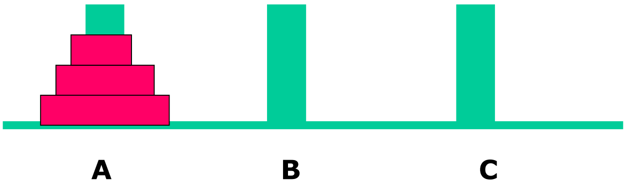
Altri algoritmi

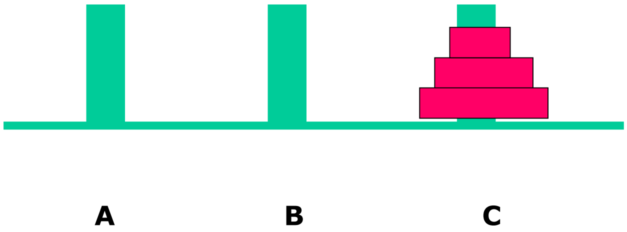
Lezione 4

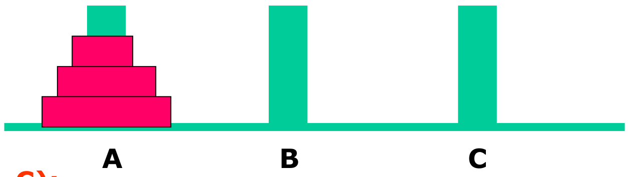
Torre di Hanoi

- 3 paletti
- bisogna spostare la torre di n cerchi dal paletto sorgente A a quello destinatario C usando un paletto ausiliario B
- un cerchio alla volta e
- mai un cerchio sopra uno piu' piccolo

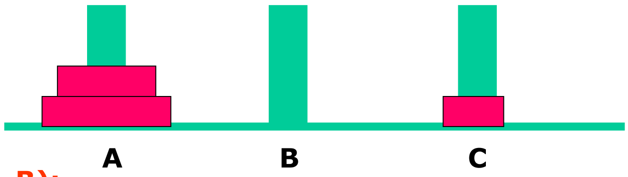


Torre di Hanoi

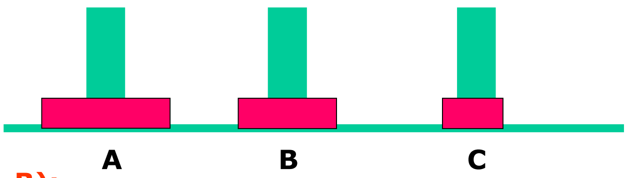




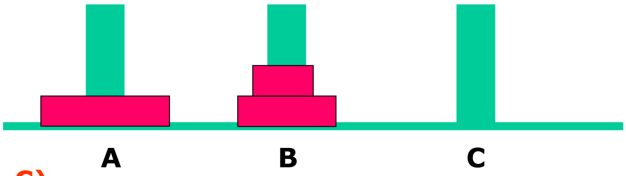
sposta(A, C);



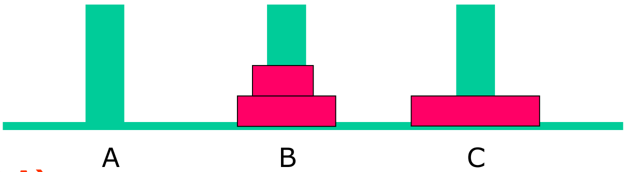
sposta(A, B);



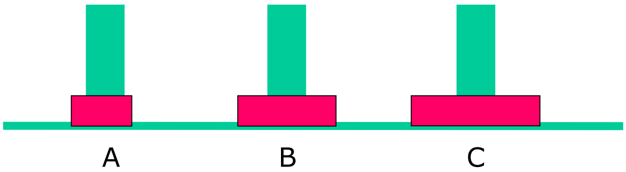
sposta(C, B);



sposta(A,C)

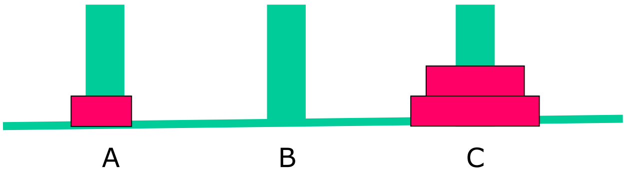


sposta(B,A)

