

*senza nome*

1) Siano  $f, g, h$  arbitrarie funzioni asintoticamente crescenti e positive. Dimostrare la verità o falsità della seguente affermazione:  
se  $f(n) = \Theta(g(n))$ , allora  $h(f(n)) = \Theta(h(g(n)))$   
nota: si ricordi che  $h(f(n))$  denota la composizione delle funzioni nn la moltiplicazione di  $f$  e  $g$

*compiuto 1*

2) Sia data la seguente equazione di ricorrenza:  
 $T(n) = 1$  se  $n=1$ ,  $T(n) = 2T(1/2*n) + n(\log_2(n^2))$  se  $n > 1$   
Calcolare la stima asintotica più vicina possibile a  $T(n)$

3) sia dato un generico albero binario  $T$ . Si definisca un algoritmo ricorsivo che costruisca un nuovo albero  $T'$  strutturalmente identico a  $T$ , tale, cioè, che ogni nodo di  $T$  abbia una controparte situata nella stessa posizione in  $T'$ . Inoltre, ciascun nodo  $u$  di  $T'$  deve contenere come chiave il numero di nodi pari contenuti nel sottoalbero di  $T$  radicato nel suo nodo controparte  $u$  in  $T$ .

*esercizio 2*  
*esercizio 2*

purtroppo non ho lo scanner...mo te la scrivo qui  
esercizio 1 [5 punti]

Siano  $f$  e  $g$  due arbitrarie funzioni asintoticamente crescenti e positive.  
Si dimostri la verità o falsità della seguente affermazione:  
se  $\log(\log f(n)) = \Theta(\log(\log g(n)))$  allora  $\log f(n) = \Theta(\log g(n))$   
ps il log è in base due

esercizio 2 [7 punti]

$T(n) =$   
1 se  $n=1$   
 $\sqrt{n} * T(\sqrt{n}) + n$  se  $n > 1$

esercizio 3 [7 punti]

Si scriva un algoritmo ricorsivo efficiente che, dato un albero binario  $T$ , verifichi (in una singola visita dell'albero) se per ogni nodo dell'albero i suoi sottoalberi sinistro e destro hanno lo stesso numero di nodi con chiave pari.

Non è ammesso l'uso di variabili globali né di parametri per riferimento

esercizio 4 [11 punti]

Un percorso semplice (quindi non ciclico) in un grafo orientato  $G$  si dice [A] massimale [B] se non vi si può aggiungere "alla fine" nessun altro nodo senza renderlo un percorso ciclico o fargli perdere la proprietà di essere un percorso.

Si scriva un algoritmo efficiente che, dato un grafo orientato  $G$  e un nodo  $s$  di  $G$ , stampi tutti i percorsi massimali di  $G$  che si dipartono da  $s$ .

[B] Suggerimento: [B] è possibile risolvere il problema tramite un'opportuna variante della visita in profondità e l'impiego di una coda o stack esplicito

# Tema d'esame di Algoritmi e Strutture Dati

## Modulo A

### 22/01/2001

**Tempo a disposizione: 3 ore.**

1. Ordinare in modo crescente secondo il tasso di crescita asintotico le seguenti funzioni, esplicitando e dimostrando per esteso le relazioni asintotiche ( $o(\cdot)$ ,  $\omega(\cdot)$ , oppure  $\Theta(\cdot)$ ) esistenti tra le funzioni adiacenti nell'ordinamento risultante:

$$\begin{array}{ll} n^{1/a} & \log(\log n^a) \\ 2^{\log(2e^{\ln n})} & n^a \\ \log(\log n) & n + 2 \end{array}$$

**Nota:**  $a$  è da considerarsi una costante arbitraria che soddisfa come **unico vincolo** quello di essere **maggiori di 0**.

2. Sia data la seguente equazione di ricorrenza:

$$T(n) = \begin{cases} 5 & \text{se } n = 2 \\ 2\sqrt{n} T(\sqrt{n}) + n & \text{se } n > 2 \end{cases}$$

- (a) Trovare la stima asintotica più vicina possibile a  $T(n)$ , utilizzando il **metodo iterativo**.
- (b) Risolvere l'esercizio utilizzando il **metodo di sostituzione**, utilizzando come ipotesi di soluzione i risultati calcolati per il punto (a)

3. Considerate l'algoritmo di ordinamento Quicksort.

- (a) Illustrare dettagliatamente i passi eseguiti dall'algoritmo sulla seguente sequenza di numeri interi in input:

$$\langle 95, 90, 76, 45, 20, 25, 34, 38 \rangle$$

- (b) Mostrare una permutazione degli 8 numeri della sequenza al punto (a) che costituisce il **caso migliore** per tale algoritmo di ordinamento per sequenze di 8 elementi, e motivare la scelta della sequenza.

4. Sia dato un albero binario di ricerca  $T$  qualsiasi con  $N$  nodi. Ciascun nodo  $n$  contiene un campo `colore[n]` che può assumere valori `rosso` o `nero`. Si assuma che  $T$  sia stato colorato assegnando a ciascun nodo  $n$  un valore per il campo `colore[n]`. Scrivere un algoritmo che decide se  $T$  è un albero Red–Black oppure no. In caso affermativo, l'algoritmo deve ritornare l'altezza nera; in caso negativo, deve ritornare il primo sottoalbero che viola la condizione e segnalare il motivo per cui l'albero non è un albero Red–Black.

# Tema d'esame di Algoritmi e Strutture Dati

## Modulo A

### 01/03/2001

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

1. Date le seguenti coppie di funzioni, dimostrare per esteso le relazioni asintotiche più restrittive possibili esistenti tra le funzioni di ciascuna coppia:

(i)	$(3/2)^n$	$(7/5)^n$
(ii)	$\frac{n^2}{\sqrt{\log n}}$	$n\sqrt{\log n}$
(iii)	$n \log^2 n$	$\log(4^n) \log(n^4)$
(iv)	$n^{(\log n)}$	$2^{\log^4 n}$

2. Sia data la seguente equazione di ricorrenza:

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

dove  $k$  è una costante positiva. Trovare la stima asintotica più vicina possibile di  $T(n)$ , utilizzando il **metodo iterativo**. È ammesso l'uso di un albero di ricorrenza.

3. Sia dato un array di  $n$  elementi le cui chiavi possono assumere solo valore **vero**, **forse** o **falso**. Scrivere un algoritmo che in tempo  $O(n)$  riarrangi gli elementi dell'array in modo che tutte le chiavi con valore **falso** precedano tutte le chiavi con valore **forse**, e queste ultime precedano tutte le chiavi con valore **vero**.
4. Partendo dalla definizione di **albero AVL minimo**, dimostrare la seguente affermazione per induzione o argomentarene la falsità, esibendo un controesempio:

in un **albero AVL minimo** di altezza  $h$ , l'**altezza minima** di un nodo foglia è esattamente  $\lceil h/2 \rceil$ .

# Tema d'esame di Algoritmi e Strutture Dati

## Modulo B

### 18/06/2001

**Tempo a disposizione: 3 ore.**

Considerate il problema di memorizzare una collezione di elementi tramite un albero di ricerca binario in modo da minimizzare il numero di confronti necessari per trovare un dato elemento. Se tutti gli elementi avessero uguale probabilità di essere ricercati, allora la soluzione ottima per memorizzare gli elementi sarebbe tramite un albero perfettamente bilanciato. Se però alcuni elementi vengono ricercati con maggiore frequenza di altri, un albero non bilanciato può essere preferibile.

Sopponete di avere  $N$  elementi distinti con chiavi  $k_1 < k_2 < \dots < k_N$ , e che l'elemento  $i$ -esimo abbia probabilità  $p_i \geq 0$  di essere ricercato. Supponete, inoltre, che ogni ricerca acceda effettivamente ad un elemento nell'albero, in altre parole che valga  $\sum_{i=1}^N p_i = 1$ . Se l'elemento  $i$ -esimo è memorizzato a livello  $l_i$  dell'albero (la radice è a livello 1, i suoi figli a livello 2, ecc.), allora per trovare l'elemento  $i$  saranno necessari  $l_i$  confronti.

Quindi, per un dato albero, il numero medio di confronti necessari per effettuare una ricerca è  $\sum_{i=1}^N p_i l_i$ . L'obiettivo è determinare l'albero binario che minimizza tale quantità.

1. Considerate l'algoritmo “greedy” che inserisce le chiavi in ordine decrescente di frequenza. Fornire un esempio minimo (con il più piccolo numero di elementi) che mostri che questa strategia greedy non sempre fornisce l'albero ottimo di ricerca.
2. Fornire un algoritmo che risolve correttamente il problema, e se ne analizzi la complessità.

# Tema d'esame di Algoritmi e Strutture Dati

## Modulo B

### 18/06/2001

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

Considerare il problema di disporre  $N$  libri su degli scaffali in una libreria.

Sia  $b_i$  (con  $1 \leq i \leq N$ ) il libro  $i$ -esimo nell'ordinamento, e siano  $s_i$  la misura del suo spessore e  $a_i$  la misura della sua altezza. Ogni scaffale ha la stessa lunghezza  $L$ .

**L'ordine in cui i libri devono essere disposti è fissato secondo un sistema di catalogazione (ad esempio in ordine alfabetico per autore, e se dello stesso autore per ordine alfabetico del titolo) e non può essere cambiato.**

Le altezze dei libri possono essere differenti da libro a libro, e le distanze in cui fissare gli scaffali nella libreria possono essere aggiustati secondo l'altezza del libro più alto che verrà riposto sullo scaffale stesso.

Il **costo** di una particolare disposizione dei libri sugli scaffali sia definita come la somma su tutti gli scaffali delle altezze del libro più alto su ciascuno scaffale.

Il problema è quello di trovare la disposizione dei libri sugli scaffali in modo da minimizzare il costo.

1. Fornire un esempio che mostri che l'algoritmo “greedy” per disporre i libri sugli scaffali, utilizzando la strategia di riempire il più possibile gli scaffali, non sempre fornisce la soluzione ottima.
2. Fornire un algoritmo che risolve correttamente il problema e ne si analizzi la complessità.

# Tema d'esame di Algoritmi e Strutture Dati

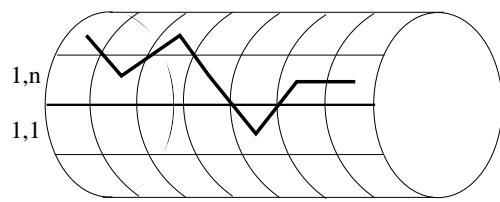
## Modulo B

### 21/11/2001

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

Considerare una matrice  $A : n \times n$  di interi positivi ( $a_{i,j}$ ), con  $1 \leq i, j \leq n$ . Si supponga di unire i due bordi (o estremi) orizzontali della matrice in modo da formare un cilindro, cioè in modo che la prima e l'ultima riga siano tra loro contigue (vedi figura).

j \ i	1	2	3	4	...
1					
2					
3					
4					
:					
n					



Si consideri ora il problema di tracciare un *percorso* a partire da una cella della prima colonna fino ad una cella nell'ultima colonna, sotto le seguenti restrizioni: da ogni cella  $(i, j)$ , è possibile muoversi solo nelle celle  $(i + 1, j)$ ,  $(i + 1, j - 1)$  o  $(i + 1, j + 1)$ . Il percorso può partire da una qualsiasi cella della prima colonna, e terminare in una qualsiasi cella dell'ultima colonna (vedi figura). Il *costo* di un percorso è costituito dalla somma dei valori delle celle attraversate dal percorso stesso. Il problema è di calcolare il percorso di peso minimo nella matrice  $A$  in input.

1. Esibire la *sottostruttura ottima* per il problema in questione e dimostrarne l'ottimalità.
2. Esibire l'*equazione di ricorrenza* che caratterizza il costo della soluzione ottima.
3. Fornire un algoritmo per calcolare la *soluzione ottima* (e non solamente per il costo della soluzione ottima) e studiarne la complessità.

# Tema d'esame di Algoritmi e Strutture Dati

## Modulo A

### 28/02/2002

**Tempo a disposizione: 3 ore.**

1. Studiare la relazione asintotica tra le seguenti funzioni, esplicitando e dimostrando per esteso la relazione asintotica ( $o(\cdot)$ ,  $\omega(\cdot)$ , oppure  $\Theta(\cdot)$ ) esistente tra di esse:

$$n^a \quad \log |\log n^a|$$

**Nota:**  $a$  è da considerarsi una **costante arbitraria**, mentre  $|f(n)|$  indica il valore assoluto della funzione  $f(n)$ .

2. Sia data la seguente equazione di ricorrenza:

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

Trovare la stima asintotica più vicina possibile a  $T(n)$ , utilizzando il **metodo iterativo** (è ammesso l'impiego degli alberi di ricorrenza).

