four vertexes u, v, w and e, design an efficient algorithm to compute the shortest path from u to v that includes vertex w but excludes vertex e. Analyse the complexity of the proposed algorithm.


```
f(G(V,E), u, w, v, e)
    V = V \ {e}
    BFS(u)
    d = w.d
    BFS(w)
    d += v.d
    return d
```

```
c = w          if you want
p = []         the path
while c != NIL
        p.append(c)
        c = w.π
```