

# Tema d'esame di Algoritmi e Strutture Dati I

## (Traccia A)

### 04/03/2020

**Tempo a disposizione: 2 ore e 30 minuti.**

1. Si risolva la seguente equazione di ricorrenza, utilizzando il metodo degli alberi di ricorrenza:

$$T(n) = \begin{cases} 1 & \text{se } n \leq 3 \\ 3\sqrt[3]{n} T(\sqrt[3]{n}) + \sqrt{n} & \text{se } n > 3 \end{cases}$$

2. Sia dato il seguente algoritmo ricorsivo:

**Algorithm 1** ALGO( $T, h, k$ )

```
1  ret = 0
2  if  $T \neq NIL$  &&  $h > 0$  then
3      If  $T \rightarrow key < k$  then
4          ret = ALGO( $T \rightarrow dx, h - 2, k$ )
5      else
6          if  $T \rightarrow key \% 2 = 0$  then
7              ret = 1
8          x = ret + ALGO( $T \rightarrow sx, h/2, k$ )
9          ret = x + ALGO( $T \rightarrow dx, h - 1, k$ )
10 return ret
```

Scrivere un algoritmo **iterativo** che **simuli precisamente** il comportamento ricorsivo dell'algoritmo sopra riportato.

3. Siano dati un grafo orientato  $G = (V, E)$  e due sottoinsiemi di vertici  $V_1, V_2 \subseteq V$  (rappresentati come array, cioè  $V_1[j] \in V$  e  $V_2[j] \in V$ , per ogni  $j \in \{1, \dots, |V_1|\}$ ).  
 $\forall j \in \{1, \dots, |V_1|\}$

Si vuole verificare, in tempo lineare sulla dimensione di  $G$ , se **ogni percorso che si diparte da un qualsiasi vertice di  $V_1$  e non passa da alcun vertice di  $V_2$  è necessariamente finito**.

Si illustri brevemente a parole l'idea della soluzione che si intende proporre e spieghi perché risolve il problema in tempo lineare. Successivamente, si scriva l'algoritmo che risolve il problema.

# Tema d'esame di Algoritmi e Strutture Dati I

## 23/06/2016

**Tempo a disposizione: 3 ore.**

1. A) Si dimostri la verità o la falsità della seguente affermazione, explicitando per esteso il procedimento seguito:  
se  $h(n) = \Theta(f(n))$  e  $f(n) = \Theta(g(n))$ , allora  $g(n) + h(n) = \Theta(f(n) + g(n))$ .  
B) Si individuino, se esistenti, le costanti necessarie a dimostrare la seguente relazione:  
 $2^n = \log_2 \left(\frac{n}{\epsilon}\right) = \Theta(n)$ . Il procedimento di soluzione deve essere riportato per esteso.
2. Calcolare, explicitando il procedimento completo di soluzione seguito, il tempo di esecuzione del seguente algoritmo in funzione del parametro  $n$ .

1 ALGO( $n$ )

```
1    $x = n^2$ 
2   while  $x > 1$  do
3        $y = x/3$ 
4       while  $y < x$  do
5            $y = y + 2$ 
6            $x = x - 2$ 
7        $x = \frac{x}{3}$ 
8   return( $x$ )
```

3. Sia dato un albero binario di ricerca  $T$  (i cui nodi contengono esclusivamente un campo chiave, un campo figlio sinistro e un campo figlio destro) e un array ordinato  $A$  contenente  $m$  chiavi. Scrivere un algoritmo **ricorsivo efficiente** che inserisca nell'albero  $T$  tutti le chiavi di  $A$  che non siano già presenti in  $T$ .

Nota: per motivi di efficienza, non sarà ammessa alcuna soluzione che applichi l'algoritmo di inserimento in ABR alle singole chiavi di  $A$ . Non è, inoltre, ammesso l'uso di passaggi di parametri per riferimento né l'impiego di variabili globali.

4. Dato un insieme di etichette  $P = \{p_1, p_2, \dots, p_t\}$ , un grafo ormaiato etichettato rispetto a  $P$  è definito come una tripla  $G = (V, E, L)$ , dove  $(V, E)$  è un grafo ormaiato ( rappresentato con liste di adiacenza) e  $L : V \rightarrow P$  è una funzione totale che associa a ogni vertice una etichetta nell'insieme  $P$  (cioè  $L(v) \in P$  per ogni  $v \in V$ ).

Si scriva un algoritmo che, dati in ingresso il grafo ormaiato etichettato  $G$  rispetto a  $P$ , un vertice  $u \in V$ , e una etichetta  $p \in P$ , verifica in tempo  $O(|V| + |E|)$  se tutti i percorsi infiniti che partono da  $u$  passano da almeno un vertice etichettato con  $p$ .

# Tema d'esame di Algoritmi e Strutture Dati I

## 15/07/2016

Tempo a disposizione: 3 ore.

1. A) Si dimozi la verità o la falsità della seguente affermazione, esplicitando per esteso il procedimento seguito:  
$$\text{se } h(n) = \Theta(t(n)) \text{ e } f(n) = \Theta(g(n)), \text{ allora } \log_2(h(n) \cdot f(n)) = \Theta(\log_2(t(n) \cdot f(n))).$$
B) Si individuano, se esistono, le costanti necessarie a dimostrare la seguente relazione:  
$$2 \log_2 n - \frac{3}{2} = \Theta(\log_2 n).$$
 Il procedimento di soluzione deve essere riportato per esteso.
2. Calcolare, esplicitando il procedimento completo di soluzione seguito, il tempo di esecuzione del seguente algoritmo in funzione del parametro  $n$ .

```
1 ALGO( $n$ )
2    $x = 2^n$ 
3   while  $x > 2$  do
4      $y = \frac{x}{2}$ 
5      $z = 1$ 
6     while  $x - 1 \geq y + 1$  do
7        $y = x - z$ 
8        $z = 2 \cdot z$ 
9    $x = \frac{x}{2}$ 
9  return
```

3. Sia dato un albero binario di ricerca  $T$ , ogni nodo  $x$  del quale contiene esclusivamente un campo chiave, un campo figlio sinistro, un campo figlio destro e un campo  $N.N.$  che indica il numero di nodi contenuti nell'albero radicato nel nodo  $x$  stesso. Scrivere un algoritmo riconosciutore efficiente che cancelli dall'albero  $T$  tutti le chiavi esterne all'intervallo chiuso  $[k_1, k_2]$  e, contemporaneamente, inserisca in un array  $A$  (di dimensione pari al numero di nodi in  $T$ ) tutte quelle comprese nell'intervallo  $[k_1, k_2]$ , salvando l'informazione contenuta nel campo  $N.N.$  dei nodi. Al termine, l'array  $A$  deve risultare ordinato in modo crescente. Inoltre, l'albero risultante deve sempre rispettare il vincolo che il campo  $N.N.$  di ogni nodo indica il numero esatto di nodi contenuti nei sottosalberi in esso radicato.

L'algoritmo ricorsivo può prendere in ingresso esclusivamente la radice corrente  $T$ , i valori  $k_1$ ,  $k_2$ , l'array  $A$  e l'indice iniziale  $p$  della porzione ancora libera dell'array. Lo stesso algoritmo deve restituire come valore di ritorno la radice del nuovo albero corrente.

Nota: Non è ammesso l'uso di passaggio di parametri per riferimento ad l'impiego di variabili globali.

4. Data un grafo orientato  $G = (V, E)$ , rappresentato con liste di adiacenza. Si scriva un algoritmo che, dati in ingresso il grafo orientato  $G$ , un numero naturale  $k$  e un vertice  $u$ ,  $u \in V$ , verifichi in tempo  $O(|V| + |E|)$  se tutti i percorsi da  $u$  a  $v$  che passano da  $z$  hanno lunghezza maggiore di  $k$ .