3. Dimostrare per induzione la validità della seguente uguaglianza:

$$\sum_{i=1}^{n-1} \frac{1}{i(i+1)} = 1 - \frac{1}{n}$$

4. Sia dato un albero binario di ricerca  $T$  nei cui nodi **non** sia presente il puntatore al padre. Si descriva prima e si sviluppi poi un algoritmo **ricorsivo** che realizzi la cancellazione di un nodo dell'albero (se esiste). L'algoritmo dovrà ricevere in input:

- un puntatore  $T$  alla radice del (sotto)-albero su cui eseguire l'operazione di cancellazione;
- il valore della chiave  $k$  (e **non** il nodo) da eliminare (se presente).

**Nota:** ogni ulteriore algoritmo di supporto sviluppato dovrà essere di tipo ricorsivo.

# Tema d'esame di Algoritmi e Strutture Dati Mod. B

## 31/01/2003

Tempo a disposizione: **3 ore**

Il conduttore di un programma radiofonico deve ogni giorno compilare il palinsesto del suo programma musicale, pevedendo la programmazione di alcune delle canzoni dalla lista  $I$  di  $m$  canzoni della stazione, indicizzate con i numeri da 1 ad  $m$ , in modo da massimizzare il gradimento dei suoi ascoltatori. La radio per cui il conduttore lavora lancia ogni giorno un sondaggio col quale gli ascoltatori possono votare le canzoni da loro preferite. Il conduttore ha quindi ogni giorno a disposizione i risultati del sondaggio di gradimento del giorno prima, in forma di valori da 0 ad 1 ( $g_i$  indica il gradimento della canzone  $i$ ). Il programma radiofonico ha la durata di  $N$  minuti e ciascuna canzone ha la sua durata ( $d_i$  è la durata in secondi della canzone  $i$ ).

Definire un algoritmo che, dati in ingresso la durata  $N$  in minuti del programma, la lista  $I$  delle canzoni dalla prima alla  $m$ -esima, la lista associata  $D[1..m]$  delle durate in secondi delle canzoni e la lista  $G[1..m]$  dei valori di gradimento del giorno prima, fornisca in uscita il palinsesto per il programma che massimizza il gradimento del pubblico (nota che il palinsesto non può contenere più occorrenze della stessa canzone).

# Tema d'esame di Algoritmi e Strutture Dati

## Modulo B

### 024/02/2003

**Tempo a disposizione: 3 ore.**

Un turista in viaggio deve visitare una serie di località di interesse artistico sitate nei pressi della città  $X$ . Prima di partire per il viaggio ha già definito l'ordine di visita delle località (dato dalla sequenza  $l[1], \dots, l[n]$ ), una per ogni giornata di viaggio. Una volta giunto nella città  $X$ , il turista deve iniziare il suo viaggio tra le località. Viene informato che in ogni località  $i$  (con  $1 \leq i \leq n$ ) di suo interesse sono a disposizione differenti mezzi di trasporto tra una località e la successiva (l'insieme  $M[i]$  elenca i mezzi di trasporto che il turista ha a disposizione nella località  $i$  per raggiungere la località  $i + 1$ , mentre  $M[0]$  sono i mezzi a disposizione per raggiungere la località 1 dalla città  $X$  di partenza). Ogni mezzo di trasporto disponibile in ciascuna località ha un suo costo di utilizzo (sia  $price[i, j]$  il costo per l'utilizzo del mezzo di trasporto  $j$  nella località  $i$ ). Inoltre, il turista ha tra le sue preferenze di viaggio il non voler mai utilizzare lo stesso mezzo di trasporto per due giorni di seguito (anche se è disposto a riutilizzare uno stesso mezzo di trasporto in giornate non consecutive).

Si definisca un algoritmo di Programmazione Dinamica che risolva il problema di pianificare quali mezzi di trasporto convenga utilizzare al turista per spostarsi tra le varie località, in modo da ridurre al minimo il costo complessivo del viaggio, ma anche da rispettare le preferenze di viaggio del turista.

# Tema d'esame di Algoritmi e Strutture Dati

## Modulo B

### 27/06/2003

**Tempo a disposizione: 3 ore.**

Sia dato un **grafo orientato aciclico**  $G = \langle V, E \rangle$  rappresentato tramite liste di adiacenza. Ad ogni arco  $(i, j)$  sia associato in costo  $\text{peso}(i, j)$  come valore intero. Il problema è quello di calcolare il percorso di peso minimo tra un vertice sorgente  $s_1$  ed un vertice destinazione  $s_z$ . Il costo di un percorso sia definito come la somma dei pesi assicati agli archi che lo costituiscono. Definire un algoritmo per risolvere il problema dato. Più specificatamente:

1. scomporre il problema in sottoproblemi e dimostrare la proprietà di sottostruttura ottima;
2. esibire l'equazione di ricorrenza per il calcolo del costo del percorso ottimo;
3. fornire un algoritmo di programmazione dinamica che calcoli il costo della soluzione ottima, e se ne analizzi la complessità;  
[**Suggerimento:** al fine di poter riempire correttamente la struttura tabellare che rappresenta i (costi dei) sottoproblemi, può risultare utile calcolare un qualche ordinamento tra i vertici, utilizzando, ad esempio, una variante opportuna di uno degli algoritmi su grafi visti a lezione.]
4. infine, fornire un algoritmo che stampi il percorso ottimo tra  $s_1$  e  $s_z$ .

# Tema d'esame di Algoritmi e Strutture Dati

## Modulo A

19/06/2006



**Tempo a disposizione: 3 ore.**

1. [5 punti] Si dimostri la verità o la falsità della seguente affermazione:

$$\text{se } 2^{f(n)} = \Theta(2^{g(n)}), \text{ allora } f(n) = \Theta(g(n))$$

2. [5 punti] Sia data la seguente equazione di ricorrenza:

$$T(n) = \begin{cases} 1 & \text{se } n = 1 \\ 3T(n/4) + \sqrt{n} & \text{se } n > 1 \end{cases}$$

Risolvere l'equazione di ricorrenza utilizzando il metodo iterativo (alberi di ricorrenza).

3. [10 punti] Si scriva un *algoritmo ricorsivo* efficiente che cancelli da un albero binario di ricerca  $T$  tutte le chiavi contenute in un altro albero binario di ricerca  $T'$ .
4. [10 punti] Si scriva un algoritmo dettagliato che, dato un grafo orientato  $G$ , ne calcoli le *componenti fortemente connesse* e stampi i vertici contenuti in ciascuna componente e il numero delle componenti del grafo.

# Tema d'esame di Algoritmi e Strutture Dati I

## 11/09/2006

**Tempo a disposizione: 3 ore.**

1. **(4 punti)** Dimostrare o falsificare la seguente affermazione:

$$\text{Se } f(n) = O(g(n)) \text{ allora } \sqrt{g(n)} = \Omega(\sqrt{f(n)})$$

2. **(6 punti)** Risolvere la seguente equazione di ricorrenza:

$$T(n) = \begin{cases} 1 & \text{se } n \leq 1 \\ 4T(n/9) + \sqrt{n} & \text{se } n > 1 \end{cases}$$

3. **(10 punti)** Si definisca un algoritmo ricorsivo efficiente che, ricevuti in ingresso un (riferimento ad un) Albero Binario di Ricerca  $T$  e tre valori  $k_{min}, k_{max}$  (con  $k_{min} \leq k_{max}$ ) e  $z \geq 0$  cancelli dall'albero  $T$  tutti i nodi che o hanno chiave con valore **esterno all'intervallo**  $[k_{min}, k_{max}]$  o stanno a **distanza maggiore uguale a**  $z$  dalla radice. Non è ammesso l'uso di variabili globali né del passaggio di parametri per riferimento.

4. **(10 punti)** Scrivere un algoritmo che, in tempo lineare sulla dimensione del grafo in input, ne calcoli un ordinamento topologico. **Non è ammesso l'impiego della visita in profondità (DFS).**

# Tema d'esame di Algoritmi e Strutture Dati I

## 25/06/2007

**Tempo a disposizione: 3 ore.**

1. [6 punti] Si supponga che  $h(n) = \Theta(t(n))$  e che  $f(n)/h(n) = \Theta(g(n))$ . Si dimostri la verità o la falsità della seguente affermazione:

$$\frac{f^2(n)}{h(n) t(n)} = \Theta(g^2(n))$$

2. [7 punti] Sia data la seguente equazione di ricorrenza:

$$T(n) = \begin{cases} 1 & \text{se } n = 1 \\ 15 T(n/4) + n^2 \log n & \text{se } n > 1 \end{cases}$$

Trovare la stima asintotica più vicina possibile a  $T(n)$ .

3. [11 punti] Scrivere un algoritmo ricorsivo efficiente che, dato un albero binario di ricerca  $T$  e un due valori  $k_1$  e  $k_2$  (con  $k_1 \leq k_2$ ), cancelli da  $T$  tutti nodi con chiavi comprese tra  $k_1$  e  $k_2$  e restituisca il numero di nodi cancellati dall'albero.

**Non è ammesso l'uso di passaggio di parametri per riferimento né l'impiego di variabili globali.**

4. [6 punti] Un *tour di Eulero* in un grafo orientato  $G = \langle V, E \rangle$  è un percorso ciclico che attraversa ogni arco una e una sola volta. Un tour di Eulero in un grafo orientato  $G$  esiste se il grado entrante di ogni vertice è uguale al suo grado uscente. Si scriva un algoritmo che, dato un grafo orientato  $G$ , verifichi se esista o meno un tour di Eulero in tempo  $\Theta(|V| + |E|)$ .

# Tema d'esame di Algoritmi e Strutture Dati I

## 16/07/2007

**Tempo a disposizione: 3 ore.**

1. [7 punti] Usando la definizione di  $\Theta$ , si domostri la verità o la falsità della seguente affermazione:

se  $f(n) = \Theta(n)$  e  $g(n) = \Theta(2^{n^2})$ , allora  $2^{2 \cdot \log f(n)} = \Theta(\log(g(n)))$

2. [7 punti] Sia data la seguente equazione di ricorrenza:

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

Trovare la stima asintotica più vicina possibile a  $T(n)$ .

3. [10 punti] Scrivere un algoritmo ricorsivo efficiente che, dato un albero binario di ricerca  $T$ , due valori  $k_1$  e  $k_2$  (con  $k_1 \leq k_2$ ) e un valore  $c$ , restituisca (se esiste), effettuando una sola visita dell'albero, il puntatore al nodo di  $T$  che ha chiave compresa tra  $k_1$  e  $k_2$  e al tempo stesso che sia la più vicina possibile al (ma diversa dal) valore  $c$ .

**Non è ammesso l'uso di passaggio di parametri per riferimento né l'impiego di variabili globali.**

4. [6 punti] Si scriva un algoritmo che, dato un grafo orientato  $G$  e un vertice  $v$  di  $G$ , verifichi in tempo lineare sulla dimensione del grafo se  $G$  è un albero radicato in  $v$ .

**Non è ammesso l'uso di passaggio di parametri per riferimento né l'impiego di variabili globali.**

# Tema d'esame di Algoritmi e Strutture Dati I

## (12/02/2008)

**Tempo a disposizione: 3 ore.**

1. [7 punti] Siano  $f(n)$  e  $g(n)$  due funzioni asintoticamente positive e crescenti. Dimostrare la verità o la falsità della seguente affermazione:

$$\log(f(n) \cdot g(n)) = O(\max\{\log f(n), \log g(n)\})$$

2. [6 punti] Si consideri la seguente equazione di ricorrenza:

$$T(n) = \begin{cases} 1 & \text{se } n = 1 \\ 17T(n/2) + n^4 & \text{se } n > 1 \end{cases}$$

Trovare la stima più vicina possibile a  $T(n)$ .

3. [10 punti] Si consideri un albero binario  $T$  che soddisfa la seguente proprietà: *il valore della chiave di ogni nodo è non minore del valore delle chiavi dei suoi due figli*.

Si definisca un algoritmo ricorsivo che dato (il riferimento a) l'albero  $T$  e due valori di chiave  $k_{min}$  e  $k_{max}$  (con  $k_{min} < k_{max}$ ), cancelli dall'albero  $T$  tutti i nodi con chiave compresa tra  $k_{min}$  e  $k_{max}$ , preservando la proprietà sopra riportata.

**Suggerimento:** Può risultare utile sviluppare un algoritmo ricorsivo di appoggio che esegua la cancellazione della radice di un (sotto) albero del tipo descritto sopra e ritorni l'albero risultante.

