Introduzione agli Algoritmi
Esame Scritto a canali unificati con spunti per la soluzione

docenti: T. Calamoneri, A. Monti
Sapienza Università di Roma
Ottobre 2023

Esercizio 1 (10 punti): *Siano:*

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

una funzione di costo di un algoritmo ricorsivo $\mathcal{A}$, e

$$T(n) = aT(n/2) + \Theta(1)$$

una funzione di costo di un altro algoritmo ricorsivo $\mathcal{A}'$, dove a è una costante intera positiva maggiore di 1, e per entrambe le ricorrenze vale $T(1) = \Theta(1)$.

Qual è il minimo valore intero della costante a che rende $\mathcal{A}$ asintoticamente più veloce di $\mathcal{A}'$?

Dettagliare il ragionamento ed i passaggi del calcolo, giustificando ogni affermazione.

Applicando alla prima ricorrenza il metodo iterativo si ha $T(n) = \Theta(n^2)$.
Applicando alla seconda ricorrenza il teorema principale si ha $T'(n) = \Theta(n^{log_2 a})$.
Per quanto detto Il primo algoritmo è asintoticamente più efficiente del secondo solo se $n^2 < n^{log_2 a}$, vale a dire $2 < log_2 a$ cioè $4 < a$; quindi, il minimo intero per cui questo accade è $a = 5$.

È necessario dettagliare tutti i passaggi per arrivare all'equazione di ricorrenza e per risolverla.

Esercizio 2 (10 punti):
Un array A di interi positivi si dice valido se i numeri nelle posizioni potenza di due non superano i numeri nelle posizioni potenza di tre ed inoltre i numeri in queste posizioni sono tutti pari.

Ad esempio,

- *l'array* $A = [1, 50, 20, 70, 6, 11, 10, 21, 40, 80, 1, 1, 13, 1, 22, 64, 30, 1]$ *è valido in quanto* $A[2^0] = 50, A[2^1] = 20, A[2^2] = 6, A[2^3] = 40, A[2^4] = 30, A[3^0] = 50, A[3^1] = 70, A[3^2] = 80$.
  Inoltre i numeri nelle posizioni potenza di due non superano i numeri nelle posizioni potenza di tre.
- *l'array* $B = [1, 50, 20, 70, 6, 11, 10, 21, 40, 85, 1, 1, 13, 1, 22, 64, 30, 1]$ *non è valido in quanto* $B[3^2] = 85$ *che è dispari*
- *l'array* $C = [1, 50, 20, 70, 6, 11, 10, 21, 40, 80, 1, 1, 13, 1, 22, 64, 90, 1]$ *non è valido in quanto* $C[2^4] = 90 > C[3^0] = 50$

si scriva un algoritmo **iterativo** *che, dato un array di n elementi, in tempo $O(log n)$ restituisce 1 se A è valido, 0 altrimenti.*

Dell'algoritmo proposto:

a) si scriva lo pseudocodice opportunamente commentato;

b) si calcoli formalmente il costo computazionale.

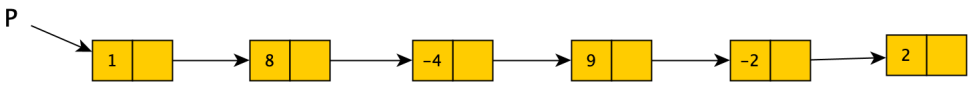
Controlliamo nell'array le sole posizioni potenza di due e potenza di tre per assicurarci che queste contengano solo numeri pari. Inoltre poiché i numeri in posizione potenza di due non superano quelli in posizione potenza di tre se e solo se il numero massimo in posizione potenza di due $massimo2$ non supera il numero minimo in posizione potenza di tre $minimo3$ calcoliamo valori di $massimo2$ e di $minimo3$ e controlliamo che $massimo2$ sia minore o uguale a $minimo3$.
Segue un possibile codice Python

```python
def es2(A):
    if len(A)<1: return 1
    if A[1]%2==1:return 0
    massimo2=minimo3=A[1]
    i=2
    while i < len(A):
        if A[i]%2==1: return 0
        massimo2=max(massimo2,A[i])
        i = 2*i
    i=3
    while i < len(A):
        if A[i]%2==1: return 0
        minimo3=min(minimo3,A[i])
        i = 3*i
    if massimo2<=minimo3: return 1
    return 0
```

Per quanto riguarda la complessità:
ad ogni iterazione del primo while l'indice i raddoppia quindi il primo while termina dopo al più $log_2 n$ iterazioni. Ad ogni iterazione del secondo while l'indice i triplica quindi il secondo while termina dopo al più $log_3 n$ iterazioni. Il costo di entrambi i while è dunque $\Theta(logn)$ e questa è anche la complessità dell'algoritmo.

Esercizio 3 (10 punti): *Abbiamo una lista a puntatori contenente nodi con campo chiave contenente interi e vogliamo sapere il valore massimo ed il valore minimo delle chiavi dei nodi della lista.*

Ad esempio per la lista a puntatori in figura la risposta è la coppia di interi 9 e -4.



Dato il puntatore p al nodo testa della lista di $n \geq 1$ nodi, progettare un algoritmo **ricorsivo** *che, in tempo $\Theta(n)$, risolva il problema.*

Ciascun nodo della lista ha due campi: il campo key contenente il valore chiave ed il campo $next$ al nodo seguente ($next$ è pari a $None$ per l'ultimo nodo della lista).
Dell'algoritmo proposto:

a) si scriva lo pseudocodice opportunamente commentato;

b) si giustifichi formalmente il costo computazionale.

NOTA BENE: nello pseudocodice dell'algoritmo ricorsivo **non** *si deve far uso di variabili globali.*

Sia p il puntatore alla lista, se la lista è vuota si restituisce $None$, altrimenti se la lista e' composta da un unico nodo si restituisce $p.key$ altrimenti si risolve ricorsivamente il problema sulla lista $p.next$ e ottenuti i valori a come massimo e b per la sottolista $p.next$ si restituisce come massimo il massimo tra $p.key$ e a e come minimo il minimo tra $p.key$ e b.
Ecco un possibile codice Python dell'algoritmo che non utilizza variabili globali:

```
def es3(p):
    if p == None:
        return None, None
    if p.next == None:
        return p.key, p.key
    a, b = es3(p.next)
    return max(a, p.key), min(b, p.key):
```

Il costo computazionale è quello della visita di una lista con n nodi. L'equazione di ricorrenza relativa alla visita è:

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

L'equazione si può risolvere con il metodo iterativo (che va esplicitamente svolto) dando come soluzione $\Theta(n)$.
Di conseguenza il costo dell'algoritmo è, come richiesto, $\Theta(n)$.
Nel compito, ogni passaggio va dettagliato.