Introduzione agli Algoritmi

Esame Scritto a canali unificati

con idee per la soluzione

docenti: T. CALAMONERI, A. MONTI
Sapienza Università di Roma
27 Giugno 2022

Esercizio 1 (10 punti): Si consideri la seguente funzione:

```
def Exam(A, n):
    b=1
    if n <= 2:  return b
    i = 1
    while i ≤ 8:
        b=b*Exam(A, n//2)
        i+=1
    for i in range(n-1):
        A[i]+=A[i+1]
    return b
```

che viene richiamata la prima volta così: `Exam`$(A, len(A))$.

a) Si imposti la relazione di ricorrenza che ne definisce il tempo di esecuzione giustificando dettagliatamente l'equazione ottenuta.

b) Si risolva la ricorrenza usando due metodi a scelta, dettagliando i passaggi del calcolo e giustificando ogni affermazione.

a) La chiamata ricorsiva viene effettuata 8 volte su un input la cui dimensione è la metà; le operazioni non ricorsive costano $\Theta(n)$. Quindi l'equazione di ricorrenza è: $T(n) = 8T(n/2) + \Theta(n)$ se $n > 2$
$T(n) = \Theta(1)$ altrimenti.

b) Possiamo risolvere con il metodo principale trovando che la soluzione è $\Theta(n^3)$ e verificarla con un altro metodo.

Esercizio 2 (10 punti): Un array A ordinato di $n > 1$ interi distinti ha subito una rotazione di k posizioni verso sinistra, $1 \leq k < n$. Ad esempio, per $A = [5, 7, 9, 2, 3]$ il valore di k è 2 mentre per $A = [9, 2, 3, 5, 7]$ è 4.
Progettare un'algoritmo che, dato l'array A, in tempo $O(\log n)$ restituisca il valore di k.

Dell'algoritmo proposto:

a) si dia la descrizione a parole;

b) si scriva lo pseudocodice;

c) si giustifichi il costo computazionale.

a) Nota che dopo una rotazione verso sinistra di k, $1 \leq k < n$, il valore minimo dell'array A si troverà in posizione $n - k$ (ricorda che le posizioni dell'array di n elementi vanno da 0 ad $n - 1$). Di conseguenza, per risolvere il problema basterà individuare in A la posizione i dell'elemento minimo e restituire il valore $n-i$. Per trovare l'elemento minimo in tempo $O(\log n)$ possiamo applicare la ricerca binaria (infatti l'elemento minimo di A a seguito della rotazione è l'unico che risulta essere preceduto da un elemento di valore maggiore).

b) di seguito una possibile implementazione dell'algoritmo in Python:

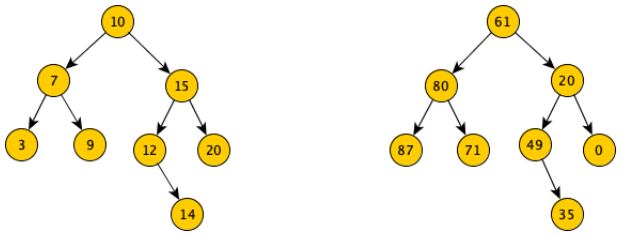
```
def es2(A):
    i, j = 0, len(A) - 1:
    while True:
        m = (i + j)//2
        if A[m] > A[m + 1]:
            return len(A) - (m + 1)
        if A[i] < A[m]:
            i = m + 1
        else:
            j = m
```

c) il costo computazionale dell'algoritmo dipende dal numero di iterazioni del $while$. Ad ogni iterazione del $while$ l'intervallo $[i, j]$ in cui cercare il minimo si dimezza quindi dopo al più $\log n$ iterazioni il minimo viene trovato e l'algoritmo termina. Il costo computazionale dell'algoritmo è dunque $O(\log n)$.

Esercizio 3 (10 punti): Progettare un algoritmo che, dato il puntatore alla radice di un **albero binario di ricerca** T, modifica il valore di ciascun nodo di T in modo

che il nuovo valore del nodo risulti la somma di tutte le chiavi(che, in quanto tali, sono tutte distinte) che in T avevano un valore maggiore della sua chiave originaria.

Ad esempio l'albero sulla destra è il risultato dell'applicazione dell'algoritmo sull'albero binario di ricerca T riportato a sinistra.



Il costo computazionale dell'algoritmo proposto deve essere $\Theta(n)$ dove n è il numero di nodi dell'albero.
Dell'algoritmo proposto:

a) si dia la descrizione a parole;

b) si scriva lo pseudocodice;

c) si giustifichi il costo computazionale.

a) Per ottenere le chiavi dobbiamo visitare i nodi dell'albero; se effettuiamo una visita in una sorta di in-order rovesciato, cioè in modo tale che per ogni nodo visitiamo prima il sottoalbero di destra (contenente le chiavi più grandi) poi il nodo e poi il sottoalbero di sinistra, allora tenendo traccia della somma delle chiavi dei nodi via via visitati, al momento della visita del generico nodo x è disponibile la somma di tutte le chiavi maggiori della chiave di x. Il nodo, grazie a questa informazione, prima di continuare la visita sul suo sottoalbero di sinistra, può aggiornare opportunamente la somma delle chiavi con la sua chiave e, nello stesso tempo, sostituire la sua chiave con la somma ricevuta.
b) nella funzione ricorsiva che segue per semplicità viene utilizzata la variabile globale $somma$ inizializzata a zero, ma è ovviamente possibile farne a meno.

```
def es3(r):
    if r:
        es3(r.right)
        global somma
        somma = somma + r.key
        r.key = somma − r.key
        es3(r.left)
```

c) Il costo computazionale è, come è ovvio, quello della visita di un albero, l'equazione di ricorrenza relativa allo pseudocodice è:

- $T(n) = T(k) + T(n-1-k) + \Theta(1)$
- $T(0) = \Theta(1)$

che si può risolvere con il metodo di sostituzione ottenendo come soluzione $\Theta(n)$.