

**ESERCIZI ONLINE  
ESAME  
ALGORITMI 2019-20**

ES1

Si valuti con si/no le seguenti affermazioni:

$$800n + n^2 + n \log n \in O(n \log n)$$

si	
----	--

$$\frac{n^2+6n+9}{n+1} \in \Theta(n^2)$$

no	
----	--

$$\frac{n^2+6n+9}{n+1} \in \Theta(n)$$

si	
----	--

$$3^{n-2} + 2^{n+5} \in \Omega(3^n)$$

si	
----	--

La risposta corretta è:  $800n + n^2 + n \log n \in O(n \log n)$

- no,  $\frac{n^2+6n+9}{n+1} \in \Theta(n^2)$
- no,  $\frac{n^2+6n+9}{n+1} \in \Theta(n)$
- si,  $3^{n-2} + 2^{n+5} \in \Omega(3^n)$
- si

1. No, no, si, si

ES2

- Riportare le caratteristiche di un *heap minimo*.
- Il seguente array rappresenta un *heap minimo*.

(1, 2, 3, 10, 8, 9, 7, 14, 15, 16, 13)

Effettuare l'estrazione del minimo riportando l'array che rappresenta lo heap dopo ogni scambio di elementi.

Uno heap minimo è un albero binario semi-completo sinistro, cioè è completo sino al penultimo livello mentre ciascun vertice dell'ultimo non ha vuoti alla sua sinistra; inoltre la chiave di ogni vertice padre è  $\geq$  di quella dei suoi figli nel caso ci fossero. Nella rappresentazione in forma di array H l'eventuale figlio sinistro di  $H[i]$  è  $H[2i]$  ed il suo eventuale figlio destro è  $H[2i + 1]$ .

quindi:

(13, 2, 3, 10, 8, 9, 7, 14, 15, 16)

(2, 13, 3, 10, 8, 9, 7, 14, 15, 16)

(2, 8, 3, 10, 13, 9, 7, 14, 15, 16)

Nella prima riga il 13 e il 2

Nella seconda riga il 13 e l'8

questi elementi sopraindicati sono coinvolti nello scambio che produce la riga successiva.

ES3

Si consideri l'algoritmo:

```

ALG( $n$ )
  if  $n \leq 50$  then return 1
  else
    if  $n \leq 100$  then
      return  $4 \cdot \text{ALG}(\lfloor n/4 \rfloor) + 7$ 
    else
       $m \leftarrow 0$ 
      for  $i \leftarrow 1$  to 3 do
         $m \leftarrow m + \text{ALG}(\lfloor n/3 \rfloor)$ 
  return  $m$ 

```

La sua funzione tempo in termini di  $n$  soddisfa la relazione di ricorrenza:

- $T(n) = T(n/4) + 1$
- $T(n) = 4T(n/4) + 1$
- $T(n) = 3T(n/3) + 1$  ✓

Punteggio ottenuto 1,0 su 1,0

La risposta corretta è:  $T(n) = 3T(n/3) + 1$

che dopo  $k$  svolgimenti ha la forma

- a:

$$3^k T(n/3^k) + \sum_{i=0}^{k-1} 3^i$$

- b:

$$4^k T(n/4^k) + \sum_{i=0}^{k-1} 4^i$$

- c:

$$T(n/4^k) + k$$

- a
- b ✗
- c

Punteggio ottenuto 0,0 su 1,0

La risposta corretta è: a

da cui risulta che  $T(n)$  ha complessità

- $\Theta(n)$
- $\Theta(n \log_3 n)$
- $\Theta(\log_4 n)$  ✗

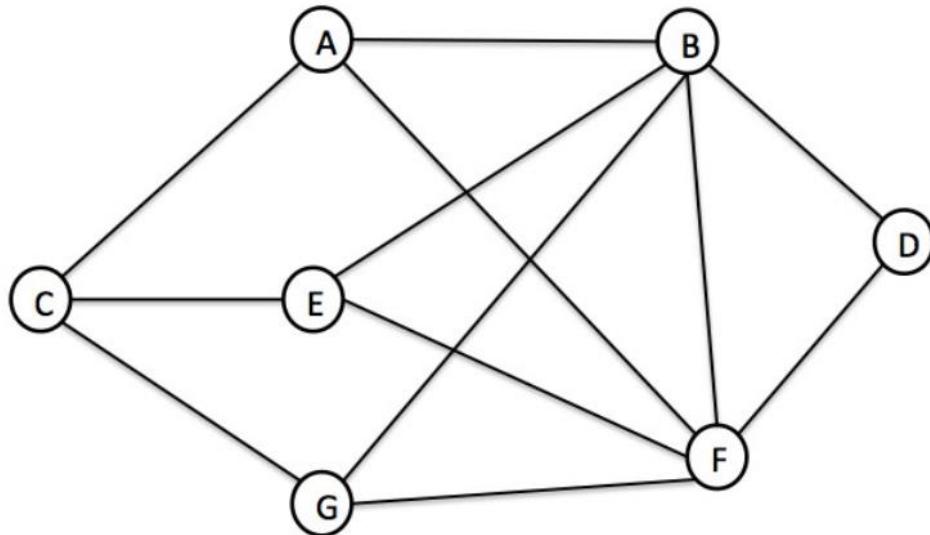
Punteggio ottenuto 0,0 su 1,0

La risposta corretta è:  $\Theta(n)$

1.  $T(n)=3T(n/3)+1$
2. A
3. Theta( $n$ )

ES4

Un grafo  $G = (V, E)$  si dice *bipartito* se esiste una partizione di  $V$  in due sottoinsiemi  $V_1$  e  $V_2$  tale che nessun arco  $(u, v) \in E$  abbia i vertici  $u, v$  contenuti nella stessa parte, ossia per ogni  $(u, v) \in E$  o  $u \in V_1$  e  $v \in V_2$ , oppure  $u \in V_2$  e  $v \in V_1$ . Si decida se il seguente grafo è bipartito oppure no, fornendo una partizione  $V_1, V_2$  di  $V$  nel caso affermativo; si spieghi perché nessuna partizione di  $V$  soddisfa la proprietà succitata nel caso negativo.



il grafo non è bipartito perchè contiene cicli di lunghezza dispari, come ad esempio BEF. Un grafo è bipartito se ogni suo sottografo lo è ; d'altra parte il sottografo BEF non è bicolorabile o non è bipartito, poichè almeno due vertici adiacenti devono avere lo stesso colore.

ES5

Si dica quali dei seguenti asserti equivale a  $g(n) \in \Theta(f(n))$ :

$\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = l > 0 \wedge l < \infty$  Scegli... ▾

$g(n) \in O(f(n)) \wedge f(n) \in \Omega(g(n))$  Scegli... ▾

$\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = 0$  Scegli... ▾

$g(n) \in O(f(n)) \wedge g(n) \in \Omega(f(n))$  Scegli... ▾

1. vero
2. falso
3. falso
4. vero

ES6

Siano  $f(n), g(n)$  funzioni asintoticamente positive; si dica quali delle seguenti

se  $f(n) \in \Theta(g(n))$  allora  $g(n) \in \Theta(f(n))$  no ▾

se  $f(n) \in O(g(n))$  allora  $2^{f(n)} \in O(2^{g(n)})$  si ▾

$\min\{f(n), g(n)\} \in \Theta(f(n) + g(n))$  Scegli... ▾

$f(n) \in \Theta(f(n)/2)$  si ▾

1. Si
2. Si
3. no
4. no

## ES7

Si considerino le funzioni  $f(n) = n$  e  $g(n) = n^{1+\sin n}$ . Si valuti con si/no le seguenti affermazioni:

$f(n) \notin \Omega(g(n))$

$f(n) \notin O(g(n))$

$g(n) \in O(f(n))$

$f(n) \in \Omega(g(n))$

1. no
2. no
3. no
4. si

## ES8

Si consideri l'algoritmo:

```
ALG(n)
  if  $n \leq 50$  then return 1
  else
    if  $n \leq 100$  then
      return  $4 \cdot \text{ALG}(n - 1) + 7$ 
    else
       $m \leftarrow \text{ALG}(n - 1)$ 
      for  $i \leftarrow 1$  to  $n$  do
        for  $j \leftarrow 1$  to  $n$  do
           $m \leftarrow m + i + j$ 
      return  $m$ 
```

La sua funzione tempo in termini di  $n$  soddisfa la relazione di ricorrenza:

- $T(n) = T(n - 1) + n$
- $T(n) = 2T(n - 1) + n^2$
- $T(n) = T(n - 1) + n^2$

che dopo  $k$  svolgimenti ha la forma

\* a:

$$2^k T(n - k) + \sum_{i=0}^{k-1} (n - i)^2$$

\* b:

$$T(n - k) + \sum_{i=0}^{k-1} (n - i)$$

\* c:

$$T(n - k) + \sum_{i=0}^{k-1} (n - i)^2$$

- a
- b
- c

da cui risulta che  $T(n)$  ha complessità

- $\Theta(n^2)$
- $\Theta(2^n)$
- $\Theta(n^3)$

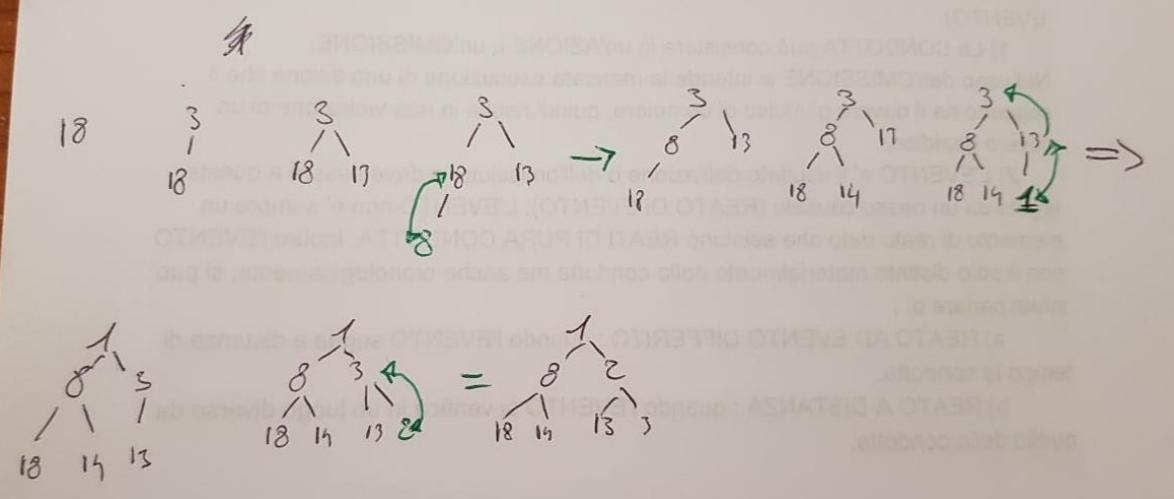
1. C
2. C
3. C

### ES9

Si inseriscano in un heap minimo inizialmente vuoto le seguenti chiavi: 18, 3, 13, 8, 14, 1, 2. Si riporti il vettore che rappresenta lo heap dopo ogni inserimento.

[18] -> [3,18] -> [3, 18, 13] -> [3, 8, 13, 18] -> [3, 8, 13, 18, 14] -> [1, 8, 3, 18, 14, 13] -> [1, 8, 2, 18, 14, 13]

18  
3  
13  
8  
14  
1  
2



### ES10

Si inseriscano in un heap minimo inizialmente vuoto le seguenti chiavi: 22, 7, 17, 12, 18, 5, 6. Si riporti il vettore che rappresenta lo heap dopo ogni inserimento.

1.  $[22] \rightarrow [7,22] \rightarrow [7,22,17] \rightarrow [7,12,17,22] \rightarrow [7,12,17,22,18] \rightarrow [5,12,7,22,18,17] \rightarrow [5,12,6,22,18,17,7]$

## ~~ES11~~

Si consideri il seguente algoritmo.

```

ALG( $A[1..n]$ )
  for  $i \leftarrow n - 1$  down to 1 do
     $j \leftarrow i$ 
    while  $j < n$  and  $A[j] < A[j + 1]$  do
      scambia  $A[j + 1]$  con  $A[j]$ 
       $j \leftarrow j + 1$ 

```

Si risponda succintamente alle seguenti domande:

1. Cosa fa l'algoritmo?
2. Si indichino l'invariante del ciclo esterno e l'invariante del ciclo interno.
3. Quali sono il caso peggiore e la sua complessità in termini di  $\Theta$ ?
4. Sapreste indicare un algoritmo che risolva lo stesso problema in tempo asintoticamente migliore?

1 ordina l'array in ordine decrescente

2. esterno  
 $i < n \wedge i \geq 1$

interno

$\forall A[i] \geq A[i+1]$   
in  $A[j] \dots A[n-2]$

3 caso peggiore: elementi disposti in ordine crescente  $= \Theta(n^2)$

4 merge-sort

## ~~ES12~~

domanda 4  
non completato  
integro max.: 100  
contrassegna domanda

Supponendo che la funzione  $\text{Foo}(n)$  richieda tempo  $O(\log n)$ , si consideri il seguente algoritmo, di cui  $T(n)$  è la funzione tempo:

```

Gremlins( $n$ )
  // pre: n intero > 1
  if ( $n = 2$ ) then return 1
  else
     $m := \text{Gremlins}(n - 1) + \text{Foo}(n)$ 
    return  $m$ 

```

Scegli una o più alternative:

- a.  $T(n) \in O(\log n)$
- b. se  $0 \leq k \leq n$  allora  $T(n) = T(n - k) + \sum_{i=0}^{k-1} \log(n - i)$
- c. se  $0 \leq k \leq n$  allora  $T(n) = T(n - k) + \log(n - k)$
- d.  $T(n) \in O(n \log n)$

Verifica risposta

$$\begin{aligned}
 T(n) &= T(n-1) + \log n \\
 &\stackrel{k \leq n}{=} T(n-2) + \log(n-1) + \log n \\
 &= T(0) + \sum_{i=0}^{n-1} \log(n-i) = O(n \log n)
 \end{aligned}$$

### ES13

Si considerino le funzioni hash

$$h_1(k) = k \bmod m, \quad h_2(k) = 1 + (k \bmod m)$$

con cui è definita il doppio hashing:

$$h(k, i) = (h_1(k) + i h_2(k)) \bmod m$$

Lo stato della tabella hash ad indirizzamento aperto  $T[0..12]$ , di dimensione  $m = 13$  dopo l'inserimento delle chiavi: 50, 69, 72, 98, 79, 14 nell'ordine dato, risulta essere (indicando con -1 una locazione libera di  $T$ ):

- [-1, 79, 98, 14, 69, -1, -1, 72, -1, -1, -1, 50, -1]
- [-1, 79, -1, -1, 69, 98, -1, 72, -1, 14, -1, 50, -1]
- [-1, -1, 98, 14, 69, -1, -1, 72, -1, 79, -1, 50, -1]

1. Soluzione è la prima, vedi quaderno per soluzione completa

### ES14

Sia  $f(n)$  una funzione non decrescente e positiva per qualunque  $n$ .

$f(n) \in O(g(n))$  implica  $f(n) \notin \Omega(g(n))$ ? Scegli... ▾

$f(n) \in O(g(n))$  implica  $g(n) \in \Omega(f(n))$ ? Scegli... ▾

$f(n) \in O(g(n))$  implica  $f(n) \notin \Theta(g(n))$ ? Scegli... ▾

$f(n) \notin O(g(n))$  implica  $f(n) \notin \Theta(g(n))$ ? Scegli... ▾

1. Falso
2. Vero
3. Falso
4. Vero

### ES15

Siano  $f(n), g(n)$  funzioni asintoticamente positive; si dica quali delle seguenti asserzioni è vera:

$\max\{f(n), g(n)\} \in \Theta(f(n) + g(n))$  Scegli... ▾

se  $f(n) \in O(g(n))$  allora  $g(n) \in \Omega(f(n))$  Scegli... ▾

$f(n) \in \Theta(f(n)/3)$  Scegli... ▾

se  $f(n) \leq g(n)$  per ogni  $n$  allora  $g(n) - f(n) \in \Omega(g(n))$  Scegli... ▾

1. Vera
2. Vera
3. Falsa
4. Falsa

## ES16

Si effettui la visita in profondità, a partire dal nodo b, del seguente grafo di otto nodi dato tramite le liste di adiacenza (i nodi adiacenti devono essere considerati nell'ordine in cui appaiono nella lista di adiacenza):

b: d, e, f  
 c: b, h, g  
 d: f, b  
 e: d, f  
 f: b  
 g: c, f  
 h: e, f  
 a: c, g

Si riporti per ogni nodo i tempi di inizio e fine visita (ossia i valori degli attributi *d* ed *f*) e per ogni arco il suo tipo secondo la classificazione degli archi durante una visita in profondità.

## ES17

Sia dato un albero *T* (non vuoto), rappresentato con puntatori *child* e *sibling*, nel quale ogni nodo è etichettato con un numero intero. Si dia un algoritmo che, per ogni nodo che non è una foglia, stampi la somma delle etichette dei suoi figli. (Una foglia è un nodo che non ha figli.)

## ES18

Si effettui la visita in profondità, a partire dal nodo A, del seguente grafo di otto nodi dato tramite le liste di adiacenza (i nodi adiacenti devono essere considerati nell'ordine in cui appaiono nella lista di adiacenza):

A: C, D, E  
 B: A, G, F  
 C: E, A  
 D: C, E  
 E: A  
 F: B, E  
 G: D, E  
 H: B, F

Si riporti per ogni nodo i tempi di inizio e fine visita (ossia i valori degli attributi *d* ed *f*) e per ogni arco il suo tipo secondo la classificazione degli archi durante una visita in profondità.

A: d = 1, f = 8  
 B: d = 9, f = 14  
 C: d = 2, f = 5  
 D: d = 6, f = 7  
 E: d = 3, f = 4  
 F: d = 12, f = 13  
 G: d = 10, f = 11  
 H: d = 15, f = 16

(A, C) -> Arco foresta  
 (C, E) -> Arco foresta  
 (E, A) -> Arco indietro  
 (C, A) -> Arco indietro  
 (A, D) -> Arco foresta  
 (D, C) -> Arco attraversamento  
 (D, E) -> Arco attraversamento  
 (A, E) -> Arco avanti  
 (B, A) -> Arco attraversamento  
 (B, G) -> Arco foresta  
 (G, D) -> Arco attraversamento  
 (G, E) -> Arco attraversamento  
 (B, F) -> Arco foresta  
 (F, B) -> Arco indietro  
 (F, E) -> Arco attraversamento  
 (H, B) -> Arco attraversamento  
 (H, F) -> Arco attraversamento

## ES19

Sia dato un albero  $k$ -ario  $T$  non vuoto, rappresentato con puntatori *child* e *sibling* nel quale ogni nodo è etichettato con un numero intero. Si dia un algoritmo che stampi per ogni livello dell'albero la somma delle etichette dei nodi che ne fanno parte.

```

PRINT-SUM-LIV(T)
If T != nil then
  If T.child = nil then PRINT(T.key)
  While T.child != nil do
    sum <- T.key
    while T.sibling != nil do
      sum <- sum + T.sibling.key
      T <- T.sibling
    PRINT(sum)
    T <- T.child
  
```

## ES20

Si consideri il seguente algoritmo.

```

ALGO(A[0..n - 1])
Pre: A array di interi
a ← 1
for i ← 0 to n - 2 do
  j ← i + 1
  while j < n and A[j - 1] ≥ A[j] do
    j ← j + 1
  a ← max(a, j - i)
return a
  
```

Si risponda alle seguenti domande:

1. Cosa calcola *Algo(A)*?
2. Quali sono il caso peggiore e la sua complessità in termini di  $\Theta$ ?
3. Sapreste indicare un algoritmo che risolva lo stesso problema in tempo asintoticamente migliore?

