Alberi

Per fare un albero (binario di ricerca) ci vuole...

Dato un vettore V di n interi ordinati e distinti, scrivere una procedura che costruisce un albero binario di ricerca di altezza minima.
Discuterne la correttezza e la complessità.

Divide-et-impera

Chi manca?

Sia dato un vettore ordinato $A[1 \ldots n]$ contenente n elementi interi distinti appartenenti all'intervallo $1 \ldots n+1$. Si scriva una procedura, basata sulla ricerca binaria, per individuare in tempo $O(\log n)$ l'unico intero dell'intervallo $1 \ldots n+1$ che non compare in A.

Punto fisso

Progettare un algoritmo che, preso un vettore ordinato A di n interi distinti, determini se esiste un indice i tale che $A[i] = i$ in tempo $O(\log n)$.

Vettori uni-modulari

Un vettore di interi A è detto unimodulare se ha tutti valori distinti ed esiste un indice h tale che $A[1] > A[2] > \ldots > A[h-1] > A[h]$ e $A[h] < A[h+1] < A[h+2] < \ldots < A[n]$, dove n è la dimensione del vettore. Progettare un algoritmo $O(\log n)$ che dato un vettore unimodulare restituisce il valore minimo del vettore.

Analisi

SortinoSort

Il professor Sortino ha inventato un nuovo algoritmo di ordinamento. Il vettore di input viene diviso in tre parti, di dimensioni approssimativamente uguali $n/3$. Dopo di che, vengono ordinati ricorsivamente le prime due parti del vettore (ovvero i primi due terzi), i secondi due terzi, e di nuovo i primi due terzi.

SortinoSort(**integer**[ ] A, **integer** i, **integer** j)

```
if j − i + 1 ≤ 6 then
    InsertionSort(A, i, j)
else
    integer s ← ⌈(j − i + 1)/3⌉
    SortinoSort(A, i, i + 2s − 1)
    SortinoSort(A, i + s, j)
    SortinoSort(A, i, i + 2s − 1)
```

Analisi

1. Qual è la complessità di questo algoritmo? Il Prof. Sortino finirà sulla prossima edizione del mio libro?
2. (Difficile, Opzionale) Dimostrare per induzione che questo algoritmo è corretto. Per comodità, assumete pure che tutti i valori siano distinti.

Spoiler alert!

Alberi

Per fare un albero (binario di ricerca) ci vuole...

TREE build-tree-rec(**integer**[ ] A, **integer** i, **integer** j)

---

if $i \leq j$ **then**
    integer $m \leftarrow (i + j)/2$
    TREE $T \leftarrow$ **new** TREE
    $T.left \leftarrow$ build-tree-rec$(A, i, m - 1)$
    $T.right \leftarrow$ build-tree-rec$(A, m + 1, j)$
    $T.key \leftarrow V[m]$
    return T
else
    return nil

L'equazione di ricorrenza è pari a: $T(n) = 2T(n/2) + 1$, che dà origine

Chi manca?

```
integer missing(integer[] A, integer i, integer j)
if i = j then
    if A[i] = i then
        return i + 1
    else
        return i
m ← ⌈(i + j)/2⌉
if A[m] = m then
    return missing(A, m + 1, j)
else
    return missing(A, i, m)
```

Punto fisso

boolean puntofisso(**integer**[ ] A, **integer** i, **integer** j)

```
if j > i then
    return false
integer m ← ⌊(i + j)/2⌋
if A[m] = m then
    return true
else if A[m] < m then
    return puntofisso(A, m + 1, j)
else
    return puntofisso(A, i, m − 1)
```

Vettori unimodulari

```
integer vum(integer[] A, integer i, integer j)
if i = j then
    return A[i]
if j = i + 1 then
    return min(A[i], A[j])
integer m = (i + j)/2
if A[m − 1] > A[m] and A[m + 1] > A[m] then
    return A[m]
if A[m − 1] > A[m] then
    return vum(A, m + 1, j)
else
    return vum(A, i, m − 1)
```

SortinoSort – 12/09/10

La complessità è rappresentata dalla seguente equazione di ricorrenza:

$$T(n) = \begin{cases} 3T(\frac{2}{3}n) + 1 & n > 6 \\ 1 & n \leq 6 \end{cases}$$

Utilizzando il teorema delle ricorrenze lineari con partizione bilanciata, si ottiene che $a = 3$, $b = 3/2$, da cui $\alpha = \log_{3/2} 3$; inoltre, $\beta = 0$. Siamo quindi nel caso $T(n) = n^{\alpha}$.
Non avete una calcolatrice e non sapete quanto sia $\log_{3/2} 3$?

SortinoSort – 12/09/10

La complessità è rappresentata dalla seguente equazione di ricorrenza:

$$T(n) = \begin{cases} 3T(\frac{2}{3}n) + 1 & n > 6 \\ 1 & n \leq 6 \end{cases}$$

Utilizzando il teorema delle ricorrenze lineari con partizione bilanciata, si ottiene che $a = 3$, $b = 3/2$, da cui $\alpha = \log_{3/2} 3$; inoltre, $\beta = 0$. Siamo quindi nel caso $T(n) = n^{\alpha}$.
Non avete una calcolatrice e non sapete quanto sia $\log_{3/2} 3$?

E' semplice: $\left(\frac{3}{2}\right)^2 = \frac{9}{4} < 3$, mentre $\left(\frac{3}{2}\right)^3 = \frac{27}{8} > 3$. Quindi α è compreso fra 2 e 3, e quindi questo algoritmo è addirittura peggiore di Insertion Sort. Il prof. Sortino non finirà nel mio libro.