# Tema d'esame di Algoritmi e Strutture Dati I

26/01/2017

## Tema 1

1. A) Si dimostri la verità o la falsità, esplicitando il procedimento seguito, della seguente affermazione:  
 $\log_2 \frac{h(n)}{f(n)} = \Theta\left(\log_2 \frac{t(n)}{g(n)}\right)$  e  $\log_2 \frac{h(n)}{f(n)} = \Theta(\log_2(t(n) \cdot g(n)))$  allora  $h(n) = \Theta(t(n))$ .  
 B) Si individuino, se esistono, le costanti necessarie a dimostrare la seguente relazione asintotica estesa.  
 $2n^{\left(\frac{3}{2}\right)} - 12n + 6\sqrt{n} = \Theta(n^2\sqrt{n})$ . Il procedimento di soluzione deve essere riportato per

2. Calcolare, esplicitando il procedimento completo di soluzione seguito, il tempo di esecuzione del seguente algoritmo in funzione del parametro  $n$ .

```

1  x = n
2  y =  $\frac{x}{4}$ 
3  while x > y + 1 do
4      y =  $\frac{y}{4}$ 
5      while y > 0 do
6          y = y - 1
7          x = x - 4
8      y =  $\frac{y}{4}$ 
9  return

```

3. Sia dato un albero binario di ricerca  $T$  (i cui nodi contengono esclusivamente un campo chiave, un campo figlio sinistro e un campo figlio destro). Il seguente algoritmo ricorsivo calcola la massima profondità di un nodo pari in  $T$ :

**Algoritmo** EvenDepthRic( $T$ )

```

1  ret = -1
2  if T ≠ NIL then
3      dpsz = EVENDEPTHRIC( $T \rightarrow \alpha$ )
4      dpdz = EVENDEPTHRIC( $T \rightarrow \beta$ )
5      maxdp = MAX(dpsz, dpdz)
6      if maxdp ≥ 0 then maxdp = maxdp + 1
7      if  $T \rightarrow \text{key} \% 2 = 0$ 
8          then ret = MAX(0, maxdp)
9      else ret = maxdp
10 return ret

```

Scrivere l'algoritmo iterativo EVENDEPTHITER( $T$ ) che simula precisamente il comportamento di EVENDEPTHRIC( $T$ ) sull'albero  $T$ .

4. Sia dato grafo orientato  $G = \langle V, E \rangle$ , rappresentato con liste di adiacenza. Si scriva un algoritmo che, dato in ingresso solo il grafo  $G$ , verifichi se in esso esistono tre vertici  $u$ ,  $v$  e  $z$  tali che esiste un percorso in  $G$  che parte da  $u$  e passa prima da  $v$  e poi da  $z$  e almeno un percorso di  $G$  che passa da  $u$  e passa prima da  $z$ .

# Tema d'esame di Algoritmi e Strutture Dati I

## 26/01/2017

### Tema 2

1. A) Si dimostri la verità o la falsità, esplicitando il procedimento seguito, della seguente affermazione:  
 $\log_2 \frac{t(n)}{g(n)} = \Theta(\log_2(t(n) - g(n)))$  e  $\log_2 \frac{t(n)}{f(n)} = O\left(\log_2 \frac{t(n)}{g(n)}\right)$ , allora  $\log_2 h(n) = \Theta(\log_2 t(n))$ .  
B) Si individuino, se esistono, le costanti necessarie a dimostrare la seguente relazione asintotica:  
 $7n\sqrt{n} + 3n - 10\sqrt{n} = \Theta(n^{1.5})$ . Il procedimento di soluzione deve essere riportato per esteso.
2. Calcolare, esplicitando il procedimento completo di soluzione seguito, il tempo di esecuzione del seguente algoritmo in funzione del parametro  $n$ .

#### Algoritmo Esercizio(n)

```
1  z = 2^n
2  x = n
3  while z < x do
4      y = 2z
5      while y > z do
6          y = y/2
7          z = z + y
8      x = x + 1
9  return
```

3. Sia dato un albero binario di ricerca  $T$  (i cui nodi contengono esclusivamente un campo chiave, un campo figlio sinistro e un campo figlio destro). Il seguente algoritmo ricorsivo calcola la somma delle chiavi dispari in  $T$ :

#### Algoritmo CountOddRIC( $T$ )

```
1  ret = 0
2  If  $T \neq \text{NIL}$  then
3      if  $T \rightarrow \text{key} \% 2 = 1$ 
4          then ret =  $T \rightarrow \text{key}$ 
5       $r_{ss} = \text{COUNTODDRIC}(T \rightarrow \text{sl})$ 
6       $r_{dr} = \text{COUNTODDRIC}(T \rightarrow \text{dr})$ 
7      ret = ret +  $r_{ss} + r_{dr}$ 
8  return ret
```

Scrivere l'algoritmo iterativo COUNTODDITER( $T$ ) che simula precisamente il comportamento di COUNTODDRIC( $T$ ) sull'albero  $T$ .

4. Sia dato grafo orientato  $G = \langle V, E \rangle$ , rappresentato con liste di adiacenza. Si scriva un algoritmo che, dato in ingresso solo il grafo  $G$ , verifichi se in esso esistono due vertici  $u$  e  $v$  tali che ogni percorso di  $G$  che parte da  $u$  non passa da  $v$ .

Tema d'esame di Algoritmi e Strutture Dati I  
19/02/2019  
Traccia A

Tempo a disposizione: 2.30 ore.

1. Si individui la stima asintotica per la seguente equazione di ricorrenza, utilizzando il metodo degli alberi di ricorrenza:

$$T(n) = \begin{cases} 1 & \text{se } n \leq 1 \\ T(\frac{n}{3}) + T(\frac{n}{3}) + n^2 & \text{se } n > 1 \end{cases}$$

2. Sia dato il seguente algoritmo ricorsivo:

**Algorithm 1** ALG( $A, x, y, k$ )

```
1  val = 0
2  if  $x \leq y$  then
3       $z = \frac{x+y}{2}$ 
4      if  $A[z] = k$  then
5          val = 1
6      •  $a = \text{ALG}(A, x, z-1, k)$ 
7      if  $a > \text{val}$  then
8          val =  $a + \text{val}$ 
9       $b = \text{ALG}(A, z+1, y, k)$ 
10     val =  $\text{val} + a + b$ 
11 return val
```

Scrivere un algoritmo iterativo che simuli precisamente il comportamento ricorsivo dell'algoritmo sopra riportato.

3. Si definisca un albero orientato come un grafo nel quale: (i) esiste un unico suo vertice (detto la radice) che raggiunge ogni altro suo vertice tramite un qualche percorso orientato; (ii) per ogni vertice  $v$  del grafo, esiste un solo percorso orientato che raggiunge  $v$  dalla radice. Una foresta orientata è, dunque, un insieme di alberi orientati di cardinalità  $\geq 1$ . Si scriva un algoritmo che, dato in ingresso il solo grafo orientato  $G = (V, E)$ , verifichi, in tempo lineare sulla dimensione del grafo, se esso è una foresta orientata oppure no.

# Tema d'esame di Algoritmi e Strutture Dati I

19/02/2019

Traccia B

Tempo a disposizione: 2.30 ore.

- Si individui la stima asintotica per la seguente equazione di ricorrenza, utilizzando il metodo degli alberi di ricorrenza:

$$T(n) = \begin{cases} 1 & \text{se } n \leq 1 \\ T\left(\frac{3}{5}n\right) + T\left(\frac{1}{5}n\right) + n^2 & \text{se } n > 1 \end{cases}$$

- Sia dato il seguente algoritmo ricorsivo:

```
Algorithm 1 ALG(A, x,y,k)
1   val = 0
2   if x ≤ y then
3       z =  $\frac{x+y}{2}$ 
4       if A[z] = k then
5           val = 1
6       a = ALG(A,x,z - 1,k)
7       if a > val then
8           b = ALG(A,z + 1,y,k)
9       else
10          b = a + val
11      val = val + a + b
12  return val
```

Scrivere un algoritmo iterativo che simuli precisamente il comportamento ricorsivo dell'algoritmo sopra riportato.