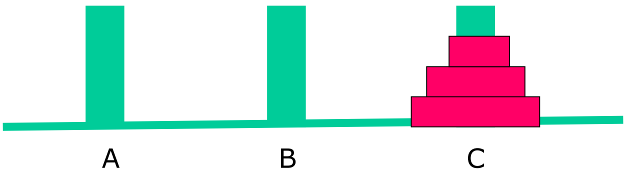


sposta(B,C)

5



sposta(A,C)



6

Torre di Hanoi

void trasferisci una torre di n cerchi da A a C {

Se n=1 sposta il cerchio da A a C;

altrimenti

{ trasferisci la torre degli n-1 cerchi più piccoli da A a B
usando C come paletto ausiliario;
sposta il cerchio più grande da A a C;
trasferisci la torre degli n-1 cerchi più piccoli da B a C
usando A come paletto ausiliario;
} }

Torre di Hanoi

```
void hanoi(int n, pal A, pal B, pal C)
{
   if (n == 1)
      sposta(A, C);
   else {
      hanoi(n - 1, A, C, B);
      sposta(A, C);
      hanoi(n - 1, B, A, C);
   }
}
```

$T(1) = a$

$T(n) = b + 2T(n-1)$

hanoi(3, A, B, C)

hanoi(2, A, C, B)

hanoi(1, A, B, C)

sposta(A, C);

sposta(A, B);

hanoi(1, C, A, B)

sposta(C, B);

sposta(A,C);

hanoi(2, B, A, C)

hanoi(1, B, C, A)

sposta(B, A);

sposta(B, C);

hanoi(1, A, B,C)

sposta(A,C);

9

soluzione

$T(1) = a$
$T(n) = b + 2T(n-1)$

$T(1) = a$

$T(2) = b + 2a$

$T(3) = b + 2b + 4a = 3b + 4a$

$T(4) = 7b + 8a$

.

.

$T(n) = (2^{(n-1)} - 1)b + 2^{(n-1)} a$

T(n) è $O(2^n)$

10

Classe di relazioni - Metodo divide et impera

```
void dividetimpera( S ) {

if ( |S|<= m )
        <risolvi direttamente il problema>;
else  {
     <dividi S in b sottoinsiemi S1.. Sb>;
      dividetimpera(S i1 );
      ...
      dividetimpera(S ia );

      <combina i risultati ottenuti>;
        }
  }
```


I problemi che hanno questa complessità richiedono un tempo cn per la combinazione dei risultati oltre alla riapplicazione dell'algoritmo su un certo numero a di sottoinsiemi, possibilmente bilanciati in dimensione.
Quando il tempo della combinazione è indipendente da n possiamo avere relazioni del tipo e m=1

$T(n) = d \qquad n \leq 1$
$T(n) = aT(n/b) + c \qquad n > 1$

$T (n) = d$
$T (n) = c + T(n/2) \qquad O(\log n) \qquad a=1$

$T (n) = d$
$T (n) = c + 2T(n/2) \qquad O(n) \qquad a=b=2$

Metodo divide et impera

Se il tempo di combinazione è invece dipendente da n e ancora a=b=2

$T(n) = d$ $\quad n \leq 1$ $\quad O(n \log n)$

$T(n) = cn + 2T(n/2)$ $\quad n>1$

Forma generale del Metodo divide et impera

$$\boxed{\begin{array}{ll} T(n) = d & \text{se } n \leq m \\ T(n) = cn^k + aT(n/b) & \text{se } n > m \end{array}}$$

$c > 0$

$T(n) \in O(n^k)$ $\quad$ se $a < b^k$

$T(n) \in O(n^k \log n)$ $\quad$ se $a = b^k$

$T(n) \in O(n^{\log_b a})$ $\quad$ se $a > b^k$

Algoritmi sui numeri

La complessità è calcolata prendendo come misura il numero di cifre che compongono il numero

Ad esempio:

- **L'addizione ha complessità O(n)**
- **la moltiplicazione che studiamo alle elementari ha complessità $O(n^2)$**
  - **Come mai?**
  - **Faccio n*n moltiplicazioni e n somme**

Applicazioni del divide et impera

Cerchiamo di ridurre la complessità della moltiplicazione tra due numeri di n cifre effettuando m moltiplicazioni tra numeri di n/2 cifre

$$A = A_s 10^{n/2} + A_d$$

$$B = B_s 10^{n/2} + B_d$$

$$A=1325 = 13*10^2 + 25 \quad n=4$$

n/2	n/2
A_s	A_d
13	25

Se faccio m=4 moltiplicazioni?

