Algoritmi e Strutture Dati (modulo I) - testo prova scritta 30/01/2024

docenti: Luciano Gualà & Andrea Clementi

Cognome:........................... Nome:............................... Matr.:...............

Esercizio 1 [16 punti]

A: notazione asintotica. Dire quali delle seguenti relazioni asintotiche sono vere:

F $n^{1/4}\log n + \sqrt{\log n} = \Omega(n^{1/3})$; F $\frac{n}{\log^2 n} = o(\frac{n+3}{\log^4 n})$; F $\frac{n^3+\log n}{\sqrt{n}} = \Theta(n^{2.5})$; F $\sqrt[4]{\log n} = O(\log\log n)$;

F $2^{\sqrt{\log n}} = o(n^3)$; V $2^n = \Theta(2^{n-10})$; F $2^{n+2} = \Theta(2^{n/2})$; V $2^{2n} = \Theta(4^n + 2^{n/2})$;

log 1
n 8 JS
t =>
n^0 vs 8

B: equazioni di ricorrenza. Fornire la soluzione asintotica alle seguenti relazioni di ricorrenza:

$T(n) = T(\frac{n}{8}) + 8$; Soluzione: log(n)

$T(n) = T(n-1) + n^3$; Soluzione: O(n4)

C: algoritmi e complessità. Quale algoritmo useresti e quanto costa se devi:

- Costruire un albero AVL contenente n chiavi prese in input: n insert che costo log(n) => n log(n)
- Ordinare n interi compresi fra 1 e n^4: Radix $T(n) = O\left(n \cdot \frac{\log n^4}{\log n}\right) = O(n)$
- Dato un BST di n nodi, restituire tutte le chiavi associate ai nodi in ordine crescente: DFS simmetrica $T(n) = O(n)$
- In un grafo orientato, capire se c'è un cammino da s a t di al più k archi che passa per uno specifico nodo w:

Esercizio 2 [8 punti]

Sia T un albero binario con n nodi, dove ad ogni nodo v è associato un valore positivo $val(v)$. Si progetti un algoritmo che dato T e un valore Δ, restituisca il numero di nodi di T la cui somma dei valori degli antenati è almeno Δ.

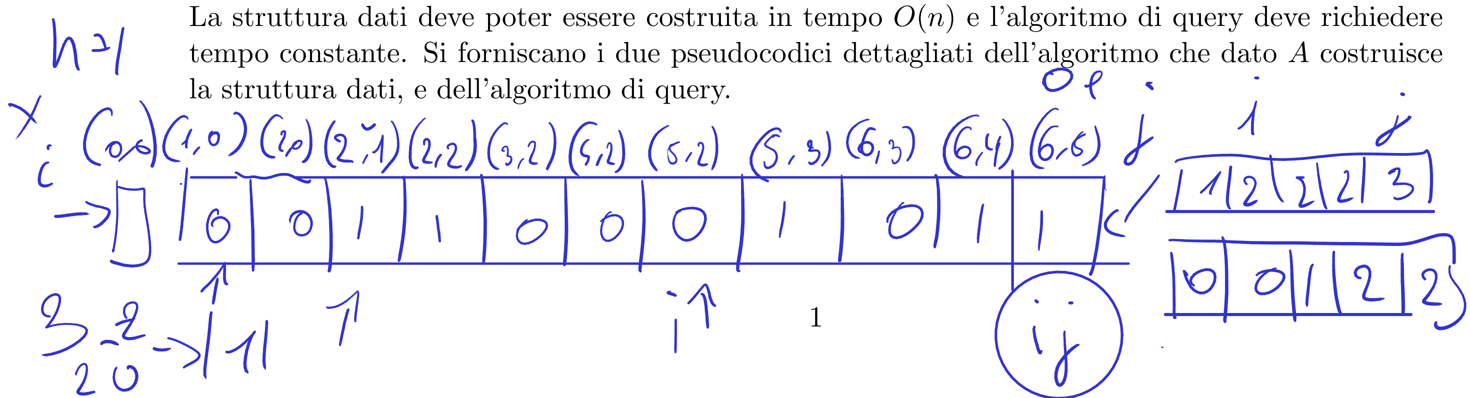
Si assuma che T è rappresentato tramite una struttura dati collegata, con record e puntatori, dove il record di ogni nodo contiene il puntatore al figlio sinistro e al figlio destro del nodo. L'algoritmo deve avere complessità $O(n)$. Si fornisca lo pseudocodice dettagliato.

