Corso di laurea in Informatica
Introduzione agli Algoritmi
Didattica blended

Esercizi per casa

Angelo Monti

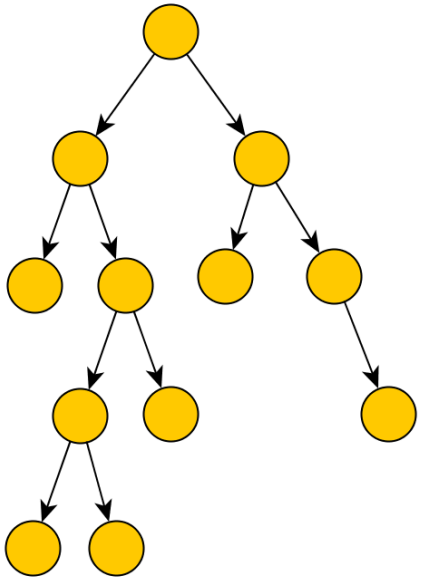
SAPIENZA
UNIVERSITÀ DI ROMA

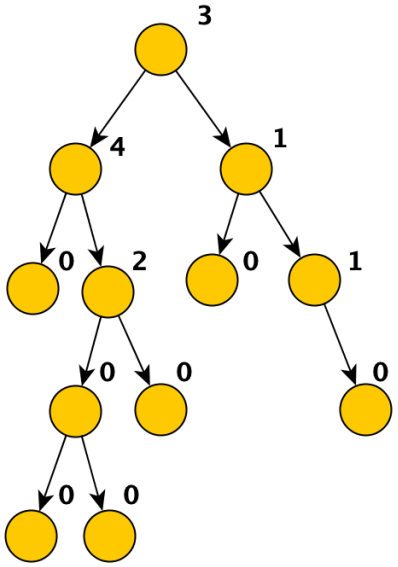
Sulla base delle slides a cura di T. Calamoneri e G. Bongiovanni per il corso di informatica generale AA 2019/2020

Esercizio 1

In un albero binario lo **sbilanciamento** di un nodo è il valore assoluto tra il numero di nodi nel suo sottoalbero sinistro ed il numero di nodi nel suo sottoalbero destro.

Esempio di albero binario e sbilanciamento dei suoi nodi:





Progettare un algoritmo che dato un albero binario di cui si conosce il puntatore alla radice restituisca il massimo tra gli sbilanciamenti dei suoi nodi.

L'algoritmo deve avere complessità $O(n)$.

Esercizio 2

Siano dati due vettori *A* e *B*, composti da $n \geq 1$ ed $m \geq 1$ interi, rispettivamente. I vettori sono entrambi ordinati in senso crescente. *A* e *B* non contengono valori duplicati; tuttavia, uno stesso valore potrebbe essere presente una volta in *A* e una volta in *B*.

Progettare un algoritmo di complessità $O(n+m)$ che stampi i numeri che appartengono all'unione dei valori di *A* e di *B*; l'unione va intesa in senso insiemistico, quindi gli eventuali valori presenti in entrambi i vettori devono essere stampati solo una volta.

Ad esempio, se $A = [2,3,4,6]$ e $B = [1,3,4,7]$, l'algoritmo deve stampare in qualche ordine gli elementi 1, *2, 3, 4, 6, 7*.

Di tale algoritmo:

Si dia una descrizione a parole e si scriva lo pseudocodice.

Esercizio 3

Abbiamo una procedura f che prende un intero $x \geq 0$ e restituisce un intero. La funzione calcolata è strettamente crescente (vale a dire $f(x) < f(x+1)$) e sappiamo che vogliamo trovare il primo intero n non negativo per cui la funzione assume un valore non-negativo.

Ad esempio:
per $f(x) = -100 + 3x$ allora il valore da trovare è 34

Progettare un algoritmo che trova questo valore in tempo $O(\log n)$. Potete assumere che la procedura f ha complessità $\Theta(1)$.

