La ricorsione

- La ricorsione è una tecnica di programmazione attraverso la quale un sottoprogramma chiama se stesso.

- Un esempio è il calcolo del fattoriale di un numero naturale. Il fattoriale di 0 è 1. Il fattoriale di n, numero naturale maggiore di 0, è dato dal prodotto di tutti i numeri interi compresi tra 1 ed n.

- Es: 0!=1, 1!=1, 5!=5*4*3*2*1=120

Fattoriale versione iterativa

Input: numero naturale n
Output: fattoriale di n (n!)

```
sottoprogramma fattoriale(in:n; out: fatt)
{
 fatt ← 1
 mentre (n>0) fai
 {
   fatt ← fatt*n
   n ← n-1
 }
}
```

Fattoriale versione ricorsiva

$$\text{fattoriale}(n) = \begin{cases} 1 & \text{se } n=0 \\ n*\text{fattoriale}(n-1) & \text{se } n>0 \end{cases}$$

Fattoriale (versione ricorsiva)

```
sottoprogramma fattoriale(in:n; out: fatt)
 1. se (n=0)allora
          1.1 fatt ← 1
altrimenti
          {
          1.2 fattoriale(in:(n-1); out: fatt1)
          1.3 fatt ← n*fatt1
          }
```

Fibonacci

Immaginiamo di chiudere una coppia di conigli in un recinto.

Sappiamo che ogni coppia di conigli:

a) inizia a generare dal secondo mese d'età
b) genera una coppia di conigli al mese
c) non muore mai.

Quanti conigli ci saranno nel recinto dopo un anno ?

Fibonacci
soluzione ricorsiva

fib(0) = 1
fib(1) = 1
fib(n) = fib(n-1)+fib(n-2) se n>1

dove fib(k)="numero di coppie di conigli al mese k"

L'algoritmo

```
sottoprogramma Fibonacci(in: n; out: num_conigli)
{
   1. se ((n=0) o (n=1)) allora
                    1.1 num_conigli ← 1
            altrimenti
                    {
                     1.2 Fibonacci(in:n-2;out:num_conigli1)
                     1.3 Fibonacci(in:n-1;out:num_conigli2)
                     1.4 num_conigli ← num_conigli1+num_conigli2
                    }
}
```

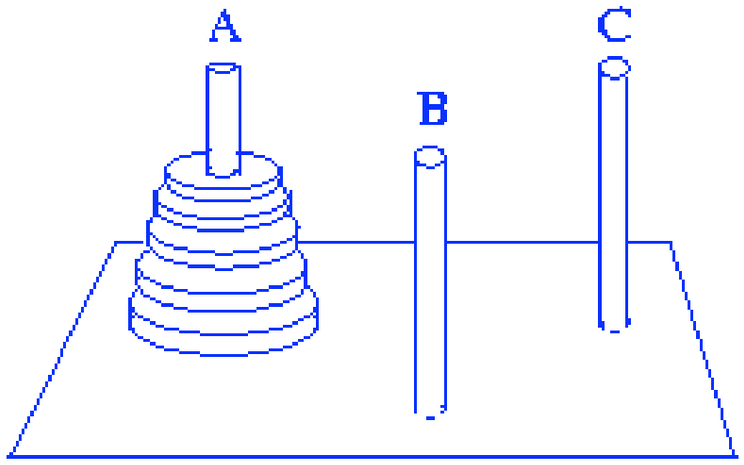
Fibonacci
Soluzione iterativa

Fibonacci
Soluzione iterativa

```
sottoprogramma Fibonacci_it(in:N;out:num_conigli)
{
  1. num_conigli_prec ← 1
  2. num_conigli ← 1
  3. I ← 2
  3. mentre (I <= N) fai {
      3.1 aux ← num_conigli
      3.2 num_conigli ← num_conigli + num_conigli_prec
      3.3 num_conigli_prec ← aux
      3.4 I ← I+1
     }
}
```