1) Algo(A) calcola il numero massimo di elementi consecutivi in ordine decrescente presenti nell'array.

$$\begin{aligned}
 2) \quad T(n) &= c_1 n + c_2(n-1) + c_3 \sum_{i=1}^{n-1} i + c_4 \sum_{i=1}^{n-1} (i-1) + c_5(n-1) = \\
 &= c_1 n + c_2(n-1) + c_3((n-1)/2(n)) + c_4((n-1)/2(n-2)) + c_5(n-1) = \\
 &= (c_1+c_2)n - c_2 + c_3(n^2 - n)/2 + c_4(n^2 - 3n + 2)/2 + c_5(n-1) = \\
 &= ((c_3 + c_4)/2)n^2 + (c_1 + c_2 + c_5 + (-c_3 - (3)c_4)/2)n - c_2 + (2)c_4 - c_5
 \end{aligned}$$

Ha una complessità temporale quadratica

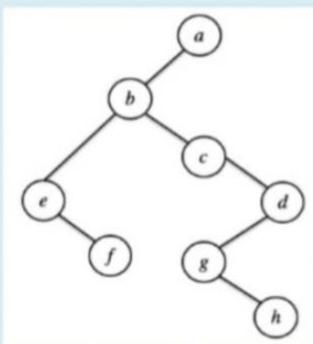
- 3) ALGO'(A)

```

a <- 1
for i <- 0 to n-2 do
  b <- 1
  if A[i] >= a[i+1] then
    b <- b+1
  else
    a <- max(a,b)
    b <- 1
return a
  
```

## ES21

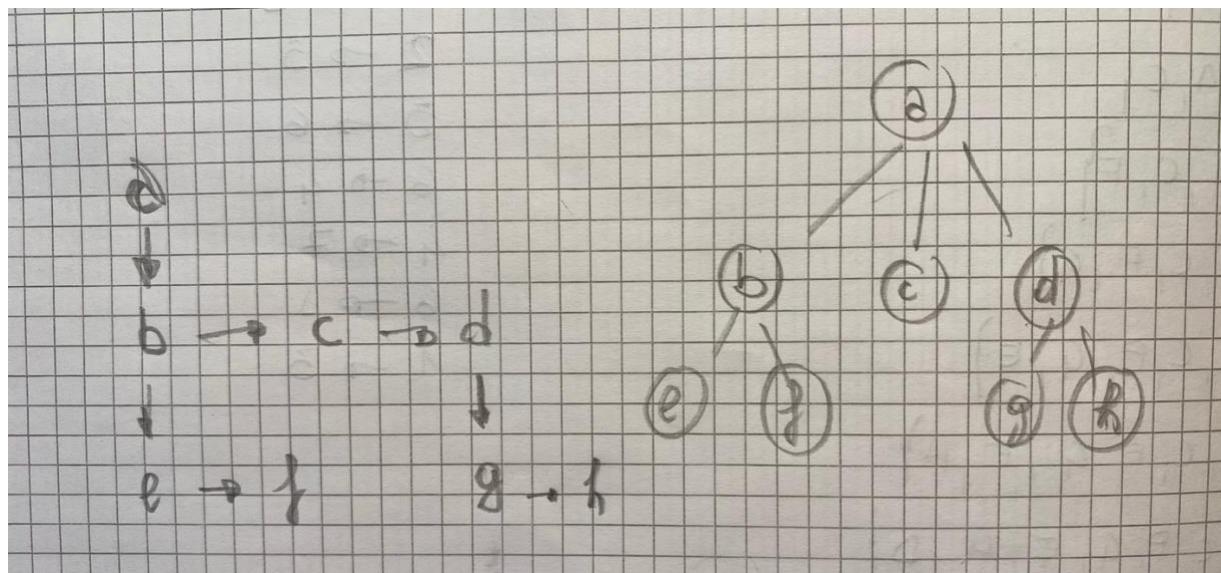
Si consideri la seguente rappresentazione di un albero  $k$ -ario con un albero binario, corrispondente alla sua memorizzazione mediante puntatori child/sibling:



Quali delle seguenti affermazioni è corretta?

Scegli una o più alternative:

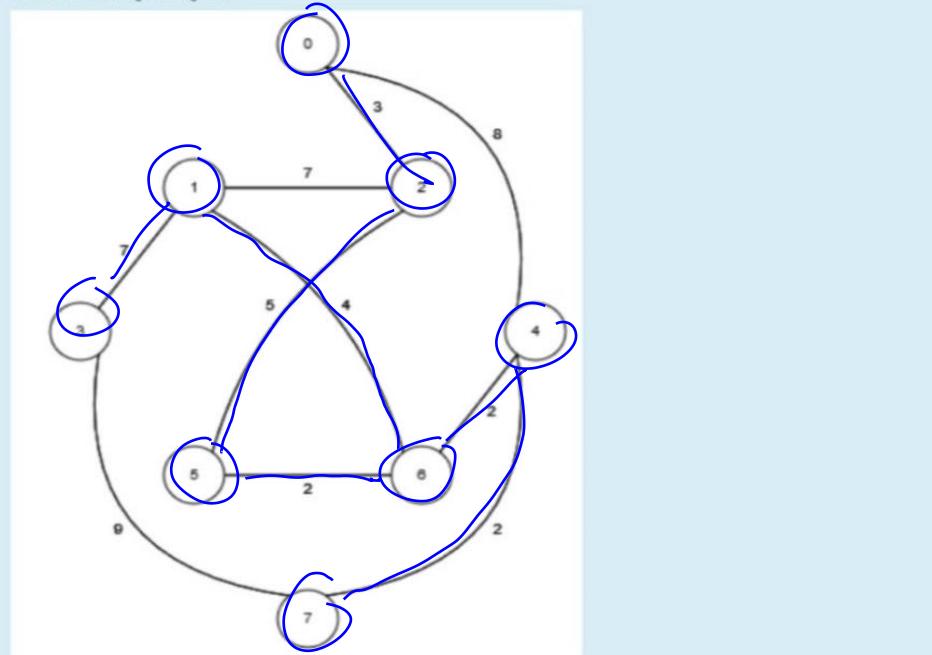
- a. l'albero ha grado 3 e l'insieme delle sue foglie è  $\{c, e, f, g, h\}$ ;
- b. l'albero ha grado 3 e i nodi  $\{a, b, c, d\}$  sono sullo stesso ramo;
- c. L'albero ha altezza 5 e l'insieme delle sue foglie è  $\{f, h\}$ .
- d. l'albero ha altezza 2 e i nodi  $\{a, d, h\}$  formano un ramo;
- e. l'albero ha grado 2 e i nodi  $\{a, b, f\}$  formano un ramo;



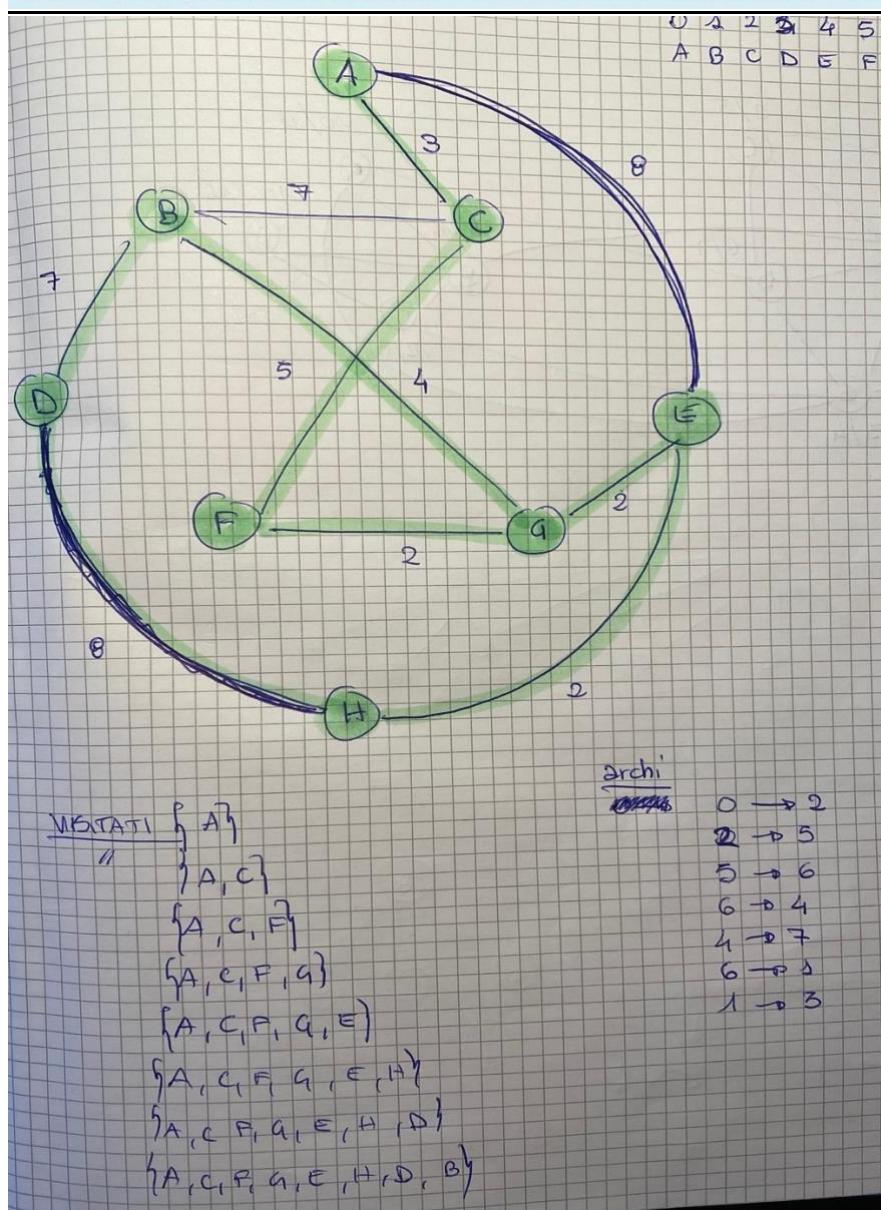
1. A, D

## ES22

Si consideri il seguente grafo:



Si simuli sul grafo l'algoritmo di Prim a partire dal nodo 0. Si riporti il risultato elencando gli archi inclusi nella soluzione.



## ES23

Si stabilisca a quali classi di complessità appartengano o meno le funzioni  $2^{n+1}$ ,  $2^{2n}$  e  $4^n$ .

Scegli una o più alternative:

- a.  $2^{n+1} \in O(2^n)$ ,  $2^{2n} \in O(4^n)$ ,  $4^n \notin O(2^n)$
- b.  $2^{n+1} \in O(2^n)$ ,  $2^{2n} \notin O(2^n)$ ,  ~~$4^n \in O(2^n)$~~
- c.  $2^{n+1} \in O(2^n)$ ,  $2^{2n} \notin O(2^n)$ ,  $4^n \notin O(2^n)$
- d.  ~~$2^{n+1} \notin O(2^n)$~~ ,  $2^{2n} \notin O(2^n)$ ,  $4^n \notin O(2^n)$

1. C poiché:

- $2^{n+1} = 2^n \times 2$  **appartiene** a  $O(2^n)$
- $2^{2n} = 2^n \times 2^n$  **non appartiene** a  $O(2^n)$
- $4^n = (2 \times 2)^n = 2^n \times 2^n$  **non appartiene** a  $O(2^n)$

La combinazione tra queste tre porta al risultato C

e >? ho?

## ES24

Sia  $T[0..m - 1]$  una tabella hash con capacità  $m = 10$  e chiavi intere. Qual è lo stato di  $T$  dopo aver inserito le chiavi 88, 12, 2, 22, 33, se si utilizza il metodo di indirizzamento aperto e la funzione hash  $h(k) = k \bmod 10$ ?

Scegli un'alternativa:

- a. {nil, nil, 12, 33, 2, 22, nil, nil, 88, nil}
- b. {nil, nil, 33, 12, 2, 22, nil, nil, 88, nil}
- c. {nil, nil, 12, 2, 22, 33, nil, nil, 88, nil}
- d. {nil, nil, 12, 22, 2, 33, nil, nil, 88, nil}

1. C

8/6/2020

Sia  $f(n)$  una funzione non decrescente e positiva per qualunque  $n$ .

$f(n) \in O(g(n))$  implica  $f(n) \notin \Omega(g(n))$ ? Scegli...

$f(n) \in O(g(n))$  implica  $g(n) \in \Omega(f(n))$ ? Scegli...

$f(n) \in O(g(n))$  implica  $f(n) \notin \Theta(g(n))$ ? Scegli...

$f(n) \notin O(g(n))$  implica  $f(n) \notin \Theta(g(n))$ ? Scegli...

Si analizzi la complessità temporale in funzione di  $n$  del seguente algoritmo:

```

ALG(n)
Pre: n > 0 intero
if n = 1 then
    return 1
else
    result ← 0
    for i ← 1 to n do
        result ← result + ALG(i - 1)
    return result
  
```

La sua funzione tempo soddisfa la relazione di ricorrenza

$T(n) = T(n - 1) \cdot n$

$T(n) = T(n - 1) + n^2$

$T(n) = 2T(n - 1) \cdot n^2$

che dopo  $k \leq n$  svolgimenti ha la forma

- Se  $f(x) = g(x)$

$$\lim_{x \rightarrow \infty} \frac{f(x)}{g(x)} = 1$$

$$0 \leq \omega < \infty \Leftrightarrow f(n) \in O(g(n))$$

$$0 < \omega < \infty \Leftrightarrow f(n) \in \Theta(g(n))$$

$$0 < \omega \leq \infty \Leftrightarrow f(n) \in \Omega(g(n))$$

- $\lim_{x \rightarrow \infty} \frac{f(x)}{g(x)} = \frac{\infty}{\infty} = \infty \quad 0 < \omega$

$$\lim_{x \rightarrow \infty} \frac{f(x)}{g(x)} = \begin{cases} f(x) > g(x) \rightarrow \infty \\ g(x) > f(x) \rightarrow 0 \end{cases}$$

- $f(n) \notin O(g(n)) \Rightarrow f(n) \notin \Theta(g(n))$

$$\lim_{x \rightarrow \infty} \frac{f(x)}{g(x)} = 0$$

21

Si analizzi la complessità temporale in funzione di  $n$  del seguente algoritmo:

```
ALG(n)
Pre:  $n > 0$  intero
if  $n = 1$  then
    return 1
else
    result  $\leftarrow 0$ 
    for  $i \leftarrow 1$  to  $n$  do
        result  $\leftarrow$  result + ALG( $i - 1$ )
    return result
```

La sua funzione tempo soddisfa la relazione di ricorrenza

$T(n) = T(n - 1) \cdot n$   $\leftarrow$

$T(n) = T(n - 1) + n^2$

$T(n) = 2T(n - 1) \cdot n^2$

che dopo  $k \leq n$  svolgimenti ha la forma

• a:

$$T(n - k) + \sum_{i=0}^{k-1} (n - i)$$

b:

$$T(n - k) \cdot \prod_{i=0}^{k-1} (n - i)$$

• c:

$$2^k T(n - k) \cdot \prod_{i=0}^{k-1} 2^i (n - i)$$

a

b

c

da cui risulta che  $T(n)$  ha complessità

$\Theta(n!)$

$\Theta(n^2)$

$\Theta(2^n n!)$

$$\begin{aligned} T(n) &= T(n-1) \cdot n \\ &= (n-1) T(n-2) \end{aligned}$$

$$\begin{aligned} k &= n \\ T(0) &\cdot \sum_{i=0}^{n-1} (n-i) \end{aligned}$$

Si consideri il seguente algoritmo.

```

ALG( $A[1..n]$ )
  for  $i \leftarrow n$  down to 2 do
     $k \leftarrow i$ 
    for  $j \leftarrow 1$  to  $i - 1$  do
      if  $A[j] < A[k]$  then
         $k \leftarrow j$ 
    scambia  $A[i]$  con  $A[k]$ 
  return  $A$ 
```

- Cosa fa l'algoritmo?
- Qual è l'invariante del ciclo esterno?
- Qual è la complessità dell'algoritmo in termini di  $\Theta$ ?
- Sapreste indicare un algoritmo che risolva lo stesso problema in tempo asintoticamente migliore?

- ordina elementi in modo non crescente
- $\forall x \in A[1..i] \quad \forall y \in A[i+1..n]. \quad x \geq y \quad \wedge \quad \forall s \in i+1..n-1. \quad A[s] \geq A[s+1]$
- $\Theta(n^2)$
- changes ~ sort  $\circ$  how sort tempo  $\Theta(n \log n)$

4

Si consideri lo heap minimo rappresentato con l'array:

[14, 32, 18, 50, 41, 23, 90, 87, 64, 53, 43]

ottenuto dopo l'estrazione del minimo 5, dallo:

[5, 14, 18, 32, 41, 23, 90, 50, 64, 53, 43, 87]

[5, 14, 23, 32, 41, 18, 90, 50, 64, 53, 43, 87]

[5, 14, 18, 50, 41, 23, 90, 32, 64, 53, 43, 87]

Da questo, dopo l'inserimento della chiave 15, si ottiene:

[14, 32, 15, 50, 41, 18, 90, 87, 64, 53, 43, 23]

[14, 32, 15, 50, 41, 23, 90, 87, 64, 53, 43, 18]

[14, 50, 15, 32, 41, 18, 90, 87, 64, 53, 43, 23]

\* Per ogni nodo, l'origine del generatore del nodo è minore dell'origine del nodo

Esercizio 4. (Punti 5) Si consideri lo heap minimo rappresentato con l'array

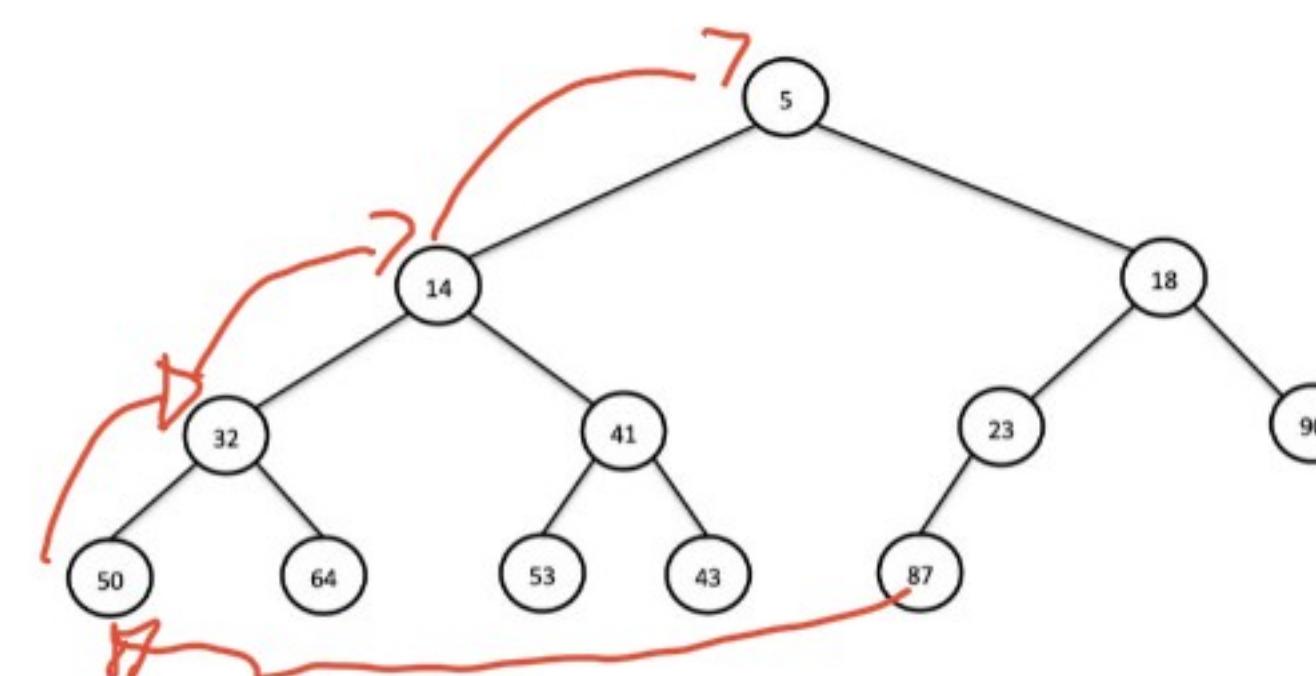
(5, 14, 18, 32, 41, 23, 90, 50, 64, 53, 43, 87)

Si effettuino le seguenti operazioni una dopo l'altra:

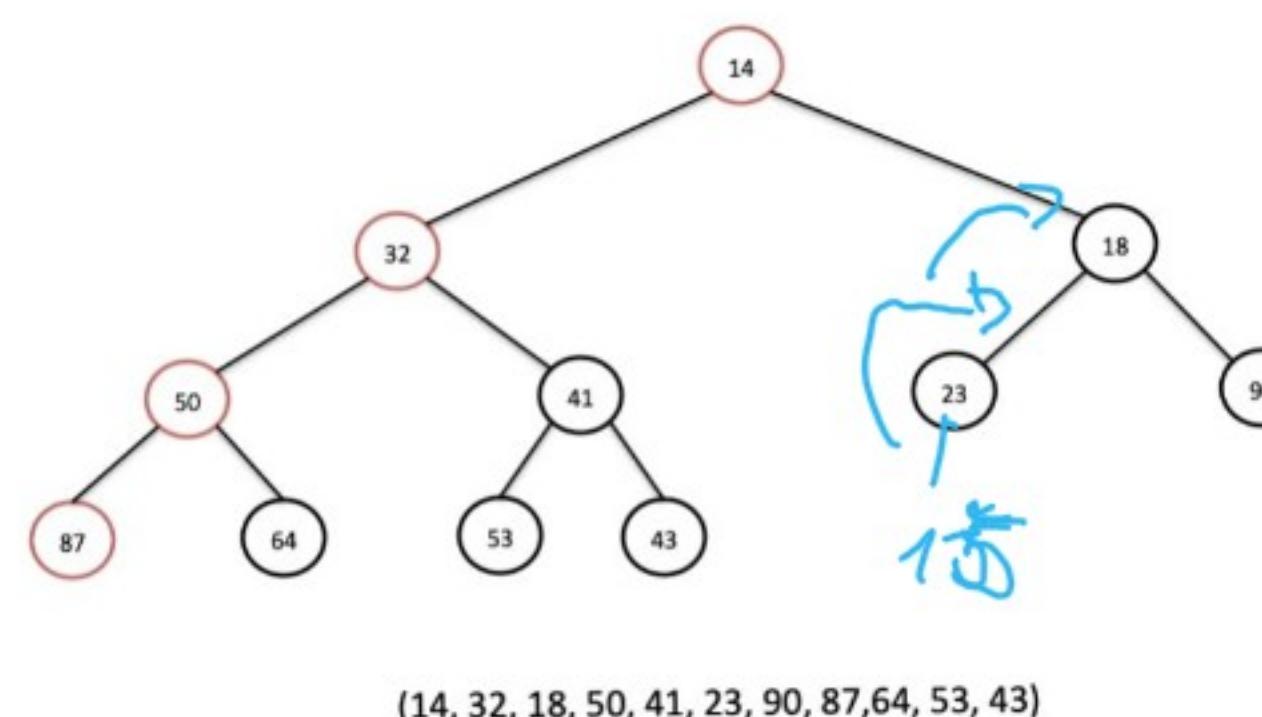
- estrazione del minimo
- inserimento della chiave 15

Si disegni lo heap risultante dopo ciascuna operazione (non dopo ciascuno scambio).