- Si scriva un algoritmo che, dato in ingresso un grafo orientato  $G = \{V, E\}$ , in valore intero  $k$ , un vertice  $s \in V$  e un insieme  $A \subseteq V$  rappresentato come array di vertici, verifichi, in tempo lineare sulla dimensione del grafo, se è vera la seguente condizione:

*ogni percorso che parte dal vertice s e raggiunge un qualche vertice di A ha lunghezza maggiore di k.*

# Tema d'esame di Algoritmi e Strutture Dati I

13/01/2020

## Traccia B

Tempo a disposizione: 2 ore 30 minuti.

1. Si individuino, se esistono, le costanti necessarie a dimostrare la seguente relazione:

$$2n^2 \log(n^2) + 10n^2 = \Theta(2^{2\log(n)} \log(n)).$$

Si fornisca per esteso il procedimento di soluzione seguito.

2. Sia dato il seguente algoritmo:

```
ALGO( $T, k_1, k_2, P$ )
1    $ret = 0$ 
2   if  $T \neq NIL$  then
3       if  $T \rightarrow k < k_1$  then
4            $ret = ALGO(T \rightarrow dx, k_1, k_2, T)$ 
5       else if  $T \rightarrow k > k_2$  then
6            $ret = ALGO(T \rightarrow sx, k_1, k_2, T)$ 
7       else
8            $ret = ALGO(T \rightarrow dx, k_1, k_2, T)$ 
9            $ret = ret \parallel ALGO(T \rightarrow sx, k_1, k_2, T)$ 
10      if  $\neg ret$  then
11          if  $P \rightarrow sx \neq NIL$  then
12               $P \rightarrow sx = CANCELAROOT(T)$ 
13          else
14               $P \rightarrow dx = CANCELAROOT(T)$ 
15  return  $ret$ 
```

Scrivere un algoritmo **iterativo** che simuli precisamente il comportamento ricorsivo dell'algoritmo sopra riportato.

Sia dato un grafo orientato  $G = (V, E)$ , un insieme di vertici  $X \subseteq V$  (rappresentato come array) e due vertici  $u, v \in V$ . Si scriva un algoritmo che, in **tempo lineare sulla dimensione del grafo G**, verifichi se le seguenti condizioni sono soddisfatte:

- ogni percorso che parte da  $v$  e termina in  $u$  passa necessariamente per un vertice in  $X$ ;
- ogni percorso che parte da  $u$  e termina in  $v$  passa necessariamente per un vertice in  $X$ .

# Tema d'esame di Algoritmi e Strutture Dati I

## 05/02/2020

**Tempo a disposizione: 2.30 ore.**

- Si dimostri, esplicitando il procedimento completo seguito, la verità o la falsità della seguente affermazione:

se  $f(n) = \Theta(\sqrt{g(n)})$  e  $f(n) = \Theta(k^2(n))$ , allora  $g(n) = \Theta(k(n)^4)$ .

- Sia dato il seguente algoritmo ricorsivo:

**Algorithm 1 ALGO( $A, i, j$ )**

```
1  ret = 0
2  if  $i \leq j$  then
3       $k = i + j$ 
4      if  $k \% 2 = 0$  then
5           $q = \frac{k}{2}$ 
6           $x = \text{ALGO}(A, i, q - 1)$ 
7          if  $A[q] \% 2 = 0$  then
8               $y = 1 + \text{ALGO}(A, q + 1, j)$ 
9          else
10              $y = \text{ALGO}(A, q + 1, j)$   $\nabla \mathcal{K} = \times \tau \gamma$ 
11         else
12             ret =  $\text{ALGO}(A, i, j - 1)$ 
13             if  $A[j] \% 2 = 0$  then
14                 ret = ret + 1
15 return ret
```

Scrivere un algoritmo **iterativo** che **simuli precisamente** il comportamento ricorsivo dell'algoritmo sopra riportato.

- Sia dato grafo orientato  $G = \langle V, E \rangle$ , rappresentato con liste di adiacenza, un array  $A$  di dimensione  $k \leq |V|$ , tale che  $A[i] \in V$  per ogni  $i \in \{1, \dots, k\}$ , e un vertice  $v \in V$ . Si scriva un algoritmo che, dati in ingresso i parametri  $G, A$  e  $v$ , calcoli in **tempo lineare sulla dimensione di  $G$**  l'insieme di vertici  $Z \subseteq V$  contenente il massimo insieme di vertici di  $G$  che soddisfano la seguente condizione:

- $z \in Z$  se e solo se in  $G$  esiste un percorso da  $z$  a  $v$  che, **prima di raggiungere  $v$** , passa per almeno un vertice  $a \in A$ .

# Tema d'esame di Algoritmi e Strutture Dati I

## 13/01/2021

1. Sia dato un grafo orientato  $G = \langle V, E \rangle$ , rappresentato con liste di adiacenza, e un vertice  $s$  e due insiemi di vertici  $B \subseteq V$  e  $C \subseteq V$ , rappresentati come array. Si scriva un algoritmo che, dati in ingresso  $G$ ,  $s$ ,  $B$  e  $C$ , collezioni in **tempo lineare sulla dimensione di  $G$**  in una lista  $L$  tutti i vertici  $v$  che soddisfano entrambe le seguenti condizioni:
  - $v$  appartiene a  $B$  e può raggiungere  $s$  tramite un percorso;
  - esiste anche un percorso da  $s$  a  $v$  che non passa per alcun vertice di  $C$ .

# Algoritmi e Strutture Dati I

22/09/2021

Un percorso finito  $\pi = v_1 \cdot v_2 \cdots v_k$  in un grafo si dice *massimale* se ogni sua estensione  $\pi' = \pi \cdot u$ , con  $u \in V$  arbitrario, non è più un percorso del grafo.

Siano dati due grafi orientati  $G_1 = \langle V, E_1 \rangle$  e  $G_2 = \langle V, E_2 \rangle$ , rappresentati con liste di adiacenza, e due vertici  $s \in V$  e  $v \in V$ .

Si scriva un algoritmo che verifichi in tempo lineare sui due grafi in ingresso se sono soddisfatte entrambe le seguenti condizioni:

1. ogni percorso *finito massimale* in  $G_1$  che parte da  $s$  non passa per  $v$ ;
2. tutti i percorsi *infiniti* in  $G_2$  che partono da  $s$  passano per  $v$ .

Si descriva prima l'idea ad alto livello, si dia poi l'algoritmo e, infine, se ne discuta la complessità.

# Tema d'esame di Algoritmi e Strutture Dati I 24/02/2017

## Traccia 2

Tempo a disposizione: 3 ore.

1. A) Si dimostri la verità o la falsità (tramite controesempio) della seguente affermazione: Se  $z(n) = \Theta(2^{g(n)})$  e  $\log_2 g(n) = \Theta(h(n))$ , allora  $\log_2(\log_2(z(n))^2) = \Theta(h(n))$ .  
Si assuma che le fuzioni  $g$ ,  $z$  e  $f$  siano asintoticamente crescenti e positive.  
B) Si individuino, se esistono, le costanti necessarie a dimostrare la seguente relazione:  
 $\log_2\left(\frac{4^n}{\sqrt{n}}\right) = \Theta(\log_2(5^n))$ .

2. Sia dato il seguente algoritmo:

**1** Algoritmo( $n$ )

```
1    $x = 0$ 
2    $z = n$ 
3   while  $x \leq n^2$  do
4        $j = 1$ 
5       while  $z > \log_2 j$  do
6            $j = j * 2$ 
7            $z = z - 2$ 
8        $x = x + 3z$ 
9   return
```

Calcolare, esplicitando il procedimento di soluzione seguito, il tempo di esecuzione dell'algoritmo in funzione del parametro  $n$ .