Esercizio 4

Dato un vettore A che contiene n interi distinti e ordinati in modo crescente e due interi x e k, con $k < n$, progettare un algoritmo che stampi l'insieme dei k interi più vicini ad x presenti in A.
Nota: *se l'elemento x appartiene all'insieme non deve essere conteggiato tra quelli da stampare.*

La complessità dell'algoritmo deve essere $O(k + \log n)$.

Ad esempio per

$A = [12, 16, 22, 30, 35, 39, 42, 45, 48, 50, 53, 55, 56]$,

$x = 35$ e $k = 4$ l'algoritmo deve stampare $30, 39, 42$ e 45.

Esercizio 5

Diciamo che un vettore di interi distinti è $k-ordinato$ *se ogni elemento del vettore si trova a distanza al più* k *dalla sua posizione corretta (vale a dire quella che assumerebbe se il vettore venisse ordinato).*

Ad esempio:

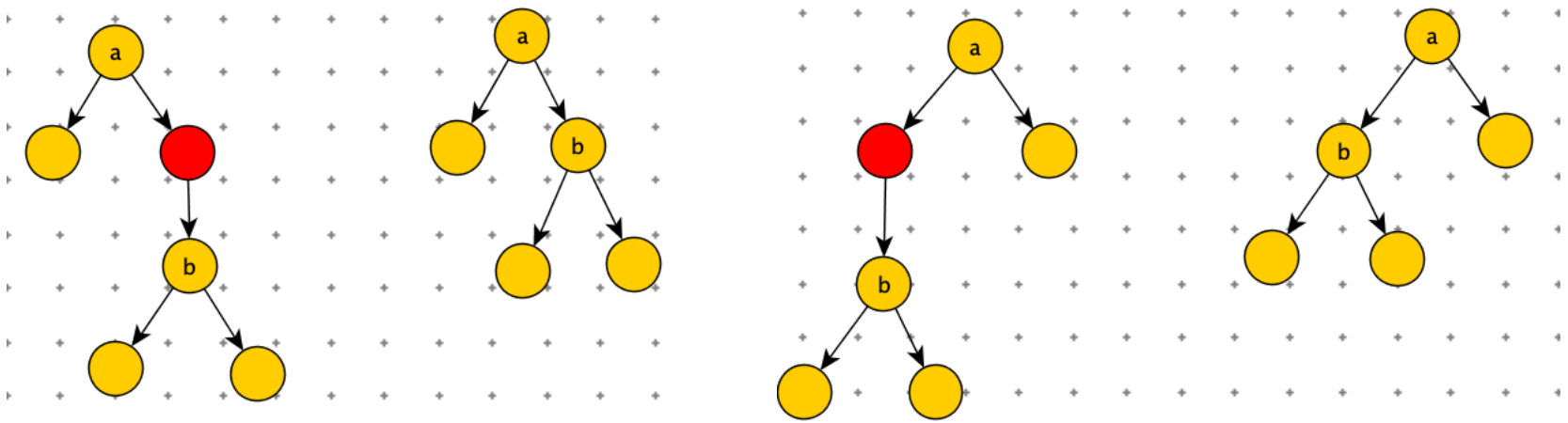
il vettore A= [10, 9, 8, 7, 4, 70, 60, 50] è 4-ordinato (basta ordinarlo per rendersene conto) mentre il vettore B=[6, 5, 3, 2, 8, 10, 9] è 3-ordinato

Progettare un algoritmo di ordinamento che dato un vettore A $k-ordinato$ *di* n *interi e l'intero* k *ordina* A *.*