Moltiplicazione veloce fra interi non negativi

$A*B = (A_s 10^{n/2} + A_d)*(B_s 10^{n/2} + B_d)$
$= A_sB_s10^n + (A_sB_d+A_dB_s)\ 10^{n/2} + A_dB_d$

poiché $(A_s+A_d)(B_s+B_d) = A_sB_d + A_dB_s + A_sB_s+A_dB_d$

$A_sB_d+A_dB_s = (A_s+A_d)(B_s+B_d) - A_sB_s - A_dB_d$ e quindi

$A*B = A_sB_s10^n + ((A_s+A_d)(B_s+B_d) - A_sB_s - A_dB_d)10^{n/2} + A_dB_d$

La moltiplicazione di due numeri di n cifre può essere ridotto a 3 moltiplicazioni di numeri di n/2 cifre più un tempo lineare per le somme le sottrazioni e gli shift.

Scriviamo il programma per calcolare

$AB = A_sB_s10^n + ((A_s+A_d)(B_s+B_d) - A_sB_s - A_dB_d)10^{n/2} + A_dB_d$

```
numero mult ( numero A, numero B, int n ) {
if ( n=1)  return A * B;        O(1)
else  {
 As = parte sinistra di A ; Ad = parte destra di A ;  O(n/2)
 Bs = parte sinistra di B ; Bd = parte destra di B ;  O(n/2)
 int x1= As+Ad ; int x2= Bs+Bd ; O(n/2)  (somma di due numeri)
 int y1= mult (x1, x2 ,n/2);
 int y2= mult (As, Bs , n/2);
 int y3= mult (Ad, Bd ,n/2);
 int z1= left_shift(y2,n);              O(n)
 int z2= left_shift (y1-y2-y3, n/2);    O(n/2)
 return z1+z2+y3;
}
```

left_shift(h,n) scorre h di n posti a sinistra facendo entrare altrettanti 0 ($*10^n$)

Moltiplicazione veloce

$$T(n) = d \qquad n \le 1$$
$$T(n) = cn + 3T(n/2) \qquad n > 1$$

$$T(n) \in O(n^{\log_2 3})$$

$$T(n) \in O(n^{1.59})$$

La classe delle Relazioni di ricorrenza lineari

$$T(n) = d \qquad n \le 1$$
$$T(n) = c + T(n-1) \qquad n > 1 \qquad O(n)$$

$$T(n) = d \qquad n \le 1$$
$$T(n) = c + 2T(n-1) \qquad n > 1 \qquad O(2^n)$$

Forma generale delle Relazioni di ricorrenza lineari

$T(n) = d$
$T(n) = cn^k + a_1T(n-1) + a_2T(n-2) + a_rT(n-r)$

polinomiale se esiste al più un solo $a_i = 1$ e gli altri a_i sono tutti 0 (c'è una sola chiamata ricorsiva). Negli altri casi sempre esponenziale.

Soluzione polinomiale

$T(n) = d$
$T(n) = cn^k + T(n-1)$

$c > 0$

$T(n) \in O(n^{k+1})$

Caso particolare della soluzione polinomiale

$T(n) = a \quad n \leq 1$

$T(n) = cn + T(n-1) \quad n>1$

$O(n^2)$

Serie di Fibonacci

$f_0 = 0$

$f_1 = 1$

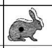
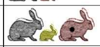
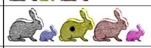
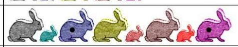
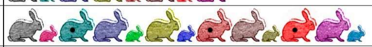
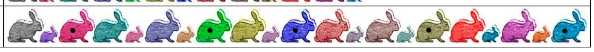
$f_n = f_{n-1} + f_{n-2}$

Ogni numero è uguale alla somma dei due precedenti

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, ...