4. [7 punti] Si definisca un algoritmo che, dato in ingresso un grafo  $G$  arbitrario e un vertice qualsiasi  $v$ , trasformi il grafo in ingresso nell'albero radicato in  $v$  dei percorsi minimi che si dipartono da  $v$ .

# Tema d'esame di Algoritmi e Strutture Dati I

## 27/03/2008

Tempo a disposizione: 3 ore.

1. (7 punti) Siano  $f$  e  $g$  due funzioni positive e asintoticamente crescenti. A seconda del caso, dimostrare la verità o la falsità delle seguenti affermazioni:

(a) se  $h^2(n) = \Theta(\min\{f(n), g(n)\})$ , allora  $\sqrt{f(n)} = O(h(n))$ .

(b) se  $\sqrt{h(n)} = O(\min\{f(n), g(n)\})$ , allora  $h(n) = O(g^2(n))$ .

2. (6 punti) Risolvere la seguente equazione di ricorrenza:

$$T(n) = \begin{cases} 1 & \text{se } n \leq 1 \\ T(3n/4) + T(n/2) + n & \text{altrimenti} \end{cases}$$

3. (9 punti) Si definisca un algoritmo ricorsivo efficiente che, ricevuti in ingresso un (riferimento ad un) Albero Binario di Ricerca  $T$  e tre valori  $k_{\min}, k_{\max}$  (con  $k_{\min} \leq k_{\max}$ ) e  $k$ , restituisca, se esiste, quel nodo di  $T$  che ha chiave con valore interno all'intervallo  $[k_{\min}, k_{\max}]$  e che, contemporaneamente, sia il più lontano possibile da  $k$ . Non è ammesso l'uso di variabili globali né del passaggio di parametri per riferimento.

4. (8 punti) Scrivere un algoritmo che dato in ingresso un vertice  $s$  e un grafo orientato  $G$ , rappresentato con liste di adiacenza, stampi tutti percorsi ciclici di  $G$  che si dipartono da  $s$ .

# Tema d'esame di Algoritmi e Strutture Dati I

## 20/06/2008

Tempo a disposizione: 3 ore.

1. [6 punti] Si domostri la verità o la falsità della seguente affermazione:

se  $2^{f(n)} = \Theta(g(n))$  e  $g(n) = \Theta(h(n)^k)$  per una qualche costante  $k > 0$ , allora  
 $f(n) = \Theta(\log h(n))$

2. [5 punti] Sia data la seguente equazione di ricorrenza:

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

Trovare la stima asintotica più vicina possibile a  $T(n)$ .

3. [10 punti] Sia dato un albero binario di ricerca  $T$  in cui ogni nodo contiene esclusivamente un campo per la chiave, uno per il puntatore al figlio destro e uno per il figlio sinistro. Siano inoltre dati in ingresso un possibile valore di chiave  $k$  e due valori  $l_1$  e  $l_2$  (con  $l_1 \leq l_2$ ). Scrivere un algoritmo ricorsivo efficiente che cerchi e stacchi, se esiste, quel "nodo dell'albero  $T$  che, tra i nodi che si trovano ad un livello di profondità esterno all'intervallo  $[l_1, l_2]$  e che sono diversi dalla radice di  $T$ , contiene la chiave più vicina possibile a  $k$  ma maggiore di  $k$ ".

L'algoritmo dovrà ritornare il riferimento al nodo cercato, se esso esiste. Non è ammesso l'uso di passaggio di parametri per riferimento né l'impiego di variabili globali.

4. [9 punti] Sia dato un grafo orientato  $G = \langle V, E \rangle$ , rappresentato tramite liste di adiacenza. Si definisca un algoritmo che verifichi in tempo lineare sulla dimensione del grafo se è vero che esiste un vertice  $v \in V$  tale che:

- se  $u \in V$  allora  $u \rightsquigarrow v$  per qualche percorso  $\pi$ ;
- per ogni  $z \in V$ ,  $v \rightsquigarrow z$  per qualche percorso  $\lambda$ ;

# Tema d'esame di Algoritmi e Strutture Dati I

(12/01/2009)

Tempo a disposizione: 3 ore.

V

- [6 punti] Assumendo che  $f$  e  $g$  siano funzioni asintoticamente positive e crescenti, si dimostri la verità o la falsità della seguente affermazione:

$$\text{se } f(n) = \Theta(n) \text{ e } g(n) = \Theta(2^n), \text{ allora } 2^{f(n)} = \Theta(g(n))$$

✓

- [6 punti] Risolvere la seguente equazione di ricorrenza:

$$T(n) = \begin{cases} 1 & \text{se } n \leq 1 \\ 6 T(n/8) + \sqrt{n} & \text{se } n > 1 \end{cases}$$

- [10 punti] Scrivere un algoritmo ricorsivo efficiente che, dato un albero binario di ricerca  $T$ , un possibile valore di chiave  $k$ , e due interi  $l_1$  e  $l_2$  (con  $1 \leq l_1 \leq l_2$ ), cerchi un nodo dell'albero che soddisfa la seguente proprietà:

"contiene la più grande chiave pari minore di  $k$  tra i nodi che si trovano a profondità compresa tra  $l_1$  e  $l_2$ "

Se il nodo cercato esiste, l'algoritmo dovrà staccare dall'albero  $T$  il nodo trovato e rimuovere il riferimento al nodo stesso. Non è ammesso l'uso di passaggio di parametri per riferimento né l'impiego di variabili globali.

- [8 punti] Si definisca un algoritmo che, dato in ingresso un grafo  $G$  arbitrario e un vertice qualsiasi  $v$ , trasformi il grafo in ingresso nell'albero radicato in  $v$  dei percorsi minimi che si dipartono da  $v$ . L'algoritmo deve risolvere il problema in tempo lineare sulla dimensione del grafo.

C

# Tema d'esame di Algoritmi e Strutture Dati I

15/07/2009

Tempo a disposizione: 3 ore.

1. Siano  $f$  e  $g$  due funzioni asintoticamente positive e crescenti. A seconda del caso, dimostrare la verità o la falsità delle seguenti affermazioni:

(i) se  $\sqrt{h(n)} = O(2^{f(n)^{p(n)}})$ , allora  $\log \log h(n) = \Theta(g(n) \cdot \log f(n))$ . (1)

(ii) se  $h(n) = \Theta(\max\{\log \log f(n), \log \log g(n)\})$ , allora  $g(n) = O(2^{(2^{h(n)})})$ .

2. Risolvete la seguente equazione di ricorrenza:

$$T(n) = \begin{cases} 1 & \text{se } n \leq 2 \\ T(n/3) + T(n/4) + n & \text{altrimenti} \end{cases}$$

✓

3. Si definisca un algoritmo ricorsivo efficiente che, noti un Albero Binario di Ricerca  $T$ , i valori  $k_{min}, k_{max}$  (con  $k_{min} \leq k_{max}$ ) e un intero  $x \geq 0$ , cancelli tutti i nodi con chiave pari compresa tra  $k_{min}$  e  $k_{max}$  che sono radici di sottoalberi di altezza non inferiore ad  $x$ . Notare che il rimbalzo sulle altezze dei sottosalberi va misurato rispetto all'albero originario, non ha quello eventualmente modificato. Non è ammesso l'uso di variabili globali, né del passaggio di parametri per riferimento.

4. Scrivere un algoritmo che dato in ingresso un vertice  $s$  e un grafo orientato  $G$ , rappresentato con liste di adiacenza, stampi tutti percorsi di  $G$  che raggiungono il vertice  $s$ .



✓

# Tema d'esame di Algoritmi e Strutture Dati I

## (Gruppo I)

22/01/2010

**Tempo a disposizione: 3 ore.**

1. Siano  $f(n)$  e  $g(n)$  due funzioni asintoticamente positive e crescenti. Dimostrare la verità o la falsità della seguente affermazione:

$$\log(f(n) \cdot g(n)) = O(\max\{\log f(n), \log g(n)\})$$

2. Risolvere la seguente equazione di ricorrenza:

$$T(n) = \begin{cases} 1 & \text{se } n \leq 1 \\ 7T(n/2) + n^3 \log n & \text{se } n > 1 \end{cases}$$

3. Scrivere un algoritmo ricorsivo efficiente che cancelli da un Albero Binario di Ricerca  $T$  (i cui nodi contengono solo il campo chiave, figlio destro e figlio sinistro) ogni nodo, diverso dalla radice dell'albero, che soddisfa la seguente proprietà:

“contiene una chiave pari minore di  $k$  ed è radice di un sottoalbero di altezza minore di  $H$ ”

dove  $H$  è un valore fornito in ingresso. Si noti che la proprietà dei nodi da cancellare è da intendersi rispetto all'albero originario  $T$  in ingresso. Non è ammesso l'uso di variabili globali né di passaggio di parametri per riferimento.

4. Dato un grafo orientato  $G$  e un vertice  $s$  di  $G$ , scrivere un algoritmo che in tempo lineare sulla dimensione del grafo, stampi, senza ripetizioni, tutti i vertici di  $G$  che o raggiungono  $s$  o sono da  $s$  raggiungibili.

# Tema d'esame di Algoritmi e Strutture Dati I

## 22/06/2011

**Tempo a disposizione: 3 ore.**

1. Dimostrare per esteso, ricavando le costanti necessarie, le relazioni asintotiche sotto riportate:

(a)  $n \log_2 n - 3n - 18 = \Omega(n)$ ;  
(b)  $3n^2 + 2n + 3 = \Theta(n^2 - 7)$ .

2. Si derivi, mostrando per esteso il procedimento seguito, l'equazione di ricorrenza per la funzione  $T(n)$  che descrive il tempo di esecuzione dell'algoritmo sotto riportato. Si risolva, poi, l'equazione, individuando la stima asintotica più vicina possibile a  $T(n)$ .

1 ALGO( $n$ )

```
1 if  $n \leq 2$  then return(0)
2 else
3   y = ALGO( $n/3$ )
4   i =  $2^n$ 
5   while  $i \geq 2$  do
6     j =  $\lfloor \frac{1}{2} \log_2 i \rfloor$ 
7     while  $j > 0$  do
8       i =  $\frac{i}{2}$ 
9     j = j - 1
10    z = ALGO( $n/2$ )
11    return (z+y)
```

3. Sia dato un albero binario di ricerca  $T$ , i cui nodi contengano esclusivamente una chiave intera, un puntatore al figlio sinistro e uno al figlio destro.

Si definisca un algoritmo ricorsivo efficiente che, dati i valori interi  $h_1 \geq 1$ ,  $h_2 \geq 1$ ,  $k_1 \geq 0$ ,  $k_2 \geq 0$ , cancelli dall'albero  $T$  tutti i nodi che, nell'albero originale fornito in ingresso, soddisfano la seguente proprietà:

*hanno chiave k pari tale che  $k_1 \leq k \leq k_2$  e sono radici di sottoalberi il cui percorso esterno ha lunghezza h che soddisfa  $h_1 \leq h \leq h_2$ .*

Si ricorda che la *lunghezza del percorso esterno* in un albero radicato nel nodo  $x$  è la somma delle lunghezze dei percorsi da  $x$  ad una foglia.

**Nota:** Non è possibile utilizzare parametri passati per riferimento né variabili globali.

4. Si supponga di voler disporre in una fila indiana  $n$  persone. Si ha a disposizione un insieme  $A$  di  $m$  affermazioni del tipo *la persona i detesta la persona j*. Se  $i$  detesta  $j$ , allora si vuole evitare di mettere  $i$  dietro a  $j$ . Definire un algoritmo che verifichi se è possibile disporre le persone in fila in modo che *nessuna persona stia davanti ad una persona che le detesta* e, in caso affermativo, produca una tale disposizione. L'algoritmo deve avere complessità  $O(n + m)$ .

# Tema d'esame di Algoritmi e Strutture Dati I

## 15/07/2011

**Tempo a disposizione: 3 ore.**

1. Dimostrare per esteso, ricavando le costanti necessarie, le relazioni asintotiche sotto riportate:

(a)  $n \cdot \log n - 10 \cdot n + 4 = \Omega(2 \cdot n \cdot \log n)$

(b)  $2 \cdot n^2 + 4 \cdot n + 3 = O(n^2 - n + 7)$ .

2. Si calcoli, svolgendo per esteso il ragionamento seguito, il tempo di esecuzione dell'algoritmo sotto riportato.

**1 ALGORITMO( $n$ )**

```
1   k = 1
2   t = 0
3   s = 1
4   while k ≤ n do
5       j = k
6       while j ≤ n do
7           s = j + k
8           j = 2 · j
9       t = t + 1
10      k = 2 · k
11  return s + t
```

3. Sia dato un albero binario di ricerca  $T$ , i cui nodi contengano esclusivamente una chiave intera, un puntatore al figlio sinistro e uno al figlio destro. Sia dato, inoltre, un array  $A$  contenente possibili chiavi intere ordinate in modo crescente.

