Ricorrenza $2T(n/8) + 2T(n/4) + n$

Trovare un limite asintotico superiore e un limite asintotico inferiore alla seguente ricorrenza, facendo uso del metodo di sostituzione:

$$T(n) = \begin{cases} 1 & n \leq 1 \\ 2T(n/8) + 2T(n/4) + n & n > 1 \end{cases}$$

Alberi – Indovina l'albero

Gli ordini di visita di un albero binario di 9 nodi sono i seguenti:

- A, E, B, F, G, C, D, I, H (anticipato)
- B, G, C, F, E, H, I, D, A (posticipato)
- B, E, G, F, C, A, D, H, I (simmetrico).

Si ricostruisca l'albero binario e si illustri brevemente il ragionamento.

Alberi – Albero livello–valore

Scrivere un algoritmo che preso in input un albero binario T i cui nodi sono associati ad un valore intero $T.value$, restituisca il numero di nodi dell'albero il cui valore è uguale al livello del nodo.

Vi ricordo che il livello del nodo è pari al numero di archi che devono essere attraversati per raggiungere il nodo dalla radice. Per cui la radice ha livello 0, i suoi figli hanno livello 1, etc.

Crazy

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

```
int crazy(int n)
if n ≤ 1 then
    return n · n
else
    int 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⌋)
```

Spoiler alert!

Ricorrenza $2T(n/8) + 2T(n/4) + n$

$$T(n) = \begin{cases} 1 & n \leq 1 \\ 2T(n/8) + 2T(n/4) + n & n > 1 \end{cases}$$

È facile dimostrare che ovviamente, la funzione $T(n)$ è $\Omega(n)$; infatti,

$$T(n) = 2T(n/8) + 2T(n/4) + n \geq n \geq c_1 n$$

per $c_1 \leq 1$. Non è necessario dimostrare il caso base, perché abbiamo rimosso gli elementi ricorsivi e quindi non abbiamo bisogno di induzione.

Ricorrenza $2T(n/8) + 2T(n/4) + n$

$$T(n) = \begin{cases} 1 & n \leq 1 \\ 2T(n/8) + 2T(n/4) + n & n > 1 \end{cases}$$

Una prima osservazione che si potrebbe fare per "indovinare" un limite superiore è la seguente:

$$\begin{aligned} T(n) &= 2T(n/8) + 2T(n/4) + n \\ &\leq 2T(n/4) + 2T(n/4) + n \\ &\leq 4T(n/4) + n \end{aligned}$$

In base a questo risultato, il master theorem dice che $T(n) = O(n \log n)$.

Ricorrenza $2T(n/8) + 2T(n/4) + n$

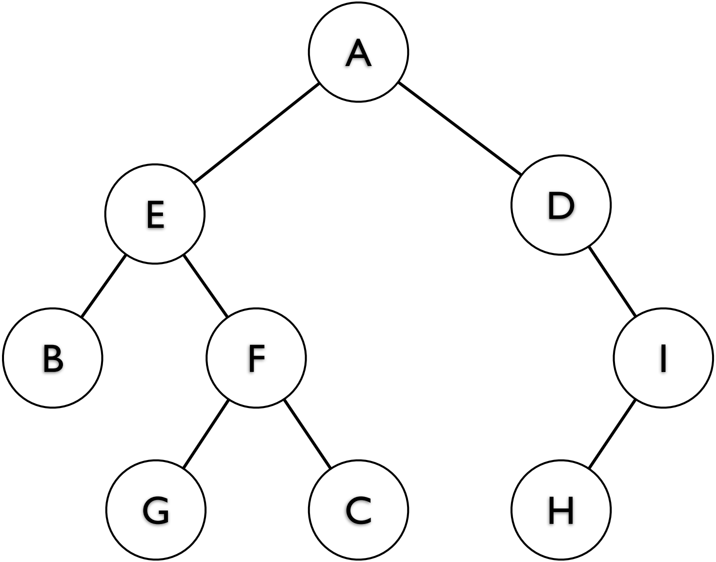
$$T(n) = \begin{cases} 1 & n \leq 1 \\ 2T(n/8) + 2T(n/4) + n & n > 1 \end{cases}$$

A questo punto, non ci resta che vedere quale dei due risultati ($\Omega(n)$ e $O(n \log n)$ è stretto. Proviamo con $O(n)$.
Ipotesi induttiva: $\forall k < n : T(k) \leq ck$. Proviamo che il risultato è valido anche per n.

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

L'ultima disequazione è vera se $3/4c + 1 \leq c$, ovvero se $c \geq 4$.

Indovina l'albero



Albero livello-valore

Una semplice visita posticipata risolve il problema. La complessità è ovviamente $O(n)$.

```
int sameLevel(TREE T)
---
return sameLevelRec(T, 0)
```

```
int sameLevelRec(TREE T, int ℓ)
---
int tot = 0
if T ≠ nil then
    tot = sameLevelRec(T.right(), ℓ + 1) + sameLevelRec(T.left(), ℓ + 1)
    if T.value == ℓ then
        tot = tot + 1
return tot
```

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 $m = 1$.