3. Sia dato il seguente algoritmo ricorsivo:

**2** ALGO( $T, P, x$ )

```
1    $ret = 0$ 
2   if  $T \neq Nil$  then
3        $a = \text{ALGO}(T \rightarrow Sx, T)$ 
4        $b = \text{ALGO}(T \rightarrow Dx, T)$ 
5        $ret = a + b + 1$ 
6   if  $T \rightarrow Key \% 2 = 0 \wedge ret < x \wedge P \neq Nil$  then
7       if  $T = P \rightarrow Dx$  then
8            $P \rightarrow Dx = \text{CANCELLA-RADICE}(T)$ 
9       else
10       $P \rightarrow Sx = \text{CANCELLA-RADICE}(T)$ 
11   return  $ret$ 
```

dove si assume che la funzione  $\text{CANCELLA-RADICE}(T)$ , qui non specificata, cancelli da un albero  $T$  la sua radice e ritorni un riferimento alla nuova radice.

Scrivere un algoritmo **iterativo** che **simuli precisamente** il comportamento ricorsivo dell'algoritmo sopra riportato.

4. Sia dato grafo orientato  $G = \{V, E\}$  e insieme di vertici  $B \subseteq V$  memorizzato in un array. Si scriva un algoritmo che, in tempo lineare sulla dimensione di  $G$ , verifichi se esiste un percorso infinito  $\pi$  in  $G$  nel quale ciascun vertice in  $B$  occorre infinite volte ( $\pi$  può ovviamente contenere anche altri vertici di  $G$ , oltre a quelli in  $B$ ).

# Tema d'esame di Algoritmi e Strutture Dati 1 28/06/2017

## Traccia 1

Tempo a disposizione: 2 ore e 30 minuti.

1. Sia data la seguente equazione di ricorrenza:

$$T(n) = \begin{cases} 1 & \text{se } n \leq 2 \\ 4 T(\sqrt{n}) + \log_2 n & \text{se } n > 2 \end{cases}$$

Si calcoli la stima più precisa possibile per  $T(n)$ , esplicitando tutti i passi dello svolgimento.

2. Sia dato il seguente algoritmo ricorsivo:

**Algorithm 1** ALGO( $T, a, b$ )

```
1  if  $T \neq \text{Nil}$  then
2      if  $T \rightarrow \text{Key} < a$  then
3           $T \rightarrow Sx = \text{ALGO}(T \rightarrow Sx, a, b)$ 
4      else if  $T \rightarrow \text{Key} > b$  then
5           $T \rightarrow Dx = \text{ALGO}(T \rightarrow Dx, a, b)$ 
6      else
7           $T \rightarrow Sx = \text{ALGO}(T \rightarrow Sx, a, b)$ 
8           $T \rightarrow Dx = \text{ALGO}(T \rightarrow Dx, a, b)$ 
9       $T = \text{CANCELLA\_RADICE}(T)$ 
10 return  $T$ 
```

dove si assuma che la funzione CANCELLA-RADICE( $T$ ), qui non specificata, cancelli da un albero  $T$  la sua radice e ritorni un riferimento alla nuova radice.

Scrivere un algoritmo iterativo che simuli precisamente il comportamento ricorsivo dell'algoritmo sopra riportato.

3. Sia dato grafo orientato  $G = (V, E)$  e quattro vertici  $a, b, c, d \in V$ . Si scriva un algoritmo che, in tempo lineare sulla dimensione di  $G$ , verifichi se ogni percorso infinito, che parte da  $a$  e passa per  $b$ , prima di raggiungere  $b$  visita necessariamente **entrambi** i vertici  $c$  e  $d$ . (NOTA: la risposta dell'algoritmo deve essere positiva se e soltanto se entrambi i vertici  $c$  e  $d$ , in qualsiasi ordine, vengono incontrati lungo ciascun percorso con le proprietà indicate sopra.)

**Traccia 2****Tempo a disposizione: 2 ore 30 minuti.**

1. Si individuino, se esistono, le costanti necessarie a dimostrare la seguente relazione:

$$8n^2 + 5 - \frac{12}{n} = \Theta\left(\log_2\left(2^{(2\cdot\log_2 n)}\right)\right).$$

Si dimostri o motivi con dettaglio ogni affermazione fatta durante lo svolgimento.

2. Sia dato il seguente algoritmo ricorsivo:

**Algorithm 1** ALGO( $T, P$ )

```

1   flag = 0
2   ret = 0
3   if  $T \neq \text{Nil}$  then
4       if  $T \rightarrow \text{Key} \% 2 = 0$  then
5           flag = 1
6        $x = \text{ALGO}(T \rightarrow Sx, T)$ 
7        $y = \text{ALGO}(T \rightarrow Dx, T)$ 
8       if flag = 1  $\wedge P \neq \text{NULL} \wedge x + y = 0$  then
9           if  $P \rightarrow Sx = T$  then
10           $P \rightarrow Sx = \text{Cancella\_Root}(T)$ 
11      else
12           $P \rightarrow Dx = \text{Cancella\_Root}(T)$ 
13      ret =  $x + y + \text{flag}$ 
14  return ret

```

dove si assume che  $P$  contenga un puntatore al padre di  $T$  e che la funzione CANCELLA-RADICE( $T$ ), qui non specificata, cancelli da un albero  $T$  la sua radice e ritorni un riferimento alla nuova radice.

Scrivere un algoritmo **iterativo** che simuli precisamente il comportamento ricorsivo dell'algoritmo sopra riportato.

3. Sia dato grafo orientato  $G = (V, E)$  e tre vertici  $a, b, c \in V$ . Si scriva un algoritmo che, in tempo lineare sulla dimensione di  $G$ , verifichi se esiste un percorso infinito che passa esattamente una volta prima per  $a$ , poi per  $b$  e infine per  $c$ . (NOTA: la risposta dell'algoritmo deve essere positiva se e soltanto se i tre vertici  $a, b$  e  $c$  vengono incontrati una sola volta e nell'ordine stabilito lungo un qualche percorso infinito.)

# Tema d'esame di Algoritmi e Strutture Dati I 18/07/2017

## Traccia 1

**Tempo a disposizione:** 2 ore e 30 minuti.

1. Sia data la seguente equazione di ricorrenza:

$$T(n) = \begin{cases} 1 & \text{se } n \leq 2 \\ 10 \cdot T\left(\frac{n}{3}\right) + n^2 & \text{se } n > 2 \end{cases}$$

Si calcoli la stima più precisa possibile per  $T(n)$ , esplicitando tutti i passi dello svolgimento.

2. Sia dato il seguente algoritmo ricorsivo:

**Algorithm 1** ALGO( $T, k, P$ )

```
1   ret = -1
2   if  $T \neq Nil$  then
3      $h_1 = \text{ALGO}(T \rightarrow Sx, k, T)$ 
4      $h_2 = h_1 + \text{ALGO}(T \rightarrow Dx, k, T)$ 
5     ret =  $h_2 + 1$ 
6     if  $ret > k \wedge P \neq NIL$  then
7        $P \rightarrow Dx = \text{CANCELLA\_RADICE}(T)$ 
8     else
9        $P \rightarrow Sx = \text{CANCELLA\_RADICE}(T)$ 
10  return ret
```

dove si assume che il parametro in input  $P$  sia il padre di  $T$  e la funzione CANCELLA-RADICE( $T$ ), qui non specificata, cancelli da un albero  $T$  la sua radice e restituisca un riferimento alla nuova radice.

Scrivere un algoritmo **iterativo** che simuli **precisamente** il comportamento ricorsivo dell'algoritmo sopra riportato.

3. Si scriva un algoritmo che, dato come **unico input** un grafo orientato  $G = (V, E)$ , verifichi in tempo lineare sulla dimensione di  $G$  se esistono tre vertici  $a, b, c \in V$  tali che **almeno una** delle seguenti condizioni è verificata:

- $a$  non raggiunge  $b$ , oppure
- $b$  non raggiunge  $c$ , oppure
- $c$  non raggiunge  $a$ .

