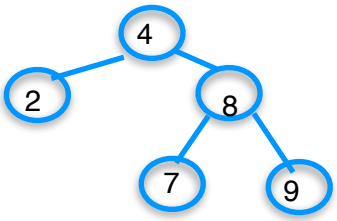
Introduzione agli algoritmi
Appello del 28/6/2017
E. Fachini - R. Petreschi

B

Le soluzioni degli esercizi scritte in modo illeggibile o in cui compaiano solo conti o pseudocodice senza commenti e risposte non motivate saranno valutati 0. Prima di descrivere un algoritmo in pseudocodice si deve delineare l'idea algoritmica. Inoltre deve essere precisato l'output atteso da eventuali singole funzioni utilizzate, oltre agli eventuali vincoli sul loro input (precondizioni).

Parte II

1. Si illustri l'operazione di rotazione a destra in un ABR, spiegando perché la proprietà di essere ABR è preservata dall'operazione. Si inserisca un nuovo nodo, di chiave 6 nell'AVL disegnato sotto e si faccia vedere come ribilanciare l'albero. Si metta in evidenza il cambiamento dei fattori di bilanciamento e delle altezze.



2. Si consideri la seguente funzione:

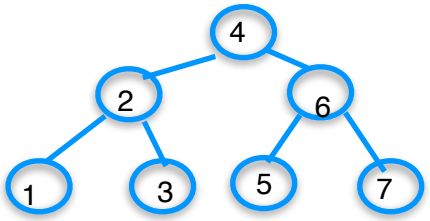
```
fun (intero n) {
i=0;
if n<2 then return i;
m = n*n
for a =1 to m do
        for b =1 to a do i++;
return 2*fun(n/2);}
```

$T(n) = T(n/2) + \Theta(n^4)$

3. Si assumi che ogni nodo u di un AVL T contenga, oltre alla chiave, ai puntatori ai figli e al padre e al fattore di bilanciamento, anche un campo sum in cui è mantenuta la somma delle chiavi nel sottoalbero radicato in u.

Esempio: se T è l'albero AVL completo sulle chiavi 1, 2, 3, 4, 5, 6, e 7, allora:

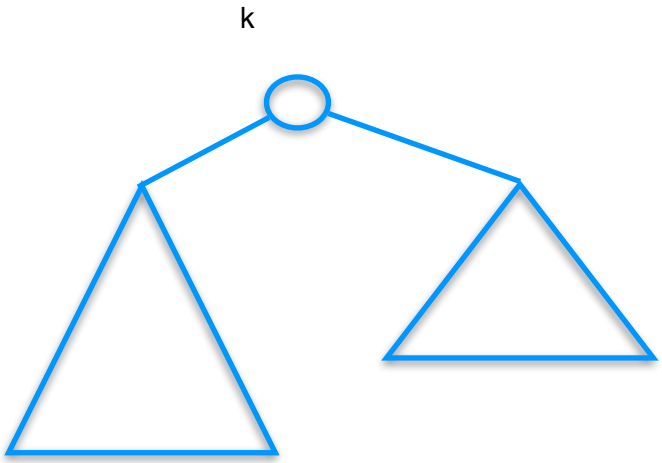
Introduzione agli algoritmi
Appello del 28/6/2017
E. Fachini - R. Petreschi

T= 

• sum(4) = 28 poiché la chiave 4 è nella radice dell'albero e 1 + 2 + 3 + 4 + 5 + 6 + 7 = 28;
• sum(6) = 18 poiché la chiave 6 è nel figlio destro della radice e 5 + 6 + 7 = 18;
• sum(3) = 3 poiché la chiave 3 è in una foglia.

Si descriva un algoritmo che realizzi una nuova operazione minSum che, dati un albero AVL T e un valore intero s, dia in output la più piccola chiave k di T tale che la somma delle chiavi nel sottoalbero radicato nel nodo di chiave k è strettamente maggiore di s. Se una tale chiave non esiste, minSum dà in output ∞. Si ricordi che la descrizione dell'idea algoritmica deve convincere della sua correttezza e che può essere illustrata con l'ausilio della grafica.
Si analizzi tempo di esecuzione, che dovrebbe essere O(log n). Vi consiglio di usare opportunamente il campo sum introdotto al punto 1.

Sol.
Si noti che se la radice dell'albero T ha chiave k, e campo sum = a, e se i figli hanno campi sum = a1 rispettivamente a2:



La somma nel nodo di chiave k è a=k+a1+a2,

se a < s allora la somma delle chiavi nei sottoalberi radicati nei figli del nodo di chiave k è minore di s, essendo minore di a, quindi è inutile scendere sui figli, dunque il risultato è ∞.
Se a > s allora T.key potrebbe essere la chiave richiesta, nel caso non ci sia un sotto albero sinistro o se a1 ≤ s, infatti in questo caso non ci sono chiavi minori di quella di T che soddisfino la condizione richiesta. Se invece a1 > s bisogna controllare il figlio sinistro, cioè bisogna fare una chiamata sul figlio sinistro, perché T.left.key < T.key e quindi T.left o un suo figlio sinistro potrebbe essere la chiave cercata.

Pseudocodice:

```
MinSum(T,s)
if T==NULL or T.sum < s then return  ∞
if T.sum > s and (not T.left or T.left.sum ≤ s) then return T
if T.sum >  s and T.left and T.left.sum > s then return MinSum(T.left,s)
else return ∞
```