Corso di laurea in Informatica
Introduzione agli Algoritmi
Didattica blended

Esercizi

Angelo Monti

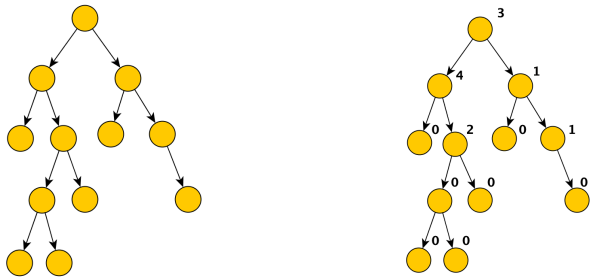
SAPIENZA
UNIVERSITÀ DI ROMA

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:



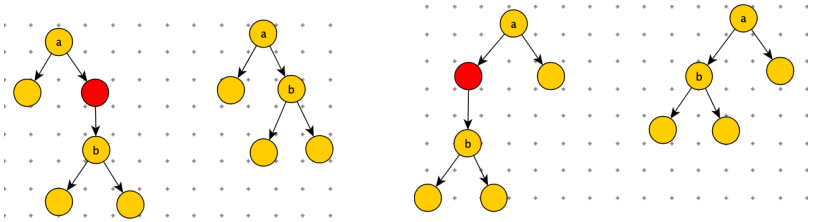
Progettare un algoritmo che dato il puntatore alla radice di un albero binario di n nodi restituisce il massimo tra gli sbilanciamenti dei suoi nodi.

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

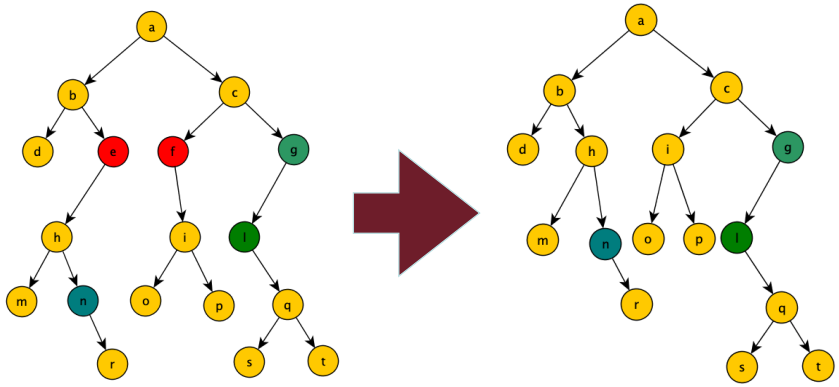
Esercizio 2

Scrivere una funzione che dato il puntatore ad un albero binario con n nodi cancella dall'albero tutti i nodi interni x con un solo figlio che hanno sia il nodo padre che il nodo 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 ogni nodo abbia anche il puntatore al padre.

La procedura deve avere complessità $O(n)$.



Esercizio 6 esempio:



Esercizio 3

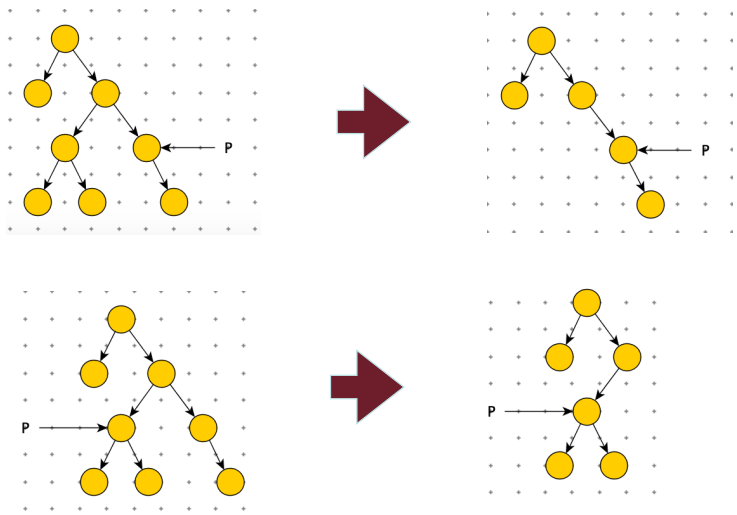
Abbiamo un albero binario di ricerca ed un suo nodo x. Vogliamo cancellare dall'albero l'eventuale nodo y fratello di x e tutti i nodi contenuti nel sottoalbero radicato in y.

Progettare una funzione $cancella()$ che risolva il problema nel caso in cui:

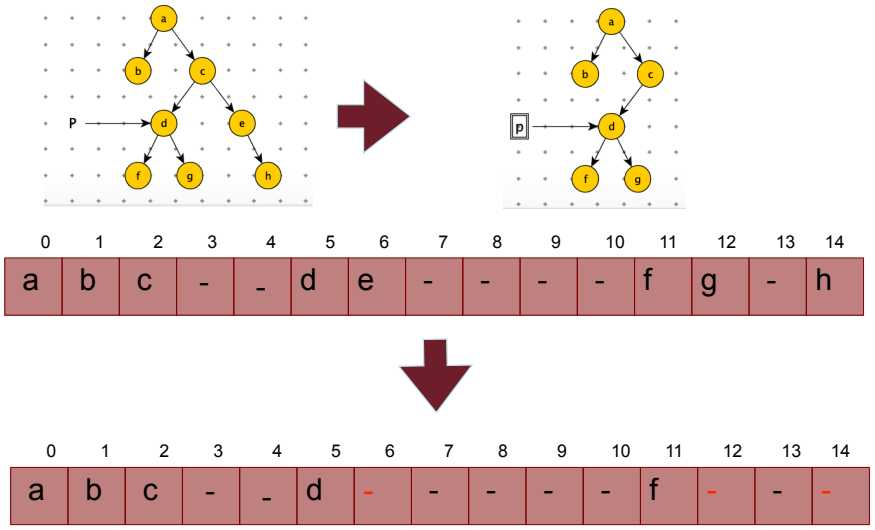
1. L'albero è rappresentato tramite nodi dove ogni nodo oltre ai puntatori ai figli ha anche il puntatore al padre ed alla funzione $cancella()$ viene fornito il puntatore il puntatore al nodo x.
2. L'albero è rappresentato con notazione posizionale e alla funzione $cancella()$ viene fornito il vettore 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 della procedura nel caso 1



Esempio di esecuzione procedura nel caso 2



Esercizio 4

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 e mantenendo gli altri nell'ordine in cui sono. .

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 rossi 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 5

Dati due numeri interi *x* ed *y*, definiamo la loro **distanza** come *dist(x, y)* = |*x*-*y*|.
Sia dato il puntatore alla radice di un albero binario di ricerca contenente chiavi intere.
Si progetti un algoritmo il più efficiente possibile che determini la distanza massima per le chiavi dell'albero e se ne calcoli il costo computazionale.
Si discuta brevemente sul modo in cui (eventualmente) cambiano l'algoritmo ed il costo computazionale nel caso in cui *l'albero binario di ricerca* sia un albero rosso-nero.

IDEA:
In un ABR le chiavi a distanza massima sono invariabilmente la chiave minima e quella massima contenute nell'ABR.