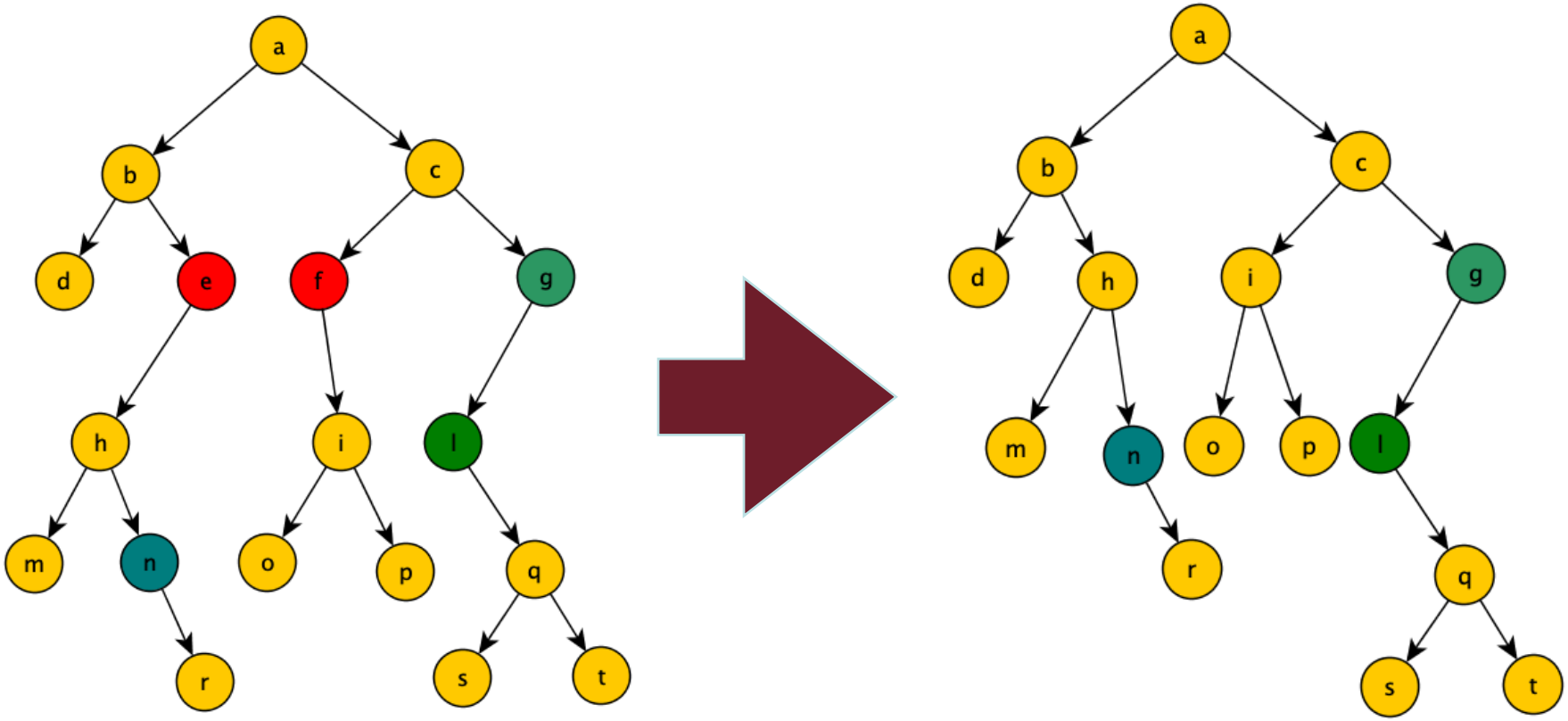
La complessità dell'algoritmo deve essere $O(n \log k)$ *.*

Esercizio 6

Scrivere una funzione che dato in input un albero binario A, con n chiavi cancella tutti i nodi interni x con un solo figlio che hanno sia il padre che il figlio dotati di due figli. La cancellazione del nodo x avviene sostituendo il puntatore dal padre di x ad x con un puntatore al figlio di x. Assumete che l'albero sia memorizzato tramite nodi e puntatori e che ogni nodo abbia anche il puntatore al padre. La procedura deve avere complessità $O(n)$.