Esercizio 3 [8 punti]

Sia $A[1:n]$ un vettore di n bit, dove quindi $A[i] \in \{0,1\}$ per ogni i. Si progetti una struttura dati che prende in input il vettore A e sia in grado poi di rispondere a query del seguente tipo:

- `Differenza(i,j)`: dati due indici $i, j \in \{1, 2, \ldots, n\}$ restituisce la differenza in modulo fra il numero di uni e di zeri nel sottovettore $A[i:j]$.

La struttura dati deve poter essere costruita in tempo $O(n)$ e l'algoritmo di query deve richiedere tempo constante. Si forniscano i due pseudocodici dettagliati dell'algoritmo che dato A costruisce la struttura dati, e dell'algoritmo di query.



$$n^{\frac{1}{4}} \log n + \sqrt{\log n} = \Omega(n^{\frac{1}{3}})$$

$$n^{\frac{1}{4}} \log n + \overset{\geq}{\underset{<}{=}} \Omega(n^{\frac{1}{3}})$$

$$\frac{n}{\log^2 n} = O\left(\frac{n+3}{\log^4 n}\right) \qquad \lim_{n \to +\infty} \frac{n}{\log^2 n} \cdot \frac{\log^4 n}{n+3}$$

$$\frac{\cancel{n}}{\cancel{\log^2 n}} \cdot \frac{\log^4 n}{\cancel{n}} \xrightarrow{n \to +\infty} +\infty \qquad \log^2 n$$

$$n^3 \sim \frac{n^3 + \log(n)}{n} = \Theta(n^{\frac{5}{2}})$$

$$n^2 = \Theta(n^{\frac{5}{2}}) \quad F$$

$$\sqrt[4]{\log n} = O(\log(\log(n)))$$

$$\log(n) = x$$

$$\frac{\sqrt[4]{x}}{\log(x)} \xrightarrow{x \to +\infty} +\infty$$

creaOracolo (A)

Sia B un array di dim $n+1$ contenente coppie (nZeri, nUni)

1	2	3	4	5	6	7
0	0	1	1	0	0	1

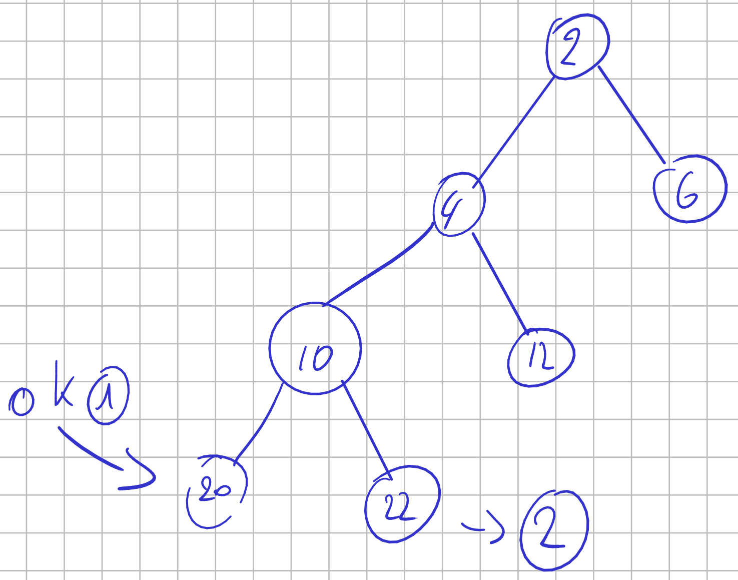
i

0,0	1,0	2,0	2,1	2,2	3,2	4,2	4,3
1	2	3	4	5	6	7	8

i+1

```
B[1] = (0, 0)
for i = 1 to n
    if A[i] == 0
        B[i+1].zeri = B[i].zeri ++
        B[i+1].uni = B[i].uni
    else B[i+1].uni = B[i].uni ++
        B[i+1].zeri = B[i].zeri
return B

Query (B, i, j)
    nZeri = B[j].nZeri - B[i-1].nZeri
    nUni = B[j].nUni - B[i-1].nUni
    return nZeri - nUni if nZeri > nUni
           else nUni - nZeri
```



$\Delta = 20$

Dal alto il la somma dei valori degli antenati
dal basso il numero di nodi che soddisfano la condizione

```
ContaSumAnt(nodo v, sum, delta)
  if v == null return 0
  Sum += val(v)
  sx = ContaSumAnt(sx(v), sum, delta)
  dx = ContaSumAnt(dx(v), sum, delta)
   if sum >= delta then retur 1+ Sx + dx
   else return sx + dx
```