Ricorsione

martedì 5 aprile 2022 10:58

- Per risolvere un problema la funzione chiama se stessa
- Non sempre è vantaggiosa utilizzarla, bisogna prestare attenzione a:
  - Facilità di implementazione
  - Vantaggi ottenuti (calcolando che noi continuiamo a fare chiamate)
- Abbiamo bisogno dei cosiddetti "casi basi" per darci uno stop.
  E, dopo il caso base abbiamo un caso che ci permette di avvicinarci
  Al nostro caso base
  If casoBase:
  Return valoreBase;
  Else.
  Return funzioneRicorsiva() ## Regole di costruzione
- Si base sull'induzione:
  - Se P(n) è vero
  - Per induzione dico che P(n) è sempre vero e,
    Se dimostro che P(n+1) è vero allora sarà sempre vero $\forall n$

Esempi:

- $n^3 + 2n \,\%\, 3 == 0 \; \forall n \geq 0$
  - N = 0 -> 0 -> vero
  - $(n+1)^3 + 2(n+1) = n^3 + 3n^2 + 3n + 1 + 2n + 2$
    $n^3 + 2n + \; 3n + 1 + 3n + 3 = 3k + 3(n^2 + n + 1)$

- $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$
  - $n = 1 \rightarrow \frac{1 * 2}{2} = 1 \rightarrow vero$
  - $Dimostrare: \frac{(n+1)(n+2)}{2}$
  - $\sum_{i=1}^{n+1} i = \sum_{i=1}^{n} i + (n+1) = (calcoli) = \frac{(n+1)(n+2)}{2}$

- Dimostrare che, partendo da 8 centesimi è possibile sempre ottenerlo sommando 3 e 5 centesimi
  $\forall n \geq 8 \quad n = k_1 * 3 + k_2 * 5$

  In questo caso, proviamo a fare tanti casi e vediamo se è possibile ritrovare un pattern
  N = 8 -> 5+3
  N = 9 -> 3+3
  N = 10 -> 5+5
  N = 11 -> 5+3+3 = 8+3
  N = 12 -> 3+3+3 = 9 + 3
  N = 13 -> 5+5+3 = 10 + 3
  Possiamo ora notare un pattern:
  I nostri casi base sono 8, 9, 10 su cui sappiamo valore
  E tutti gli altri valori maggiori non sono altro che f(n-3)+3

- Fattoriale
  Int fat(n):
  If N == 0: ## Il nostro caso base
  Return 1
  Else:
  Parziale = fat(n-1)

```
        Totale = N*parziale
        Return totale;
```

- Potenza di A dato N

```
Int potenza(A, N):
    If N == 0:
        Return 1
    Else:
        Ris = A * potenza(A, N - 1)
        Return ris;
```

- Eseguire i seguenti comandi ricorsivamente:
  MCD(0, N) = N -> Caso base
  MCD(M, N) = MC(M, N - M)
  MCD(M, N) = MCD(N, M)

  Detto questo, noi vogliamo raggiungere 0, però notare che NON dobbiamo
  Andare in negativo quindi, il primo valore deve essere più grande del secondo.

```
Int MCD(M, N):
    If n < m:
        App = n
        N = m
        M = App
    If n == 0:
        Return M
    Else:
        ris = MCD(m, n-m)
        Return ris;
```

- Data una stringa e un carattere, determinare ricorsivamente
  Quante volte quel carattere viene ripetuto.
  Useremo un indice e scorreremo questo indice.
  Il caso base è indice = 1

```
Int trova(char car, char A[], int pos):
    If pos == 1:
        If A[1] == 'z':
            Return 1
        Else:
            Return 0
    Else:
        Ris = trova(car, A, pos - 1)
        If A[pos] == 'z':
            Ris++;
        Return ris;
```

- Fibunacci (int n)

```
    If n == 1 or n == 2
        Return 1
    Else
        Ris = fibunacci(n-1)+fibunacci(n-2)
        Return Ris
```