Si definisca un algoritmo ricorsivo efficiente che cancelli dall'albero  $T$  tutti i nodi che sono a distanza  $h \geq 1$  dalla radice dell'albero fornito in ingresso e che contengono una chiave presente nell'array  $A$ .

**Nota:** Non è possibile utilizzare parametri passati per riferimento né variabili globali.

4. Sia dato un grafo orientato  $G = (V, E)$  e un insieme  $A \subseteq V$ . Si definisca la distanza di  $v \in V$  da  $A$  quel valore  $k$  tale che:

- $\delta(x, v) = k$  per qualche  $x \in A$  e
- $\delta(y, v) \geq k$  per ogni  $y \in A$ .

dove  $\delta(x, y)$  denota l'usuale funzione di distanza tra i vertici  $x$  e  $y$  nel grafo  $G$ . Si definisca un algoritmo che, in tempo  $O(|V| + |E|)$ , calcoli la distanza di un vertice  $v \in V$  dall'insieme  $A$ .

# Tema d'esame di Algoritmi e Strutture Dati I

## 22/09/2011

Tempo a disposizione: 3 ore.

1. Dimostrare per esteso, ricavando le costanti necessarie, le relazioni asintotiche sotto riportate:
  - (a)  $2n = O(n + 3 \log_2 n)$ ;
  - (b)  $3n^3 - 6n^2 + 8n = \Theta(n^3)$ .
2. Si calcoli, mostrando per esteso il procedimento seguito, il tempo di esecuzione dell'algoritmo sotto riportato.

I ALGO( $n$ )

```
1   i = 4
2   z = 0
3   while i < n do
4       j =  $\frac{i}{2}$ 
5       while j ≥ 1 do
6           i = i + 2
7           z = z + 1
8           j = j - 2
9   return (z)
```

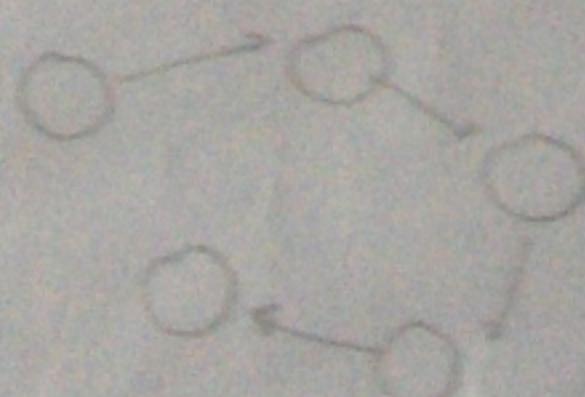
3. Sia dato un albero binario  $T$  (i cui nodi contengono esclusivamente una chiave intera, un puntatore al figlio sinistro e uno al figlio destro) che sia parzialmente ordinato, cioè tale che ogni nodo di  $T$  contiene una chiave non minore delle chiavi di entrambi i suoi figli.

Si definisca un algoritmo ricorsivo efficiente che, dati i valori interi  $k_1 \leq k_2$  e un valore intero  $x$ , cancelli dall'albero  $T$  tutti i nodi che, nell'albero originale  $T$  fornito in ingresso, soddisfano la seguente proprietà:

*hanno chiave k pari e tale che  $k_1 \leq k \leq k_2$ , e sono radici di sottoalberi contenenti almeno x nodi con chiave compresa tra  $k_1$  e  $k_2$ .*

Nota: Non è possibile utilizzare parametri passati per riferimento né variabili globali.

4. Sia dato un grafo  $G = (V, E)$  rappresentato tramite liste di adiacenza. Si definisca un algoritmo iterativo che verifichi se  $G$  è aciclico o no. In caso la risposta sia negativa, l'algoritmo deve fornire in output un possibile percorso contenente un ciclo. Il tempo impiegato dall'algoritmo deve essere  $O(|V| + |E|)$ .



# Tema d'esame di Algoritmi e Strutture Dati I

## 23/02/2012

**Tempo a disposizione: 3 ore.**

1. Dimostrare per esteso la verità o la falsità delle seguenti affermazioni:

(a)  $n^2 = \Omega(n^{\log_2 4/5} \cdot 5^{\log_2 n})$ ;

- (b) Si assuma che  $f$  e  $g$  siano funzioni asintoticamente crescenti e positive. Allora:

$$\log(f(n)) = \Theta(g(n)) \text{ implica } f(n) = \Theta(2^{g(n)})$$

2. Si calcoli, mostrando per esteso il procedimento seguito, il tempo di esecuzione dell'algoritmo sotto riportato.

```
1 ALGO(n)
1   y = 2^n
2   i = n
3   x = log_2 y
4   while x > 1 do
      y = y * 2^x
5   while y > 2^x do
6     i = i + 2
7     y = y/2
8     x = x/2
9   return i
```

3. Si definisca un algoritmo ricorsivo efficiente che, dati un albero binario di ricerca  $T$  (i cui nodi contengono esclusivamente un campo chiave, un campo figlio sinistro e un campo figlio destro), un intero positivo  $x > 0$  e un valore  $k$ , cancelli da  $T$  il nodo che soddisfa la seguente proprietà:

*contiene la più grande chiave minore di  $k$  che si trova in  $T$  a profondità non minore di  $x$ .*

**Nota:** non è ammesso l'utilizzo del passaggio di parametri per riferimento né l'impiego di variabili globali.

4. Si consideri un grafo  $G = (V, E)$  e un sottoinsieme  $A \subseteq V$  di vertici e si definisca distanza  $\delta(v, A)$  di un vertice  $v$  dall'insieme  $A$  come la minima tra le distanze del vertice  $v$  da un vertice di  $A$ . Formalmente,  $\delta(v, A) = \min\{\delta(v, a) \mid a \in A\}$ .

Dato un grafo  $G = (V, E)$  rappresentato tramite liste di adiacenza, un insieme  $A$  di vertici di  $G$  rappresentato come una lista puntata e due vertici  $u$  e  $v$ , si scriva lo pseudocodice di un algoritmo che, in tempo lineare sulla dimensione di  $G$  (quindi indipendentemente dalla dimensione del sottoinsieme  $A$ ) verifichi se i due vertici  $u$  e  $v$  sono alla stessa distanza da  $A$ .

# Tema d'esame di Algoritmi e Strutture Dati I

## 25/06/2012

**Tempo a disposizione: 3 ore.**

1. (a) Dimostrare per esteso la verità o la falsità della seguente affermazione:  
siano  $f$  e  $g$  due funzioni asintoticamente crescenti e positive e  $j$  e  $k$  due valori non negativi.  
Allora,  $f^k(n) = \Theta(2^{g^j(n)})$  implica  $\log_2 f(n) = \Theta(g(n))$ ;  
(b) Dimostrare per esteso, ricavando le costanti necessarie, la verità della seguente affermazione:

$$n \log_2 n - 3n - 18 = \Theta(n \log_2 n).$$

2. Si calcoli, mostrando per esteso il procedimento seguito, il tempo di esecuzione dell'algoritmo sotto riportato.

```
1 ALGO( $n$ )
1    $i = 1$ 
2   while  $i < n$  do
3      $j = \frac{1}{2}$ 
4     while  $j > 0$  do
5        $i = i + 2$ 
6        $j = j - 3$ 
7   return
```

3. Sia dato un Albero binario Parzialmente Ordinato  $T$ , in cui ogni nodo ha chiave non maggiore di quella dei suoi due figli. I nodi di  $T$  contengono esclusivamente un campo chiave, un campo figlio sinistro e un campo figlio destro. Si definisca un algoritmo ricorsivo efficiente che, dati tre valori interi strettamente positivi  $x$ ,  $k_1 \leq k_2$  e l'albero  $T$ , cancelli da  $T$  ogni nodo che soddisfa la seguente proprietà:

*contiene una chiave  $k$  di valore compreso tra  $k_1$  e  $k_2$  ed è radice di un albero che contiene al massimo  $x$  nodi con chiave compresa tra  $k_1$  e  $k_2$ .*

**Nota:** Non è possibile utilizzare parametri passati per riferimento né variabili globali.

4. Sia dato un grafo  $G = (V, E)$  rappresentato tramite liste di adiacenza e due sottoinsiemi di vertici  $A$  e  $B$ . Si definisca la distanza  $\delta(A, B)$  tra  $A$  e  $B$  nel seguente modo:

$$\delta(A, B) = \begin{cases} i & \text{se per ogni } a \in A \text{ esiste un } b \in B \text{ tale che } \delta(a, b) = i \\ \infty & \text{altrimenti.} \end{cases}$$

Si definisca un algoritmo che, dati in ingresso il grafo  $G$ , gli insiemi  $A$  e  $B$ , e un intero  $x$ , verifichi in tempo lineare sulla dimensione del grafo se  $\delta(A, B) = x$ .

# Tema d'esame di Algoritmi e Strutture Dati I

11/09/2012

1. Siano  $f$ ,  $g$ ,  $h$  e  $t$  funzioni asintoticamente positive e crescenti. Dimostrare la verità o la falsità della seguente affermazione:

$$\text{se } \log\left(\frac{f(n)}{t(n)}\right) = \Theta(\log g(n)) \text{ e } \frac{h(n)}{f(n)} = \Theta\left(\frac{t(n)}{g(n)}\right), \text{ allora } h(n) = \Theta(t^2(n)).$$

2. Sia dato il seguente algoritmo:

```
ALGORITMO( n )
    x = 2
    while x < n3 do
        y = x/2
        while y < x do
            y = y + 2
            x = x + 1
        x = x * 2
    return(x)
```

Si calcoli la stima asintotica *più vicina possibile* alla funzione  $T(n)$  che descrive il tempo di esecuzione dell'algoritmo.

3. Sia dato un albero binario di ricerca  $T$ , i cui nodi contengano esclusivamente una chiave intera, un puntatore al figlio sinistro e uno al figlio destro. Si definisca un algoritmo ricorsivo efficiente che, dati cinque valori interi  $h_1 \geq 1$ ,  $h_2 \geq 1$ ,  $n_1 \geq 0$ ,  $n_2 \geq 0$  e  $k$ , cancelli dall'albero  $T$  tutti i nodi che, nell'albero in ingresso, soddisfano la seguente proprietà:

*ha chiave pari minore di  $k$ , la sua distanza dalla radice è compresa tra i valori  $h_1$  e  $h_2$  ed è radice di un sottoalbero i cui nodi con chiave minore di  $k$  sono in numero compreso tra  $n_1$  e  $n_2$ .*

**Nota:** Non è possibile utilizzare parametri passati per riferimento né variabili globali.

4. Sia dato un grafo orientato  $G$ . Sia  $VAL[\cdot]$  un array che associa ad ogni vertice  $v$  un valore intero  $VAL[v]$ . Si definisca un algoritmo che, in tempo **lineare** sulla dimensione del grafo  $G$ , verifichi se esistono due vertici  $u$  e  $z$  tali che:  $VAL[u]$  sia pari,  $VAL[z]$  sia dispari e che siano tra loro reciprocamente raggiungibili.

# Tema d'esame di Algoritmi e Strutture Dati I

## 09/09/2013

**Tempo a disposizione: 3 ore.**

1. A) Si dimostri la verità o la falsità, esplicitando il procedimento seguito, della seguente affermazione: Se  $f(n) = \Theta(\sqrt{g(n)})$  e  $\log g(n) = \Theta(2^n)$ ; allora  $\log f(n) = \Theta(\log n)$ .  
B) Si dimostri o si confuti la seguente affermazione, esplicitando il procedimento seguito e le costanti necessarie, se esse esistono:  $n^2 - n + \frac{1}{n} = \Theta(n^2)$ .
2. Calcolare, **esplicitando il procedimento completo di soluzione seguito**, il tempo di esecuzione del seguente algoritmo in funzione del parametro  $n$ .

**1 Algoritmo( $n$ )**

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

3. Sia dato un albero binario  $T$  (i cui nodi contengono esclusivamente un campo chiave intero, un campo figlio sinistro e un campo figlio destro).

Scrivere un algoritmo ricorsivo efficiente che elimini da  $T$  tutti i nodi che contengono una chiave pari e, contemporaneamente, costruisca un albero binario di ricerca  $T'$  contenente tutti i nodi eliminati da  $T$ . L'algoritmo richiesto deve restituire l'albero  $T'$  e non può avere  $T'$  tra i suoi parametri di ingresso.