... e dopo un anno ci saranno 233 coppie di conigli

La torre di Hanoi

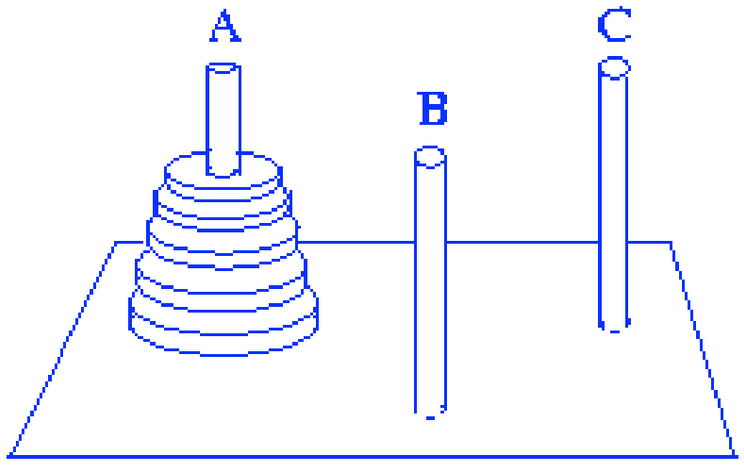


Il fine del gioco è trasferire i dischi dal piolo A al piolo C.

Regole:

- muovere un disco alla volta
- mai mettere un disco più largo su uno più stretto.

La torre di Hanoi



Il fine del gioco è trasferire i dischi dal piolo A al piolo C.

Regole:

- muovere un disco alla volta
- mai mettere un disco più largo su uno più stretto.

Soluzione ricorsiva:

- trasferisci N-1 dischi da A a B (C d'ausilio)
- muovi il disco più largo da A a C.
- trasferisci N-1 dischi da B a C (A d'ausilio)

La torre di Hanoi

La torre di Hanoi

```
sottoprogramma MuoviDisco(in:X,Y){
   1. stampa “Muovi il disco da”
   2. stampa X
   3. stampa “a”
   4. stampa Y
}
```

sposta 1 disco da X a Y

La torre di Hanoi

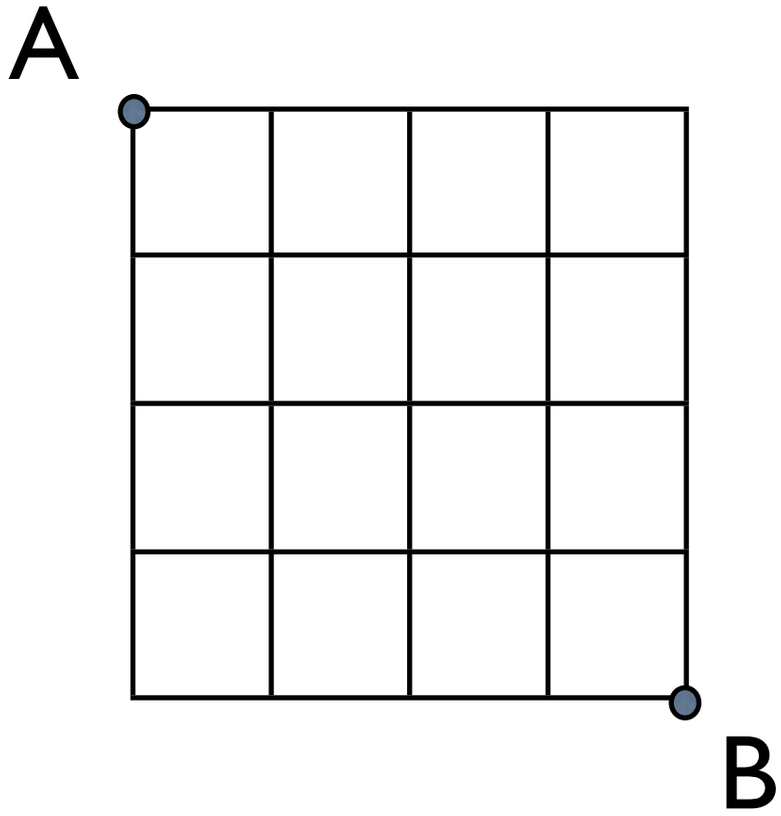
```
sottoprogramma MuoviDisco(in:X,Y){
  1. stampa “Muovi il disco da”
  2. stampa X
  3. stampa “a”
  4. stampa Y
}
```