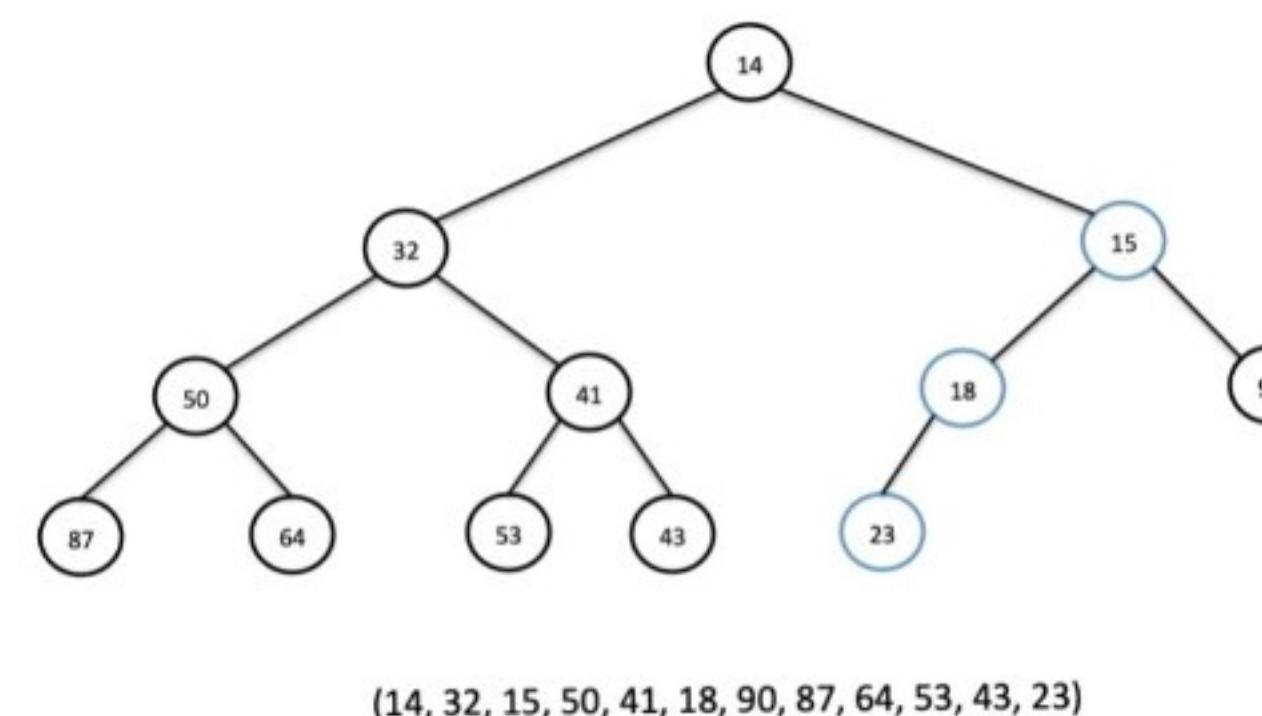
Soluzione 4. Lo heap dato risulta:



Dopo l'estrazione del minimo, ossia di 5, lo heap risulta:



Infine lo heap dopo l'inserimento della chiave 15 risulta:



Domanda 5

Risposta non ancora data

Punteggio max.:  
6,00

Sia dato un albero  $k$ -ario  $T$  non vuoto, rappresentato con puntatori *child* e *sibling*. Si dia un algoritmo  $\text{Shortest}(T)$ , dove  $T \neq \text{nil}$  punta alla radice, che restituisca il numero di nodi lungo il ramo più corto dell'albero. (Un ramo è un cammino dalla radice ad una foglia. Una foglia è un nodo che non ha figli.)

DFS DELL'ALBERO:

SHORTEST( $T$ )

Pre:  $T \neq \text{nil}$

if  $T.\text{child} = 1$  then  
return 1

else

$m \leftarrow \text{SHORTEST}(T.\text{child})$

$c \leftarrow T.\text{child}.\text{sibling}$

while  $c \neq \text{nil}$  do

$m \leftarrow \min(m, \text{SHORTEST}(c))$

$c \leftarrow c.\text{sibling}$

return  $m+1$

Durante la visita in profondità di un grafo orientato si registrano i seguenti tempi di *inizio visita* e *fine visita*:

- nodo 1: 1,8
- nodo 2: 2,7
- nodo 3: 3,6
- nodo 4: 9,10
- nodo 5: 11,12
- nodo 6: 4,5

Si classifichino i seguenti archi:

$2 \rightarrow 3$	Scegli... Albero	Arco IN AVANTI
$1 \rightarrow 3$	Scegli... Avanti	Arco verso Punteggia GENERAZIONE
$4 \rightarrow 2$	Scegli... attr.	Arco IMPOSSIBILE nel GRAFO
$6 \rightarrow 1$	Scegli... Indietro	Arco su' NODO
$6 \rightarrow 5$	Scegli... ImposS!	IMPOSSIBILE nel GRAFO
$1 \rightarrow 4$	Scegli... Impo	IMPOSSIBILE nel GRAFO



~~Domanda 1~~  
Non completato

Punteggio max.:  
4,00



Contrassegna  
domanda

Si stabilisca a quali classi di complessità appartengano o meno le funzioni  $2^{n+1}$ ,  $2^{2n}$  e  $4^n$ .

Scegli una o più alternative:

- a.  $2^{n+1} \in O(2^n)$ ,  $2^{2n} \in O(4^n)$ ,  $4^n \notin O(2^n)$
- b.  $2^{n+1} \in O(2^n)$ ,  $2^{2n} \notin O(2^n)$ ,  $4^n \in O(2^n)$
- c.  $2^{n+1} \in O(2^n)$ ,  $2^{2n} \notin O(2^n)$ ,  $4^n \notin O(2^n)$
- d.  $2^{n+1} \notin O(2^n)$ ,  $2^{2n} \notin O(2^n)$ ,  $4^n \notin O(2^n)$

Verifica risposta

$$f(n) = 2^{n+1}, \quad g(n) = 2^n \quad f(n) = 2 \cdot 2^n \quad \text{con } c=2 \quad f(n) = c \cdot g(n) \Rightarrow \in O(2^n)$$

$$f(n) = 2^{2n}, \quad g(n) = 2^n \quad f(n) = 2^n \cdot 2^n \Rightarrow \notin O(2^n)$$

$$f(n) = 4^n, \quad g(n) = 2^n \quad f(n) = (2 \cdot 2)^n \Rightarrow 4^n \notin O(2^n)$$

Domanda 2

Non compilato

Punteggio max.:

6,00

Contrassegna  
domandaSupponendo che la funzione  $\text{Foo}(n)$  richieda tempo  $O(\log n)$ , si consideri il seguente algoritmo, di cui  $T(n)$  è la funzione tempo:

```
Gremlins(n)
\\ pre: n intero > 1
if (n = 2) then return 1
else
    m := Gremlins(n - 1) + Foo(n)
    return m
```

Scegli una o più alternative:

- a.  $T(n) \in O(\log n)$
- b. se  $0 \leq k \leq n$  allora  $T(n) = T(n - k) + \sum_{i=0}^{k-1} \log(n - i)$
- c. se  $0 \leq k \leq n$  allora  $T(n) = T(n - k) + \log(n - k)$
- d.  $T(n) \in O(n \log n)$

Verifica risposta

$$\begin{aligned}
 T(n) &= T(n-1) + \log n \\
 &\stackrel{k \leq n}{=} T(n-2) + \log(n-1) + \log n \\
 &= T(0) + \sum_{i=0}^{n-1} \log(n-i) = O(n \log n)
 \end{aligned}$$

Domanda 3

Risposta non ancora data

Punteggio max.:

8,00



Contrassegna domanda

Si consideri il seguente algoritmo:

```
Casper(A[0..n-1])
\\ pre: A array di interi
for i := 0 to n - 2
    for j := i + 1 to n - 1
        if A[i] = A[j] then
            return false
return true
```

Si risponda succintamente alle seguenti domande:

1. Quando Casper(A) ritorna true?
2. Qual è la sua complessità in termini di  $\Theta$ ?
3. Sapreste indicare un algoritmo che risolva lo stesso problema in tempo asintoticamente migliore?

- Sei now now due esercizi oggi
- $\Theta(n^2)$

Casper ( A[0..n-1] )  
Pre: A array numeri

i = 0

j = i + 1.

WHILE(= i ≤ n - 2  
if A[i] = A[j] then  
 return false

i + 1;

j = i + 1;

return true

Domanda 4

Non completato

Punteggio max.:  
4,00



Contrassegna  
domanda

Sia  $T[0..m - 1]$  una tabella hash con capacità  $m = 10$  e chiavi intere. Qual è lo stato di  $T$  dopo aver inserito le chiavi 88, 12, 2, 22, 33, se si utilizza il metodo di indirizzamento aperto e la funzione hash  $h(k) = k \bmod 10$ ?

Scegli un'alternativa:

- a.  $\{nil, nil, 12, 33, 2, 22, nil, nil, 88, nil\}$
- b.  $\{nil, nil, 33, 12, 2, 22, nil, nil, 88, nil\}$
- c.  $\{nil, nil, 12, 2, 22, 33, nil, nil, 88, nil\}$
- d.  $\{nil, nil, 12, 22, 2, 33, nil, nil, 88, nil\}$

Verifica risposta

$$h(88) = 88 \bmod 10 = 8$$

$$h(12) = 12 \bmod 10 = 2$$

$$h(2) = 2 \bmod 10 = 2 \rightarrow \text{Ocuperà la posizione successiva}$$

$$h(22) = 22 \bmod 10 = 2$$

$$h(33) = 33 \bmod 10 = 3$$

Domanda 5

Non completato

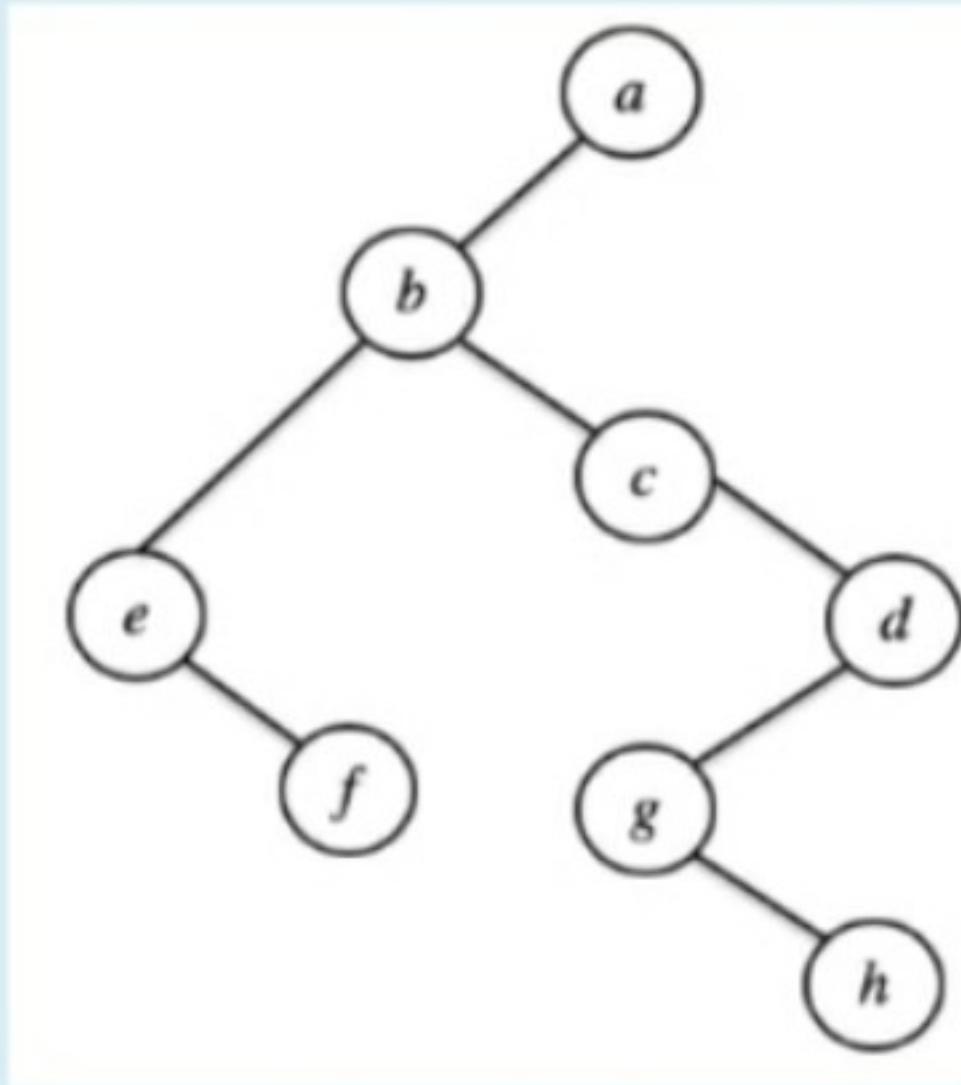
Punteggio mass.

6,00



Contrassegna  
domanda

Si consideri la seguente rappresentazione di un albero  $k$ -ario con un albero binario, corrispondente alla sua memorizzazione mediante puntatori child/sibling:



Quali delle seguenti affermazioni è corretta?

Scegli una o più alternative:

- a. l'albero ha grado 3 e l'insieme delle sue foglie è  $\{c, e, f, g, h\}$ ;
- b. l'albero ha grado 3 e i nodi  $\{a, b, c, d\}$  sono sullo stesso ramo;
- c. L'albero ha altezza 5 e l'insieme delle sue foglie è  $\{f, h\}$ .
- d. l'albero ha altezza 2 e i nodi  $\{a, d, h\}$  formano un ramo;
- e. l'albero ha grado 2 e i nodi  $\{a, b, f\}$  formano un ramo;

Verifica risposta

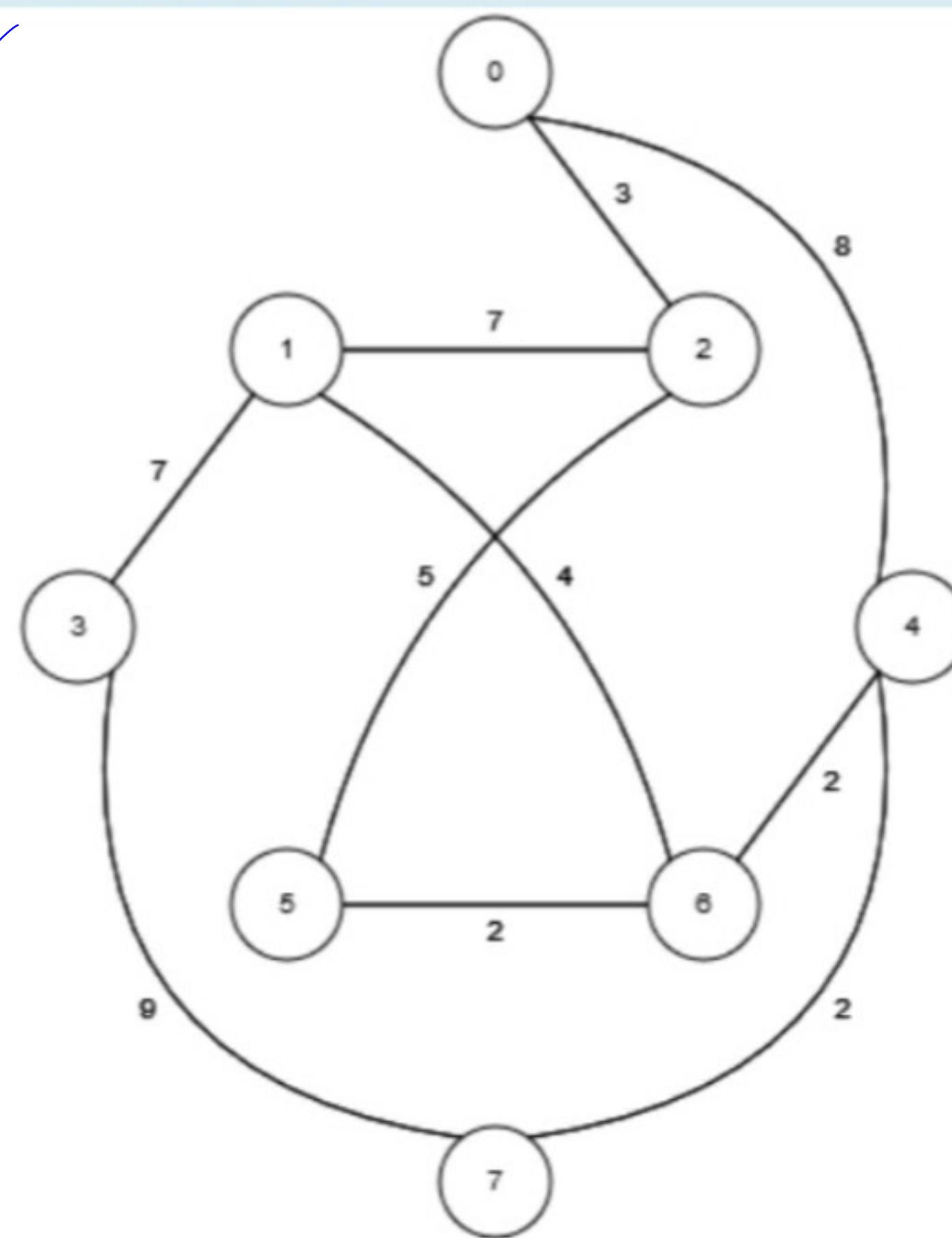
Domanda 6

Risposta non ancora data

Punteggio max.: 8,00



Si consideri il seguente grafo:



Si simuli sul grafo l'algoritmo di Prim a partire dal nodo 0. Si riporti il risultato elencando gli archi inclusi nella soluzione.