**Non è ammesso l'uso di passaggio di parametri per riferimento né l'impiego di variabili globali.**

4. 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, in tempo lineare sulla dimensione di  $G$ , verifichi se esiste un percorso di  $G$  della forma  $v_1 \rightarrow v_2 \rightarrow \dots \rightarrow v_k \rightarrow v_1$  tale che  $A \subseteq \{v_1, \dots, v_k\}$ .

# Tema d'esame di Algoritmi e Strutture Dati I

26/06/2014

Tempo a disposizione: 3 ore.

Si dimostri per esteso la verità o la falsità di ciascuna delle seguenti affermazioni:

- A) Si dimostri, esplicitando per esteso il procedimento seguito, la verità o la falsità della seguente affermazione: Se  $f(n) = \Theta(\log_2 g(n))$  e  $h(n) = \Theta(2^n)$ , allora  $\log_2 h(f(n)) = \Theta(\log_2 g(n))$ . Si assuma che le funzioni  $f$  e  $g$  siano asintoticamente crescenti e positive.
- B) Calcolare, se esistono, le costanti necessarie a dimostrare la seguente affermazione:  $e^{n-1} - 2^n = \Theta(e^n)$  (si espliciti per esteso il procedimento seguito).

2. Sia dato il seguente algoritmo:

1 Algoritmo( $n$ )

```
1  x = 2n
2  y = 0
3  while x > 2 do
4      z = y + 1
5      y = log2x
6      while y > 0 do
7          y = y - 2
8          z = z * 2
9      x = x/z
10 return
```

Calcolare, esplicitando tutto il procedimento di soluzione seguito, la stima asintotica più vicina possibile al tempo di esecuzione  $T(n)$  dell'algoritmo.

C) Sia dato un Albero Binario di Ricerca  $T$ , i cui nodi contengono i seguenti 4 campi: (i) un campo chiave di tipo intero; (ii) due campi uno per figlio sinistro e uno per figlio destro; e (iii) un campo addizionale di tipo intero, che associa ad ogni nodo il numero di foglie contenute nel sottoalbero di cui esso è radice. Siano, inoltre, dati due interi  $k_1$  e  $k_2$  (con  $k_1 \leq k_2$ ) e due interi  $x > 0$  e  $h > 0$ . Definire un algoritmo ricorsivo efficiente che cancelli da  $T$  tutti nodi che, nell'albero iniziale  $T$  in ingresso, soddisfano la seguente condizione:

"hanno chiave pari compresa tra  $k_1$  e  $k_2$ , hanno profondità minore o uguale ad  $h$  e il percorso esterno del sottoalbero di cui sono radici è maggiore o uguale a  $x$ ."

L'albero risultante deve essere dello stesso tipo dell'albero in ingresso (si faccia, quindi, attenzione a calcolare correttamente i valori del quarto campo dei nodi).

Non è ammesso l'uso di passaggio di parametri per riferimento né l'impiego di variabili globali.

D) Sia dato un arbitrario grafo orientato  $G = (V, E)$  e un vertice  $v \in V$ . Un grafo  $G' = (V, E')$  di  $G$  si dice "sottografo massimale di  $G$  che ammette almeno un ordinamento topologico che parte da  $v$ " se: (i)  $E' \subseteq E$  e (ii) per ogni  $(u, u') \in E \setminus E'$ , il grafo  $(V, E' \cup \{(u, u')\})$  non ammette un ordinamento topologico che parte da  $v$ .

Si scriva un algoritmo che, in tempo lineare sulla dimensione di  $G$ , costruisca un sottografo massimale  $G'$  di  $G$  che ammette almeno un ordinamento topologico che parte da  $v$  (il grafo  $G'$  risultante deve essere un nuovo grafo, cioè un grafo distinto da  $G$ ).

# Tema d'esame di Algoritmi e Strutture Dati I

## 18/07/2014

**Tempo a disposizione: 3 ore.**

1.  A) Si dimostri la verità o la falsità (tramite controesempio) della seguente affermazione: Se  $z(n) = \Theta(\log n^3)$  e  $g(n) = \Theta(\log(\log n)^3)$ , allora  $\log z(n) = \Theta(2^{g(n)})$ .  
B) Si individuino, se esistono, le costanti necessarie a dimostrare la seguente relazione:  
 $\log\left(\frac{n}{10}\right) = \Theta(\log n^5)$
2. Sia dato il seguente algoritmo:

1 Algoritmo( $n$ )

```
1    $x = n$ 
2    $j = \frac{x}{2}$ 
3   while  $x > 1$  do
4      $z = j$ 
5     while  $j > 1$  do
6        $j = \frac{j}{4}$ 
7        $x = x - 2$ 
8        $x = \frac{2^{(2z)}}{2^x}$ 
9      $j = \frac{x}{2}$ 
10  return
```

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

3.  Sia dato un albero binario parzialmente ordinato  $T$  (in cui i figli di ogni nodo hanno chiave non maggiore di quella del nodo stesso).

- A) scrivere un algoritmo ricorsivo efficiente che, dato in ingresso  $T$  e un valore  $k > 0$ , restituisca la **lista ordinata delle chiavi pari e minori di  $k$**  contenute nell'albero ( $T$  non deve essere necessariamente preservato).  
B) come modifichereste la procedura del punto A) se venisse richiesto di risolvere il problema senza modificare l'albero originario  $T$ ?

**Non è ammesso l'uso di passaggio di parametri per riferimento né l'impiego di variabili globali.**

4. 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 che abbia cardinalità  $\geq 1$ .

Si scriva un algoritmo che, in tempo lineare sulla dimensione del grafo orientato  $G = \{V, E\}$  fornito in ingresso, verifichi se il grafo in ingresso è una foresta orientata oppure no.

# Tema d'esame di Algoritmi e Strutture Dati I

## 17/09/2014

**Tempo a disposizione: 3 ore.**

1. Per i seguenti esercizi si riporti per esteso il procedimento di soluzione seguito:

- A) Si dimostri la verità o la falsità (tramite controesempio) della seguente affermazione:  
Se  $z(n) = \Theta(n^5)$  e  $g(n) = \Theta(\log(\log n)^2)$ , allora  $\log z(n) = \Theta(2^{g(n)})$ ;  
B) Si individuino, se esistono, le costanti necessarie a dimostrare la seguente relazione:  
 $2n \log n + n - 8 \log n = \Theta(n \log n)$ .

2. Sia dato il seguente algoritmo:

```
1 Algoritmo( $n$ )
1    $x = 2^n$ 
2   while  $x > 2$  do
3      $y = 0$ 
4     while  $\log x > y$  do
5        $y = y + 1$ 
6        $x = \frac{x}{2}$ 
7      $z = \frac{n}{x}$ 
8   return Algoritmo( $z$ )
```

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

3. Sia dato un Albero Binario di Ricerca  $T$  a chiavi intere (in cui i campi di ogni nodo prevedono solo la chiave e i puntatori ai due figli) e due valori interi  $k_1$  e  $k_2$ . Scrivere un algoritmo **iterativo** che cancelli da  $T$  tutti i nodi che abbiano chiave  $k$  tale che  $k_1 \leq k \leq k_2$ .

**È ammesso l'uso di chiamate di procedura ma non di chiamate ricorsive. Non è ammesso l'uso di passaggio di parametri per riferimento né l'impiego di variabili globali.**

4. Un grafo orientato  $G = \langle V, E \rangle$  si dice **completo** se ogni suo vertice è adiacente ad ogni altro vertice. Si ricorda, inoltre, che se  $V' \subseteq V$ , il sottografo di  $G$  **indotto** da  $V'$  è il massimo sottografo di  $G$  che ha  $V'$  come insieme dei vertici.

Siano dati in ingresso un grafo orientato  $G = \langle V, E \rangle$  rappresentato come matrice di adiacenza e una sequenza di vertici distinti  $V' \subseteq V$  ( contenuta in un array di lunghezza  $|V'|$ ).

- A) si scriva un algoritmo che verifichi, in tempo  $O(|V'|^2)$ , se il sottografo di  $G$  indotto da  $V'$  è completo oppure no;
- B) si scriva un algoritmo che in tempo  $O(|V'|^3)$  calcoli la più lunga **sottostringa**  $V''$  di  $V'$  (cioè la massima sottosequenza di elementi contigui nella sequenza  $V'$ ) tale che il sottografo di  $G$  indotto da  $V''$  è completo.

# Tema d'esame di Algoritmi e Strutture Dati I

## 29/01/2015

**Tempo a disposizione: 3 ore.**

1. X A) Si dimostri per esteso la verità o la falsità (tramite conto esempio) della seguente affermazione:

$$\text{Se } g(n) \cdot h(n) = \Theta(n^3) \text{ e } \frac{f(n)}{g(n)} = \Theta(n), \text{ allora } \log(f(n)) + \log(h(n)) = \Theta(\log(n)).$$

Si assuma che tutte le funzioni siano asintoticamente crescenti e positive.

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

$$n^2 \log(n^2) + 15n^2 = \Theta(n^2 \log(n)).$$

Si fornisca per esteso il procedimento di dimostrazione seguito.

2. Sia dato il seguente algoritmo:

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

Calcolare, esplicitando il procedimento di soluzione seguito, la stima asintotica più vicina possibile al tempo di esecuzione  $T(n)$  dell'algoritmo.

- X 3. Sia dato un albero binario di ricerca  $T$  (i cui nodi contengono esclusivamente un campo chiave di tipo intero, un campo figlio sinistro e un campo figlio destro) e due interi  $k_1$  e  $k_2$  (con  $k_1 \leq k_2$ ). Definire un algoritmo ricorsivo efficiente che cancelli da  $T$  tutti e soli i nodi a profondità massima con chiave compresa tra  $k_1$  e  $k_2$ . Più precisamente, l'algoritmo deve cancellare tutti e soli i nodi che soddisfano **entrambe** le seguenti condizioni:

- (a) hanno chiave pari e compresa tra  $k_1$  e  $k_2$ ;
- (b) non hanno alcun discendente che soddisfi la condizione (a).

Non è ammesso l'uso di passaggio di parametri per riferimento né l'impiego di variabili globali.

4. Diciamo che due vertici  $v$  e  $u$  di un grafo orientato  $G = \langle V, E \rangle$  sono *separati* da un insieme  $X \subseteq V$ , se ogni possibile percorso da  $v$  a  $u$  passa necessariamente per un vertice di  $X$  e, viceversa, ogni percorso da  $u$  a  $v$  passa necessariamente per un vertice di  $X$ . Si scriva un algoritmo che, in **tempo lineare sulla dimensione del grafo** orientato  $G$ , fornito in ingresso e rappresentato mediante liste di adiacenza, verifichi se due vertici in ingresso  $u$  e  $v$  sono separati dall'insieme di vertici  $X \subseteq V$ , anch'esso dato in ingresso come array di vertici.

# Tema d'esame di Algoritmi e Strutture Dati I

## 20/02/2015

**Tempo a disposizione: 3 ore.**

1. A) Si dimostri per esteso la verità o la falsità (tramite controesempio) della seguente affermazione: Se  $z(n) = \Theta(2^{g(n)})$  e  $h(n) = \Theta(\log g(n))$ , allora  $\log z(n) = \Theta(2^{h(n)})$ .  
Si assuma che le funzioni  $g$ ,  $z$  e  $f$  siano asintoticamente crescenti e positive.
- B) Si individuino, se esistono, le costanti necessarie a dimostrare la seguente relazione:  
 $5n^2 - 8\sqrt{n} + 1 = \Theta(n^2)$ . Il procedimento di soluzione deve essere riportato per esteso.

2. Sia dato il seguente algoritmo:

```
1 Algoritmo( $n$ )
2    $x = n$ 
3   while  $x > 4$  do
4      $z = 3 * \sqrt{x}$ 
5      $j = 1$ 
6     while  $j > \sqrt{x}$  do
7        $j = j * 2$ 
8        $z = z - j$ 
9      $x = \frac{x}{(z+1)}$   $\frac{x}{2-2}$ 
10    return
```

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

3. Sia dato un albero binario di ricerca  $T$  (i cui nodi contengono esclusivamente un campo chiave intero, un campo figlio sinistro e un campo figlio destro).

Scrivere un algoritmo **iterativo** efficiente che costruisca una lista ordinata (in ordine crescente), contenente tutti i nodi di  $T$  le cui chiavi siano contenute nell'intervallo di interi  $[k_1, k_2]$  e che si trovino a profondità comprese nell'intervallo di interi  $[h_1, h_2]$  (dove  $T, k_1, k_2, h_1$  e  $h_2$  sono tutti parametri di input dell'algoritmo).

