A

Introduzione agli algoritmi
Prova di esame del 30/6/2016
Prof.sse E. Fachini - R. Petreschi

Si consideri l'operazione di inserimento in un AVL. Si supponga di trovarsi nel caso in cui una rotazione a sinistra è sufficiente a ribilanciare l'albero. Si illustri il processo di aggiornamento dei fattori di bilanciamento. Si descriva, anche aiutandosi con dei disegni, la situazione del sotto albero di radice x, T(x), su cui viene eseguita la rotazione, prima dell'inserimento, dopo l'inserimento e dopo la rotazione. Si mostri come vengono modificati i fattori di bilanciamento dei nodi implicati nella rotazione. Si spieghi perché non è necessario aggiornare i fattori di bilanciamento di tutti gli altri nodi nel cammino da x alla radice, così come dei nodi che non appartengono al cammino nuovo nodo inserito-radice. Si concluda con l'analisi del tempo asintotico dell'operazione di inserimento in questo caso.

Sol. Si consultino libro di testo e lucidi.

2. Si determini il tempo di esecuzione asintotico T(n) della seguente funzione:

```
analizzami(A,i,j)
n = j - i + 1
c = 1
m = (i + j +1)/2
d = m
while d >1 do
        c++ ; d=d/2
if n>1 then
        return analizzami(A,i,m-1) + analizzami(A,m+1,j)
```

Sol. La relazione di ricorrenza da risolvere è
$T(n)=2T(n/2) + c \lg n$ se $n>1$ $T(n) = d$ altrimenti, dove c e d sono costanti positive.
Si dimostra facilmente che ha soluzione in $O(n\lg n)$, meno facilmente che è in $O(n)$.

3.

Sia dato un albero binario di ricerca, ABR, implementato con strutture a puntatore in modo che per ogni nodo x siano presenti oltre al campo chiave e ai campi puntatori ai figli e al padre, un campo puntatore, succ, al nodo di chiave successiva nell'albero a quella di x. Si mostri come deve essere modificato l' algoritmo di inserimento di una nuova chiave, in modo che tutti i campi del nuovo nodo siano opportunamente valorizzati. Si valuti il tempo asintotico di esecuzione dell'algoritmo presentato. Quale operazione su un ABR, tra quelle nel programma del corso, verrebbe effettuata in modo più efficiente avendo a disposizione anche il campo successivo? Si argomenti senza necessariamente fornire lo pseudocodice dell'operazione.

Sol. Dovendo inserire un nuovo nodo, dovremo oltre a individuare la sua corretta posizione nell'albero aggiornare i campi al successivo dei nodi coinvolti. I nodi coinvolti sono il nuovo nodo inserito, suo padre e il suo precedente nell'albero.

Supponiamo di disporre del puntatore al precedente del padre del nuovo nodo da inserire, prec e sia y il puntatore al nodo che diventerà padre del nuovo nodo.

Quindi abbiamo prec.key < y.key e nessun'altra chiave è compresa tra queste due.

Inseriamo il nodo di chiave k e distinguiamo due casi:

1. y.key < k , in tal caso il nodo di chiave k diventa figlio destro del nodo y e quindi si ha

   prec.key < y.key < k < y.succ.key.

   In tal caso il nodo di chiave k è anche il successivo di suo padre, mentre il successivo di suo padre deve diventare il suo successivo. Infine il campo succ di prec non cambia, perché è ancora y il nodo con la chiave successiva a quella di prec.
2. y.key > k, in tal caso il nodo di chiave k diventa figlio sinistro allora si ha:

   prec.key < k < y.key < y.succ.key.

Quindi si deve aggiornare il campo succ di prec al nuovo nodo inserito, il cui successivo è il padre, per cui il campo succ del padre non cambia.

Per determinare il puntatore prec, si può utilizzare un parametro che viene aggiornato nella discesa verso y. Il parametro prec conterrà il puntatore all'ultimo nodo padre di un figlio destro lungo il cammino verso y. Questo nodo contiene la chiave che precede immediatamente, nell'albero, quella di y.
Il tempo dell'inserimento resta in O(h), dove h è l'altezza dell'albero, perché le modifiche effettuate localmente sono di costo costante ma il tempo per la ricerca della posizione in cui inserire il nodo è al più pari all'altezza.

Pseudocodice:

Insert(T,k)
Crea un nodo x di chiave k, con i campi puntatori a NIL
InsertSucc(T,x,NIL)

La funzione chiamata è così specificata:

insertSucc(T,x,prec)
input: T è un puntatore alla radice di un albero binario, x è un nodo da inserire e prec è un puntatore alla radice di un albero binario
precondizione: T è un ABR non nullo e la chiave di x non è presente in T
output: l'albero T viene modificato con l'inserimento di x e l'opportuno aggiornamento dei campi succ dei vari nodi coinvolti.

```
if T.key < x.key then
        if T.right == NIL  then
                T.right = x;  x.p = T, x.succ = T.succ; T.succ = x;
```

```
        else InsertSucc(T.right,x,T)
if T.key > x.key then
       if T.left == NIL  then
            T.left = x; x.p = T;  if prec ≠ NIL then prec.succ = x;
            x.succ = T;
         else InsertSucc(T.left,x,prec)
```

L'operazione che in modo più evidente si avvantaggia della presenza del campo al successivo è la visita inorder, che pur essendo sempre in Θ(n), dove n è il numero dei nodi, si avvallerebbe dell'individuazione in tempo costante del successivo di ogni nodo.
Anche la cancellazione migliora, pur rimanendo in O(h), tenendo conto del tempo necessario a individuare il nodo da cancellare.