Esercizio 6 esempio:



Esercizio 7

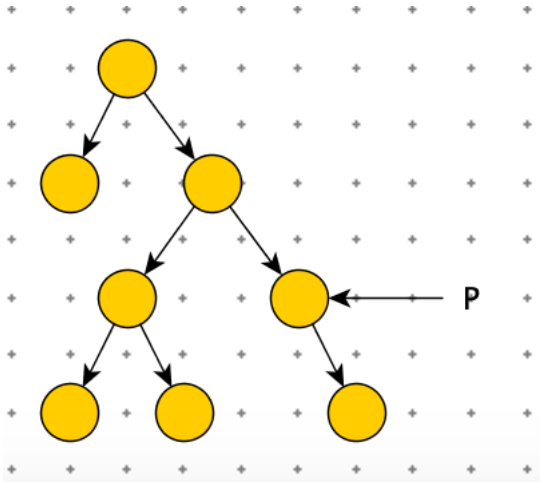
Abbiamo un albero binario di ricerca ed un suo nodo x. Vogliamo cancellare dall'albero l'eventuale nodo y fratello di x e tutti i nodi contenuti nel sottoalbero radicato in y.

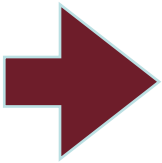
Progettare una funzione che risolva il problema nel caso in cui:

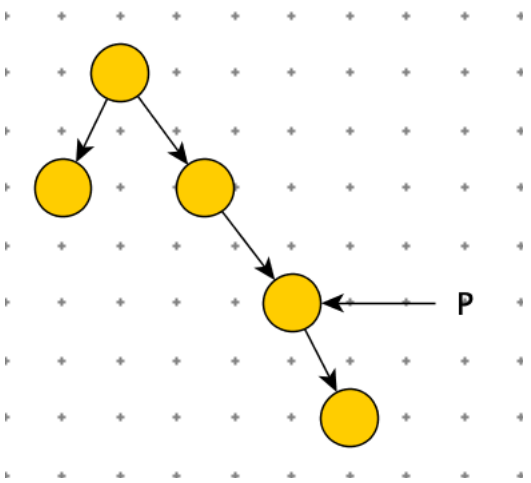
1. *L'albero e' rappresentato tramite nodi e puntatori dove ogni nodo ha anche il puntatore al padre ed alla funzione viene fornito il puntatore all'albero ed il puntatore al nodo x .*
2. *L'albero e' rappresentato con notazione posizionale e alla funzione viene passato il vettore A e l'indice i di A in cui si trova la chiave del nodo x.*

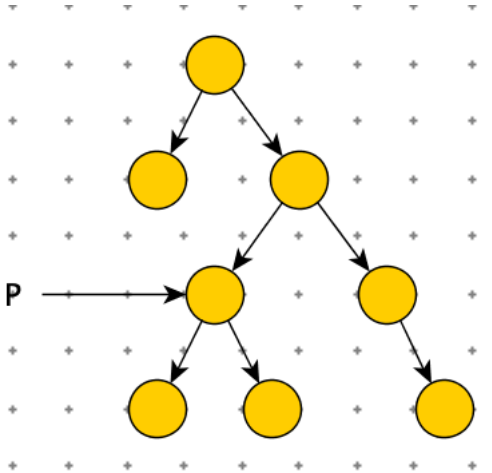
In entrambi i casi la procedura deve avere complessità $O(n)$ dove n è il numero di nodi presenti nell'albero.