**Non è ammesso l'uso di passaggio di parametri per riferimento né l'impiego di variabili globali.**

4. Sia dato grafo orientato  $G = \{V, E\}$ , rappresentato con liste di adiacenza, un insieme di nodi  $X \subseteq V$  memorizzato in un array e un intero non negativo  $k$ .  
Si scriva un algoritmo che, in tempo lineare sulla dimensione di  $G$ , raccolga in una lista  $L$  i nodi di  $G$  a partire dai quali **tutti i percorsi** che raggiungono un nodo di  $X$  hanno lunghezza maggiore di  $k$ .

# Tema d'esame di Algoritmi e Strutture Dati I

## 25/06/2015

**Tempo a disposizione: 3 ore.**

1. Si svolgano i seguenti esercizi:

(a) Si individuino, se esistono, le costanti necessarie a dimostrare la seguente relazione:  
 $\log\left(\frac{2^n}{n}\right) - 1 = \Theta(\log(2^{2n}))$ .

(b) Si dimostri la verità o la falsità (tramite controesempio) della seguente affermazione: Se  $z(n) = \Theta(2^{g(n)})$  e  $h(n) = \Theta(\log \log g(n))$ , allora  $\log \log z(n) = \Theta(2^{h(n)})$ .

Si assuma che le funzioni  $g$ ,  $z$  e  $f$  siano asintoticamente crescenti e positive.

2. Si consideri il seguente algoritmo:

I Algoritmo( $n$ )

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

Si individui la stima asintotica più vicina possibile al tempo di esecuzione dell'algoritmo in funzione del parametro  $n$ .

3. Sia dato un albero binario di ricerca  $T$ , i cui nodi contengano esclusivamente una chiave intera, un puntatore al figlio sinistro e uno al figlio destro.

Si definisca un **algoritmo iterativo** efficiente che, dato il riferimento all'albero  $T$  e un valore intero  $k$ , restituisca il riferimento al nodo di  $T$  contenente la più piccola chiave strettamente maggiore di  $k$ .

4. Sia dato un grafo orientato  $G$  e un array  $VAL[\cdot]$  che associa a ogni vertice  $v$  un numero naturale  $VAL[v]$ . Si definisca un algoritmo che, in tempo lineare sulla dimensione del grafo  $G$ , verifichi se esistono due vertici della stessa parità (cioè i cui numeri naturali associati siano o entrambi pari o entrambi dispari) tali che entrambe le seguenti condizioni valgano:

- nel grafo esiste almeno un ciclo che contiene sia il primo che il secondo vertice;
- ogni percorso che connette i due vertici passa solo per vertici della loro stessa parità.

# Tema d'esame di Algoritmi e Strutture Dati I

## 11/09/2015

Tempo a disposizione: 3 ore.

1. Si svolgano i seguenti esercizi, esplicitando lo svolgimento completo:

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

$$5n^2 - n + \log n = \Theta(n^2)$$

- b) Si dimostri la verità o la falsità (tramite controesempio) della seguente affermazione:

se  $h(n) = \Theta(2^n)$  e  $\log_2 h(f(n)) = \Theta(\log_2 g(n))$ , allora  $f(n) = \Theta(\log_2 g(n))$ .

Si assuma che le funzioni  $g$  e  $f$  siano asintoticamente crescenti e positive.

2. Si consideri il seguente algoritmo:

```
1 Algoritmo( $n$ )
2    $x = 2^n$ 
3    $y = x^2$ 
4   while  $x > 2$  do
5      $z = 1$ 
6     while  $y > z$  do
7        $y = \frac{y}{2}$ 
8        $z = z * 2$ 
9   return
```

Si individui, esplicitando lo svolgimento completo, la stima asintotica più vicina possibile al tempo di esecuzione dell'algoritmo in funzione del parametro  $n$ .

3. Sia dato un albero binario di ricerca  $T$ , i cui nodi contengano esclusivamente una chiave intera, un puntatore al figlio sinistro e uno al figlio destro.

Si definisca un **algoritmo iterativo** efficiente che, dato il riferimento all'albero  $T$  e un valore intero  $k$ , cancelli da  $T$  il nodo contenente la più piccola chiave pari maggiore di  $k$ .

4. Siano dato un grafo orientato  $G = (V, E)$ , e una permutazione  $\pi$  (in forma di array) di un qualche sottoinsieme  $V' \subseteq V$ . Si definisca un algoritmo che, in tempo lineare sulla dimensione del grafo  $G$ , verifichi se  $\pi$  è un ordinamento topologico del sottografo di  $G$  indotto da  $V'$ , senza costruire il sottografo indotto.

# Tema d'esame di Algoritmi e Strutture Dati I

## 28/01/2016

**Tempo a disposizione: 3 ore.**

1. A) 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  $\log f(n) = \Theta(\log n)$ .  
B) Si individuino, se esistono, le costanti necessarie a dimostrare la seguente relazione:  
$$\log\left(\frac{n}{\pi}\right) = \Theta(\log n^4)$$
2. Calcolare, esplicitando il procedimento completo di soluzione seguito, il tempo di esecuzione del seguente algoritmo in funzione del parametro  $n$ .

1 Algoritmo( $n$ )

```
1   x = 2n
2   y = 0
3   while x > 2 do
4       z = y + 1
5       y = log2x
6       while y > 0 do
7           y = y / 2
8           z = z * 2
9       x = x / z
10  Algoritmo(n / x)
11  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).

Scrivere un algoritmo ricorsivo efficiente che cancelli dall'albero  $T$  tutti i nodi che stiano a profondità compresa tra i valori  $l_1 \geq 1$  e  $l_2$  (dati in ingresso), le cui chiavi siano pari e si trovino in posizione multipla di 7 nell'ordinamento totale delle chiavi. Tutte le condizioni da verificare si riferiscono all'albero originario fornito in ingresso.

**Non è ammesso l'uso di passaggio di parametri per riferimento né l'impiego di variabili globali.**

4. Sia dato grafo orientato  $G = (V, E)$ , 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, in tempo lineare sulla dimensione di  $G$ , verifichi se esiste un percorso di  $G$  della forma  $v_1 \rightarrow v_2 \rightarrow \dots \rightarrow v_k \rightarrow v_1$  tale che  $A \subseteq \{v_1, \dots, v_k\}$ .

# Tema d'esame di Algoritmi e Strutture Dati I

## 19/02/2016

**Tempo a disposizione: 3 ore.**

1. A) Si dimostri la verità o la falsità, esplicitando il procedimento seguito, della seguente affermazione: Se  $h(n) = \Theta(t(n))$  e  $f(n)/h(n) = \Theta(g(n))$ , allora  $\frac{f^2(n)}{h(t) \cdot t(n)} = \Theta(g^2(n))$ .

- B) Si individuino, se esistono, le costanti necessarie a dimostrare la seguente relazione:  $4n^2 - 7\sqrt{n} + 2 = \Theta(n^2)$ . 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 Algoritmo( $n$ )

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

Scrivere un algoritmo **iterativo** efficiente che cancelli dall'albero  $T$  tutti i nodi che stanno a profondità compresa tra i valori  $l_1$  e  $l_2$  (dati in ingresso) e le cui chiavi siano pari e comprese tra i valori  $k_1$  e  $k_2$  (anch'essi dati in ingresso). Tutte le condizioni da verificare si riferiscono sempre all'albero originario fornito in ingresso.

**Non è ammesso l'uso di passaggio di parametri per riferimento né l'impiego di variabili globali.**

4. 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 due vertici  $u, v \in V$ , 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  $u \rightsquigarrow a \rightsquigarrow v$ .

# 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, **esplicitando per esteso** il procedimento seguito:

se  $h(n) = \Theta(t(n))$  e  $f(n) = \Theta(g(n))$ , allora  $g(n) + h(n) = \Theta(t(n) + f(n))$ .

- B) Si individuino, se esistono, le costanti necessarie a dimostrare la seguente relazione:  $2n - \log_2\left(\frac{n}{4}\right) = \Theta(n)$ . 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 paramentro  $n$ .

```
1 ALGO(n)
1   x = n3
2   while x > 1 do
3     y = x/3
4     while y < x do
5       y = y + 3
6       x = x - 2
7     x = x/2
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 passaggio di parametri per riferimento né l'impiego di variabili globali**.

4. Dato un insieme di etichette  $P = \{p_1, p_2, \dots, p_q\}$ , un *grafo orientato etichettato* rispetto a  $P$  è definito come una tripla  $G = \langle V, E, L \rangle$ , dove  $\langle V, E \rangle$  è un grafo orinetato (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 orinetato etichettato  $G$  rispetto a  $P$ , un vertice  $u \in V$ , e una etichetta  $p \in P$ , verifichi 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 dimostri la verità o la falsità della seguente affermazione, **esplcitando per esteso il procedimento seguito:**  
se  $h(n) = \Theta(t(n))$  e  $f(n) = \Theta(g(n))$ , allora  $\log_2(g(n) \cdot h(n)) = \Theta(\log_2(t(n) \cdot f(n)))$ .  
B) Si individuino, se esistono, le costanti necessarie a dimostrare la seguente relazione:  
 $2\log_2 n - \frac{4}{n} = \Theta(\log_2 n)$ . Il procedimento di soluzione deve essere riportato per esteso.
2. Calcolare, **esplcitando il procedimento completo di soluzione seguito**, il tempo di esecuzione del seguente algoritmo in funzione del parametro  $n$ .

```
1 ALGO(n)
1   x = 2^n
2   while x > 2 do
3     y = x/2
4     z = 1
5     while x - 1 > y + 1 do
6       x = x - z
7       z = 2 * z
8     x = (x-2)/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 **ricorsivo 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**  $[k_1, k_2]$ , sfruttando 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 nel sottoalbero 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** né l'impiego di variabili globali.

4. Dato un *grafo orientato*  $G = \langle V, E \rangle$ , rappresentato con liste di adiacenza. Si scriva un algoritmo che, dati in ingresso il grafo orinetato  $G$ , un numero naturale  $k$  e tre vertici  $u, z, v \in V$ , 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

09/09/2016

Tempo a disposizione: 3 ore.

1. ✓ A) Dimostrare la verità o la falsità della seguente affermazione, assumendo la funzione  $g$  monotonicamente crescente e positiva:

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

- B) Verificare, riportando il **procedimento completo**, se la seguente affermazione è vera e, in caso affermativo, esibire le costanti necessarie a verificare la relazione:

$$n - \log n + 1 = \Theta(n)$$

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

I Algoritmo( $n$ )

```
1    $x = n$ 
2   while  $x > 4$  do
3        $z = 3 * \sqrt{x}$ 
4        $j = 1$ 
5       while  $j > \sqrt{x} + \frac{1}{2}$  do
6            $z = z - j$ 
7            $j = 3 * j$ 
8        $x = \frac{z}{z-j}$ 
9   return
```

- ✓ 3. Sia dato un **generico albero binario**  $T$  (i cui nodi contengono esclusivamente un campo chiave, un campo figlio sinistro e un campo figlio destro).

- ✓ (a) Scrivere un algoritmo **iterativo** che in **tempo lineare** sul numero di nodi di  $T$  crei una copia esatta (cioè isomorfa)  $T'$  dell'albero in  $T$  ingresso. L'algoritmo deve ricevere in ingresso esclusivamente il riferimento alla radice dell'albero  $T$  e restituire il nuovo albero  $T'$ .
- ~ (b) Si discuta la complessità dell'algoritmo proposto, mostrando che risolve il problema nel tempo richiesto.

**Non è ammesso l'uso di passaggio di parametri per riferimento né l'impiego di variabili globali.**

4. Sia un dato grafo orientato  $G = (V, E)$  e quattro nodi  $u, v, k, z \in V$ .

- ✓ (a) Si scriva un algoritmo che verifichi in tempo  $O(|V| + |E|)$  se ogni percorso  $\pi$ , non necessariamente semplice, che va da  $v$  a  $u$  soddisfa la seguente condizione:
- se  $\pi$  passa per il vertice  $k$  allora passa (prima o dopo  $k$ ) anche per il vertice  $z$ .
- (b) Può l'algoritmo proposto per il punto (a) risolvere anche la versione dello stesso problema in cui è richiesto di considerare solo percorsi semplici? Si motivi la risposta.

# 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:  
Se  $\log_2 \frac{f(n)}{t(n)} = \Theta(\log_2(t(n) \cdot g(n)))$  e  $\log_2 \frac{t(n)}{g(n)} = \Theta\left(\log_2 \frac{h(n)}{f(n)}\right)$ , allora  $\log_2 h(n) = \Theta(\log_2 t(n))$ .  
B) Si individuino, se esistono, le costanti necessarie a dimostrare la seguente relazione asintotica:  
 $7n\sqrt{n} + 3n - 10\sqrt{n} = \Theta(n^{\frac{3}{2}})$ . 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   $x = 2^n$ 
2   $z = n$ 
3  while  $z < x$  do
4       $y = 2z$ 
5      while  $y > 2$  do
6           $y = y/2$ 
7           $z = z + y$ 
8       $z = z + 1$ 
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       $r_{sx} = \text{COUNTODDRIC}(T \rightarrow sx)$ 
6       $r_{dx} = \text{COUNTODDRIC}(T \rightarrow dx)$ 
7       $ret = ret + r_{sx} + r_{dx}$ 
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

26/01/2017

## Tema 2

1. A) Si dimostri la verità o la falsità, esplicitando il procedimento seguito, della seguente affermazione:  
Se  $\log_2 \frac{f(n)}{t(n)} = \Theta(\log_2(t(n) \cdot g(n)))$  e  $\log_2 \frac{t(n)}{g(n)} = \Theta\left(\log_2 \frac{h(n)}{f(n)}\right)$ , allora  $\log_2 h(n) = \Theta(\log_2 t(n))$ .  
B) Si individuino, se esistono, le costanti necessarie a dimostrare la seguente relazione asintotica:  
 $7n\sqrt{n} + 3n - 10\sqrt{n} = \Theta(n^{\frac{3}{2}})$ . 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   $x = 2^n$ 
2   $z = n$ 
3  while  $z < x$  do
4       $y = 2z$ 
5      while  $y > 2$  do
6           $y = y/2$ 
7           $z = z + y$ 
8       $z = z + 1$ 
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       $r_{sx} = \text{COUNTODDRIC}(T \rightarrow sx)$ 
6       $r_{dx} = \text{COUNTODDRIC}(T \rightarrow dx)$ 
7       $ret = ret + r_{sx} + r_{dx}$ 
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 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          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 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  $a$ .

Domenico Pirone N86001906

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

## Traccia 2

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

1. 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.

2. 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.

3. 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:

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

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( $T, l$ )

```
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.

