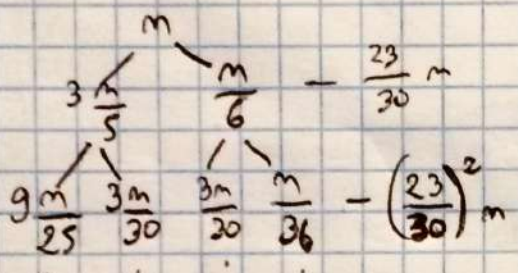
CORREZIONE I° APPELLO

$$T(n) = \begin{cases} 1 & n = 1 \\ 3T\left(\frac{n}{5}\right) + T\left(\frac{n}{6}\right) + n & n > 1 \end{cases} = (H)(n)$$



$$\sum_{k=0}^{\log_{\frac{6}{5}} n} a^k n \leq \sum_{k=0}^{\infty} a^k n \qquad a < 1$$

$$= \frac{1}{1-a}$$

$$T(n) = an + b$$

oppure

$$T(n) \leq cn \quad e \quad T(n) \geq dn \qquad \forall n \geq n_0, \exists c, d > 0$$

$$T(n) = 3T\left(\frac{n}{5}\right) + T\left(\frac{n}{6}\right) + n$$

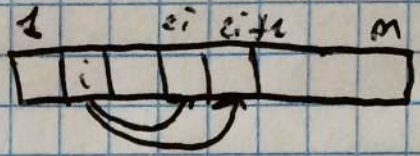
ip. ind. $T\left(\frac{n}{5}\right) \leq c\frac{n}{5}$ $\qquad T\left(\frac{n}{6}\right) \leq c\frac{n}{6}$

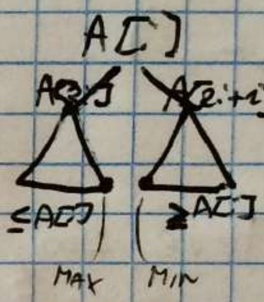
$$\leq 3\frac{n}{5}c + \frac{n}{6}c + n$$

$\forall c > 0$

$= \frac{23}{30} nc + n \le cn$

si vuole che $\frac{7}{30} c \ge 1$ → $c \ge \frac{30}{7}$ $\forall n \ge 1$






```
IsABR(A, n, i)
  if i > n
    return true
  else
    okL = IsABR(A, n, 2i)
    okR = IsABR(A, n, 2i+1)
    m = MAX(A, n, 2i)
    M = MIN(A, n, 2i+1)
    return okL AND okR AND A[i] ≥ m AND A[i] ≤ M

MIN(A, i, n)
  if i > n
    return +∞
  else
    while 2i ≤ n
      i = i·2
    return A[i]
```

$T(n) = 2T(\frac{n}{2}) + 2\log \ldots \quad O(n \log \ldots)$

return OK1 AND OK2 AND A[j] ≥ m AND A[

```
MIN(A, i, n)
  if i > n
    return +∞
  else
    while 2i ≤ n
      i = i · 2
    return A[i]
```

$$T(n) = 2T\left(\frac{n}{2}\right) + 2\log n = O(n \log n)$$

$n^{\log_2 2} = n$ vs $\log n$

$f(n) = \log n = O(n)$

$= O(n^{1-\varepsilon}) \quad \varepsilon > 0$

CASO 1

$T(n) = \Theta(n^{\log_2 2}) = \Theta(n)$

IsABRRec (A, n, i)

→ a) è ABR ? T/F
b) min
c) max

```
IsABRRec (A, n, i)
    if i > n
        return True, +∞, -∞
    else
        OKL, minL, maxL = IsABRRec (A, n, 2i)
        OKR, minR, maxR = IsABRRec (A, n, 2i+1)
        return OKL AND OKR AND maxL ≤ A[i] AND A[i] ≤ minR,
                    minL,
                    maxR
```