Home ► Anno Accademico 2019/2020 ► Corsi di Laurea in Informatica ► ASD-2020 ► Appello 13 Luglio 2020 ► Esame scritto 13 Luglio 2020



Olivia Samy Sobhi Hanna

**Iniziato** lunedì, 13 luglio 2020, 10:01

**Stato** Completato

**Terminato** lunedì, 13 luglio 2020, 11:26

**Tempo impiegato** 1 ora 25 min.

**Valutazione** 17,0 su un massimo di 36,0 (47%)

**Domanda 1**

Parzialmente

corretta

Punteggio

ottenuto 3,0 su  
4,0

Si valuti con si/no le seguenti affermazioni:

 $800n + n^2 + n \log n \in O(n \log n)$   si X $\frac{n^2+6n+9}{n+1} \in \Theta(n^2)$   no ✓ $\frac{n^2+6n+9}{n+1} \in \Theta(n)$   si ✓ $3^{n-2} + 2^{n+5} \in \Omega(3^n)$   si ✓La risposta corretta è:  $800n + n^2 + n \log n \in O(n \log n)$ - no,  $\frac{n^2+6n+9}{n+1} \in \Theta(n^2)$ - no,  $\frac{n^2+6n+9}{n+1} \in \Theta(n)$ - si,  $3^{n-2} + 2^{n+5} \in \Omega(3^n)$ 

- si

**Storico delle risposte**

Passo	Data/Ora	Azione	Stato	Punteggio
1	13/07/2020 10:01	Iniziato	Risposta non ancora data	
2	13/07/2020 10:04	Salvato: $\backslash(800n+n^2+n\log n$ $\backslashin O(n \log n)\backslash) ->$ si; $\backslash(\backslashfrac$ $\{n^2+6n+9\}{n+1} \backslashin$ $\backslashTheta (n^2)\backslash) ->$ no; $\backslash(\backslashfrac$ $\{n^2+6n+9\}{n+1} \backslashin$ $\backslashTheta (n)\backslash) ->$ si; $\backslash(3^{n-2}+2^{n+5}) \backslashin$ $\backslashOmega (3^n)\backslash) ->$ si	Risposta salvata	
3	13/07/2020 11:26	Tentativo terminato	Parzialmente corretta	3,0

**Domanda 2**

Parzialmente

corretta

Punteggio

ottenuto 2,0 su  
6,0

Si consideri l'algoritmo:

```

ALG( $n$ )
  if  $n \leq 50$  then return 1
  else
    if  $n \leq 100$  then
      return  $4 \cdot \text{ALG}(\lfloor n/4 \rfloor) + 7$ 
    else
       $m \leftarrow 0$ 
      for  $i \leftarrow 1$  to 3 do
         $m \leftarrow m + \text{ALG}(\lfloor n/3 \rfloor)$ 
  return  $m$ 

```

La sua funzione tempo in termini di  $n$  soddisfa la relazione di ricorrenza:

- $T(n) = T(n/4) + 1$
- $T(n) = 4T(n/4) + 1$
- $T(n) = 3T(n/3) + 1$  ✓

Punteggio ottenuto 1,0 su 1,0

La risposta corretta è:  $T(n) = 3T(n/3) + 1$ che dopo  $k$  svolgimenti ha la forma

- a:

$$3^k T(n/3^k) + \sum_{i=0}^{k-1} 3^i$$

- b:

$$4^k T(n/4^k) + \sum_{i=0}^{k-1} 4^i$$

- c:

$$T(n/4^k) + k$$

- a
- b ✗
- c

Punteggio ottenuto 0,0 su 1,0

La risposta corretta è: a

da cui risulta che  $T(n)$  ha complessità

- $\Theta(n)$
- $\Theta(n \log_3 n)$
- $\Theta(\log_4 n)$  ✗

Punteggio ottenuto 0,0 su 1,0

La risposta corretta è:  $\Theta(n)$

### Storico delle risposte

Passo	Data/Ora	Azione	Stato	Punteggio
1	13/07/2020 10:01	Iniziato	Risposta non ancora data	
2	13/07/2020 10:10	Salvato: parte 1: $\backslash(T(n) = 3 T(n/3) + 1\backslash);$ parte 2: b; parte 3: $\backslash(\Theta (\log_4 n)\backslash)$	Risposta salvata	
3	13/07/2020 11:26	Tentativo terminato	Parzialmente corretta	2,0

**Domanda 3**

Completo

Punteggio

ottenuto 0,0 su

8,0

Si consideri il seguente algoritmo.

```
ALGO( $A[0..n - 1]$ )
Pre:  $A$  array di interi
 $a \leftarrow 1$ 
for  $i \leftarrow 0$  to  $n - 2$  do
     $j \leftarrow i + 1$ 
    while  $j < n$  and  $A[j - 1] \geq A[j]$  do
         $j \leftarrow j + 1$ 
     $a \leftarrow \max(a, j - i)$ 
return  $a$ 
```

Si risponda alle seguenti domande:

1. Cosa calcola  $\text{Algo}(A)$ ?
2. Quali sono il caso peggiore e la sua complessità in termini di  $\Theta$ ?
3. Sapreste indicare un algoritmo che risolva lo stesso problema in tempo asintoticamente migliore?

1- l'algoritmo calcola il massimo degli elementi ( $a$ ,  $j-i$ ) in assoluto.  
2- il caso peggiore si verifica quando  $\Theta(n^2)$   
3- no perché calcola il problema nel minor tempo possibile

**Commento:**

completamente travisato

**Storico delle risposte**

Passo	Data/Ora	Azione	Stato	Punteggio
1	13/07/2020 10:01	Iniziato	Risposta non ancora data	

Passo	Data/Ora	Azione	Stato	Punteggio
2	13/07/2020 10:35	Salvato: 1- lalgoritmo calcola il massimo degli elementi (a, j-i) in valore assoluto. 2- il caso peggiore si verifica quando $o(n^2)$ 3- no perché calcola il problema nel minor tempo possibile	Risposta salvata	
3	13/07/2020 11:26	Tentativo terminato	Completo	
4	16/07/2020 11:55	Valutato manualmente 0 con il commento: <b>completamente travisato</b>	Completo	0,0

**Domanda 4**

Completo

Punteggio  
ottenuto 4,0 su  
4,0

- Riportare le caratteristiche di un *heap minimo*.
- Il seguente array rappresenta un *heap minimo*.

(1, 2, 3, 10, 8, 9, 7, 14, 15, 16, 13)

Effettuare l'estrazione del minimo riportando l'array che rappresenta lo heap dopo ogni scambio di elementi.

Uno heap minimo è un albero binario semi-completo sinistro, cioè è completo sino al penultimo livello mentre ciascun vertice dell'ultimo non ha vuoti alla sua sinistra; inoltre la chiave di ogni vertice padre è  $\geq$  di quella dei suoi figli nel caso ci fossero. Nella rappresentazione in forma di array H l'eventuale figlio sinistro di  $H[i]$  è  $H[2i]$  ed il suo eventuale figlio destro è  $H[2i + 1]$ .

quindi:

(13,2,3,10,8,9,7,14,15,16)

(2,13,3,10,8,9,7,14,15,16)

(2,8,3,10,13,9,7,14,15,16)

Nella prima riga il 13 e il 2

Nella seconda riga il 13 e l'8

questi elementi sopraindicati sono coinvolti nello scambio che produce la riga successiva.

Commento:

**Storico delle risposte**

Passo	Data/Ora	Azione	Stato	Punteggio
1	13/07/2020 10:01	Iniziato	Risposta non ancora data	

Passo	Data/Ora	Azione	Stato	Punteggio
2	13/07/2020 10:45	Salvato: Uno heap minimo è un albero binario semi-completo sinistro, cioè è completo sino al penultimo livello mentre ciascun vertice dell'ultimo non ha vuoti alla sua sinistra; inoltre la chiave di ogni vertice padre è $>=$ di quella dei suoi figli nel caso ci fossero. Nella rappresentazione in forma di array H l'eventuale figlio sinistro di $H[i]$ è $H[2i]$ ed il suo eventuale figlio destro è $H[2i + 1]$ . quindi: (13,2,3,10,8,9,7,14,15,16) (2,13,3,10,8,9,7,14,15,16) (2,8,3,10,13,9,7,14,15,16) Nella prima riga il 13 e il 2 Nella seconda riga il 13 e l8 questi elementi sopraindicati sono coinvolti nello scambio che produce la riga successiva.	Risposta salvata	
3	13/07/2020 11:26	Tentativo terminato	Completo	
4	13/07/2020 18:22	Valutato manualmente 4 con il commento:	Completo	4,0

**Domanda 5**

Completo

Punteggio

ottenuto 0,0 su

6,0

Sia dato un albero  $T$  (non vuoto), rappresentato con puntatori *child* e *sibling*, nel quale ogni nodo è etichettato con un numero intero. Si dia un algoritmo che restituisca *true* se l'etichetta di tutte le foglie è dispari, e *false* altrimenti.

```
FatherChildOdd(T )
if T.child = nil then return true
else
    for i ← 0 to n – 1 do
        if (V[i] mod2)=1then
            p←p +1
        else T ← T.child
    return T.key = s and b
```

**Commento:**

`if T.child = nil then return true` non controlla la parità dell'etichetta della foglia  $T$

`else`

`for i ← 0 to n – 1 do ...`

non ha senso; oltretutto cosa è  $n$ ?

**Storico delle risposte**

Passo	Data/Ora	Azione	Stato	Punteggio
1	13/07/2020 10:01	Iniziato	Risposta non ancora data	
2	13/07/2020 10:57	Salvato: FatherChildOdd( $T$ ) $)$ if $T.child = \text{nil}$ then $\quad \text{return true}$ else    for $i$ $\leftarrow 0$ to $n - 1$ do        if $\quad (V[i] \text{ mod} 2) = 1$ then $\quad \quad p \leftarrow p + 1$ else $T$ $\leftarrow T.child$ $\text{return } T.key = s \text{ and } b$	Risposta salvata	
3	13/07/2020 11:26	Tentativo terminato	Completo	

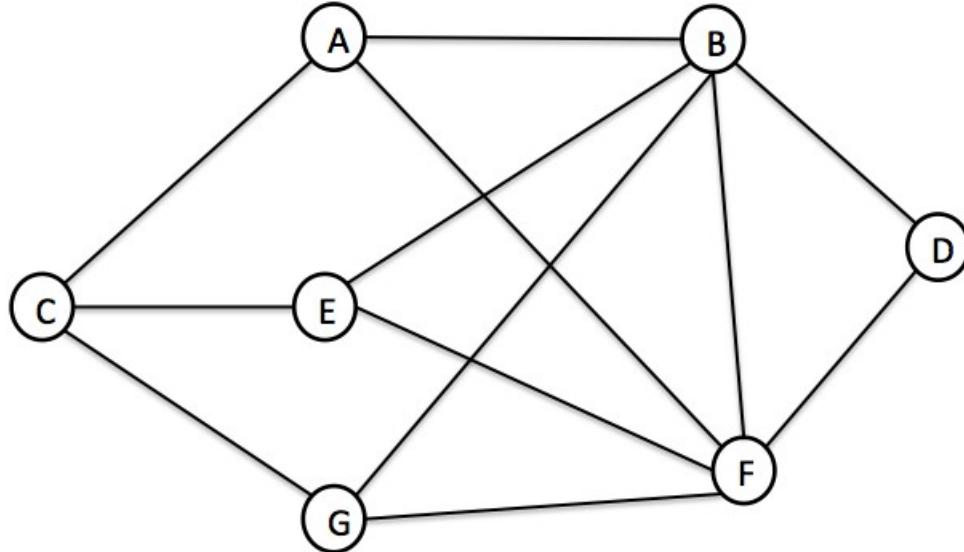
Passo	Data/Ora	Azione	Stato	Punteggio
4	16/07/2020 20:02	Valutato manualmente 0 con il commento: if T.child = nil then return true non controlla la parità dell'etichetta della foglia T else for i ← 0 to n – 1 do ... non ha senso; oltretutto cosa è n?	Completo	0,0

~~Domanda 6~~

Completo

Punteggio  
ottenuto 8,0 su  
8,0

Un grafo  $G = (V, E)$  si dice *bipartito* se esiste una partizione di  $V$  in due sottoinsiemi  $V_1$  e  $V_2$  tale che nessun arco  $(u, v) \in E$  abbia i vertici  $u, v$  contenuti nella stessa parte, ossia per ogni  $(u, v) \in E$  o  $u \in V_1$  e  $v \in V_2$ , oppure  $u \in V_2$  e  $v \in V_1$ . Si decida se il seguente grafo è bipartito oppure no, fornendo una partizione  $V_1, V_2$  di  $V$  nel caso affermativo; si spieghi perché nessuna partizione di  $V$  soddisfa la proprietà succitata nel caso negativo.



il grafo non è bipartito perchè contiene cicli di lunghezza dispari, come ad esempio BEF. Un grafo è bipartito se ogni suo sottografo lo è ; d'altra parte il sottografo BEF non è bicolorabile o non è bipartito, poichè almeno due vertici adiacenti devono avere lo stesso colore.

Commento:

### Storico delle risposte

Passo	Data/Ora	Azione	Stato	Punteggio
1	13/07/2020 10:01	Iniziato	Risposta non ancora data	

Passo	Data/Ora	Azione	Stato	Punteggio
2	13/07/2020 11:26	Salvato: il grafo non è bipartito perchè contiene cicli di lunghezza dispari, come ad esempio BEF. Un grafo è bipartito se ogni suo grafo lo è ; daltra parte il sottografo BEF non è bicolorabile o non è bipartito, poichè almeno due vertici adiacenti devono avere lo stesso colore.	Risposta salvata	
3	13/07/2020 11:26	Tentativo terminato	Completo	
4	15/07/2020 10:55	Valutato manualmente 8 con il commento:	Completo	8,0

**Domanda 1**

Parzialmente corretta

Punteggio ottenuto 2,00 su 4,00



Contrassegna domanda

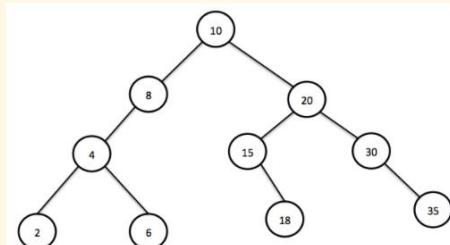
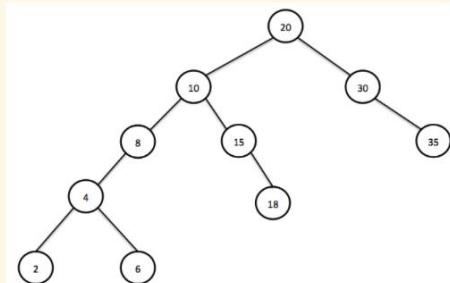
Si considerino le funzioni  $f(n) = \sqrt[3]{n^2}$  e  $g(n) = \log_2 n^3$ .

Scegli una o più alternative:

- a.  $f(n) \in \Omega(g(n))$
- b.  $f(n) \in \Theta(g(n))$
- c.  $g(n) \in o(f(n))$
- d.  $g(n) \in \Theta(f(n))$

La risposta corretta è:  $f(n) \in \Omega(g(n))$ ,  $g(n) \in o(f(n))$

La risposta corretta è:

**Domanda 2**

Parzialmente corretta

Punteggio ottenuto 3,00 su 6,00



Contrassegna domanda

Si considerino le funzioni  $T_0$  e  $T_1$  definite attraverso le seguenti relazioni di ricorrenza:

$$T_0(n) = T_0(n/2) + 5n, \quad T_1(n) = 2T_1(n/2) + \frac{1}{2}.$$

Scegli una o più alternative:

- a.  $T_0(n) \in O(T_1(n))$  e  $T_0(n) \in \Omega(T_1(n))$ .
- b.  $T_0(n) \in O(n)$ , ma  $T_1(n) \in \Omega(2^n)$ .  
✗  $T_0(n) \in \Theta(T_1(n)) = \Theta(n)$ , dunque è vero che  $T_0(n) \in O(n)$ , ma  $T_1(n) \notin \Omega(2^n)$
- c.  $T_0(n) \in \Theta(T_1(n))$ .  
✓
- d.  $T_0(n) \in \Omega(n \log n)$ , mentre  $T_1(n) \in O(\log n)$ .

La risposta corretta è:  $T_0(n) \in \Theta(T_1(n))$ ,  $T_0(n) \in O(T_1(n))$  e  $T_0(n) \in \Omega(T_1(n))$ .

<b>Domanda 1</b>
Parzialmente corretta
Sintegrazione oraria 2,00 su 4,00
Contrassegna domanda

Si stabilisca a quali classi di complessità appartengano o meno le funzioni  $2^{n+1}$ ,  $2^{2n}$  e  $4^n$ .

Scegli una o più alternative:

- a.  $2^{n+1} \notin O(2^n)$ ,  $2^{2n} \notin O(2^n)$ ,  $4^n \notin O(2^n)$
- b.  $2^{n+1} \in O(2^n)$ ,  $2^{2n} \notin O(2^n)$ ,  $4^n \notin O(2^n)$
- c.  $2^{n+1} \in O(2^n)$ ,  $2^{2n} \notin O(2^n)$ ,  $4^n \in O(2^n)$
- d.  $2^{n+1} \in O(2^n)$ ,  $2^{2n} \in O(4^n)$ ,  $4^n \notin O(2^n)$

La risposta corretta è:  $2^{n+1} \in O(2^n)$ ,  $2^{2n} \notin O(2^n)$ ,  $4^n \notin O(2^n)$ ,  
 $, 2^{n+1} \in O(2^n)$ ,  $2^{2n} \in O(4^n)$ ,  $4^n \notin O(2^n)$

Si considerino le funzioni  $T_0$  e  $T_1$  definite attraverso le seguenti relazioni di ricorrenza:

$$T_0(n) = T_0(n/2) + 5n, \quad T_1(n) = 2T_1(n/2) + \frac{1}{2}.$$

Scegli una o più alternative:

- a.  $T_0(n) \in \Theta(T_1(n))$ .
- b.  $T_0(n) \in O(T_1(n))$  e  $T_0(n) \in \Omega(T_1(n))$ .
- c.  $T_0(n) \in O(n)$ , ma  $T_1(n) \in \Omega(2^n)$ .
- d.  $T_0(n) \in \Omega(n \log n)$ , mentre  $T_1(n) \in O(\log n)$ .  
✗  $T_0(n) \in \Theta(T_1(n)) = \Theta(n)$ , dunque nessuna delle due affermazioni è vera

La risposta corretta è:  $T_0(n) \in \Theta(T_1(n))$ ,  
 $, T_0(n) \in O(T_1(n))$  e  $T_0(n) \in \Omega(T_1(n))$ .

Si consideri il seguente algoritmo:

```
Casper(A[0..n-1])
  \\\ pre: A array di interi
  for i := 0 to n - 2
    for j := i + 1 to n - 1
      if A[i] = A[j] then
        return false
  return true
```

Si risponda succintamente alle seguenti domande:

1. Quando Casper(A) ritorna true?
2. Qual è la sua complessità in termini di  $\Theta$ ?
3. Sapreste indicare un algoritmo che risolva lo stesso problema in tempo asintoticamente migliore?



1. Casper ritorna **true** se per ogni  $i, j \in 0..n - 1$  si abbia che  $A[i] \neq A[j]$ .
2. La complessità di Casper è dello stesso ordine del numero delle ripetizioni dell'**if**, ossia  $\Theta(n^2)$ .
3. Se ordiniamo  $A[0..n - 1]$  con un algoritmo di costo  $\Theta(n \log n)$ , come **MergeSort** o **HeapSort**, ogni elemento ripetuto nell'array originale sarà contiguo ad una delle sue ripetizioni, cosa che può verificarsi con un semplice algoritmo  $\Theta(n)$ . In questo modo si ottiene un algoritmo di costo  $\Theta(n \log n) + \Theta(n) = \Theta(n \log n)$ , asintoticamente migliore del quadratico Casper.

~~X~~ Sia  $T[0..m - 1]$  una tabella hash con capacità  $m = 10$  e chiavi intere. Qual è lo stato di  $T$  dopo aver inserito le chiavi 88, 12, 2, 22, 33, se si utilizza il metodo di indirizzamento aperto e la funzione hash  $h(k) = k \bmod 10$ ?