# Tema d'esame di Algoritmi e Strutture Dati I 18/07/2017

## Traccia 2

**Tempo a disposizione: 2 ore e 30 minuti.**

- Si individuino, se esistono, le costanti necessarie a dimostrare la seguente relazione:

$$5n^3 + 5n - 10 = \Theta(2^{(3 \cdot \log_2 n)}).$$

Si dimostri o motivi con dettaglio ogni affermazione fatta durante lo svolgimento.

- Sia dato il seguente algoritmo ricorsivo:

**Algorithm 1** ALGO( $T, h$ )

```
1   ret = -1
2   s = 0
3   d = 0
4   if  $T \neq Nil$  then
5     if  $T \rightarrow Sx \neq Nil$  then
6       s = ALGO( $T \rightarrow Sx, h + 1$ )
7       if  $T \rightarrow Dx \neq Nil$  then
8         d = ALGO( $T \rightarrow Dx, h + 1$ )
9         if  $s = 0 \wedge d = 0$  then
10          ret = h
11        else
12          ret = s + d
13      return ret
```

Scrivere un algoritmo iterativo che simuli precisamente il comportamento ricorsivo dell'algoritmo sopra riportato.

- Sia dato grafo orientato  $G = (V, E)$  e tre vertici  $a, b, c \in V$ . Si scriva un algoritmo che verifichi in tempo lineare sulla dimensione di  $G$  se ogni percorso infinito che parte da  $a$  e passa per  $b$  passa necessariamente anche dal vertice  $c$  prima di raggiungere  $b$ . **NOTA:** la risposta dell'algoritmo deve essere positiva se e soltanto se il vertice  $c$  compare prima di  $b$  lungo ogni percorso infinito che parte da  $a$  e passa per  $b$ .

# Tema d'esame di Algoritmi e Strutture Dati I 16/10/2017

**Tempo a disposizione: 2 ore e 30 minuti.**

1. Sia data la seguente affermazione: Se  $2^{f(n)} = \Theta(2^{h(n)})$  e  $\log_2(h(n)) = \Theta(g(n))$ , allora  $f(n) = \Theta(2^{g(n)})$ .

Se ne stabilisca la verità o la falsità, **esibendo per esteso il procedimento seguito**. Si assuma che le fuzioni  $f, g$  e  $h$  siano asintoticamente crescenti e positive.

2. Sia dato il seguente algoritmo ricorsivo:

**Algorithm 1** ALGO( $T, a, b$ )

```
1  if  $T \neq \text{Nil}$  then
2      if  $T \rightarrow \text{Key} \geq a$  and  $T \rightarrow \text{Key} \leq b$  then
3           $T \rightarrow Sx = \text{ALGO}(T \rightarrow Sx, a, T \rightarrow \text{Key})$ 
4           $T \rightarrow Dx = \text{ALGO}(T \rightarrow Dx, T \rightarrow \text{Key}, b)$ 
5      else
6          CANCELLA_TREE( $T$ )
7           $T = \text{NULL}$ 
8  return  $T$ 
```

dove si assume che la funzione CANCELLA\_TREE( $T$ ), qui non specificata, cancelli e deallochi tutto l'albero  $T$ .

Scrivere un algoritmo **iterativo** che **simuli precisamente** il comportamento ricorsivo dell'algoritmo sopra riportato.

3. Siano dati un grafo orientato  $G = (V, E)$  e un insieme di vertici  $B \subseteq V$ . Si vuole risolvere, in tempo lineare sulla dimensione del grafo, il seguente problema: verificare se i vertici nell'insieme  $V \setminus B$  possono essere partizionati in **due insiemi disgiunti**,  $V_1$  e  $V_2$ , tali che ogni vertice di  $V_1$  può raggiungere un qualche vertice di  $B$  e tutti i vertici di  $V_2$  sono raggiungibili da qualche vertice di  $B$ . In caso di risposta affermativa, si vogliono stampare i due insiemi  $V_1$  e  $V_2$  che soddisfano la condizione.

Si illustri brevemente a parole l'idea della soluzione che si intende proporre e, successivamente, se ne scriva il relativo l'algoritmo.

# Tema d'esame di Algoritmi e Strutture Dati I 25/01/2018

## Traccia 1

Tempo a disposizione: 2 ore e 30 minuti.

1. Sia data la seguente equazione di ricorrenza:

$$T(n) = \begin{cases} 2 & \text{se } n \leq 2 \\ T\left(\frac{n}{3}\right) + 3T\left(\frac{n}{2}\right) + n^2 & \text{se } n > 2 \end{cases}$$

Si calcoli la stima più precisa possibile per  $T(n)$ , esplicitando tutti i passi dello svolgimento.

2. Sia dato il seguente algoritmo ricorsivo:

**Algorithm 1** ALGO( $T, l$ )

```
1    $X = Nil$ 
2   if  $T \neq Nil$  and  $l \geq 0$  then
3      $X = ALLOCA\_NODO()$ 
4      $X \rightarrow Key = T \rightarrow Key$ 
5     if  $T \rightarrow Dx \neq Nil$  then
6        $X \rightarrow Dx = ALGO(T \rightarrow Dx, l - 1)$ 
7     if  $T \rightarrow Sx \neq Nil$  then
8        $X \rightarrow Sx = ALGO(T \rightarrow Sx, l - 1)$ 
9   return  $X$ 
```

dove si assuma che la funzione `ALLOCA_NODO()`, qui non specificata, allochi un nuovo nodo per un albero binario.

Scrivere un algoritmo **iterativo** che simuli precisamente il comportamento ricorsivo dell'algoritmo sopra riportato.

3. Dati in ingresso un grafo  $G = \{V, E\}$ , un insieme di vertici  $B \subseteq V$  (contenuto in un array) e un vertice  $s \in V$ , si vuole costruire, in tempo lineare sul grafo  $G$ , il massimo sottografo  $G'$  di  $G$  che contiene tutti i vertici che in  $G$  raggiungono (cioè che hanno almeno un percorso che raggiunge) il vertice  $s$  senza passare da alcun vertice in  $B$ .

Si illustri brevemente a parole l'idea della soluzione che si intende proporre e, successivamente, si scriva l'algoritmo che risolve il problema.

## Traccia 2

Tempo a disposizione: 2 ore e 30 minuti.

1. Sia data la seguente equazione di ricorrenza:

$$T(n) = \begin{cases} 2, & \text{se } n \leq 2 \\ 2T\left(\frac{n}{3}\right) + T\left(\frac{n}{2}\right) + n^2 & \text{se } n > 2 \end{cases}$$

Si calcoli la stima più precisa possibile per  $T(n)$ , esplicitando tutti i passi dello svolgimento.

2. Sia dato il seguente algoritmo ricorsivo:

**Algorithm 1** ALGO( $T, l$ )

```
1    $X = Nil$ 
2   if  $T \neq Nil$  and  $l \geq 0$  then
3       if  $T \rightarrow Key \% 2 = 0$  then
4            $X = ALLOCANODO()$ 
5            $X \rightarrow Sx = ALGO(T \rightarrow Sx, l - 1)$ 
6            $X \rightarrow Dx = ALGO(T \rightarrow Dx, l - 1)$ 
7            $X \rightarrow Key = T \rightarrow Key$ 
8   return  $X$ 
```

dove si assume che la funzione `ALLOCANODO()`, qui non specificata, allochi un nuovo nodo per un albero binario.

Scrivere un algoritmo iterativo che simuli precisamente il comportamento ricorsivo dell'algoritmo sopra riportato.

3. Dati in ingresso un grafo  $G = \{V, E\}$ , e tre vertici  $s$ ,  $v$  e  $u$ , si vuole verificare, in tempo lineare sul grafo  $G$ , se esiste un percorso in  $G$  che parte da  $s$ , passa per  $v$  ma non per  $u$ .

Si illustri brevemente a parole l'idea della soluzione che si intende proporre e, successivamente, si scriva l'algoritmo che risolve il problema.