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

## Traccia 2

Tempo a disposizione: 2 ore e 30 minuti.

1. Sia data la seguente equazione di ricorrenza:

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

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( $T, l$ )**

```
1    $X = Nil$ 
2   if  $T \neq Nil$  and  $l \geq 0$  then
3       if  $T \rightarrow Key \% 2 = 0$  then
4            $X = ALLOCATE\_NODO()$ 
5            $X \rightarrow Sr = ALGO(T \rightarrow Sr, l - 1)$ 
6            $X \rightarrow Dr = ALGO(T \rightarrow Dr, l - 1)$ 
7            $X \rightarrow Key = T \rightarrow Key$ 
8   return  $X$ .
```

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

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

3. Dati in ingresso un grafo  $G = \langle V, E \rangle$ , e tre vertici  $s$ ,  $t$  e  $u$ , si vuole verificare, in tempo lineare sul grafo  $G$ , se esiste un percorso in  $G$  che parte da  $s$ , passa per  $t$  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    $y = A[j]$  metto
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  $= ret + x$ 
8       if  $q + 1 < j$  then
9            $y = \text{ALGO}(A, q + 1, j)$ 
10      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.

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

## Traccia 12

Tempo a disposizione: 2 ore e 30 minuti.

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

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

Esplcitare 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  $T \neq Nil$  then
3        $x = T \rightarrow k \% 2$ 
4        $y = ALGO(T \rightarrow sx, k)$ 
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       $L_D = ALGO(T \rightarrow dx, L)$ 
4      if  $p = 0$  then
5           $x = NEW-NODE(T \rightarrow k)$ 
6           $L = ALGO(T \rightarrow sx, x)$ 
7           $x \rightarrow next = L_D$ 
8      else
9           $L = ALGO(T \rightarrow sx, L_D)$ 
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  $ALGO(T, Nil)$ .
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.**

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

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

2. 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.

3. 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.**

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

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

- 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
6            $a = \text{CHECKBST}(T, x, val - 1)$ 
7           if  $a \neq 0$  then
8                $b = \text{CHECKBST}(T, val + 1, y)$ 
9       else
10       $a = 0$ 
11       $ret = a \wedge b$ 
12      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 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 ≠ NIL then
5       if T→key > k then
6           a = ALG(T→sx,k)
7           b = T
8       else if T→key < k then
9           a = T
10          b = ALG(T→dx,k)
11      else
12          a = ALG(T→sx,k)
13          b = ALG(T→dx,k)
14      ret = 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

19/02/2019

Traccia A

Tempo a disposizione: 2,30 ore.

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

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

2. Sia dato il seguente algoritmo ricorsivo.

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

```
1   val = 0
2   if  $x \leq y$  then
3        $x = \frac{x+2}{2}$ 
4       if  $A[x] = k$  then
5           val = 1
6        $a = \text{ALG}(A, x, x-1, k)$ 
7       if  $a > val$  then
8           val =  $a + val$ 
9        $b = \text{ALG}(A, x+1, y, k)$ 
10      val = 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, verifica, 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 29/03/2019

## Traccia A

**Tempo a disposizione: 2.30 ore.**

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

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

- 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  $u \rightsquigarrow a \rightsquigarrow v$ .

# 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        $L' =$  alloca nodo
4        $L' \rightarrow key = A[q]$  (Q è primo della f)
5        $q = \left\lfloor \frac{p+r}{2} \right\rfloor$ 
6       if  $A[q] \% 2 = 0$  then
7            $L' \rightarrow next = ALGO(A, q+1, r, L)$ 
8           ret = ALGO( $A, p, q-1, L'$ )
9       else
10           $L' \rightarrow next = ALGO(A, p, q-1, L)$ 
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  $\mathcal{A}(T, k_1, k_2)$

```
1 if  $T \neq Nil$  do
2    $\neg L' = ALGORITMO(T \rightarrow f_1, k_1, k_2, L)$ 
3    $L' = ALGORITMO(T \rightarrow f_2, k_1, k_2, L')$ 
4    $L' = ALGORITMO(T \rightarrow f_3, k_1, k_2, L')$ 
5   if  $k_1 \leq T \rightarrow key \leq k_2$  do
6      $L = ALLOCA\_NODO()$ 
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  $z$ .

**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.**

1. 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  $g(n) = \Theta(k(n)^4)$ .

2. 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           $y = \text{ALGO}(A, q + 1, j)$   $\pi_{\mathcal{K}} = X \cap Y$ 
11      else
12          ret = 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.

3. 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

## (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

17/06/2020

**Tempo a disposizione: 1 ora.**

Sia dato il seguente algoritmo ricorsivo:

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

```
1   ret = 0
2   if ( $p \leq r$ ) then
3        $q = \lfloor \frac{p+r}{2} \rfloor$ 
4       if ( $k == A[q]$ ) then
5           ret =  $A[q]$ 
6       ret = ALGO( $A, q + 1, r, k$ ) + ret
7       if  $ret > 0$  then
8           ret = ret + ALGO( $A, p, q - 1, k$ )
9   return ret
```

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

# Tema d'esame di Algoritmi e Strutture Dati I

07/07/2020

**Tempo a disposizione: 1 ora.**

Traccia A

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

Traccia B

```
algo (A, i, j):
    ret = i
    if (i <= j):
        k = i+1
        while (k <= j) and A[k] <= A[i]:
            k = k+1
        ret = algo (A, i, (i+j) / 2)
        if k % 2 == 1:
            ret = algo (A, k/2, j) +ret-k
        else:
            ret = algo (A, k, j) +ret-k/2
    return ret
```

# 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  $C$ .

# Tema d'esame di Algoritmi e Strutture Dati I

## 13/04/2021

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

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

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

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

# Algoritmi e Strutture Dati I

## Traccia A 16/06/2021

Si definisca un **albero orientato** come un grafo orinetato con le seguenti proprietà: (i) esiste un unico vertice  $s$  (detto radice) che raggiunge ogni altro vertice tramite un qualche percorso orientato; (ii) se un qualsiasi vertice  $v$  è raggiungibile da un arbitrario vertice  $u$ , allora **esiste un solo percorso orientato da  $u$  a  $v$** . Una **foresta orientata** è, allora, un grafo orientato formato da uno o più alberi orientati tra loro non connessi.

Si scriva un algoritmo che, in **tempo lineare sulla dimensione del grafo** orientato  $G = (V, E)$  fornito in ingresso, verifichi se  $G$  è una **foresta orientata** oppure no.

# Algoritmi e Strutture Dati I

## Traccia A 16/06/2021

Si definisca un **albero orientato** come un grafo orinetato con le seguenti proprietà: (i) esiste un unico vertice  $s$  (detto radice) che raggiunge ogni altro vertice tramite un qualche percorso orientato; (ii) se un qualsiasi vertice  $v$  è raggiungibile da un arbitrario vertice  $u$ , allora **esiste un solo percorso orientato da  $u$  a  $v$** . Una **foresta orientata** è, allora, un grafo orientato formato da uno o più alberi orientati tra loro non connessi.

Si scriva un algoritmo che, in **tempo lineare sulla dimensione del grafo** orientato  $G = (V, E)$  fornito in ingresso, verifichi se  $G$  è una **foresta orientata** oppure no.

# Algoritmi e Strutture Dati I

## Traccia B 16/06/2021

Scrivere un algoritmo **iterativo** che **simuli precisamente** il comportamento ricorsivo dell'algoritmo sotto riportato, dove i parametri  $i$  e  $j$  sono valori interi,  $A$  un array di interi e  $x$  un valore Booleano.

```
1 ALGORITMO( $A,i,j,x$ )
1   if  $j - i >= 1$  then
2      $y = Rand() \% 2$ 
3     if ( $x = 1$ ) then
4        $ret = ALGORITMO(A, \frac{i+j}{2} + 1, j, y)$ 
5       if  $ret \% 2 = 0$  then
6          $ret = ALGORITMO(A, i, \frac{i+j}{2}, 1 - y)$ 
7     else
8        $ret = ALGORITMO(A, i, \frac{i+j}{2}, y)$ 
9     if  $ret \% 2 = 1$  then
10       $ret = ret + ALGORITMO(A, \frac{i+j}{2} + 1, j, 1 - y)$ 
11   else
12      $ret = A[i]$ 
13 return  $ret$ 
```

# Algoritmi e Strutture Dati I

## 22/09/2021

Sia dato un grafo orientato  $G$  e un array  $color[]$  che associa a ogni vertice  $v$  un colore, *rosso* o *verde*. Si definisca un algoritmo che, in tempo **lineare sulla dimensione del grafo  $G$** , verifichi se esistono due vertici  $u$  e  $v$  tali che:

- $color[u] \neq color[v]$ ;
- esiste un percorso da  $u$  a  $v$ ;
- esiste un percorso da  $v$  a  $u$ .

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

# 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.

# Equazioni di ricorrenza

## 1. [ Cormen ]

1.  $T(n) = 2T(n/2) + n^4$
2.  $T(n) = T(7n/10) + n$
3.  $T(n) = 16T(n/4) + n^2$
4.  $T(n) = 7T(n/3) + n^2$
5.  $T(n) = 7T(n/2) + n^2$
6.  $T(n) = 2T(n/4) + \sqrt{n}$
7.  $T(n) = T(n - 2) + n^2$
8.  $T(n) = 4T(n/3) + n \log_2 n$
9.  $T(n) = 3T(n/3) + n / \log_2 n$
10.  $T(n) = 4T(n/2) + n^2\sqrt{n}$
11.  $T(n) = 3T(n/3 - 2) + n/2$
12.  $T(n) = 2T(n/2) + n / \log_2 n$
13.  $T(n) = T(n/2) + T(n/4) + T(n/8) + n$
14.  $T(n) = T(n - 1) + 1/n$
15.  $T(n) = T(n - 1) + \log_2 n$
16.  $T(n) = T(n - 2) + 1 / \log_2 n$
17.  $T(n) = \sqrt{n}T(\sqrt{n}) + n$
18.  $T(n) = \sqrt[3]{n}T(\sqrt[3]{n}) + \sqrt{n}$

## 2. [ Jefferson ]

1.  $T(n) = 2T(n/4) + \sqrt{n}$
2.  $T(n) = 2T(n/4) + n$
3.  $T(n) = 2T(n/4) + n^2$
4.  $T(n) = 3T(n/3) + \sqrt{n}$
5.  $T(n) = 3T(n/3) + n$
6.  $T(n) = 3T(n/3) + n^2$
7.  $T(n) = 4T(n/2) + \sqrt{n}$
8.  $T(n) = 4T(n/2) + n$
9.  $T(n) = 4T(n/2) + n^2$
10.  $T(n) = T(n/2) + T(n/3) + T(n/6) + n$
11.  $T(n) = T(n/2) + 2T(n/3) + 3T(n/4) + n^2$
12.  $T(n) = T(n/15) + T(n/10) + 2T(n/6) + \sqrt{n}$
13.  $T(n) = 2T(n/2) + O(n \log n)$
14.  $T(n) = 2T(n/2) + O(n / \log n)$
15.  $T(n) = \sqrt{n}T(\sqrt{n}) + n$
16.  $T(n) = \sqrt{2n}T(\sqrt{2n}) + \sqrt{n}$