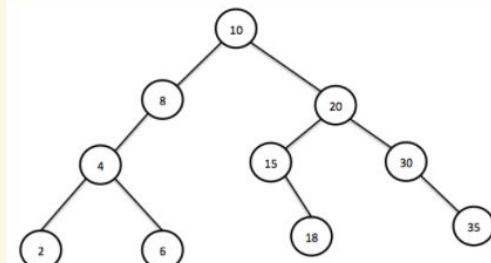
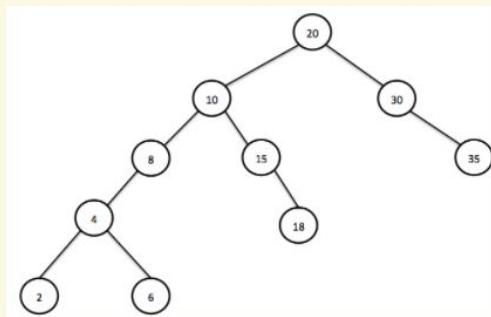
Scegli un'alternativa:

- a. {nil, nil, 12, 22, 2, 33, nil, nil, 88, nil}
- b. {nil, nil, 33, 12, 2, 22, nil, nil, 88, nil}
- c. {nil, nil, 12, 33, 2, 22, nil, nil, 88, nil}
- d. {nil, nil, 12, 2, 22, 33, nil, nil, 88, nil}

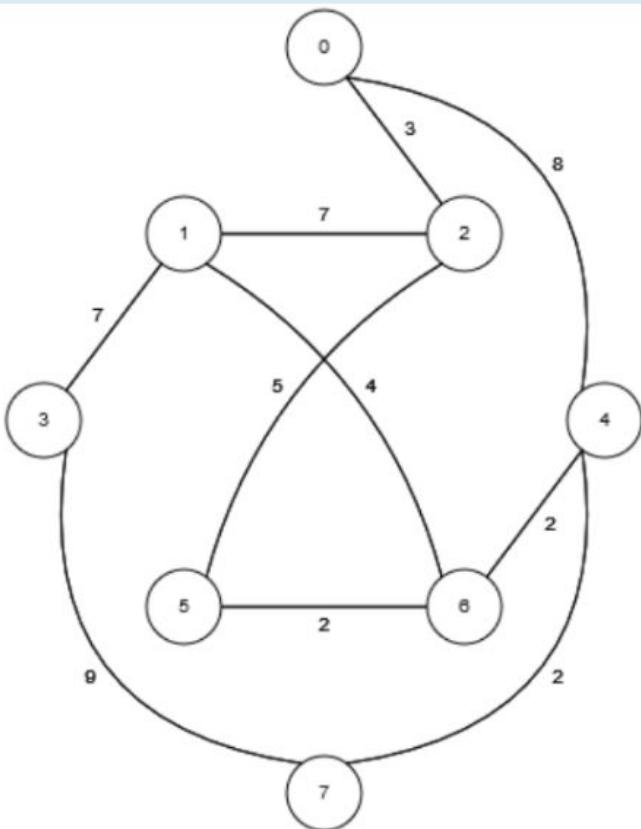
La risposta corretta è: {nil, nil, 12, 2, 22, 33, nil, nil, 88, nil}

Si effettui l'inserimento delle seguenti chiavi in un albero binario di ricerca inizialmente vuoto: 20, 10, 30, 35, 8, 4, 2, 6, 15, 18, 35. Si effettui quindi una rotazione intorno alla radice in modo tale che l'albero diventi più bilanciato. Si indichi quali dei seguenti alberi siano il risultato dell'inserimento e del successivo bilanciamento.

La risposta corretta è:



~~X~~ Si consideri il seguente grafo:



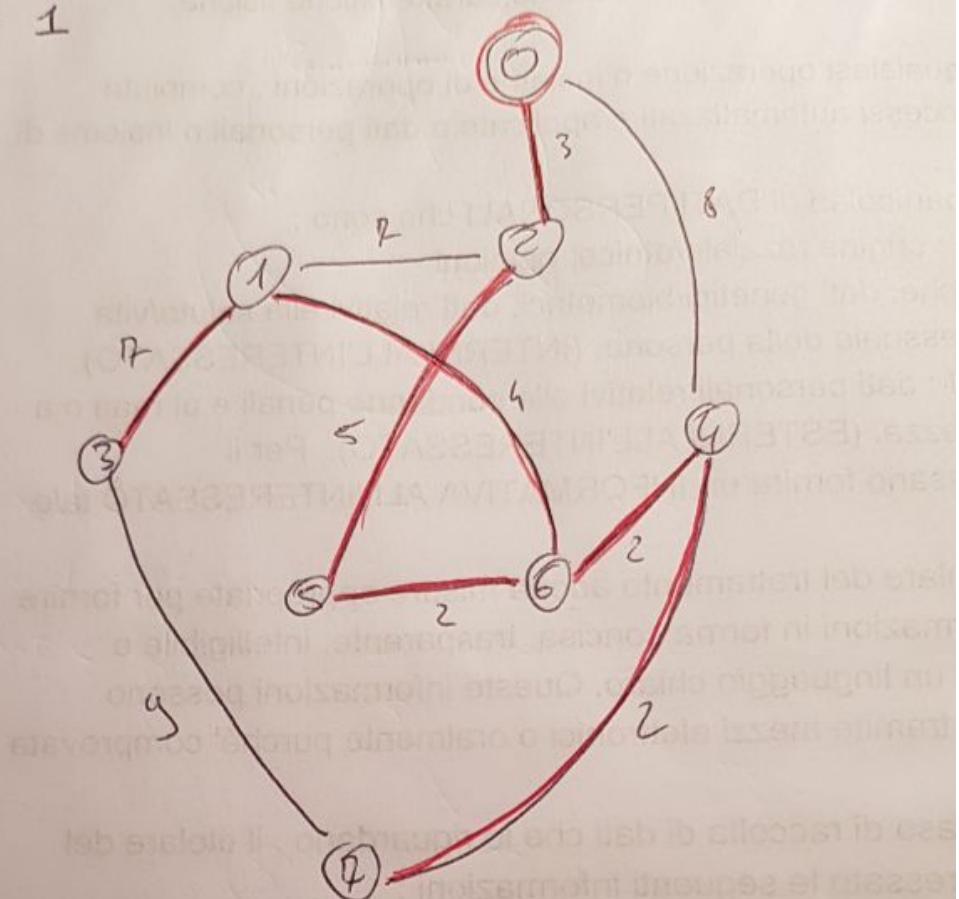
Si simuli sul grafo l'algoritmo di Prim a partire dal nodo 0. Si riporti il risultato elencando gli archi inclusi nella soluzione.



```
0->2 costo 3  
2->5 costo 5  
5->6 costo 2  
6->1 costo 4  
1->3 costo 7  
6->4 costo 2
```

$$0 \xrightarrow{3} 2$$

$$2 \xrightarrow{2} 1$$



**X**

**Domanda 1**  
Risposta errata  
Punteggio ottenuto 0,00 su 4,00  
 Contrassegna domanda

Si considerino le funzioni  $f(n) = \sqrt[3]{n^2}$  e  $g(n) = \log_2 n^3$ .

Scegli una o più alternative:

- a.  $g(n) \in o(f(n))$  
- b.  $f(n) \in \Omega(g(n))$
- c.  $f(n) \in \Theta(g(n))$
- d.  $g(n) \in \Theta(f(n))$  
- e.  $g(n) \notin \Omega(f(n))$ .

La risposta corretta è:  $f(n) \in \Omega(g(n))$ ,  $g(n) \notin \Omega(f(n))$

**X**

**Domanda 2**  
Parzialmente corretta  
Punteggio ottenuto 3,00 su 6,00  
 Contrassegna domanda

Supponendo che la funzione  $\text{Foo}(n)$  richieda tempo  $O(\log n)$ , si consideri il seguente algoritmo, di cui  $T(n)$  è la funzione tempo:

```
Gremlins(n)
  \\\ pre: n intero > 1
  if (n = 2) then return 1
  else
    m := Gremlins(n - 1) + Foo(n)
    return m
```

Scegli una o più alternative:

- a. se  $0 \leq k \leq n$  allora  $T(n) = T(n - k) + \sum_{i=0}^{k-1} \log(n - i)$
- b. se  $0 \leq k \leq n$  allora  $T(n) = T(n - k) + \log(n - k)$  
- c.  $T(n) \in O(\log n)$
- d.  $T(n) \in O(n \log n)$  

La risposta corretta è:  $T(n) \in O(n \log n)$ , se  $0 \leq k \leq n$  allora  $T(n) = T(n - k) + \sum_{i=0}^{k-1} \log(n - i)$

**Domanda 3**  
Risposta non ancora data  
Punteggio max.: 8,00  
 Contrassegna domanda

Si consideri l'algoritmo:

```
Moo(A[0..n-1])
  \\\ pre: A array di interi, \\\(n > 0\)
  d := 0
  for i := 0 to n - 2
    for j := i + 1 to n - 1
      if |A[i] - A[j]| > d then
        d := |A[i] - A[j]|
  return d
```

Si richiede di:

1. spiegare brevemente che cosa calcola  $\text{Moo}(A)$ , senza dire come lo calcoli;
2. stabilire l'ordine di grandezza  $\Theta$  del tempo di questo algoritmo;
3. definire un secondo algoritmo che ritorni lo stesso risultato, ma in tempo asintoticamente inferiore.



1. Calcola la massima differenza tra gli elementi di un array

2.  $\Theta(n^2)$

3.

```
Moo(A[1..n])
  \\\ A array di interi
  max ← min ← A[1]
  for i ← 2 to n do
    if max < A[i] then
      max ← A[i]
    else if min > A[i] then
      min ← A[i]
  return |max - min|
```

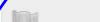
~~Domanda 4~~

Risposta corretta

Punteggio

ottenuto 4,00 su

4,00



Contrassegna domanda

In uno heap minimo  $H$

1. si inseriscono, nell'ordine, le chiavi: 50, 12, 33, 21, 50;
2. si estraie il minimo;
3. si inserisce 40.

Si indichi la sequenza corretta degli stati di  $H$  dopo queste tre fasi.

Scegli un'alternativa:

- a. prima: [12, 21, 33, 50, 50]; seconda: [21, 50, 33, 50]; terza: [21, 40, 50, 33, 50]
- b. prima: [12, 21, 33, 50, 50]; seconda: [21, 50, 33, 50]; terza: [21, 40, 33, 50, 50]
- c. prima: [21, 12, 33, 50, 50]; seconda: [21, 50, 33, 50]; terza: [21, 40, 33, 50, 50]
- d. prima: [12, 21, 33, 50, 50]; seconda: [21, 33, 50, 50]; terza: [21, 40, 33, 50, 50]

La risposta corretta è: prima: [12, 21, 33, 50, 50]; seconda: [21, 50, 33, 50]; terza: [21, 40, 33, 50, 50]

~~Domanda 5~~

Risposta corretta

Punteggio

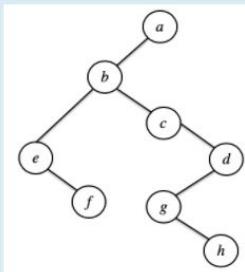
ottenuto 6,00 su

6,00



Contrassegna domanda

Si consideri la seguente rappresentazione di un albero  $k$ -ario con un albero binario, corrispondente alla sua memorizzazione mediante puntatori child/sibling:



Quali delle seguenti affermazioni è corretta?

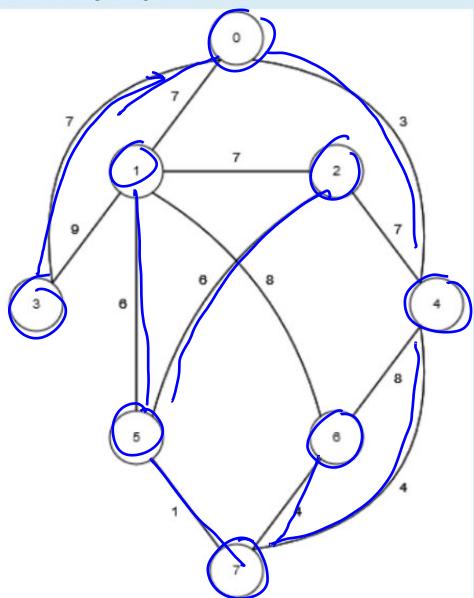
Scegli una o più alternative:

- a. L'albero ha grado 3 e l'insieme delle sue foglie è  $\{c, e, f, g, h\}$ ;
- b. L'albero ha grado 2 e i nodi  $\{a, b, f\}$  formano un ramo;
- c. L'albero ha altezza 5 e l'insieme delle sue foglie è  $\{f, h\}$ .
- d. L'albero ha altezza 2 e i nodi  $\{a, d, h\}$  formano un ramo;
- e. L'albero ha grado 3 e i nodi  $\{a, b, c, d\}$  sono sullo stesso ramo;

La risposta corretta è: l'albero ha grado 3 e l'insieme delle sue foglie è  $\{c, e, f, g, h\}$ ;  
l'albero ha altezza 2 e i nodi  $\{a, d, h\}$  formano un ramo;

**Domanda 6**  
Risposta non ancora data  
Punteggio max.: 8,00  
 contrassegna domanda

Si consideri il seguente grafo:



57  
40  
47  
46  
51  
52  
03

Si simuli sul grafo l'algoritmo di Kruskal. Si riporti il risultato elencando gli archi inclusi nella soluzione.

[ ]

[5-7],[4-0],[7-4],[7-6],[5-2],[5-1][]

0



Domanda 1

Risposta errata

Punteggio  
ottenuto 0,00 su  
4,00Contrassegna  
domanda $\Theta(n^2)$ Sia  $p(n) = 2n^2 + 10n + 4$ .

Scegli una o più alternative:

X a.  $p(n) \notin \Omega(n^3)$  perché  $n^3$  non è un confine inferiore di  $p(n)$  per alcuna costante  $c > 0$  e da alcun  $n_0$  in poi b.  $p(n) \in O(n^3)$  perché  $\lim_{n \rightarrow \infty} \frac{p(n)}{n^3} = 0$ X Proprio perché  $\lim_{n \rightarrow \infty} \frac{p(n)}{n^3} = 0$ ,  $p(n)$  è un infinitesimo di  $n^3$  e quindi deve essere vero che  $p(n) \notin \Omega(n^3)$  c.  $p(n) \notin \Omega(n^3)$  perché  $n^3$  non è un confine inferiore di  $p(n)$  per ogni costante  $c > 0$  ed ogni  $n$  d.  $p(n) \notin \Omega(n^3)$  perché  $\lim_{n \rightarrow \infty} \frac{p(n)}{n^3} = 0$ .La risposta corretta è:  $p(n) \notin \Omega(n^3)$  perché  $n^3$  non è un confine inferiore di  $p(n)$  per alcuna costante  $c > 0$  e da alcun  $n_0$  in poi.  
 $p(n) \notin \Omega(n^3)$  perché  $\lim_{n \rightarrow \infty} \frac{p(n)}{n^3} = 0$ .Si considerino le funzioni  $T_0$  e  $T_1$  definite attraverso le seguenti relazioni di ricorrenza:

$$T_0(n) = T_0(n/2) + 5n, \quad T_1(n) = 2T_1(n/2) + \frac{1}{2}.$$

Scegli una o più alternative:

 a.  $T_0(n) \in \Omega(n \log n)$ , mentre  $T_1(n) \in O(\log n)$ . b.  $T_0(n) \in O(n)$ , ma  $T_1(n) \in \Omega(2^n)$ . c.  $T_0(n) \in O(T_1(n))$  e  $T_0(n) \in \Omega(T_1(n))$ . d.  $T_0(n) \in \Theta(T_1(n))$ .La risposta corretta è:  $T_0(n) \in \Theta(T_1(n))$ .  
 $T_0(n) \in O(T_1(n))$  e  $T_0(n) \in \Omega(T_1(n))$ .

Si consideri l'algoritmo:

```
Moo(A[0..n-1])
\\ pre: A array di interi, \n > 0\
d := 0
for i := 0 to n - 2
    for j := i + 1 to n - 1
        if |A[i] - A[j]| > d then
            d := |A[i] - A[j]|
return d
```

Si richiede di:

1. spiegare brevemente che cosa calcola Moo(A), **senza** dire come lo calcoli;
2. stabilire l'ordine di grandezza  $\Theta$  del tempo di questo algoritmo;
3. definire un secondo algoritmo che ritorni lo stesso risultato, ma in tempo asintoticamente inferiore.



1. L'algoritmo Moo(A) calcola il massimo delle differenze degli elementi di A[1..n] in valore assoluto

2.  $\Theta(n^2)$ 

3.

Moo(A[1..n])

 $\backslash\backslash A$  array di interimax  $\leftarrow$  min  $\leftarrow A[1]$ for i  $\leftarrow$  2 to n doif max  $<$  A[i] then    max  $\leftarrow$  A[i]else if min  $>$  A[i] then    min  $\leftarrow$  A[i]

return |max - min|



Sia  $T[0..m - 1]$  una tabella hash con capacità  $m = 10$  e chiavi intere. Qual è lo stato di  $T$  dopo aver inserito le chiavi 88, 12, 2, 22, 33, se si utilizza il metodo di indirizzamento aperto e la funzione hash  $h(k) = k \bmod 10$ ?

Scegli un'alternativa:

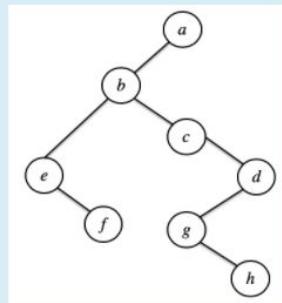
- a.  $\{nil, nil, 12, 33, 2, 22, nil, nil, 88, nil\}$
- b.  $\{nil, nil, 12, 22, 2, 33, nil, nil, 88, nil\}$
- c.  $\{nil, nil, 33, 12, 2, 22, nil, nil, 88, nil\}$
- d.  $\{nil, nil, 12, 2, 22, 33, nil, nil, 88, nil\}$



La risposta corretta è:  $\{nil, nil, 12, 2, 22, 33, nil, nil, 88, nil\}$



Si consideri la seguente rappresentazione di un albero  $k$ -ario con un albero binario, corrispondente alla sua memorizzazione mediante puntatori child/sibling:



Quali delle seguenti affermazioni è corretta?

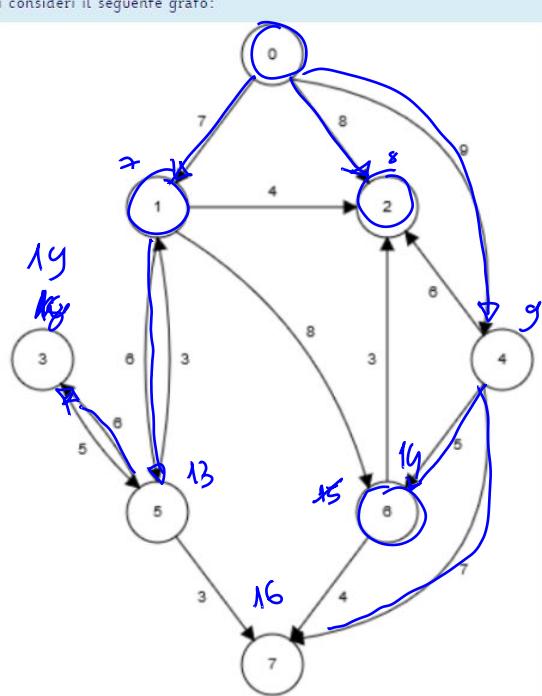
Scegli una o più alternative:

- a. l'albero ha grado 2 e i nodi  $\{a, b, f\}$  formano un ramo;
- b. l'albero ha grado 3 e l'insieme delle sue foglie è  $\{c, e, f, g, h\}$ ;
- c. l'albero ha grado 3 e i nodi  $\{a, b, c, d\}$  sono sullo stesso ramo;
- d. l'albero ha altezza 2 e i nodi  $\{a, d, h\}$  formano un ramo;
- e. L'albero ha altezza 5 e l'insieme delle sue foglie è  $\{f, h\}$ .

La risposta corretta è: l'albero ha grado 3 e l'insieme delle sue foglie è  $\{c, e, f, g, h\}$ ;  
, l'albero ha altezza 2 e i nodi  $\{a, d, h\}$  formano un ramo;

X

Si consideri il seguente grafo:



$$\rightarrow f(0)=0$$

$$01 \rightarrow f(1)=7$$

$$02 \rightarrow f(2)=8$$

$$04 \rightarrow f(4)=9$$

$$15 \rightarrow f(5)=13$$

$$46 \rightarrow f(6)=14$$

$$47 \rightarrow f(7)=16$$

f(0)

$$53 \rightarrow f(3)=19$$

Si simuli sul grafo l'algoritmo di Dijkstra a partire dal nodo 0. Si riporti il risultato fornendo tutti i cammini minimi identificati.

**ESERCIZI ONLINE  
ESAME  
ALGORITMI 2019-20**

### ~~ES1~~

Si valuti con si/no le seguenti affermazioni:

$800n + n^2 + n \log n \in O(n \log n)$   si X

$\frac{n^2+6n+9}{n+1} \in \Theta(n^2)$   no ✓

$\frac{n^2+6n+9}{n+1} \in \Theta(n)$   si ✓

$3^{n-2} + 2^{n+5} \in \Omega(3^n)$   si ✓

La risposta corretta è:  $800n + n^2 + n \log n \in O(n \log n)$

- no,  $\frac{n^2+6n+9}{n+1} \in \Theta(n^2)$
- no,  $\frac{n^2+6n+9}{n+1} \in \Theta(n)$
- si,  $3^{n-2} + 2^{n+5} \in \Omega(3^n)$
- si

1. No, no, si, si

### ~~ES2~~

- Riportare le caratteristiche di un *heap minimo*.
- Il seguente array rappresenta un *heap minimo*.



(1, 2, 3, 10, 8, 9, 7, 14, 15, 16, 13)

Effettuare l'estrazione del minimo riportando l'array che rappresenta lo heap dopo ogni scambio di elementi.