Tema d'esame di Algoritmi e Strutture Dati I 23/02/2018

## Traccia 1

Tempo a disposizione: 2 ore e 30 minuti.

1. Si verifichi, calcolando le costanti necessarie se esistono, la seguente relazione asintotica:

$$\log_2 \left( \frac{4^n}{n^2} \right) = \Theta(\log_3(2^{2n}))$$

Esplicitare per esteso il procedimento di soluzione seguito.

2. Sia dato il seguente algoritmo ricorsivo:

**Algorithm 1** ALGO( $A, i, j$ )

```
1   $x = A[i]$ 
2   $y = A[j]$  metzo
3  if  $i \leq j$  then
4     $q = \frac{i+j}{2}$ 
5    if  $i < q$  then
6       $x = \text{ALGO}(A, i, q)$ 
7      ret  $ret = ret + x$ 
8    if  $q + 1 < j$  then
9       $y = \text{ALGO}(A, q + 1, j)$ 
10     ret  $ret = ret + y$ 
11  return  $ret$ 
```

Scrivere un algoritmo **iterativo** che simuli precisamente il comportamento ricorsivo dell'algoritmo sopra riportato.

3. Siano dati un grafo  $G = (V, E)$ , due insiemi di vertici  $A \subseteq V$  e  $B \subseteq V$  (entrambi rappresentati come array) e un intero  $k$ . Si vuole verificare, in tempo **lineare sulla dimensione del grafo**  $G$ , se ogni percorso che parte da un vertice in  $A$  e raggiunge un vertice in  $B$  ha lunghezza maggiore o uguale a  $k$ .

Si illustri brevemente a parole l'idea della soluzione che si intende proporre e spieghi perché risolve il problema in tempo lineare. Successivamente, si scriva l'algoritmo che risolve il problema.

**Traccia 12**

**Tempo a disposizione: 2 ore e 30 minuti.**

1. Si verifichi, calcolando le costanti necessarie se esistono, la seguente relazione asintotica:

$$\log_2 \left( 2^n \cdot \frac{4^n}{n} \right) = \Theta(\log_2(3^{3n}))$$

Esplicitare per esteso il procedimento di soluzione seguito.

2. Sia dato il seguente algoritmo ricorsivo:

**Algorithm 1 ALGO( $T$ )**

```

1   ret = 1
2   if  $T \neq NIL$  then
3        $x = T \rightarrow key \% 2$ 
4        $a = \text{ALGO}(T \rightarrow dx)$ 
5        $x = a + x$ 
6        $y = \text{ALGO}(T \rightarrow sx)$ 
7        $ret = x * b * T \rightarrow key$ 
8   return ret

```

Scrivere un algoritmo **iterativo** che simuli precisamente il comportamento ricorsivo dell'algoritmo sopra riportato.

3. Siano dati un grafo  $G = (V, E)$  e un sottoinsieme  $I \subseteq V$  di vertici (rappresentato come array). Sia  $F_G$  l'insieme dei vertici di  $G$  che non hanno archi uscenti.

Si vuole verificare, in tempo lineare sulla dimensione di  $G$ , se per ogni vertice  $v \in I$  esiste un vertice  $u \in F_G$  tale che  $u$  è raggiungibile da  $v$ .

Si illustri brevemente a parole l'idea della soluzione che si intende proporre e spieghi perché risolve il problema in tempo lineare. Successivamente, si scriva l'algoritmo che risolve il problema.

# Tema d'esame di Algoritmi e Strutture Dati I

## 28/06/2018

**Tempo a disposizione: 2.30 ore.**

1. Si dimostri la verità o la falsità, esplicitando il procedimento seguito, della seguente affermazione:

Se  $f(n) = \Theta(\sqrt{g(n)})$  e  $2^{g(n)} = \Theta(2^n)$ , allora  $\log f(n) = \Theta(\log n)$ .

2. Sia dato il seguente algoritmo ricorsivo:

**Algorithm 1** ALGO( $T, k$ )

```
1   ret = ∞
2   if  $T \neq \text{Nil}$  then
3        $x = T \rightarrow k \% 2$ 
4        $y = \text{ALGO}(T \rightarrow sx, k)$ 
5       if  $x = k \% 2$  then
6           ret = 0
7       else
8           ret = y
9       z = ALGO( $T \rightarrow dx, k$ )
10      ret = MIN(ret, z) + 1
11  return ret
```

Scrivere un algoritmo **iterativo** che **simuli precisamente** il comportamento ricorsivo dell'algoritmo sopra riportato.

3. Sia dato grafo orientato  $G = \langle V, E \rangle$ , rappresentato con liste di adiacenza, e un array  $A$  contenente un sottoinsieme dei vertici di  $G$  (quindi,  $A \subseteq V$ ). Si scriva un algoritmo che, dati in ingresso  $G$ ,  $A$  e un vertice  $v \in V$ , calcoli, in **tempo lineare sulla dimensione di  $G$** , l'insieme di vertici  $Z \subseteq V$  contenente **tutti e soli** i vertici di  $G$  che soddisfano la seguente condizione:

- $z \in Z$  se e solo se esiste un percorso in  $G$  da  $z$  a  $v$  che passa per un qualche  $a \in A$  (non fornito in ingresso), cioè se esiste  $a \in A$  tale che  $z \rightsquigarrow a \rightsquigarrow v$  (dove  $x \rightsquigarrow y$  indica che  $x$  raggiunge  $y$  tramite un qualche percorso).

# Torna d'esame di Algoritmi e Strutture Dati I

## 24/07/2018

**Tempo a disposizione: 2.30 ore.**

1. Si risolva la seguente equazione di ricorrenza:

$$T(n) = \begin{cases} 1 & \text{se } n \leq 2 \\ 2n T(\sqrt{n}) + n^2 & \text{se } n > 2 \end{cases}$$

2. Sia dato il seguente algoritmo ricorsivo:

**Algorithm 1** ALGO( $T, L$ )

```
1  if  $T \neq Nil$  then
2       $p = T \rightarrow k \% 2$ 
3       $L_D = \text{ALGO}(T \rightarrow dx, L)$ 
4      if  $p = 0$  then
5           $x = \text{NEW-NODE}(T \rightarrow k)$ 
6           $L = \text{ALGO}(T \rightarrow sx, x)$ 
7           $x \rightarrow next = L_D$ 
8      else
9           $L = \text{ALGO}(T \rightarrow sx, L_D)$ 
10     return  $L$ 
```

- (a) Scrivere un algoritmo **iterativo** che **simuli precisamente** il comportamento ricorsivo dell'algoritmo sopra riportato. Si assuma che  $L$  sia una lista puntata e che NEW-NODE( $k$ ) restituisca un nuovo elemento di lista contenente  $k$  come chiave e il puntatore  $next$  impostato a  $Nil$ ;  
(b) Supponendo che  $T$  sia un ABR, descrivere cosa contiene il risultato della chiamata  $\text{ALGO}(T, Nil)$ .
3. Sia dato un grafo orientato  $G = \langle V, E \rangle$ , rappresentato con liste di adiacenza, e due array  $A$  e  $B$ , ciascuno contenente un sottoinsieme dei vertici di  $G$  (quindi,  $A \subseteq V$  e  $B \subseteq V$ ). Si scriva un algoritmo che, dati in ingresso  $G$ ,  $A$  e  $B$ , calcoli in **tempo lineare sulla dimensione di  $G$**  l'insieme dei vertici  $Z \subseteq V$  contenente **tutti e soli** i vertici di  $G$  che soddisfano la seguente condizione:
  - $z \in Z$  se e solo se esistono in  $G$  due percorsi da  $z$  tali che: uno porta a un qualche vertice  $a \in A$  e non passa per alcun vertice di  $B$ ; l'altro porta a un qualche vertice  $b \in B$  e non passa per alcun vertice di  $A$ .