## 3. [ Esami ]

1.  $T(n) = 2\sqrt{n}T(\sqrt{n}) + n$
2.  $T(n) = 2T(\sqrt{n}) + \log n$

3.  $T(n) = 4T(n/4) + \sqrt{n}$
4.  $T(n) = 3T(n/4) + \sqrt{n}$
5.  $T(n) = 4T(n/9) + \sqrt{n}$
6.  $T(n) = 15T(n/4) + n^2 \log n$
7.  $T(n) = 2T(n/2) + T(n/4) + n$
8.  $T(n) = 17T(n/2) + n^4$
9.  $T(n) = T(3n/4) + T(n/2) + n$
10.  $T(n) = 6T(n/8) + \sqrt{n}$
11.  $T(n) = T(n/3) + T(n/4) + n$
12.  $T(n) = 7T(n/2) + n^3 \log n$
13.  $T(n) = T(2n/3) + T(n/3) + n^3 ?$
14.  $T(n) = 3T(\sqrt{n}) + \log n$

## Asintoticità

### 1. [ Esami ]

1. Si dimostri la verità o la falsità della seguente affermazione:

1. Se  $2^{f(n)} = \Theta(2^{g(n)})$ , allora  $f(n) = \Theta(g(n))$
2. Se  $f(n) = O(g(n))$ , allora  $\sqrt{g(n)} = \Omega(\sqrt{f(n)})$
3. Se  $h(n) = \Theta(t(n))$  e  $f(n)/h(n) = \Theta(g(n))$ , allora  $\frac{f^2(n)}{h(n) \cdot t(n)} = \Theta(g^2(n))$
4. Se  $f(n) = \Theta(n)$  e  $g(n) = \Theta(2^{n^2})$ , allora  $2^{2 \cdot \log f(n)} = \Theta(\log(g(n)))$
5. Se  $f, g$  sono due funzioni asintoticamente positive e crescenti, allora  $\log(f(n) \cdot g(n)) = O(\max\{\log(f(n)), \log(g(n))\})$
6. Se  $h^2(n) = \Theta(\min\{f(n), g(n)\})$ , allora  $\sqrt{f(n)} = O(h(n))$
7. Se  $\sqrt{h(n)} = O(\min\{f(n), g(n)\})$ , allora  $h(n) = O(g^2(n))$
8. Se  $2^{f(n)} = \Theta(g(n))$  e  $g(n) = \Theta(h(n)^k)$  per una costante  $k > 0$ , allora  $f(n) = \Theta(\log h(n))$
9. Se  $f(n) = \Theta(n)$  e  $g(n) = \Theta(2^n)$ , allora  $2^{f(n)} = \Theta(g(n))$
10. Se  $\sqrt{h(n)} = O(2^{f(n) \cdot g(n)})$ , allora  $\log(\log h(n)) = \Theta(g(n) \cdot \log f(n))$
11. Se  $h(n) = \Theta(\max\{\log \log f(n), \log \log g(n)\})$ , allora  $g(n) = O(2^{2^{h(n)}})$
12. Se  $z(n) = \Theta(2^{g(n)})$  e  $h(n) = \Theta(\log g(n))$ , allora  $\log z(n) = \Theta(2^{h(n)})$
13. Se  $f(n) = \Theta(\sqrt{g(n)})$  e  $2^{g(n)} = \Theta(2^n)$ , allora  $\log f(n) = \Theta(\log n)$
14. Se  $h(n) = \Theta(t(n))$  e  $f(n) = \Theta(g(n))$ , allora  $g(n) + h(n) = \Theta(t(n) + f(n))$
15. Se  $h(n) = \Theta(t(n))$  e  $f(n) = \Theta(g(n))$ , allora  $\log_2(g(n) \cdot h(n)) = \Theta(\log_2(t(n) \cdot f(n)))$
16. Se  $\log \frac{f(n)}{g(n)} = \Theta(\log(t(n) \cdot g(n)))$  e  $\log \frac{t(n)}{g(n)} = \Theta(\log \frac{h(n)}{f(n)})$ , allora  $\log h(n) = \Theta(\log t(n))$
17. Se  $f(n) = \Theta(\sqrt{g(n)})$  e  $f(n) = \Theta(k^2(n))$ , allora  $g(n) = \Theta(k(n)^4)$
2. Si trovino, se esistono, le costanti per soddisfare la seguente relazione asintotica:
  1.  $5n^2 - 8\sqrt{n} + 1 = \Theta(n^2)$
  2.  $\log \frac{n}{7} = \Theta(\log n^4)$
  3.  $4n^2 - 7\sqrt{n} + 2 = \Theta(n^2)$

- 
- 4.  $2n - \log \frac{n}{4} = \Theta(n)$
  - 5.  $2 \log_2(n) - 4/n = \Theta(\log_2 n)$
  - 6.  $n - \log_2(n) + 1 = \Theta(n)$
  - 7.  $n^2 \log(n^2) + 15n^2 = \Theta(n^2 \log(n))$
  - 8.  $7n\sqrt{n} + 3n - 10\sqrt{n} = \Theta(n^{3/2})$
  - 9.  $\log_2(2^n \cdot \frac{4^n}{n}) = \Theta(\log_2(3^{3n}))$
- 

68

- [Link Forum](#)
- [Link Github](#)

# 1 Esercizi

## Esercizio 1 - Esame del 25-03-2022

Si risolva la seguente equazione di ricorrenza calcolandone l' **andamento asintotico**

$$T(n) = \begin{cases} 1 & \text{se } n \leq 2 \\ 3 \cdot T(\sqrt{n}) + 2 \cdot T(\sqrt[4]{n}) + \log n & \text{altrimenti} \end{cases}$$

## Esercizio 2 - Esame del 25-03-2022

Si scriva un algoritmo che, dati un grafo  $G = \langle V, E, \rangle$  e due vertici  $s, u \in V$ , verifichi in **tempo lineare** sulla dimensione del grafo se **tutti i percorsi infiniti che partono da s non passano infinite volte per u**

## Esercizio 1 - Esame del 21-06-2022

Si individuino, nel caso esistano le **costanti moltiplicative** atte a mostrare la seguente relazione asintotica

$$\ln\left(\frac{n}{e}\right) = \Theta\left(\ln(n^e)\right)$$

(Si ricorda che con "ln" si indica il logaritmo naturale e con "e" la costante di Eulero, detta anche di Nepero)

In caso contrario mostrare la falsità della relazione

## Esercizio 2 - Esame del 21-06-2022

Si scriva un **algoritmo ricorsivo** che, dati in ingresso un albero binario di ricerca su interi  $\mathcal{T}$  e due valori  $k_1, k_2 \in \mathbb{N}$ , inserisca in una lista  $\mathcal{L}$  le chiavi  $k$  contenute in  $\mathcal{T}$  comprese tra  $k_1$  e  $k_2$  ( $k_1 \leq k \leq k_2$ ), in modo che al termine  $\mathcal{L}$  contenga valori ordinati in modo decrescente.

Tale algoritmo dovrà avere **complessità lineare** nella dimensione dell'albero.

Infine si scriva un **algoritmo iterativo** che **simuli precisamente** l'algoritmo di cui sopra

### Esercizio 1 - Esame del 21-07-2022

Si individuino nel caso esistano, le **costanti moltiplicative** atte a mostrare la seguente relazione asintotica

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

In caso contrario mostrare la falsità della relazione

### Esercizio 2 - Esame del 21-07-2022

Si scriva un **algoritmo ricorsivo** che, dati in ingresso un albero binario di ricerca su interi  $\mathcal{T}$  e due valori  $k_1, k_2 \in \mathbb{N}$ , cancelli da  $\mathcal{T}$  le chiavi  $k$  comprese tra  $k_1$  e  $k_2$  ( $k_1 \leq k \leq k_2$ ).

Tale algoritmo dovrà essere efficiente e non far uso **nè di variabili globali nè di parametri passai per riferimento**.

Infine si scriva un **algoritmo iterativo** che **simuli precisamente** l'algoritmo ricorsivo di cui sopra.

### Esercizio 1 - Esame del 22-09-2022

Si individuino, nel caso esistano le **costanti moltiplicative** atte a mostrare la seguente relazione asintotica

$$n \cdot \ln(e^2 \cdot n) = \Theta(\ln \sqrt{(n)^n})$$

(Si ricorda che con "ln" si indica il logaritmo naturale e con "e" la costante di Eulero, detta anche di Nepero)

In caso contrario mostrare la falsità della relazione. Il procedimento seguito va riportato per esteso

### Esercizio 2 - Esame del 22-09-2022

Si scriva un algoritmo che, dato in ingresso un grafo orientato  $G = \langle V, E, \rangle$ , verifichi **in tempo lineare sulla dimensione del grafo** se la seguente condizione è soddisfatta:

- $G$  contiene 3 vertici distinti, di seguito denominati  $a$ ,  $b$ ,  $c$ , tali che:
  - ◊  $a$  e  $b$  sono entrambi raggiungibili da  $c$ ;
  - ◊  $a$  e  $c$  sono entrambi raggiungibili da  $b$ ;
  - ◊  $b$  e  $c$  sono entrambi raggiungibili da  $a$ ;

Notare che i vertici  $a$ ,  $b$ ,  $c$  **non sono input** del problema

### Esercizio 1 - Esame del 25-01-2023

Si risolva la seguente equazione di ricorrenza, calcolandone l' **andamento asintotico**

$$T(n) = \begin{cases} 1 & \text{se } n \leq 2 \\ 2 \cdot T \sqrt[4]{n} + \log(2n) & \text{altrimenti} \end{cases}$$

### Esercizio 2 - Esame del 25-01-2023

Si scriva un **algoritmo iterativo** che simuli precisamente l'algoritmo ricorsivo di seguito riportato, dove  $Z$  è una funzione esterna non meglio specificata

```
1 function Algoritmo(T, h)
2   if T = Nil then
3     return Z(0, h)
4   else
5     a ← 0
6     if T → key = 0 mod 2 then
7       a ← a + Algoritmo(T → dx, 2 * h)
8
9     if T → key = 1 mod 3 then
10      a ← a + Algoritmo(T → sx, 3 * h)
11
12    return Z(T → key, a)
```

### Esercizio 1 - Esame del 22-02-2023

Si risolva la seguente equazione di ricorrenza, calcolandone l' **andamento asintotico**

$$T(n) = \begin{cases} 1 & \text{se } n \leq 27 \\ 3 n?2 \cdot T \sqrt[3]{n} + 2 n^3 & \text{altrimenti} \end{cases}$$

### Esercizio 2 - Esame del 25-01-2023

Si scriva un **algoritmo iterativo** che simuli precisamente l'algoritmo ricorsivo di seguito riportato, dove  $Z_t$  e  $Z_r$  sono 2 funzioni esterne non meglio specificate che soddisfano la seguente proprietà

$$p < Z_t(A, p, s) < Z_r(A, p s) \leq s$$

quando  $p + 1 < s$

```
1 function Algoritmo(A, p, s)
2   if s ≤ p + 1 then
3     return 0
4   else
```

```
5     q ←  $Z_t(A, p, s)$ 
6     r ←  $Z_r(A, p, s)$ 
7     a ← Algoritmo( $A, p, q$ )
8     a ← a – Algoritmo( $A, q, s$ )
9     a ← a + Algoritmo( $A, r, s$ )
10    return a + (r – q)
```

# Tema d'esame di Algoritmi e Strutture Dati I

## 21/01/2022

**Tempo a disposizione: 1.30 ore.**

1. Si risolva la seguente equazione di ricorrenza:

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

2. Sia dato il seguente algoritmo:

```
1 ALGORITMO( $T, i$ )
1    $a = G(i)$ 
2   if  $T = Nil$  then
3     return  $a$ 
4   else
5      $z = ALGORITMO(T \rightarrow sx, 2 \cdot i)$ 
6      $a = z + (T \rightarrow key) \cdot i + a$ 
7      $z = a + ALGORITMO(T \rightarrow dx, 2 \cdot i + 1)$ 
8     return  $3 \cdot z$ 
```

dove  $G(\cdot)$  è una funzione esterna. Scrivere un algoritmo **iterativo** che **simuli precisamente** il comportamento dell'algoritmo ricorsivo sopra riportato.