sposta 1 disco da X a Y

```
sottoprogramma Hanoi(in:n,X,Z,Y)
{
1. se (n=1) allora
     1.1 MuoviDisco(in: X,Z)
   altrimenti
   {
     1.2 Hanoi(in: (n-1),X,Y,Z)
     1.3 MuoviDisco(in:X,Z)
     1.4 Hanoi(in: (n-1),Y,Z,X)
   }
}
```

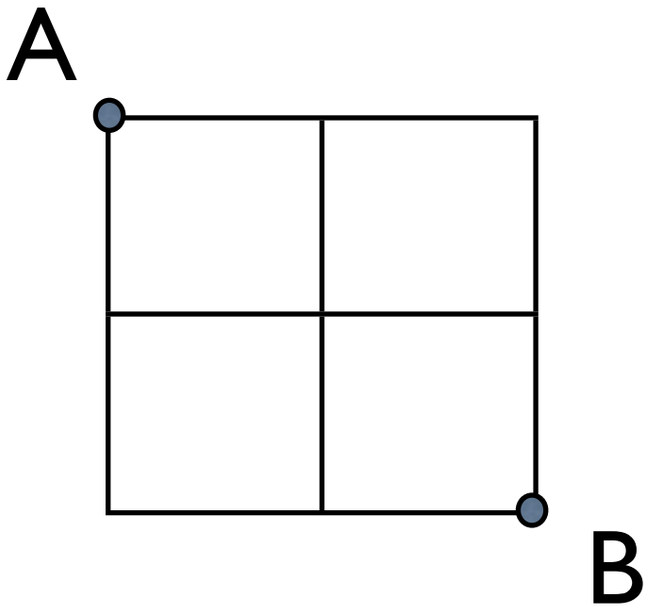
sposta n dischi da X a Z
passando per Y

Strade di Manhattan



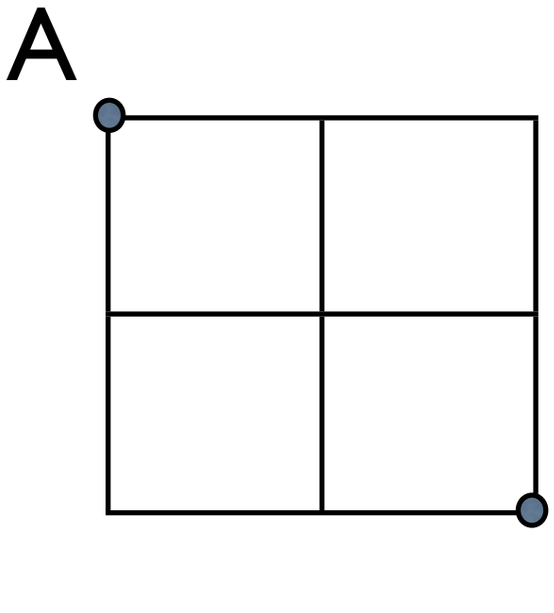
Data una griglia nxn, calcolare tutti i cammini di lunghezza minima da A (angolo superiore sx) a B (angolo inferiore dx)

Esempio



Quanti cammini minimi ci sono in una griglia 2x2?

Esempio



Quanti cammini minimi ci sono in una griglia 2x2?

6 cammini minimi da A a B!

Relazione di ricorsione

man(0,n) = 1

man(m,0) = 1

man(m,n) = man(m-1,n)+man(m,n-1) se m,n>0

L'algoritmo

```
sottoprogramma man(in:n,m;out:streets)
{
1. se (n=0) or (m=0) allora
     1.1 streets=1
   altrimenti
   {
     1.2 man(in:(n-1),m;out:str1)
     1.3 man(in:n,(m-1);out:str2)
     1.4 streets=str1+str2
   }
}
```