**Tempo a disposizione: 2.30 ore.**

- Si individuino, se esistono, le costanti necessarie a dimostrare la seguente relazione:

$$\frac{1}{n} + \log_2 n^2 = \Theta\left(3^{\frac{\log_2(\log_2 n)}{\log_2 3}}\right)$$

- Sia dato il seguente algoritmo ricorsivo:

**Algorithm 1** ALGO( $A, i, j, k$ )

```

1   ret = -1
2   if  $i < j$  then
3        $m = \frac{i+j}{2}$ 
4       if  $A[m] = k$  then
5           ret = m
6       else
7           if  $i < m$  then
8               ret = ALGO( $A, i, m-1, k$ )
9           if  $m < j$  and  $A[ret] \neq k$  then
10              ret = ALGO( $A, m+1, j, k$ )
11      else
12          if  $A[i] = k$  then
13              ret = i
14  return ret

```

Scrivere un algoritmo **iterativo** che **simuli precisamente** il comportamento ricorsivo dell'algoritmo sopra riportato.

- Sia dato un grafo orientato  $G = \langle V, E \rangle$ , rappresentato con liste di adiacenza, e due vertici  $s$  e  $u$ . Sia inoltre  $VAL[]$  un array che assegna ad ogni vertice un valore numerico naturale (cioè  $VAL[v] \in \mathbb{N}$ , per ogni  $v \in V$ ). Si scriva un algoritmo che, dati in ingresso  $G$ ,  $s$ ,  $u$  e  $VAL[]$ , verifichi, in **tempo lineare sulla dimensione di  $G$** , se esiste in  $G$  un percorso infinito  $\pi$  tale che:

- $\pi$  parte da  $s$  e contiene infinite occorrenze di  $u$ ;
- $\pi$  contiene **infinte** occorrenze di vertici che abbiano valore **pari** in  $VAL[]$ ;
- $\pi$  contiene **finite** occorrenze di vertici che abbiano valore **dispari** in  $VAL[]$ ;

# Tema d'esame di Algoritmi e Strutture Dati I

## 22/10/2018

**Tempo a disposizione: 2.30 ore.**

1. Si risolva la seguente equazione di ricorrenza, utilizzando il metodo degli alberi di ricorrenza:

$$T(n) = \begin{cases} 1 & \text{se } n \leq 3 \\ \sqrt{n} T(\sqrt{n}) + \sqrt{n} & \text{se } n > 3 \end{cases}$$

2. Sia dato il seguente algoritmo ricorsivo:

**Algorithm 1** CHECKBST( $T, x, y$ )

```
1   a = 1
2   b = 1
3   if  $T \neq NIL$  then
4       val =  $T \rightarrow k$ 
5       if  $x \leq val \leq y$  then SX
6           a = CHECKBST( $T, x, val - 1$ )
7           if  $a \neq 0$  then dX
8               b = CHECKBST( $T, val + 1, y$ )
9       else
10      a = 0
11      ret = a  $\&\&$  b
12      return ret
```

Scrivere un algoritmo **iterativo** che simuli **precisamente** il comportamento ricorsivo dell'algoritmo sopra riportato.

3. Sia dato un grafo orientato  $G = \langle V, E \rangle$ , rappresentato con liste di adiacenza, e un vertice  $s$  e due insiemi di vertici  $B \subseteq V$  e  $C \subseteq V$ , rappresentati come array. Si scriva un algoritmo che, dati in ingresso  $G, s, B$  e  $C$ , collezioni in **tempo lineare sulla dimensione di  $G$**  in una lista  $L$  tutti i vertici  $v$  che soddisfano entrambe le seguenti condizioni:

- $v$  appartiene a  $B$  e può raggiungere  $s$  tramite un percorso;
- esiste anche un percorso da  $s$  a  $v$  che non passa per alcun vertice di  $C$ .

# Tema d'esame di Algoritmi e Strutture Dati I

## 24/01/2019

**Tempo a disposizione: 2.30 ore.**

1. Si individuino, se esistono, le costanti necessarie a dimostrare la seguente relazione asintotica:

$$e^n = \Theta(e^n - 2^n).$$

Il procedimento seguito va riportato per esteso.

2. Sia dato il seguente algoritmo ricorsivo, in cui sia assunta data la funzione  $\text{BEST}(a,b,k)$ :

**Algorithm 1**  $\text{ALG}(T, k)$

```
1    $a = NIL$ 
2    $b = NIL$ 
3    $ret = NIL$ 
4   if  $T \neq NIL$  then
5       if  $T \rightarrow key > k$  then
6            $a = \text{ALG}(T \rightarrow sx, k)$ 
7            $b = T$ 
8       else if  $T \rightarrow key < k$  then
9            $a = T$ 
10           $b = \text{ALG}(T \rightarrow dx, k)$ 
11      else
12           $a = \text{ALG}(T \rightarrow sx, k)$ 
13           $b = \text{ALG}(T \rightarrow dx, k)$ 
14       $ret = \text{BEST}(a, b, k)$ 
15  return  $ret$ 
```

Scrivere un algoritmo **iterativo** che simuli precisamente il comportamento ricorsivo dell'algoritmo sopra riportato.

3. Siano dati due grafi orientati  $G_1 = \langle V, E_1 \rangle$  e  $G_2 = \langle V, E_2 \rangle$ , rappresentati con liste di adiacenza, e due vertici  $s \in V$  e  $v \in V$ . Si scriva un algoritmo che verifichi in tempo lineare sui due grafi in ingresso se sono soddisfatte entrambe le seguenti condizioni:

- nessun percorso infinito in  $G_1$  che parte da  $s$  passa per  $v$ ;
- tutti i percorsi infiniti in  $G_2$  che partono da  $s$  passano per  $v$ .

# Tema d'esame di Algoritmi e Strutture Dati I 29/03/2019

## Traccia A

Tempo a disposizione: 2.30 ore.

1. Posto  $f(n) = \sum_{i=1}^n \log_2 i$ , si dimostri per esteso la verità o la falsità della seguente affermazione:  $f(n) = \Theta(n \log_2 n)$ .

2. Sia dato il seguente algoritmo ricorsivo:

**Algorithm 1** ALGO( $A, p, r, k$ )

```
1  ret = false
2  if ( $p \leq q$ ) then if ( $p \leq r$ ) then
3    if ( $p = r$ ) then
4      ret = ( $k == A[p]$ )
      else
6       $q = \lfloor \frac{p+r}{2} \rfloor$ 
6      ret = ( $k == A[q]$ ) || ALGO( $A, p, q - 1, k$ )
7      if ret = false then
8        ret = ALGO( $A, q + 1, r, k$ )
9  return ret
```

dove l'espressione  $(a == b)$  assume valore *true* se e solo se i valori di  $a$  e  $b$  sono uguali.  
Scrivere un algoritmo iterativo che simuli precisamente il comportamento ricorsivo dell'algoritmo sopra riportato.

3. Sia dato grafo orientato  $G = \langle V, E \rangle$ , rappresentato con liste di adiacenza, e un array  $A$  contenente un sottoinsieme dei vertici di  $G$  (quindi,  $A \subseteq V$ ) e due vertici  $u, v \in V$ . Si scriva un algoritmo che, dati in ingresso unicamente  $G$ ,  $A$ ,  $u$  e  $v$ , calcoli, in tempo lineare sulla dimensione di  $G$ , l'insieme di vertici  $A' \subseteq A$  contenente tutti e soli vertici di  $A$  che soddisfano la seguente condizione:

- $a \in A'$  se e solo se esiste un percorso  $\pi$  di  $G$  che parte da  $u$ , arriva a  $v$  e passa per  $a$ , cioè  $\pi$  è della forma  $u \rightsquigarrow a \rightsquigarrow v$ .

# Strutture Dati I 2

## Traccia B

**Tempo a disposizione:** 2.30 ore.

1. Posto  $f(n) = \sum_{i=1}^n i^2$ , si dimostri per esteso la verità o la falsità della seguente asserzione:  
 $f(n) = \Theta(n^3)$ .