Serie di Fibonacci

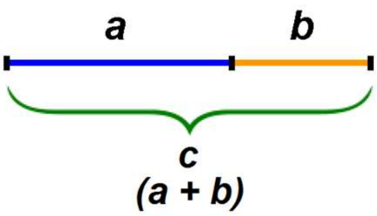
LIBER ABBACI di Leonardo Fibonacci (1200 circa)

Mesi	Coppie nate	Coppie adulte *	Totale coppie	Evoluzione nascite nei primi sei mesi
Inizio	0	1	1	
1°	1	1	2	
2°	1	2	3	
3°	2	3	5	
4°	3	5	8	
5°	5	8	13	
6°	8	13	21	

Sezione aurea

Nelle arti figurative si parla di sezione aurea quando c'è un certo rapporto tra due segmenti

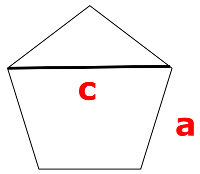
a+b: a = a: b **a è medio proporzionale fra c=a + b e b (a> b)**



a+b/a=a/b=**φ**

Sezione aurea nelle figure geometriche

φ ad esempio, è il rapporto fra la diagonale c e il lato a del pentagono



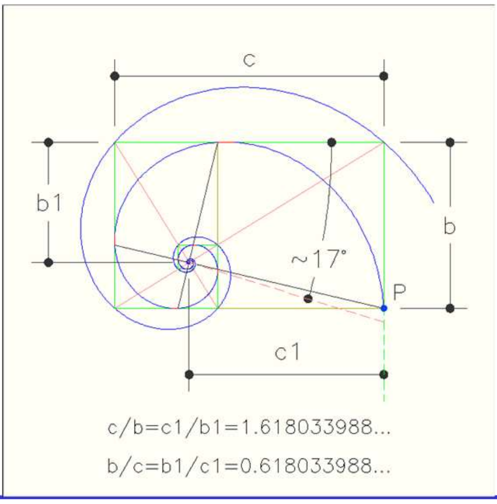
Sezione aurea

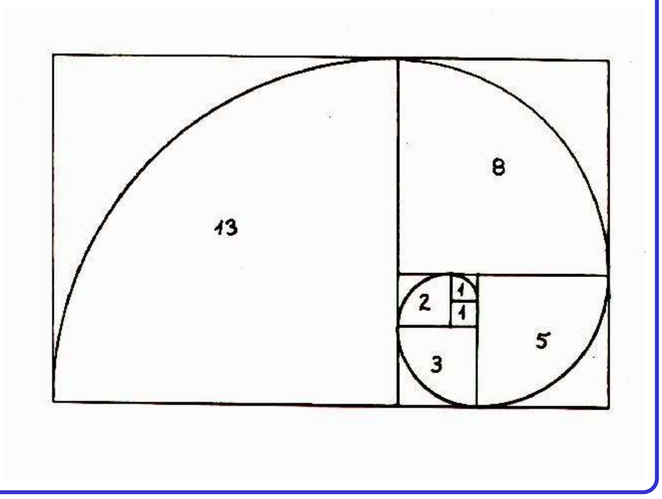
Φ è un numero irrazionale che può essere approssimato come limite del rapporto fra ogni numero della serie di Fibonacci e il precedente

$$\varphi = \lim_{n \to \infty} \frac{f_n}{f_{n-1}} = \frac{1+\sqrt{5}}{2} = 1,61803 \ldots\ldots$$

Spirale logaritmica

Sempre in geometria si ha la spirale aurea o logaritmica, ovvero un tipo particolare di spirale con fattore di accrescimento dipendente dalla sezione aurea φ





Serie di Fibonacci

$f_0 = 0$

$f_1 = 1$

$f_n = f_{n-1} + f_{n-2}$

```
int fibonacci(int n) {
  if (n == 0)  return 0;
  if (n == 1)  return 1;
  return   fibonacci(n-1) + fibonacci(n-2) ;
}
```

$T(0) = T(1) = d$

$T(n) = b + T(n-1) + T(n-2)$

$T(n) \in O(2^n)$

Serie di Fibonacci

```
int fibonacci(int n) {
   int k; int j=0; int f=1;
   for (int i=1; i<=n; i++) {
     k=j; j=f; f=k+j;
   }
   return j;
 }
```

$T(n) \in O(n)$

Serie di Fibonacci

```
int fibonacci( int n, int a = 0, int b = 1 ) {
   if (n == 0) return a;
   return fibonacci( n-1, b, a+b );
 }
```

$$T(0) = d$$
$$T(n) = b + T(n-1)$$

$T(n) \in O(n)$