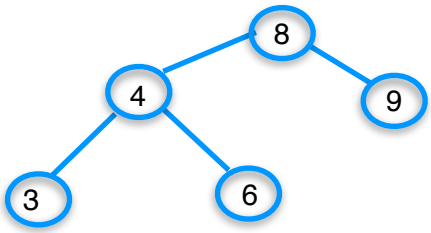
Introduzione agli algoritmi
Appello del 28/6/2017
E. Fachini - R. Petreschi

A

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 sinistra in un ABR, spiegando perché la proprietà di essere ABR è preservata dall'operazione. Si inserisca un nuovo nodo, di chiave 7 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 (array A, int i, int f) {
    m = f-i+1
    if (m ≤ 1) then return 1;
    t = m^2;
    while t ≥ 2 do t = t/3;
    j = i + m/3
    return fun(A,i,j) + fun(A,j+1,j+m/3)+ fun(A,j+m/3+1,f);}
```

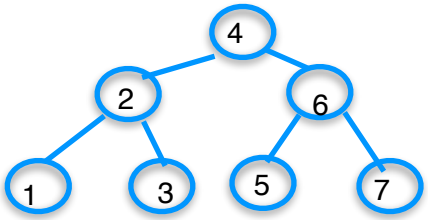
La relazione di ricorrenza è $T(n) = 3T(n/3) + \Theta(\log_3 n)$.
La soluzione è $T(n) = O(n \lg n)$.

3. Si assuma 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

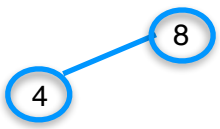
T=



• 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.

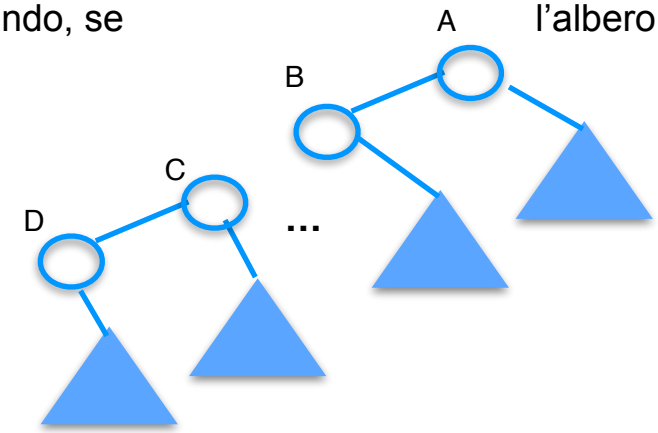
Si descriva un algoritmo che realizzi una nuova operazione SommaMinori che, dati un albero AVL T e un valore intero s, dia in output **la più piccola chiave k** di T tale che **la somma delle chiavi minori di k sia strettamente maggiore di s**. Se una tale chiave non esiste, SommaMinori dà in output ∞.
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). Si consiglia di usare opportunamente il campo sum introdotto al punto 1.

Sol.
Assumiamo che 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 minori di k, a1, è anch'essa minore di s, essendo minore di a, e il risultato è ∞.
Se a1 > s allora T.key potrebbe essere la chiave richiesta, ma bisogna controllare il figlio sinistro, T.left, se c'è, perché T.left.key < T.key e quindi T.left o un suo figlio sinistro potrebbe essere la chiave cercata. Qui il più piccolo albero è un albero con due nodi come per esempio:



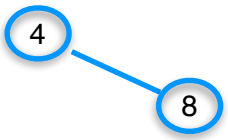
se s = 3 allora la soluzione è la chiave 8, mentre se s ≥ 4 la soluzione è ∞.
Potremmo allora direttamente controllare l'esistenza del figlio sinistro del figlio sinistro e nel caso non ci sia, e con s = 3 dare la chiave 8 come risposta.

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

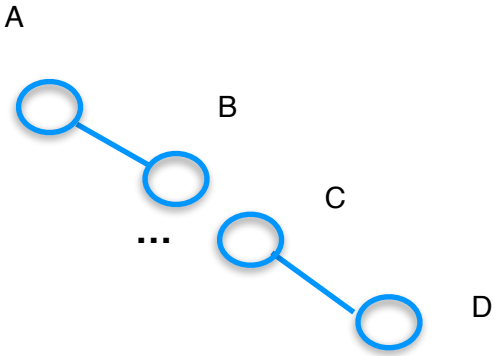
Generalizzando, se 

l'albero è più grande:

La chiave cercata deve trovarsi sul cammino più a sinistra ed è la prima per la quale il figlio sinistro ha sum maggiore di s mentre il nipote sinistro ha sum ≤ s o che non ha nipote sinistro.
Se invece a1 ≤ s dovremo dirigere la ricerca nel sotto albero destro. Però il confronto va fatto non più con s, ma con s - (T.key + T.left.sum) perché quelle chiavi sono tutte minori della chiave del figlio destro di T, in questo modo si può chiamare la funzione su questo nodo e far ripartire la ricerca nel sotto albero destro. Se T non ha il figlio sinistro allora confronteremo s con T.key. Il più piccolo albero per il quale dobbiamo andare a destra è