2. Sia dato il seguente algoritmo ricorsivo:

```
Algorithm I ALGO(A,p,r,k)
1   ret = 0
2   z = 0
3   if (p ≤ r) then
4       q = ⌊ $\frac{p+r}{2}$ ⌋
5       if (k == A[q]) then
6           z = A[q]
7       ret = z + ALGO(A,q + 1,r,k)
8   if ret > 0 then
9       ret = ret + ALGO(A,p,q - 1,k)
10  return ret
```

Scrivere un algoritmo iterativo che simuli precisamente il comportamento ricorsivo del ritmo sopra riportato.

3. Sia dato grafo orientato  $G = \langle V, E \rangle$ , rappresentato con liste di adiacenza e un array  $k$ . Si scriva un algoritmo che, dati in ingresso unicamente  $G$ ,  $B$ ,  $u$ ,  $v$  e  $k$ , calcoli, in tempo lineare sulla dimensione di  $G$ , l'insieme di vertici  $B' \subseteq B$  contenente tutti e soli vertici  $b \in B$  che soddisfano entrambe le seguenti condizioni:

- $b$  è raggiungibile da  $v$  senza passare da  $u$
- $b$  raggiunge  $u$  con un percorso di lunghezza  $< k$ .

# Tema d'esame di Algoritmi e Strutture Dati I

## 13/06/2019

Tempo a disposizione: 2,30 ore.

1. Sia  $f(n) = \sum_{i=1}^n i^k$ , con  $k$  una qualsiasi costante intera positiva. Si dimostri per esteso la verità o la falsità della seguente affermazione:  $f(n) = \Theta(n^{k+1})$ .
2. Sia dato il seguente algoritmo ricorsivo:

**Algorithm 1** ALGO( $A, p, r, L$ )

```
1   ret = L
2   if ( $p \leq r$ ) then
3        $L' = \text{allocare nodo}$ 
4        $L' \rightarrow key = A[q]$  r (da 5 primo delle 4)
5        $q = \lfloor \frac{p+r}{2} \rfloor$ 
6       if  $A[q] \% 2 = 0$  then
7            $L' \rightarrow next = \text{ALGO}(A, q+1, r, L)$ 
8           ret = ALGO( $A, p, q-1, L'$ )
9       else
10           $L' \rightarrow next = \text{ALGO}(A, p, q-1, L)$ 
11          ret = ALGO( $A, q+1, r, L'$ )
12 return ret
```

Scrivere un algoritmo iterativo che simuli precisamente il comportamento ricorsivo dell'algoritmo sopra riportato.

3. Un corso di studi prevede un insieme di corsi semestrali, alcuni dei quali sono indicati come propedeutici per altri. La relazione di propedeuticità richiede che i corsi propedeutici a un qualsiasi corso  $c$  debbano essere stati superati in un semestre precedente a quello in cui si sostiene l'esame del corso  $c$ . In altre parole, se il corso  $c_1$  è propedeutico al corso  $c_2$ , gli esami dei due corsi devono essere sostenuti in due semestri differenti: l'esame  $c_1$  deve essere stato superato prima di sostenere l'esame  $c_2$ .

Si consideri ora un grafo  $G$  i cui corsi sono rappresentati dai nodi e le propedeuticità dagli archi. Scrivere lo pseudo-codice di un algoritmo che, in tempo lineare sulla dimensione di  $G$ , calcoli il numero minimo di semestri necessario a superare tutti gli esami del corso di studi, assumendo che non esistano limiti al numero di esami che uno studente può sostenere nello stesso semestre, esclusi quelli imposti dalle propedeuticità.

**Suggerimento:** Considerando che il grafo risultante deve essere aciclico, si adatti uno degli algoritmi studiati a lezione in modo da assegnare a ogni corso il primo semestre in cui può essere sostenuto senza violare alcun vincolo di propedeuticità.

03/07/2019

## Tema d'esame di Algoritmi e Strutture Dati I

Tempo a disposizione: 2.30 ore.

1. Posto  $f(n) = \sum_{i=1}^n (\log_2 i)^2$ , si dimostri per esteso la verità o la falsità della seguente affermazione:  $f(n) = \Theta(n (\log_2 n)^2)$ .

2. Sia dato il seguente algoritmo ricorsivo:

**Algorithm 1 ALGO( $A, p, r$ )**

```
1   ret = 0
2   if ( $p \leq r$ ) then
3       if ( $p = r$ ) then
4           ret =  $A[p]$ 
5       else
6            $q_1 = \left\lfloor \frac{p+2r}{3} \right\rfloor$ 
7           ret = ALGO( $A, p, q_1$ )
8            $q_2 = \left\lfloor \frac{2p+r}{3} \right\rfloor$ 
9           ret = ret + ALGO( $A, q_1 + 1, q_2$ )
10      ret = ret + ALGO( $A, q_2 + 1, r$ )
11  return ret
```

Scrivere un algoritmo iterativo che simuli precisamente il comportamento ricorsivo dell'algoritmo sopra riportato.

3. Sia dato grafo non orientato  $G = \langle V, E \rangle$ , rappresentato con liste di adiacenza. Si scriva lo pseudo-codice di un algoritmo che, dato in ingresso unicamente  $G$ , verifichi in **tempo lineare sulla dimensione di  $G$**  se è possibile partizionare l'insieme di vertici in due insiemi  $V_1 \subseteq V$  e  $V_2 \subseteq V$  che soddisfano le seguenti condizioni:

- $V = V_1 \cup V_2$ ;
- $V_1 \cap V_2 = \emptyset$ ;
- $v \in V_1$  se e solo se per ogni arco  $(v, u) \in E$  vale  $u \in V_2$ ;

In caso affermativo, l'algoritmo deve fornire i due insiemi  $V_1$  e  $V_2$  risultanti.

# Tema d'esame di Algoritmi e Strutture Dati I 04/09/2019

Tempo a disposizione: 2.30 ore.

1. Si individuino, se esistono, le costanti necessarie a dimostrare la seguente relazione:

$$3n^2 - 3\sqrt{n} + 5 = \Theta(n^2).$$

Il procedimento di soluzione deve essere riportato per esteso.

2. Scrivere un algoritmo **iterativo** che **simuli precisamente** l'algoritmo ricorsivo sotto riportato, dove:  $T$  è un albero ternario, i cui nodi contengono oltre al campo  $key$ , tre puntatori,  $f_1, f_2$  e  $f_3$ , uno per ogni figlio;  $k_1$  e  $k_2$  sono due chiavi; e  $L$  una lista.

1 Algoritmo  $\text{ALGORITMO}(T, k_1, k_2, L)$

```
1 if  $T \neq \text{Nil}$  do
2    $L' = \text{ALGORITMO}(T \rightarrow f_1, k_1, k_2, L)$ 
3    $L' = \text{ALGORITMO}(T \rightarrow f_2, k_1, k_2, L')$ 
4    $L' = \text{ALGORITMO}(T \rightarrow f_3, k_1, k_2, L')$ 
5   if  $k_1 \leq T \rightarrow key \leq k_2$  do
6      $L = \text{ALLOCA\_NODO}()$ 
7      $L \rightarrow key = T \rightarrow key$ 
8      $L \rightarrow next = L'$ 
9   else
10     $L = L'$ 
11 return  $L$ 
```

3. Sia dato un grafo orientato  $G = \langle V, E \rangle$ , rappresentato con liste di adiacenza, due vertici  $u, v \in V$  e un array  $A$ , contenente un sottoinsieme dei vertici di  $G$  (quindi,  $A \subseteq V$ ). Si scriva un algoritmo che, dati in ingresso  $G$  e  $A$ , calcoli in **tempo lineare sulla dimensione di  $G$** , l'insieme dei vertici  $Z \subseteq A$  che soddisfa la seguente condizione:  $z \in Z$  se e solo se ogni percorso  $\pi$  che parte da  $u$  e raggiunge  $v$  non passa per  $z$ .