Uno heap minimo è un albero binario semi-completo sinistro, cioè è completo sino al penultimo livello mentre ciascun vertice dell'ultimo non ha vuoti alla sua sinistra; inoltre la chiave di ogni vertice padre è  $\geq$  di quella dei suoi figli nel caso ci fossero. Nella rappresentazione in forma di array H l'eventuale figlio sinistro di  $H[i]$  è  $H[2i]$  ed il suo eventuale figlio destro è  $H[2i + 1]$ .

quindi:

(13, 2, 3, 10, 8, 9, 7, 14, 15, 16)

(2, 13, 3, 10, 8, 9, 7, 14, 15, 16)

(2, 8, 3, 10, 13, 9, 7, 14, 15, 16)

Nella prima riga il 13 e il 2

Nella seconda riga il 13 e l'8

questi elementi sopraindicati sono coinvolti nello scambio che produce la riga successiva.

~~ES3~~

Si consideri l'algoritmo:

```

ALG(n)
  if  $n \leq 50$  then return 1
  else
    if  $n \leq 100$  then
      return  $4 \cdot \text{ALG}(\lfloor n/4 \rfloor) + 7$ 
    else
       $m \leftarrow 0$ 
      for  $i \leftarrow 1$  to 3 do
         $m \leftarrow m + \text{ALG}(\lfloor n/3 \rfloor)$ 
  return  $m$ 

```

La sua funzione tempo in termini di  $n$  soddisfa la relazione di ricorrenza:

- $T(n) = T(n/4) + 1$
- $T(n) = 4T(n/4) + 1$
- $T(n) = 3T(n/3) + 1$



Punteggio ottenuto 1,0 su 1,0

La risposta corretta è:  $T(n) = 3T(n/3) + 1$

che dopo  $k$  svolgimenti ha la forma

- a:

$$3^k T(n/3^k) + \sum_{i=0}^{k-1} 3^i$$



- b:

$$4^k T(n/4^k) + \sum_{i=0}^{k-1} 4^i$$

- c:

$$T(n/4^k) + k$$

- a
- b ~~X~~
- c

Punteggio ottenuto 0,0 su 1,0

La risposta corretta è: a

da cui risulta che  $T(n)$  ha complessità



- $\Theta(n)$
- $\Theta(n \log_3 n)$
- $\Theta(\log_4 n)$  ~~X~~

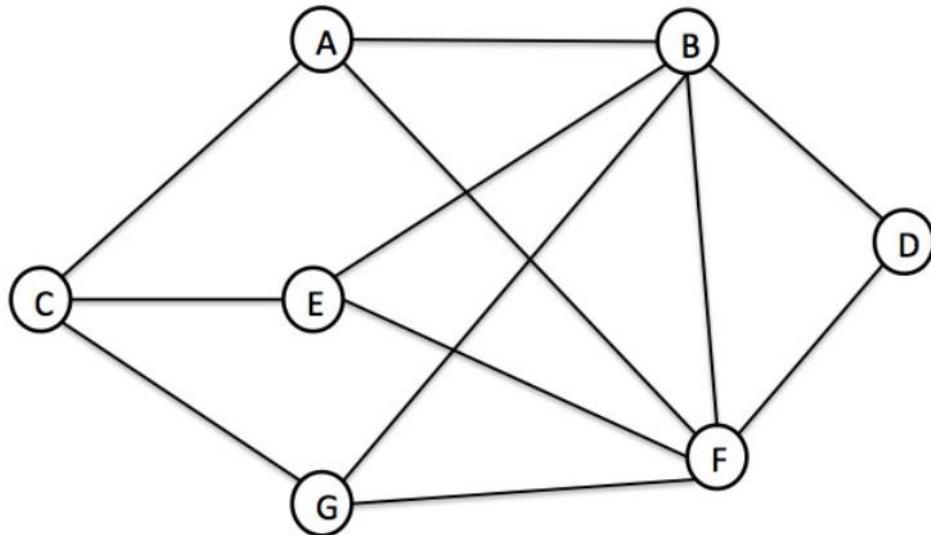
Punteggio ottenuto 0,0 su 1,0

La risposta corretta è:  $\Theta(n)$

1.  $T(n)=3T(n/3)+1$
2. A
3. Theta(n)

~~E84~~

Un grafo  $G = (V, E)$  si dice *bipartito* se esiste una partizione di  $V$  in due sottoinsiemi  $V_1$  e  $V_2$  tale che nessun arco  $(u, v) \in E$  abbia i vertici  $u, v$  contenuti nella stessa parte, ossia per ogni  $(u, v) \in E$  o  $u \in V_1$  e  $v \in V_2$ , oppure  $u \in V_2$  e  $v \in V_1$ . Si decida se il seguente grafo è bipartito oppure no, fornendo una partizione  $V_1, V_2$  di  $V$  nel caso affermativo; si spieghi perché nessuna partizione di  $V$  soddisfa la proprietà succitata nel caso negativo.



il grafo non è bipartito perchè contiene cicli di lunghezza dispari, come ad esempio BEF. Un grafo è bipartito se ogni suo sottografo lo è ; d'altra parte il sottografo BEF non è bicolorabile o non è bipartito, poichè almeno due vertici adiacenti devono avere lo stesso colore.

ES5

Si dica quali dei seguenti asserti equivale a  $g(n) \in \Theta(f(n))$ :

$\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = l > 0 \wedge l < \infty$   Scegli... x definizione

$g(n) \in O(f(n)) \wedge f(n) \in \Omega(g(n))$   Scegli... se assumo che g(n) è un numero esponenziale non è vera perché con teta devo avere lo stesso ordine di grandezza

$\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = 0$   Scegli...  $x^{2/n} \rightarrow \text{infinito} / \text{num} = \text{infinito}$

$g(n) \in O(f(n)) \wedge g(n) \in \Omega(f(n))$   Scegli...

1. vero
2. falso
3. falso
4. vero

ES6

Siano  $f(n), g(n)$  funzioni asintoticamente positive; si dica quali delle seguenti

se  $f(n) \in \Theta(g(n))$  allora  $g(n) \in \Theta(f(n))$   no

se  $f(n) \in O(g(n))$  allora  $2^{f(n)} \in O(2^{g(n)})$   si

$\min\{f(n), g(n)\} \in \Theta(f(n) + g(n))$   Scegli...

$f(n) \in \Theta(f(n)/2)$   si

implicazioni  
 $v v \rightarrow v$   
 $v f \rightarrow f$   
 $f v \rightarrow v$   
 $f f \rightarrow f$

1. Si se  $f(n)$  vale infinito non appartiene a teta implica che  $g(n)$  che non è infinito appartiene a teta
2. Si
3. no
4. no

## ES7

Si considerino le funzioni  $f(n) = n$  e  $g(n) = n^{1+\sin n}$ . Si valuti con si/no le seguenti affermazioni:

- |                            |             |                                     |
|----------------------------|-------------|-------------------------------------|
| $f(n) \notin \Omega(g(n))$ | Scegli... ▾ | <input checked="" type="checkbox"/> |
| $f(n) \notin O(g(n))$      | Scegli... ▾ | <input checked="" type="checkbox"/> |
| $g(n) \in O(f(n))$         | Scegli... ▾ | <input checked="" type="checkbox"/> |
| $f(n) \in \Omega(g(n))$    | Scegli... ▾ | <input checked="" type="checkbox"/> |

1. no
2. no
3. no
4. si

## ~~ES8~~

Si consideri l'algoritmo:

```
ALG(n)
  if  $n \leq 50$  then return 1
  else
    if  $n \leq 100$  then
      return  $4 \cdot \text{ALG}(n - 1) + 7$ 
    else
       $m \leftarrow \text{ALG}(n - 1)$ 
      for  $i \leftarrow 1$  to  $n$  do
        for  $j \leftarrow 1$  to  $n$  do
           $m \leftarrow m + i + j$ 
      return  $m$ 
```

La sua funzione tempo in termini di  $n$  soddisfa la relazione di ricorrenza:

- $T(n) = T(n - 1) + n$
- $T(n) = 2T(n - 1) + n^2$
- $T(n) = T(n - 1) + n^2$  eseguo  $n$  volte  $n-1$ , e ho 2 indici

che dopo  $k$  svolgimenti ha la forma

\* a:

$$2^k T(n - k) + \sum_{i=0}^{k-1} (n - i)^2$$

\* b:

$$T(n - k) + \sum_{i=0}^{k-1} (n - i)$$

\* c:

$$T(n - k) + \sum_{i=0}^{k-1} (n - i)^2$$

a

c

da cui risulta che  $T(n)$  ha complessità

- $\Theta(n^2)$
- $\Theta(2^n)$
- $\Theta(n^3)$

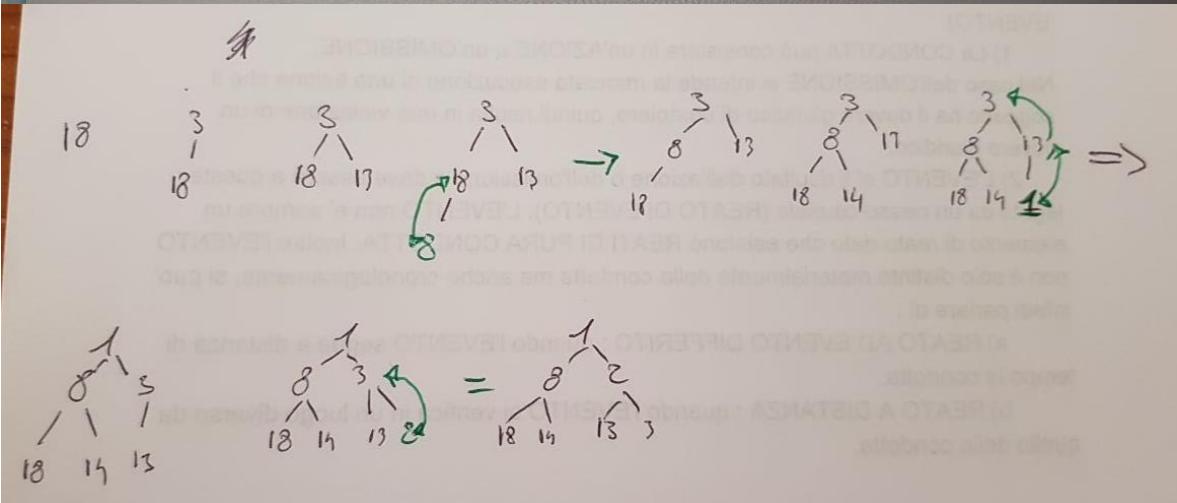
1. C
2. C
3. C

### ES9

Si inseriscano in un heap minimo inizialmente vuoto le seguenti chiavi: 18, 3, 13, 8, 14, 1, 2. Si riporti il vettore che rappresenta lo heap dopo ogni inserimento.

[18] -> [3,18] -> [3, 18, 13] -> [3, 8, 13, 18] -> [3, 8, 13, 18, 14] -> [1, 8, 3, 18, 14, 13] -> [1, 8, 2, 18, 14, 3, 13]

13 3



### ES10

Si inseriscano in un heap minimo inizialmente vuoto le seguenti chiavi: 22, 7, 17, 12, 18, 5, 6. Si riporti il vettore che rappresenta lo heap dopo ogni inserimento.

1.  $[22] \rightarrow [7,22] \rightarrow [7,22,17] \rightarrow [7,12,17,22] \rightarrow [7,12,17,22,18] \rightarrow [5,12,7,22,18,17] \rightarrow [5,12,6,22,18,17,7]$

## ES11

Si consideri il seguente algoritmo.

```

ALG( $A[1..n]$ )
  for  $i \leftarrow n - 1$  down to 1 do
     $j \leftarrow i$ 
    while  $j < n$  and  $A[j] < A[j + 1]$  do
      scambia  $A[j + 1]$  con  $A[j]$ 
       $j \leftarrow j + 1$ 

```

Si risponda succintamente alle seguenti domande:

1. Cosa fa l'algoritmo?
2. Si indichino l'invariante del ciclo esterno e l'invariante del ciclo interno.
3. Quali sono il caso peggiore e la sua complessità in termini di  $\Theta$ ?
4. Sapreste indicare un algoritmo che risolva lo stesso problema in tempo asintoticamente migliore?

1 ordina l'array in ordine decrescente

2. esterno  
 $i < n \wedge i \geq 1$

interno

$\forall A[i] \geq A[i+1]$   
in  $A[j] \dots A[n-2]$

3 caso peggiore: elementi disposti in ordine crescente  $= \Theta(n^2)$

4 merge-sort

## ES12



Supponendo che la funzione  $\text{Foo}(n)$  richieda tempo  $O(\log n)$ , si consideri il seguente algoritmo, di cui  $T(n)$  è la funzione tempo:

```

Gremlins( $n$ )
  \\\ pre: n intero > 1
  if ( $n = 2$ ) then return 1
  else
    m := Gremlins( $n - 1$ ) + Foo( $n$ )
    return m

```

Scegli una o più alternative:

- a.  $T(n) \in O(\log n)$
- b. se  $0 \leq k \leq n$  allora  $T(n) = T(n - k) + \sum_{i=0}^{k-1} \log(n - i)$
- c. se  $0 \leq k \leq n$  allora  $T(n) = T(n - k) + \log(n - k)$
- d.  $T(n) \in O(n \log n)$

Verifica risposta

$$\begin{aligned}
 T(n) &= T(n-1) + \log n \\
 &\stackrel{k \leq n}{=} T(n-2) + \log(n-1) + \log n \\
 &= T(0) + \sum_{i=0}^{n-1} \log(n-i) = O(n \log n)
 \end{aligned}$$

### ~~ES13~~

Si considerino le funzioni hash

$$h_1(k) = k \bmod m, \quad h_2(k) = 1 + (k \bmod m)$$

con cui è definita il doppio hashing:

$$h(k, i) = (h_1(k) + i h_2(k)) \bmod m$$

Lo stato della tabella hash ad indirizzamento aperto  $T[0..12]$ , di dimensione  $m = 13$  dopo l'inserimento delle chiavi: 50, 69, 72, 98, 79, 14 nell'ordine dato, risulta essere (indicando con -1 una locazione libera di  $T$ ):

- [-1, 79, 98, 14, 69, -1, -1, 72, -1, -1, -1, 50, -1]
- [-1, 79, -1, -1, 69, 98, -1, 72, -1, 14, -1, 50, -1]
- [-1, -1, 98, 14, 69, -1, -1, 72, -1, 79, -1, 50, -1]

1. Soluzione è la prima, vedi quaderno per soluzione completa

### ~~ES14~~

Sia  $f(n)$  una funzione non decrescente e positiva per qualunque  $n$ .

$f(n) \in O(g(n))$  implica  $f(n) \notin \Omega(g(n))$ ?  Scegli... ▾

$f(n) \in O(g(n))$  implica  $g(n) \in \Omega(f(n))$ ?  Scegli... ▾

$f(n) \in O(g(n))$  implica  $f(n) \notin \Theta(g(n))$ ?  Scegli... ▾

$f(n) \notin O(g(n))$  implica  $f(n) \notin \Theta(g(n))$ ?  Scegli... ▾

1. Falso
2. Vero
3. Falso
4. Vero

### ES15

Siano  $f(n), g(n)$  funzioni asintoticamente positive; si dica quali delle seguenti asserzioni è vera:

$\max\{f(n), g(n)\} \in \Theta(f(n) + g(n))$   Scegli... ▾

se  $f(n) \in O(g(n))$  allora  $g(n) \in \Omega(f(n))$   Scegli... ▾

$f(n) \in \Theta(f(n)/3)$   Scegli... ▾

se  $f(n) \leq g(n)$  per ogni  $n$  allora  $g(n) - f(n) \in \Omega(g(n))$   Scegli... ▾

1. Vera
2. Vera
3. ~~Falsa~~ *Vera*
4. Falsa

## ES16

Si effettui la visita in profondità, a partire dal nodo b, del seguente grafo di otto nodi dato tramite le liste di adiacenza (i nodi adiacenti devono essere considerati nell'ordine in cui appaiono nella lista di adiacenza):

- b: d, e, f 3
- c: b, h, g 3
- d: f, b 2
- e: d, f 2
- f: b 1
- g: c, f 2
- h: e, f 2
- a: c, g 2

Si riporti per ogni nodo i tempi di inizio e fine visita (ossia i valori degli attributi *d* ed *f*) e per ogni arco il suo tipo secondo la classificazione degli archi durante una visita in profondità.

## ~~ES17~~

Sia dato un albero *T* (non vuoto), rappresentato con puntatori *child* e *sibling*, nel quale ogni nodo è etichettato con un numero intero. Si dia un algoritmo che, per ogni nodo che non è una foglia, stampi la somma delle etichette dei suoi figli. (Una foglia è un nodo che non ha figli.)

## ~~ES18~~

Si effettui la visita in profondità, a partire dal nodo A, del seguente grafo di otto nodi dato tramite le liste di adiacenza (i nodi adiacenti devono essere considerati nell'ordine in cui appaiono nella lista di adiacenza):

- A: C, D, E
- B: A, G, F
- C: E, A
- D: C, E
- E: A
- F: B, E
- G: D, E
- H: B, F

Si riporti per ogni nodo i tempi di inizio e fine visita (ossia i valori degli attributi *d* ed *f*) e per ogni arco il suo tipo secondo la classificazione degli archi durante una visita in profondità.

- A: d = 1, f = 8
- B: d = 9, f = 14
- C: d = 2, f = 5
- D: d = 6, f = 7
- E: d = 3, f = 4
- F: d = 12, f = 13
- G: d = 10, f = 11
- H: d = 15, f = 16

- (A, C) -> Arco foresta
- (C, E) -> Arco foresta
- (E, A) -> Arco indietro
- (C, A) -> Arco indietro
- (A, D) -> Arco foresta
- (D, C) -> Arco attraversamento
- (D, E) -> Arco attraversamento
- (A, E) -> Arco avanti
- (B, A) -> Arco attraversamento
- (B, G) -> Arco foresta
- (G, D) -> Arco attraversamento
- (G, E) -> Arco attraversamento
- (B, F) -> Arco foresta
- (F, B) -> Arco indietro
- (F, E) -> Arco attraversamento
- (H, B) -> Arco attraversamento
- (H, F) -> Arco attraversamento

~~ES19~~

Sia dato un albero  $k$ -ario  $T$  non vuoto, rappresentato con puntatori *child* e *sibling* nel quale ogni nodo è etichettato con un numero intero. Si dia un algoritmo che stampi per ogni livello dell'albero la somma delle etichette dei nodi che ne fanno parte.

```

PRINT-SUM-LIV(T)
If T != nil then
  If T.child = nil then PRINT(T.key)
  While T.child != nil do
    sum <- T.key
    while T.sibling != nil do
      sum <- sum + T.sibling.key
      T <- T.sibling
    PRINT(sum)
    T <- T.child
  
```

~~ES20~~

Si consideri il seguente algoritmo.

**ALGO( $A[0..n - 1]$ )**  
**Pre:**  $A$  array di interi  
 $a \leftarrow 1$   
**for**  $i \leftarrow 0$  **to**  $n - 2$  **do**  
   $j \leftarrow i + 1$   
    **while**  $j < n$  **and**  $A[j - 1] \geq A[j]$  **do**  
       $j \leftarrow j + 1$   
     $a \leftarrow \max(a, j - i)$   
**return**  $a$

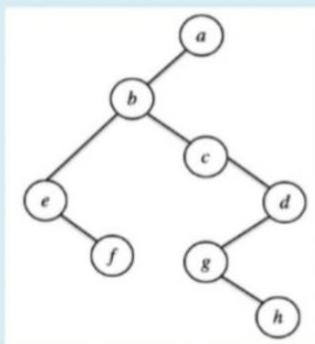
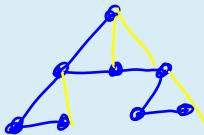
Si risponda alle seguenti domande:

1. Cosa calcola *Algo(A)*?
2. Quali sono il caso peggiore e la sua complessità in termini di  $\Theta$ ?
3. Sapreste indicare un algoritmo che risolva lo stesso problema in tempo asintoticamente migliore?