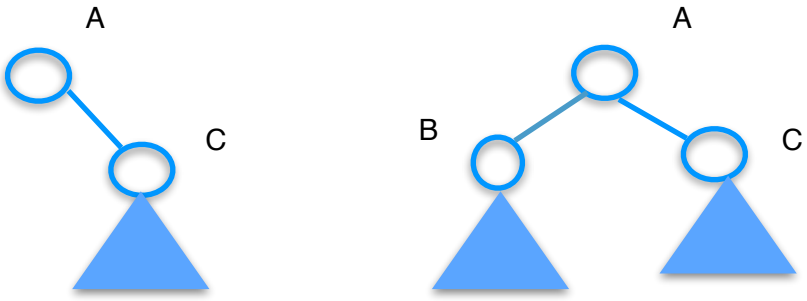


se s = 3 la soluzione deve essere la chiave 8. Infatti s = 3 < 4, quindi il figlio destro di 4 , di chiave 8 è la chiave cercata. Generalizzando consideriamo una catena destra:



se s > A.key, s - A.key > B.key … e se s - (A.key +B.key +…) < C.key questo vuol dire che s < A.key + B.key + …+ C.key e quindi D è la soluzione perché è la più piccola chiave tale che la somma delle chiavi minori è maggiore di s.

se invece anche s - (A.key +B.key +…) > C.key si è arrivati sulla foglia più a destra e allora se s - (A.key +B.key +…+C.key) < 0 allora la risposta è D perché s < (A.key +B.key +…+C.key). Se s > (A.key +B.key +…+C.key) allora la chiave cercata non c'è.
Quindi, mentre sul cammino più a sinistra si continua a scendere fino a che la somma delle chiavi nel sotto albero sinistro è minore di s, al contrario scendiamo sul cammino più a destra fino a che la somma delle chiavi dei nodi attraversati nella discesa è maggiore di s, e a quel punto il figlio destro di quel nodo, se c'è, è quello cercato, altrimenti può essere la foglia più a destra. In generale si deve scendere a destra in due casi:
manca il figlio sinistro oppure la somma delle chiavi nel sotto albero sinistro è minore di s



Una volta diminuito s della chiave di A nel primo caso e della chiave di A più la somma delle chiavi nel sotto albero radicato in B nel secondo caso, il problema su C è esattamente identico al problema iniziale e quindi si chiama la funzione su C, con il valore di s opportunamente diminuito.

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

Pseudocodice:

SommaMinori(T,s)
input: T è un riferimento a un albero binario e s un intero
precondizione: T non è nullo e non è una foglia
output: la più piccola chiave k di T tale che la somma delle chiavi minori di k sia strettamente maggiore di s.

```
SommaMinori(T,s)
if T.sum ≤ s then return  ∞
if T.left
        if T.left.sum > s then
                if T.left.left  then
                    if T.left.leftsum ≤ s then return T else SommaMinori(T.left,s)
                else return T
        else
                if T.right then SommaMinori(T.right,s - (T.left.sum + T.key))
                else return  ∞
else
        if T.right then
               if s < T.key  then return T.right
               else  SommaMinori(T.right,s -T.key)
        else
                 if s < 0  then return T else return  ∞
```

Nota. L'algoritmo presenta delle analogie con la ricerca di una chiave di un dato rango, di cui si è vista la soluzione a lezione. Quella soluzione fu scartata in favore di una iterativa che faceva uso della funzione che calcola la chiave successiva a partire dal minimo, perché non disponendo del campo sum, il calcolo ripetuto del numero dei nodi nel sotto albero sinistro portava a un algoritmo quadratico.