

**Esercizio 1** (Stringhe binarie). Data in input una stringa binaria di lunghezza  $n$ , progettare un algoritmo di complessità  $O(n \log n)$  che restituisca il numero delle sue sottostringhe che cominciano con 0 e finiscono con 1.

```

Sb(A: array) { //  $O(n)$ 
  n2 = 0
  sol = 0
  For (i = 0, 1, ..., A.length() - 1) {
    if (A[i] == 0) { n2++ }
    else { sol += n2 }
  }
  return sol
}

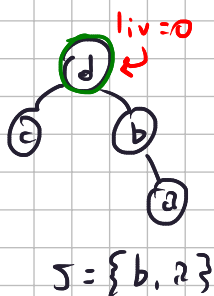
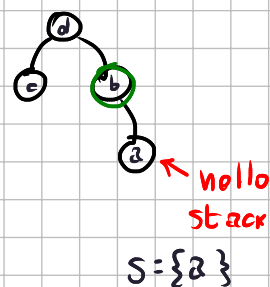
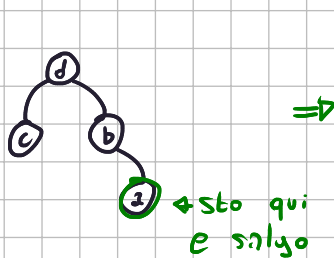
```

**Esercizio 2** (Nodi pari). Sia  $T$  un albero radicato nel nodo  $x$  e rappresentato da un vettore dei padri  $P$ . Progettare un algoritmo di complessità  $O(n)$  che dato in input il vettore  $P$  restituisca il numero dei livelli di  $T$  in cui compaiono un numero pari di nodi.

```

LivPadri(P: array) {
  n = P.length()
  S: Stack
  Liv[n] = {-1, -1, ..., -1}
  Vis[n] = {0, 0, ..., 0}
  Vis[∞] = 1
  Liv[∞] = 0
  For (i = 0, ..., n-1) { //  $O(n)$ 
    cur = i
    level = 0
    while (P[cur] ≠ cur) {
      if (Vis[cur] ≠ 1) {
        Vis[cur] = 1
        S.push(cur)
      } else {
        level = Liv[cur]
        break
      }
    }
    while (S ≠ ∅) {
      level++
      Liv[S.pop()] = level
    }
  }
  m = max(Liv)
  D[m] = {0, 0, ..., 0}
  For (i = 0, ..., m-1) {
    D[i]++
  }
  livP = 0
  For (i = 0, ..., m-1) {
    if (D[i] ≠ 0 ∧ D[i] % 2 == 0) { livP++ }
  }
  return livP
}

```



Scorro lo stack incrementando il liv

liv = 0	S = {b, a}	
liv = 1	S = {a}	b
liv = 2	S = { }	a