- 1) Algo( $A$ ) calcola il numero massimo di elementi consecutivi in ordine decrescente presenti nell'array.
- 2)  $T(n) = c_1n + c_2(n-1) + c_3 \sum_{i=1}^{n-1} i + c_4 \sum_{i=1}^{n-1} (i-1) + c_5(n-1) =$   
 $= c_1n + c_2(n-1) + c_3((n-1)/2(n)) + c_4((n-1)/2(n-2)) + c_5(n-1) =$   
 $= (c_1+c_2)n - c_2 + c_3(n^2 - n)/2 + c_4(n^2 - 3n + 2)/2 + c_5(n-1) =$   
 $= ((c_3 + c_4)/2)n^2 + (c_1 + c_2 + c_5 + (-c_3 - (3)c_4)/2)n - c_2 + (2)c_4 - c_5$   
Ha una complessità temporale quadratica
- 3) ALGO'( $A$ )  
 $a \leftarrow 1$   
**for**  $i \leftarrow 0$  **to**  $n-2$  **do**  
   $b \leftarrow 1$   
  **if**  $A[i] \geq A[i+1]$  **then**  
     $b \leftarrow b+1$   
  **else**  
     $a \leftarrow \max(a, b)$   
   $b \leftarrow 1$   
**return**  $a$

~~ES21~~

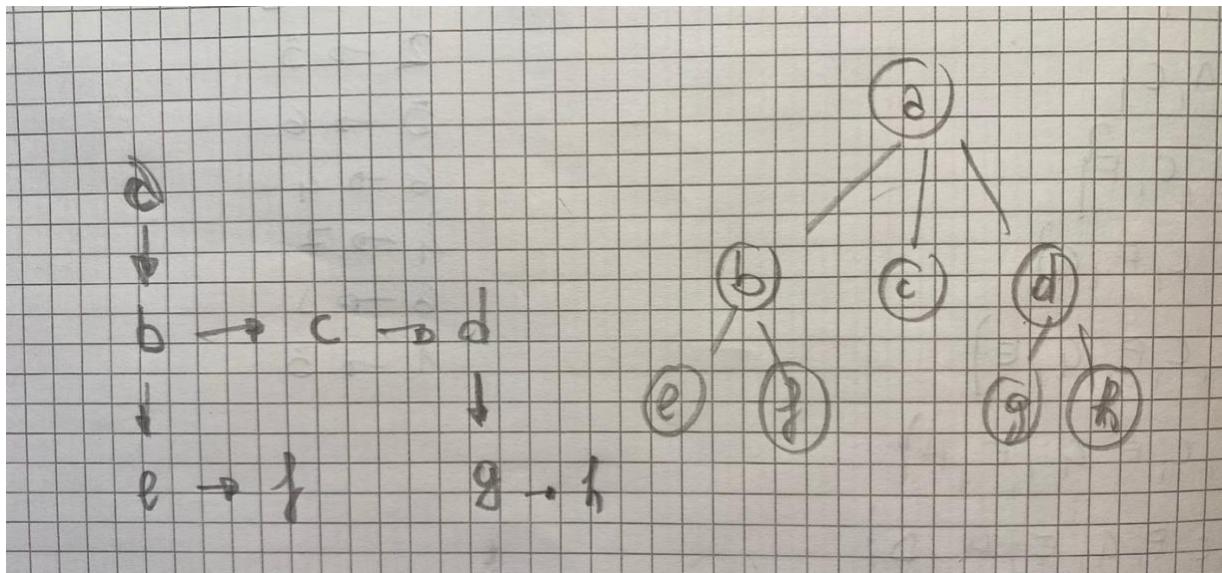
Si consideri la seguente rappresentazione di un albero  $k$ -ario con un albero binario, corrispondente alla sua memorizzazione mediante puntatori child/sibling:



Quali delle seguenti affermazioni è corretta?

Scegli una o più alternative:

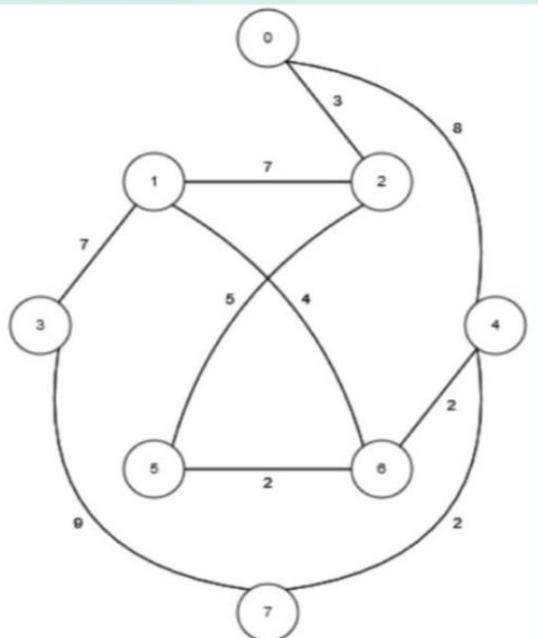
- a. l'albero ha grado 3 e l'insieme delle sue foglie è  $\{c, e, f, g, h\}$ ;
- b. l'albero ha grado 3 e i nodi  $\{a, b, c, d\}$  sono sullo stesso ramo;
- c. L'albero ha altezza 5 e l'insieme delle sue foglie è  $\{f, h\}$ .
- d. l'albero ha altezza 2 e i nodi  $\{a, d, h\}$  formano un ramo;
- e. l'albero ha grado 2 e i nodi  $\{a, b, f\}$  formano un ramo;



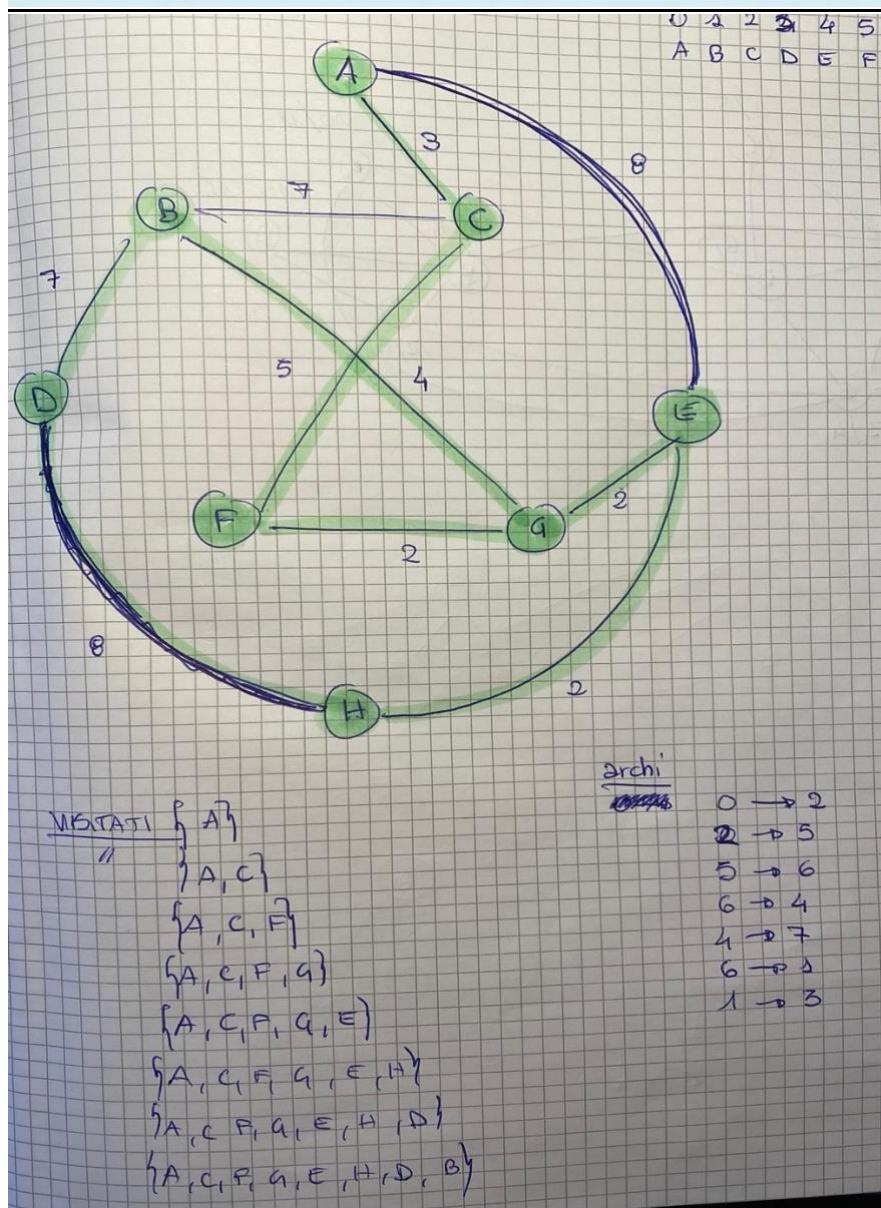
1. A, D

~~ESERCIZIO~~

Si consideri il seguente grafo:



Si simuli sul grafo l'algoritmo di Prim a partire dal nodo 0. Si riporti il risultato elencando gli archi inclusi nella soluzione.



~~ES23~~

Si stabilisca a quali classi di complessità appartengano o meno le funzioni  $2^{n+1}$ ,  $2^{2n}$  e  $4^n$ .

Scegli una o più alternative:

- a.  $2^{n+1} \in O(2^n)$ ,  $2^{2n} \in O(4^n)$ ,  $4^n \notin O(2^n)$
- b.  $2^{n+1} \in O(2^n)$ ,  $2^{2n} \notin O(2^n)$ ,  $4^n \in O(2^n)$
- c.  $2^{n+1} \in O(2^n)$ ,  $2^{2n} \notin O(2^n)$ ,  $4^n \notin O(2^n)$
- d.  $2^{n+1} \notin O(2^n)$ ,  $2^{2n} \notin O(2^n)$ ,  $4^n \notin O(2^n)$

1. C poiché:

- $2^{n+1} = 2^n \times 2$  **appartiene** a  $O(2^n)$
- $2^{2n} = 2^n \times 2^n$  **non appartiene** a  $O(2^n)$
- $4^n = (2 \times 2)^n = 2^n \times 2^n$  **non appartiene** a  $O(2^n)$

La combinazione tra queste tre porta al risultato C

~~ES24~~

Sia  $T[0..m - 1]$  una tabella hash con capacità  $m = 10$  e chiavi intere. Qual è lo stato di  $T$  dopo aver inserito le chiavi 88, 12, 2, 22, 33, se si utilizza il metodo di indirizzamento aperto e la funzione hash  $h(k) = k \bmod 10$ ?

Scegli un'alternativa:

- a. {nil, nil, 12, 33, 2, 22, nil, nil, 88, nil}
- b. {nil, nil, 33, 12, 2, 22, nil, nil, 88, nil}
- c. {nil, nil, 12, 2, 22, 33, nil, nil, 88, nil}
- d. {nil, nil, 12, 22, 2, 33, nil, nil, 88, nil}

1. C

**ESERCIZI ONLINE  
ESAME  
ALGORITMI 2019-20**

~~ES1~~

Si valuti con si/no le seguenti affermazioni:

$$800n + n^2 + n \log n \in O(n \log n)$$

si	✗
----	---

$$\frac{n^2+6n+9}{n+1} \in \Theta(n^2)$$

no	✓
----	---

$$\frac{n^2+6n+9}{n+1} \in \Theta(n)$$

si	✓
----	---

$$3^{n-2} + 2^{n+5} \in \Omega(3^n)$$

si	✓
----	---

La risposta corretta è:  $800n + n^2 + n \log n \in O(n \log n)$

- no,  $\frac{n^2+6n+9}{n+1} \in \Theta(n^2)$
- no,  $\frac{n^2+6n+9}{n+1} \in \Theta(n)$
- si,  $3^{n-2} + 2^{n+5} \in \Omega(3^n)$
- si

1. No, no, si, si

~~ES2~~

- Riportare le caratteristiche di un *heap minimo*.
- Il seguente array rappresenta un *heap minimo*.



(1, 2, 3, 10, 8, 9, 7, 14, 15, 16, 13)

Effettuare l'estrazione del minimo riportando l'array che rappresenta lo heap dopo ogni scambio di elementi.

Uno heap minimo è un albero binario semi-completo sinistro, cioè è completo sino al penultimo livello mentre ciascun vertice dell'ultimo non ha vuoti alla sua sinistra; inoltre la chiave di ogni vertice padre è  $\geq$  di quella dei suoi figli nel caso ci fossero. Nella rappresentazione in forma di array H l'eventuale figlio sinistro di  $H[i]$  è  $H[2i]$  ed il suo eventuale figlio destro è  $H[2i + 1]$ .

quindi:

(13, 2, 3, 10, 8, 9, 7, 14, 15, 16)

(2, 13, 3, 10, 8, 9, 7, 14, 15, 16)

(2, 8, 3, 10, 13, 9, 7, 14, 15, 16)

Nella prima riga il 13 e il 2

Nella seconda riga il 13 e l'8

questi elementi sopraindicati sono coinvolti nello scambio che produce la riga successiva.

~~ESS~~

Si consideri l'algoritmo:

```

ALG( $n$ )
  if  $n \leq 50$  then return 1
  else
    if  $n \leq 100$  then
      return  $4 \cdot \text{ALG}(\lfloor n/4 \rfloor) + 7$ 
    else
       $m \leftarrow 0$ 
      for  $i \leftarrow 1$  to 3 do
         $m \leftarrow m + \text{ALG}(\lfloor n/3 \rfloor)$ 
  return  $m$ 

```

La sua funzione tempo in termini di  $n$  soddisfa la relazione di ricorrenza:

- $T(n) = T(n/4) + 1$
- $T(n) = 4T(n/4) + 1$
- $T(n) = 3T(n/3) + 1$



Punteggio ottenuto 1,0 su 1,0

La risposta corretta è:  $T(n) = 3T(n/3) + 1$

che dopo  $k$  svolgimenti ha la forma

- a:

$$3^k T(n/3^k) + \sum_{i=0}^{k-1} 3^i$$



- b:

$$4^k T(n/4^k) + \sum_{i=0}^{k-1} 4^i$$

- c:

$$T(n/4^k) + k$$

- a
- b ~~X~~
- c

Punteggio ottenuto 0,0 su 1,0

La risposta corretta è: a

da cui risulta che  $T(n)$  ha complessità



- $\Theta(n)$
- $\Theta(n \log_3 n)$
- $\Theta(\log_4 n)$  ~~X~~

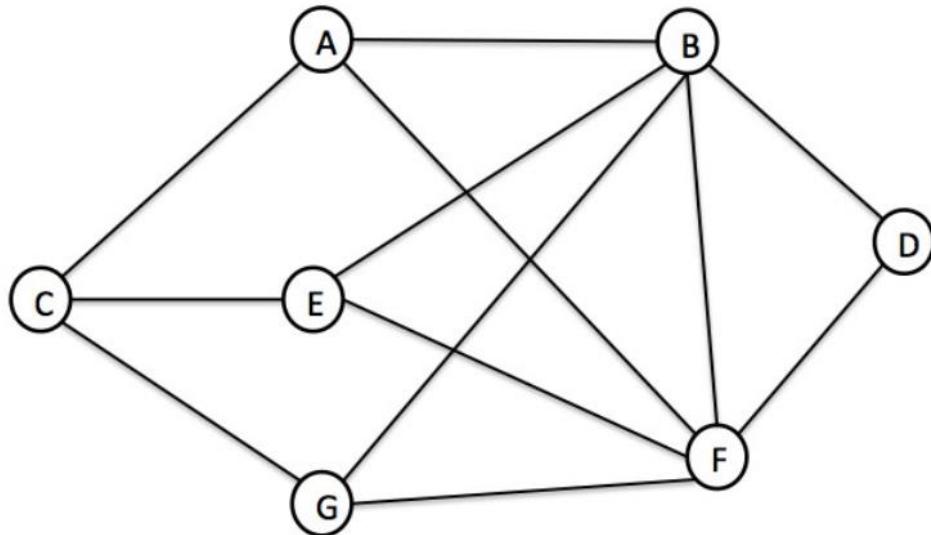
Punteggio ottenuto 0,0 su 1,0

La risposta corretta è:  $\Theta(n)$

1.  $T(n)=3T(n/3)+1$
2. A
3. Theta(n)

~~ESERCIZIO~~

Un grafo  $G = (V, E)$  si dice *bipartito* se esiste una partizione di  $V$  in due sottoinsiemi  $V_1$  e  $V_2$  tale che nessun arco  $(u, v) \in E$  abbia i vertici  $u, v$  contenuti nella stessa parte, ossia per ogni  $(u, v) \in E$  o  $u \in V_1$  e  $v \in V_2$ , oppure  $u \in V_2$  e  $v \in V_1$ . Si decida se il seguente grafo è bipartito oppure no, fornendo una partizione  $V_1, V_2$  di  $V$  nel caso affermativo; si spieghi perché nessuna partizione di  $V$  soddisfa la proprietà succitata nel caso negativo.



il grafo non è bipartito perchè contiene cicli di lunghezza dispari, come ad esempio BEF. Un grafo è bipartito se ogni suo sottografo lo è ; d'altra parte il sottografo BEF non è bicolorabile o non è bipartito, poichè almeno due vertici adiacenti devono avere lo stesso colore.

~~E5~~

Si dica quali dei seguenti asserti equivale a  $g(n) \in \Theta(f(n))$ :

$\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = l > 0 \wedge l < \infty$  Scegli... x definizione

$g(n) \in O(f(n)) \wedge f(n) \in \Omega(g(n))$  Scegli... se assumo che g(n) è un numero esponenziale non è vera perché con teta devo avere lo stesso ordine di grandezza

$\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = 0$  Scegli...  $x^{2/n} \rightarrow \text{infinito} / \text{num} = \text{infinito}$

$g(n) \in O(f(n)) \wedge g(n) \in \Omega(f(n))$  Scegli...

1. vero
2. falso
3. falso
4. vero

~~E6~~

Siano  $f(n), g(n)$  funzioni asintoticamente positive; si dica quali delle seguenti

se  $f(n) \in \Theta(g(n))$  allora  $g(n) \in \Theta(f(n))$  no

se  $f(n) \in O(g(n))$  allora  $2^{f(n)} \in O(2^{g(n)})$  si

$\min\{f(n), g(n)\} \in \Theta(f(n) + g(n))$  Scegli...

$f(n) \in \Theta(f(n)/2)$  si

implicazioni  
 v v -> v  
 v f -> f  
 f v -> v  
 f f -> f

1. Si se  $f(n)$  vale infinito non appartiene a teta implica che  $g(n)$  che non è infinito appartiene a teta
2. Si
3. no
4. no

~~ES7~~

Si considerino le funzioni  $f(n) = n$  e  $g(n) = n^{1+\sin n}$ . Si valuti con si/no le seguenti affermazioni:

$f(n) \notin \Omega(g(n))$

$f(n) \notin O(g(n))$

$g(n) \in O(f(n))$

$f(n) \in \Omega(g(n))$

1. no
2. no
3. no
4. si

~~ES8~~

Si consideri l'algoritmo:

```

ALG(n)
  if n ≤ 50 then return 1
  else
    if n ≤ 100 then
      return 4 · ALG(n - 1) + 7
    else
      m ← ALG(n - 1)
      for i ← 1 to n do
        for j ← 1 to n do
          m ← m + i + j
      return m
  
```

La sua funzione tempo in termini di  $n$  soddisfa la relazione di ricorrenza:

- $T(n) = T(n - 1) + n$
- $T(n) = 2T(n - 1) + n^2$
- $T(n) = T(n - 1) + n^2$  eseguo  $n$  volte  $n-1$ , e ho 2 indici

che dopo  $k$  svolgimenti ha la forma

\* a:

$$2^k T(n - k) + \sum_{i=0}^{k-1} (n - i)^2$$

\* b:

$$T(n - k) + \sum_{i=0}^{k-1} (n - i)$$

\* c:

$$T(n - k) + \sum_{i=0}^{k-1} (n - i)^2$$

a

c

da cui risulta che  $T(n)$  ha complessità

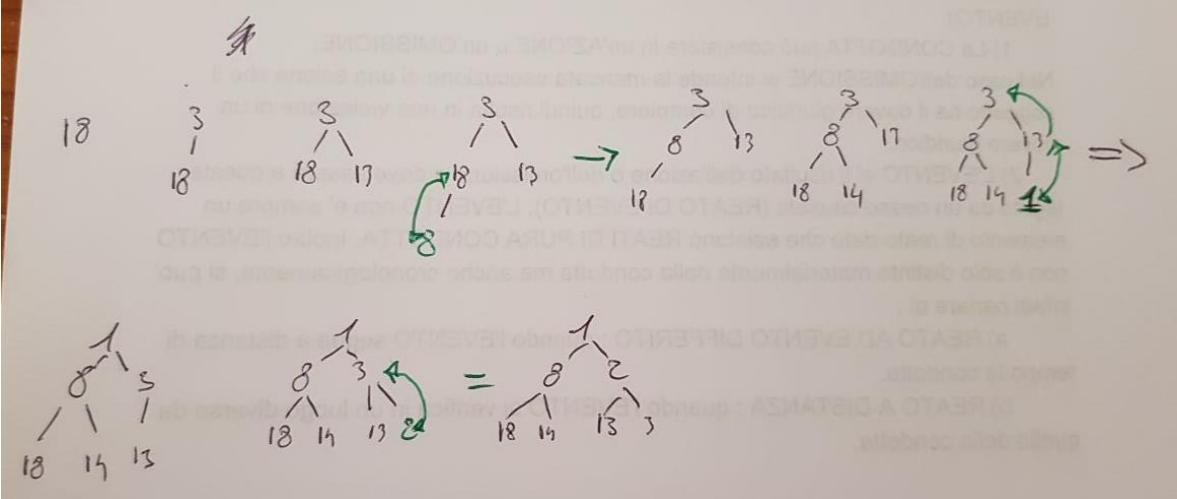
- $\Theta(n^2)$
- $\Theta(2^n)$
- $\Theta(n^3)$

1. C
2. C
3. C

### ~~ES9~~

Si inseriscano in un heap minimo inizialmente vuoto le seguenti chiavi: 18, 3, 13, 8, 14, 1, 2. Si riporti il vettore che rappresenta lo heap dopo ogni inserimento.