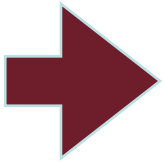
Esempio d'esecuzione procedura 7.1

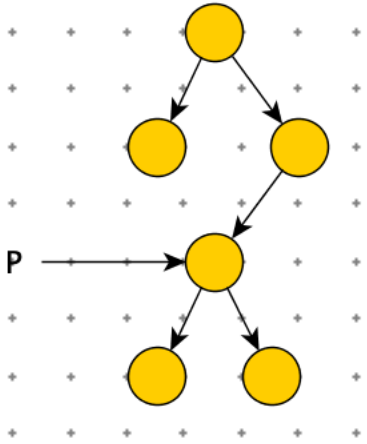




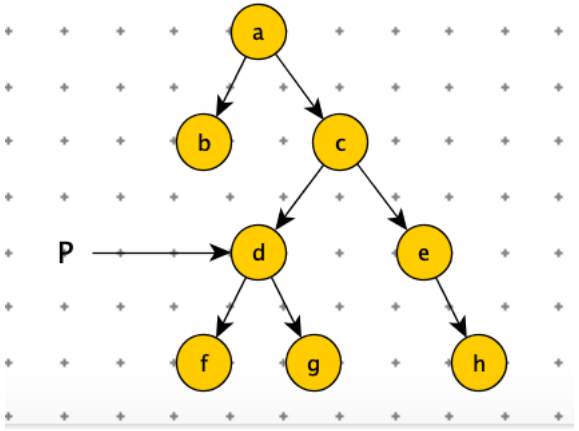


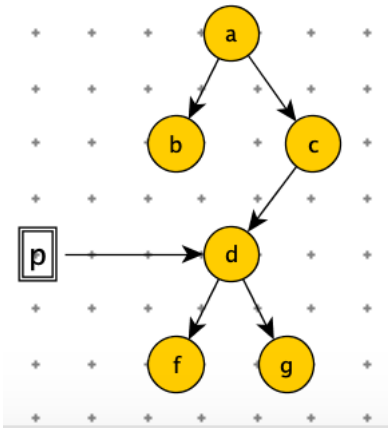






Esempio di esecuzione procedura 7.2





0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
a	b	c		_	d	e	-	-	-	-	f	g	-	h

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
a	b	c		_	d	-	-	-	-	-	f	-	-	-

Esercizio 8

Progetta un algoritmo che dato un vettore A con n interi ed un intero x determina se nel vettore esistono due interi la cui somma è x.

L'algoritmo deve avere complessità $O(n \log n)$.

Ad esempio:

- Per $A = [0, -1, 2, -3, 1]$ e $x = -2$ l'algoritmo restituisce $True$ (basta infatti considerare i due elementi -3 e 1).

- Per $A = [1, -2, 1, 0, 5]$ e $x = 0$ l'algoritmo restituisce $False$

Esercizio 9

Progettare un algoritmo che dati i puntatori p e q a due liste di interi verifica se la prima lista possa ottenersi dalla seconda cancellando eventualmente dei nodi.

L'algoritmo deve avere complessità $O(m)$ dove m sono i nodi della della seconda lista.

Ad esempio:

Per
$p -- > 1 -- > 2 -- > 3 -- > 4$ e
$q -- > 5 -- > 1 -- > 2 -- > 1 -- > 2 -- > 3 -- > 5 -- -- > 6 -- -- > 4$
l'algoritmo risponde $True$
(basta considerare gli elementi in rosso e cancellare gli altri 5->1–>2–>1–>2–>3–>5–>6–>4)

Per
$p -- > 1 -- > 2 -- > 3 -- > 4$ e
$q -- > 1 -- > 2 -- > 2 -- > 1 -- > 4 -- > 5 -- > 6 -- > 3 -- > 2$
l'algoritmo risponde $False$

Esercizio 9 Esempio di esecuzione della procedura es()

```
class Nodo:
    def __init__(self, key=None, next=None):
        self.key = key
        self.next = next

>>>p=Nodo(1); p.next=Nodo(2); p.next.next=Nodo(3); p.next.next.next=Nodo(4)

>>>t=Nodo(1); t.next=Nodo(2); t.next.next=Nodo(4); t.next.next.next=Nodo(3)

>>>q=Nodo(5); q.next=Nodo(1); q.next.next=Nodo(2);
>>>q.next.next.next=Nodo(1)
>>>q.next.next.next.next=Nodo(2);
>>>q.next.next.next.next.next=Nodo(3)
>>>q.next.next.next.next.next.next=Nodo(5)
>>>q.next.next.next.next.next.next.next=Nodo(6)
>>>q.next.next.next.next.next.next.next.next=Nodo(4)

>>> es(p,t)
False
>>> es(p,q)
True
```