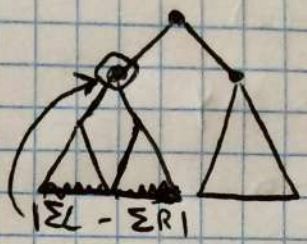
Visit & Check (A, i, last)

return OKL AND OKR AND maxL ≤ A[i] AND A[i] ≤ minR,
minL,
maxR

```
Visit&Check (A, i, last)
  if i>n
    return True, last
  else
    OKL, last = Visit&Check (A, 2i, last)
    OK = last ≤ A[i]
    OKR, last = Visit&Check (A, 2i+1, A[i])
    return OK AND OKL AND OKR, last
```

Visit&Check(A, 1, -∞)

```
degree(x)
  if x = nil
     return 0
  else
     l = sum(x.l)
     r = sum(x.r)
     return |l - r|
```




```
sum(x)
  if x = nil                    x.l = nil AND x.r = nil
     return 0
  else if leaf(x)
     return x.key
  else
     return sum(x.l) + sum(x.r)
```

somma delle chiavi delle foglie dei sottoalberi

```
sdegree(x)
  if x = nil
```

sdegree(x)

if x = nil

return 0

else

return max{degree(x), sdegree(x.l), sdegree(x.r)}

~~$T(n) = 2T(\frac{n}{2}) + n$~~

~~$\rightsquigarrow \Theta(n \log n)$~~

→ caso pessimo

$T(n) = T(n-1) + n \rightsquigarrow \Theta(n^2)$

sdegree(x) $\rightsquigarrow$ 1) max grado dei nodi del sottoalb. con radice x
2) somma delle foglie

if x = nil

sum = 0

sdegree = 0

```
else if leaf(x)
    sum = x.key
    sdegree = 0                                   O(n)
else
    suml, sdegreel = sdegree(x.l)
    sumr, sdegreer = sdegree(x.r)
    return suml + sumr,
           max{sdegreel, sdegreer, |suml - sumr|}

sdegree(T)
    -, degree = sdegree(T.root)
    return degree
```

somma S

bancnote $b_1, \dots, b_m$

$v_1, \dots, v_m$

$C(S', j)$ = "conto" per pagare S' con $b_1, \dots, b_j$

$$C(S', j) = \begin{cases} \infty & \text{se } S' > 0, j = 0 \\ 0 & \text{se } S' = 0 \\ \min\{C(S', j-1), C(S' - v_j, j-1) + 1\} & \text{se } S' > 0, j > 0 \text{ e } v_j \le S' \\ C(S', j-1) & \text{altrimenti} \end{cases}$$

$b_1 \dots b_j$

somma S

banconote $b_1 \dots b_m$

$v_1 \dots v_m$

$C(S', j)$ = "costo" per pagare S' con $b_1, \dots, b_j$

$$C(S', j) = \begin{cases} \infty & \text{se } S' > 0, j = 0 \\ 0 & \text{se } S' = 0 \\ \min \begin{cases} C(S', j-1) \\ C(S' - v_j, j-1) + 1 \end{cases} & \text{se } S' > 0, j > 0 \text{ e } v_j \leq S' \\ C(S', j-1) & \text{altrimenti} \end{cases}$$

$b_1 \dots b_j$

- soluzione ottima per il problema S' con $b_1 \dots b_m$ sono $b_{i_1} \dots b_{i_k}$ si decompone in b_{i_1} e $b_{i_2} \dots b_{i_k}$ soluzione ottima per il problema di pagare $\{S - v_{i_1}$ con $\{b_1 \dots b_m\} \setminus \{b_{i_1}\}\}$

```
banconote (S, v, m)          v[1...m]
    for j=0 to m
        for s'=0 to S          c[s', j]
            c[s', j] = -1
    return banconote_rec (S, v, m, c)

banconote_rec (s', v, j, c)
    if c[s', j] = -1
        if s' = 0
            c[s', j] = 0
        else if j = 0
            c[s', j] = ∞
        else if v[j] ≤ s'
            c[s', j] = min {banconote_rec (s', v, j-1, c), banconote_rec (s'-v[j], v, j-1, c) + 1}
        else
            c[s', j] = banconote_rec (s', v, j-1, c)
    return c[s', j]
```

costo $\Theta(S \cdot m)$

banconote:
6, 5, 5, 1, 1, 1, ^ devo pagare S = 15

la scelta greedy di usare la banconota più grossa non funziona per l'ultimo