Crazy

Trovare un limite superiore e inferiore al costo computazionale del seguente algoritmo, dando una dimostrazione formale.

---

integer crazy(**integer** n)

```
if n ≤ 1 then
    return n · n
else
    integer b ← 1
    for i ← 1 to n do
        for j ← i to n do
            b ← (b + i ∗ j) mod 1007
    return b + crazy(⌊n/4⌋) + crazy(⌊n/2⌋) + crazy(⌊n/4⌋)
```

Alberi

Sopra e sotto

Scrivere un algoritmo che prende in input un albero binario T e restituisca in output il numero di nodi di tale albero il cui numero di ascendenti (il padre, il nonno (padre del padre), il bisnonno (padre del nonno), etc) è uguale al numero di discendenti (i figli, i nipoti (figli dei figli), i bisnipoti (figli dei nipoti), etc).
Discutere correttezza e complessità dell'algoritmo proposto.

Pozzo universale

Data una rappresentazione con matrice di adiacenza, progettare un algoritmo che opera in tempo $\Theta(n)$ in grado di determinare se un grafo orientato contiene un pozzo universale: ovvero un nodo con out-degree uguale a zero e in-degree uguale a $n-1$. È possibile ottenere la stessa complessità con liste di adiacenza?

Samarcanda

Samarcanda

Nel gioco di Samarcanda, ogni giocatore è figlio di una nobile famiglia della Serenissima, il cui compito è di partire da Venezia con una certa dotazione di denari, arrivare nelle ricche città orientali, acquistare le merci preziose al prezzo più conveniente e tornare alla propria città per rivenderle.

Dato un vettore P di n interi in cui $P[i]$ è il prezzo di una certa merce al giorno i, trovare la coppia di giornate (x, y) con $x < y$ per cui risulta massimo il valore $P[y] - P[x]$. Calcolare la complessità e dimostrare la correttezza. È possibile risolvere il problema in $O(n)$.

Spoiler alert!

Crazy

L'equazione di ricorrenza della funzione crazy() è la seguente:

$$T(n) = \begin{cases} 2T(\lfloor n/4 \rfloor) + T(\lfloor n/2 \rfloor) + n^2 & n > 1 \\ 1 & n \leq 1 \end{cases}$$

E' facile vedere che $T(n) = \Omega(n^2)$; proviamo a dimostrare che $T(n) = O(n^2)$.

- Caso base: $n = 1$, $T(n) = 1 \leq cn^2 = c$, ovvero $c \geq 1$.
- Ipotesi induttiva: $\forall n' < n : T(n') \leq c(n')^2$
- Passo induttivo:

$$\begin{aligned} T(n) &= 2T(\lfloor n/4 \rfloor) + T(\lfloor n/2 \rfloor) + n^2 \\ &\leq 2c\lfloor n/4 \rfloor^2 + c\lfloor n/2 \rfloor^2 + n^2 \\ &\leq 2cn^2/16 + cn^2/4 + n^2 \\ &= 3/8cn^2 + n^2 \leq cn^2 \end{aligned}$$

L'ultima disequazione è vera per $c \geq 8/5$.

Abbiamo quindi dimostrato che $T(n) = \Theta(n^2)$, con $c \geq 8/5$ e $m = 1$.

Sopra e sotto

(**integer**, **integer**) countTreeRec(Tree t, **integer** *ancestors*)

---

if $t = \textbf{nil}$ **then**
  return (0,0)

$pred_L, count_L \leftarrow \mathsf{countTreeRec}(t.left, ancestors + 1)$
$pred_R, count_R \leftarrow \mathsf{countTreeRec}(t.right, ancestors + 1)$
return
  $(pred_L + pred_R + 1, count_L + count_R + \mathsf{iif}(ancestors = pred_L + pred_R, 1, 0))$

L'algoritmo viene invocato dalla seguente funzione wrapper:

---

countTree(Tree t)

$pred, count \leftarrow \mathsf{countTreeRec}(t, 0)$
return *count*

Pozzo universale

universalSink(**integer**[][] A)

$i \leftarrow 1$
$candidate \leftarrow$ **false**
while $i < n \wedge candidate =$ **false do**
    $j \leftarrow i + 1$
    while $j \leq n \wedge A[i, j] = 0$ **do**
        $j \leftarrow j + 1$
    if $j > n$ **then**
        $candidate \leftarrow$ **true**
    else
        $i \leftarrow j$
$rowtot = \sum_{j \in \{1 \ldots n\} - \{i\}} A[i, j]$
$coltot = \sum_{j \in \{1 \ldots n\} - \{i\}} A[j, i]$
return $rowtot = 0 \wedge coltot = n - 1$

Samarcanda

Samarcanda

Suggerimento: calcolate il vettore delle differenze fra le quotazioni, e applicate la somma massimale vista alla prima lezione.