Ripasso sugli algoritmi ricorsivi:

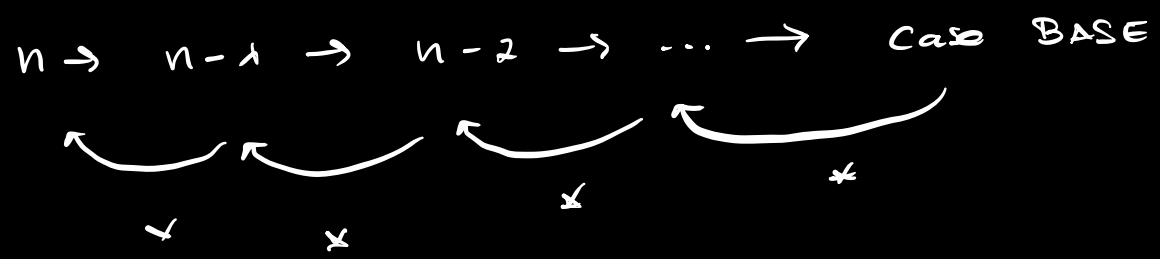
Per scrivere un problema in modo ricorsivo (se possibile), devo riuscire a trovare il caso base ed il caso passo, combinando correttamente i risultati parziali delle sottochiamate (niente di nuovo).

Esempio: Calcoliamo ricorsivamente la potenza n-esima di un numero.

```
Potenza (A , n):

    if ( n == 1 ):
        // Caso base, ce so risolvere immediatamente e richiama le funzioni sottochiamate
        return A;
    else:
        // Caso passo che crea una sottochiamata e la combina correttamente
        // con la computazione corrente.
        return A * (A,  n - 1);
```

Posso rappresentare una funzione ricorsiva con l' albero delle sottochiamate che genera:



Per il calcolo dei tempi invece posso:

Fornire la funzione di riccorrenza (in questo caso ad esempio sarebbe):

$$T(n) = \begin{cases} C & \text{Se } n == A \\ 2C + T(n-1) & \text{Se } n > A \end{cases}$$

Che può essere "srotolata" per una migliore comprensione della funzione:

$2c + T(n-1) = 2c + [\,2c + T(n-2)] = 4c + [\,2c + T(n-3)] = 6c + [2c + T(n-4)] = \ldots$. E così via

Alla fine possiamo notare un pattern: la chiamata ricorsiva "generica ha la faccia":

$2k*c + T(n-k)$.

Per un k = n (moltiplicazione di un numero per sè stesso n volte = potenza di quel numero),

otteniamo : n*2c + T(n -n) = n*2c = $\Theta(n)$

Ok, abbiamo tutti gli strumenti per parlare di ricorsione :D

Quante volte n° 5 in array:

```
count ( A[], int i )
    if ( i > A[].length : 1
        return 0;          2
} else {
    if ( A[i] == 5 )
        return 1 + count (A[], i+1)
    else {
        return count (A[], i+1)
    }
```

$$\begin{cases} 2C & \text{se } n = 0 \\ 1 & \end{cases}$$