DOMANDA A:

$0 \leq f(n) \leq cg(n)$ [illegible]

$f(n) = \Omega\, g(n)$
$g(n) = \Omega\, h(n)$ $\rightarrow f(n) = \Omega\,(h(n))$

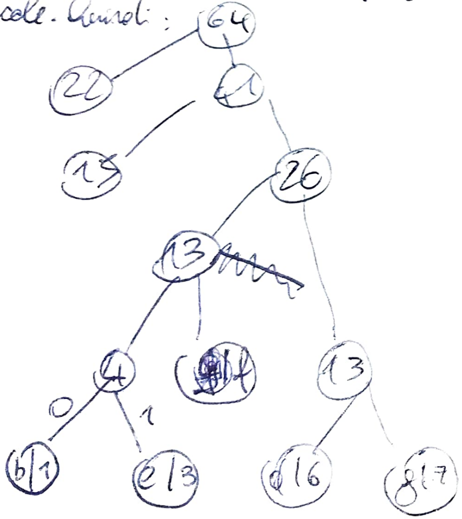
$0 \leq f(n) \leq c(g(n)) \leq c\,h(n)$

Per transitività già si nota come, per $c \geq 0$, sia verificata la disuguaglianza e sia già lim. [illegible]

DOMANDA B.

a	b	c	d	e	f	g
22	1	15	6	3	9	7

Per ciascun carattere ne si vede la frequenza, inversamente e si costruisce in ordine di ciascuna, partendo da piccole. Quindi:



DOMANDA C

Un min heap è un albero heap (albero binario quasi completo) che presenta come elemento radice l'elemento minimo di tutto l'albero. Il parent (2 * i (pos. del nodo)) è sempre minore o uguale ai discendenti (nodo left, 2i, nodo right 2i + 1)

```
                    n = A.Heapsize
isMinHeap (A, n)    for (int i = 0; i < n/2; i++)
  if (n == 0 || 2i > A.Heapsize) return true
  left = A[i] <= A[2i] and isMinHeap (A[2i+1], n)
  right = A[i] <= A[2i+1] and isMinHeap (A[2i+1], n);
   if (left && right) return true
    else return false
```

ESERCIZIO 1

ESERCIZIO 1

```
alt(A, n)
 return AltRec(A, n, 0) or Altrec(A, n, 1)
AltRec(A, i, pos)
 if i == 1 return true
 else if i == 0
 return AltRec(A, i-1, 1) and (A[i-1] ≤ A[i]
else
return AltRec(A, i-1, 0) and (A[i-1] ≥ A[i]
```

O(n) come complessità

ESERCIZIO 2

$A[1 \dots n]$ SOTTOARRAY NON VUOTO DI SOMMA MASSIMA

$i, j \mid 1 \leq i \leq j \leq n \mid A[i] + A[i+1] \dots + A[j]$ SIA MASSIMO

AD ES: $[-10, 4, 1, -1, 2, -1] \rightarrow$ SOTTOARRAY È $[4, 1, -1, 2]$

① $\begin{cases} 1 & \text{se } J_{n-1} = 0, \; i = 1 \\ \ell_{j-1} + 1 & \text{se } i > 1, \; J_{n-1} \neq 0 \end{cases}$

② MEMOIZZATO

```
MAXSUM (A, n)
L[0] = A[0]
FOR i = 1 TO N
IF(L[i+1] ≥ 0)
 L[i+1] = L[i] + A[i+1] + 1
  ELSE
 L[i+1] = A[i+1]
                    MAX = A[0]
FOR I = 1 TO N
   IF (A[i] > MAX)
     MAX = A[i]
   RETURN MAX
```

③ MEMORIZZ. DEGLI ELEMENTI (RITORNANO ANCHE IL SOTTOARRAY)

```
MAX SUM(A, N)          MAX = 0
L[0] = A[0]
S[0] = 1
  FOR I = 1 TO N
    IF(L[i+1] ≥ 0)
    L[i+1] = L[i] + A[i+1] + 1
                         S[i+1] = S[i]
     ELSE
     L[i+1] = A[i+1]
      S[i+1] = i+1

    FOR i = 1 TO N
    IF (L[i] > MAX)
      MAX = L[i]
     RETURN (S[MAX], MAX)          O(n)
```