[18] -> [3,18] -> [3, 18, 13] -> [3, 8, 13, 18] -> [3, 8, 13, 18, 14] -> [1, 8, 3, 18, 14, 13] -> [1, 8, 2, 18, 14, 3, 13]



### ~~ES10~~

Si inseriscano in un heap minimo inizialmente vuoto le seguenti chiavi: 22, 7, 17, 12, 18, 5, 6. Si riporti il vettore che rappresenta lo heap dopo ogni inserimento.

1.  $[22] \rightarrow [7,22] \rightarrow [7,22,17] \rightarrow [7,12,17,22] \rightarrow [7,12,17,22,18] \rightarrow [5,12,7,22,18,17] \rightarrow [5,12,6,22,18,17,7]$

## ~~E811~~

Si consideri il seguente algoritmo.

```

ALG( $A[1..n]$ )
  for  $i \leftarrow n - 1$  down to 1 do
     $j \leftarrow i$ 
    while  $j < n$  and  $A[j] < A[j + 1]$  do
      scambia  $A[j + 1]$  con  $A[j]$ 
       $j \leftarrow j + 1$ 

```

Si risponda succintamente alle seguenti domande:

1. Cosa fa l'algoritmo?
2. Si indichino l'invariante del ciclo esterno e l'invariante del ciclo interno.
3. Quali sono il caso peggiore e la sua complessità in termini di  $\Theta$ ?
4. Sapreste indicare un algoritmo che risolva lo stesso problema in tempo asintoticamente migliore?

1 ordina l'array in ordine decrescente

2. esterno  
 $i < n \wedge i \geq 1$

interno

$\forall A[i] \geq A[i+1]$   
in  $A[j] \dots A[n-2]$

3 caso peggiore: elementi disposti in ordine crescente  $= \Theta(n^2)$

4 merge-sort

## ~~E812~~

Domanda 4  
non completato  
integro max.: 00  
contrassegna domanda

Supponendo che la funzione  $\text{Foo}(n)$  richieda tempo  $O(\log n)$ , si consideri il seguente algoritmo, di cui  $T(n)$  è la funzione tempo:

```

Gremlins( $n$ )
  \\\ pre: n intero > 1
  if ( $n = 2$ ) then return 1
  else
    m := Gremlins( $n - 1$ ) + Foo( $n$ )
    return m

```

Scegli una o più alternative:

- a.  $T(n) \in O(\log n)$
- b. se  $0 \leq k \leq n$  allora  $T(n) = T(n - k) + \sum_{i=0}^{k-1} \log(n - i)$
- c. se  $0 \leq k \leq n$  allora  $T(n) = T(n - k) + \log(n - k)$
- d.  $T(n) \in O(n \log n)$

Verifica risposta

$$\begin{aligned}
T(n) &= T(n-1) + \log n \\
&\stackrel{k \leq n}{=} T(n-2) + \log(n-1) + \log n \\
&= T(0) + \sum_{i=0}^{n-1} \log(n-i) = O(n \log n)
\end{aligned}$$

~~ES~~3

Si considerino le funzioni hash

$$h_1(k) = k \bmod m, \quad h_2(k) = 1 + (k \bmod m)$$

con cui è definita il doppio hashing:

$$h(k, i) = (h_1(k) + i h_2(k)) \bmod m$$

Lo stato della tabella hash ad indirizzamento aperto  $T[0..12]$ , di dimensione  $m = 13$  dopo l'inserimento delle chiavi: 50, 69, 72, 98, 79, 14 nell'ordine dato, risulta essere (indicando con -1 una locazione libera di  $T$ ):

- [-1, 79, 98, 14, 69, -1, -1, 72, -1, -1, -1, 50, -1]
- [-1, 79, -1, -1, 69, 98, -1, 72, -1, 14, -1, 50, -1]
- [-1, -1, 98, 14, 69, -1, -1, 72, -1, 79, -1, 50, -1]

1. Soluzione è la prima, vedi quaderno per soluzione completa

~~ES~~14

Sia  $f(n)$  una funzione non decrescente e positiva per qualunque  $n$ .

$f(n) \in O(g(n))$  implica  $f(n) \notin \Omega(g(n))$ ? Scegli... ▾

$f(n) \in O(g(n))$  implica  $g(n) \in \Omega(f(n))$ ? Scegli... ▾

$f(n) \in O(g(n))$  implica  $f(n) \notin \Theta(g(n))$ ? Scegli... ▾

$f(n) \notin O(g(n))$  implica  $f(n) \notin \Theta(g(n))$ ? Scegli... ▾

1. Falso
2. Vero
3. Falso
4. Vero

~~ES~~15

Siano  $f(n), g(n)$  funzioni asintoticamente positive; si dica quali delle seguenti asserzioni è vera:

$\max\{f(n), g(n)\} \in \Theta(f(n) + g(n))$  Scegli... ▾

se  $f(n) \in O(g(n))$  allora  $g(n) \in \Omega(f(n))$  Scegli... ▾

$f(n) \in \Theta(f(n)/3)$  Scegli... ▾

se  $f(n) \leq g(n)$  per ogni  $n$  allora  $g(n) - f(n) \in \Omega(g(n))$  Scegli... ▾

1. Vera
2. Vera
3. Falsa
4. Falsa

~~ES16~~

Si effettui la visita in profondità, a partire dal nodo b, del seguente grafo di otto nodi dato tramite le liste di adiacenza (i nodi adiacenti devono essere considerati nell'ordine in cui appaiono nella lista di adiacenza):

b: d, e, f  
c: b, h, g  
d: f, b  
e: d, f  
f: b  
g: c, f  
h: e, f  
a: c, g

Si riporti per ogni nodo i tempi di inizio e fine visita (ossia i valori degli attributi *d* ed *f*) e per ogni arco il suo tipo secondo la classificazione degli archi durante una visita in profondità.

~~ES17~~

Sia dato un albero *T* (non vuoto), rappresentato con puntatori *child* e *sibling*, nel quale ogni nodo è etichettato con un numero intero. Si dia un algoritmo che, per ogni nodo che non è una foglia, stampi la somma delle etichette dei suoi figli. (Una foglia è un nodo che non ha figli.)

~~ES18~~

Si effettui la visita in profondità, a partire dal nodo A, del seguente grafo di otto nodi dato tramite le liste di adiacenza (i nodi adiacenti devono essere considerati nell'ordine in cui appaiono nella lista di adiacenza):

A: C, D, E  
B: A, G, F  
C: E, A  
D: C, E  
E: A  
F: B, E  
G: D, E  
H: B, F

Si riporti per ogni nodo i tempi di inizio e fine visita (ossia i valori degli attributi *d* ed *f*) e per ogni arco il suo tipo secondo la classificazione degli archi durante una visita in profondità.

A: d = 1, f = 8  
B: d = 9, f = 14  
C: d = 2, f = 5  
D: d = 6, f = 7  
E: d = 3, f = 4  
F: d = 12, f = 13  
G: d = 10, f = 11  
H: d = 15, f = 16

(A, C) -> Arco foresta  
(C, E) -> Arco foresta  
(E, A) -> Arco indietro  
(C, A) -> Arco indietro  
(A, D) -> Arco foresta  
(D, C) -> Arco attraversamento  
(D, E) -> Arco attraversamento  
(A, E) -> Arco avanti  
(B, A) -> Arco attraversamento  
(B, G) -> Arco foresta  
(G, D) -> Arco attraversamento  
(G, E) -> Arco attraversamento  
(B, F) -> Arco foresta  
(F, B) -> Arco indietro  
(F, E) -> Arco attraversamento  
(H, B) -> Arco attraversamento  
(H, F) -> Arco attraversamento

~~ES19~~

Sia dato un albero  $k$ -ario  $T$  non vuoto, rappresentato con puntatori *child* e *sibling* nel quale ogni nodo è etichettato con un numero intero. Si dia un algoritmo che stampi per ogni livello dell'albero la somma delle etichette dei nodi che ne fanno parte.

```

PRINT-SUM-LIV(T)
If T != nil then
  If T.child = nil then PRINT(T.key)
  While T.child != nil do
    sum <- T.key
    while T.sibling != nil do
      sum <- sum + T.sibling.key
      T <- T.sibling
    PRINT(sum)
    T <- T.child
  
```

~~ES20~~

Si consideri il seguente algoritmo.

**ALGO( $A[0..n - 1]$ )**  
**Pre:**  $A$  array di interi  
 $a \leftarrow 1$   
**for**  $i \leftarrow 0$  **to**  $n - 2$  **do**  
   $j \leftarrow i + 1$   
    **while**  $j < n$  **and**  $A[j - 1] \geq A[j]$  **do**  
       $j \leftarrow j + 1$   
     $a \leftarrow \max(a, j - i)$   
**return**  $a$

Si risponda alle seguenti domande:

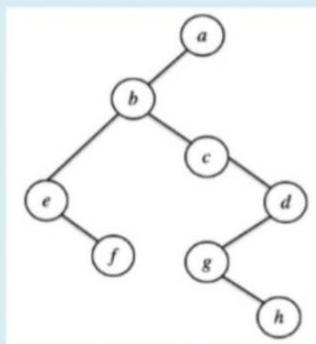
1. Cosa calcola *Algo(A)*?
2. Quali sono il caso peggiore e la sua complessità in termini di  $\Theta$ ?
3. Sapreste indicare un algoritmo che risolva lo stesso problema in tempo asintoticamente migliore?

- 1) *Algo(A)* calcola il numero massimo di elementi consecutivi in ordine decrescente presenti nell'array.
- 2) 
$$\begin{aligned} T(n) &= c_1n + c_2(n-1) + c_3 \sum_{i=1}^{n-1} i + c_4 \sum_{i=1}^{n-1} (i-1) + c_5(n-1) = \\ &= c_1n + c_2(n-1) + c_3((n-1)/2(n)) + c_4((n-1)/2(n-2)) + c_5(n-1) = \\ &= (c_1+c_2)n - c_2 + c_3(n^2 - n)/2 + c_4(n^2 - 3n + 2)/2 + c_5(n-1) = \\ &= ((c_3 + c_4)/2)n^2 + (c_1 + c_2 + c_5 + (-c_3 - (3)c_4)/2)n - c_2 + (2)c_4 - c_5 \end{aligned}$$

Ha una complessità temporale quadratica
- 3) **ALGO'(A)**  
 $a \leftarrow 1$   
**for**  $i \leftarrow 0$  **to**  $n-2$  **do**  
   $b \leftarrow 1$   
  **if**  $A[i] \geq A[i+1]$  **then**  
     $b \leftarrow b+1$   
  **else**  
     $a \leftarrow \max(a, b)$   
   $b \leftarrow 1$   
**return**  $a$

~~EZ1~~

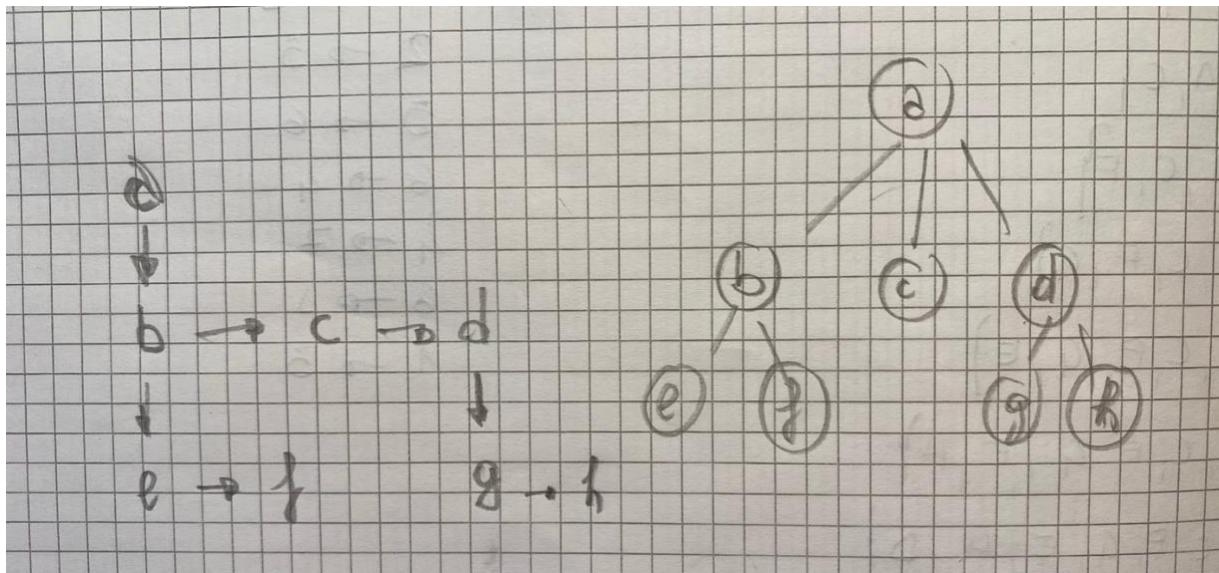
Si consideri la seguente rappresentazione di un albero  $k$ -ario con un albero binario, corrispondente alla sua memorizzazione mediante puntatori child/sibling:



Quali delle seguenti affermazioni è corretta?

Scegli una o più alternative:

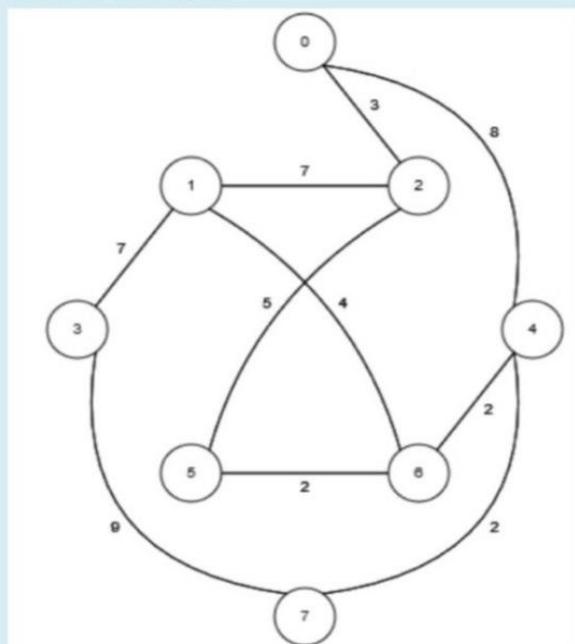
- a. l'albero ha grado 3 e l'insieme delle sue foglie è  $\{c, e, f, g, h\}$ ;
- b. l'albero ha grado 3 e i nodi  $\{a, b, c, d\}$  sono sullo stesso ramo;
- c. L'albero ha altezza 5 e l'insieme delle sue foglie è  $\{f, h\}$ .
- d. l'albero ha altezza 2 e i nodi  $\{a, d, h\}$  formano un ramo;
- e. l'albero ha grado 2 e i nodi  $\{a, b, f\}$  formano un ramo;



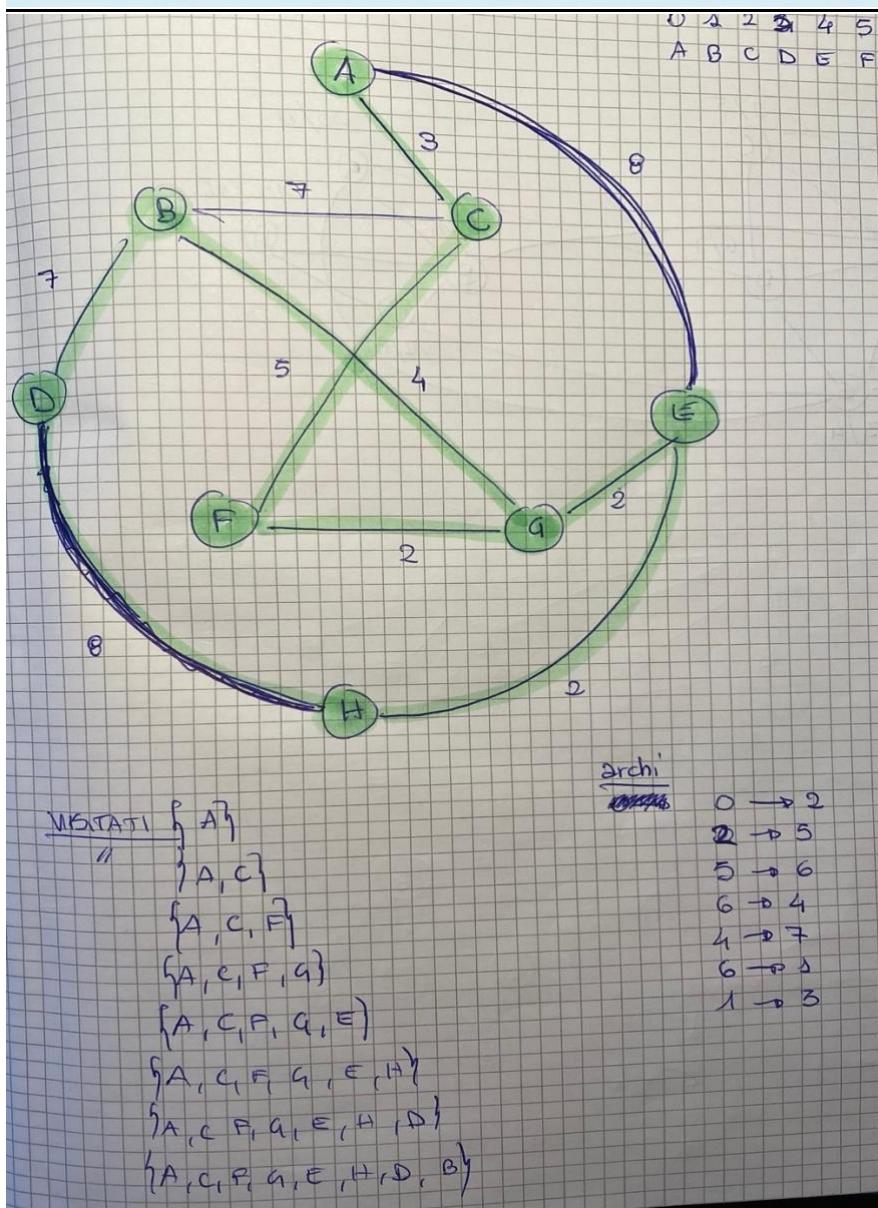
1. A, D

~~E822~~

Si consideri il seguente grafo:



Si simuli sul grafo l'algoritmo di Prim a partire dal nodo 0. Si riporti il risultato elencando gli archi inclusi nella soluzione.



### ES23

Si stabilisca a quali classi di complessità appartengano o meno le funzioni  $2^{n+1}$ ,  $2^{2n}$  e  $4^n$ .

Scegli una o più alternative:

- a.  $2^{n+1} \in O(2^n)$ ,  $2^{2n} \in O(4^n)$ ,  $4^n \notin O(2^n)$
- b.  $2^{n+1} \in O(2^n)$ ,  $2^{2n} \notin O(2^n)$ ,  $4^n \in O(2^n)$
- c.  $2^{n+1} \in O(2^n)$ ,  $2^{2n} \notin O(2^n)$ ,  $4^n \notin O(2^n)$
- d.  $2^{n+1} \notin O(2^n)$ ,  $2^{2n} \notin O(2^n)$ ,  $4^n \notin O(2^n)$

1. C poiché:

- $2^{n+1} = 2^n \times 2$  **appartiene** a  $O(2^n)$
- $2^{2n} = 2^n \times 2^n$  **non appartiene** a  $O(2^n)$
- $4^n = (2 \times 2)^n = 2^n \times 2^n$  **non appartiene** a  $O(2^n)$

La combinazione tra queste porta al risultato C

### ~~ES24~~

Sia  $T[0..m - 1]$  una tabella hash con capacità  $m = 10$  e chiavi intere. Qual è lo stato di  $T$  dopo aver inserito le chiavi 88, 12, 2, 22, 33, se si utilizza il metodo di indirizzamento aperto e la funzione hash  $h(k) = k \bmod 10$ ?

Scegli un'alternativa:

- a. {nil, nil, 12, 33, 2, 22, nil, nil, 88, nil}
- b. {nil, nil, 33, 12, 2, 22, nil, nil, 88, nil}
- c. {nil, nil, 12, 2, 22, 33, nil, nil, 88, nil}
- d. {nil, nil, 12, 22, 2, 33, nil, nil, 88, nil}

1. C