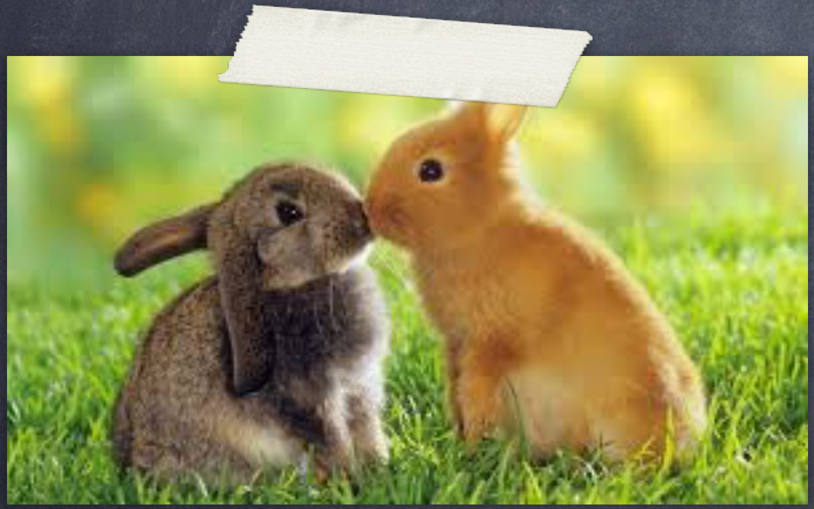
Successione di Fibonacci.. quando i conigli contano



Un percorso accattivante alla ricerca dell'ottimo

1 1 2 3 5 8 13 21 34 55…

Un po' di storia

Leonardo Pisano, detto Fibonacci, nasce a Pisa nel 1170: è un grande matematico. Oggi lo ricordiamo per la serie che porta il suo nome, ma... se oggi usiamo i numeri 0...9 lo dobbiamo a lui!

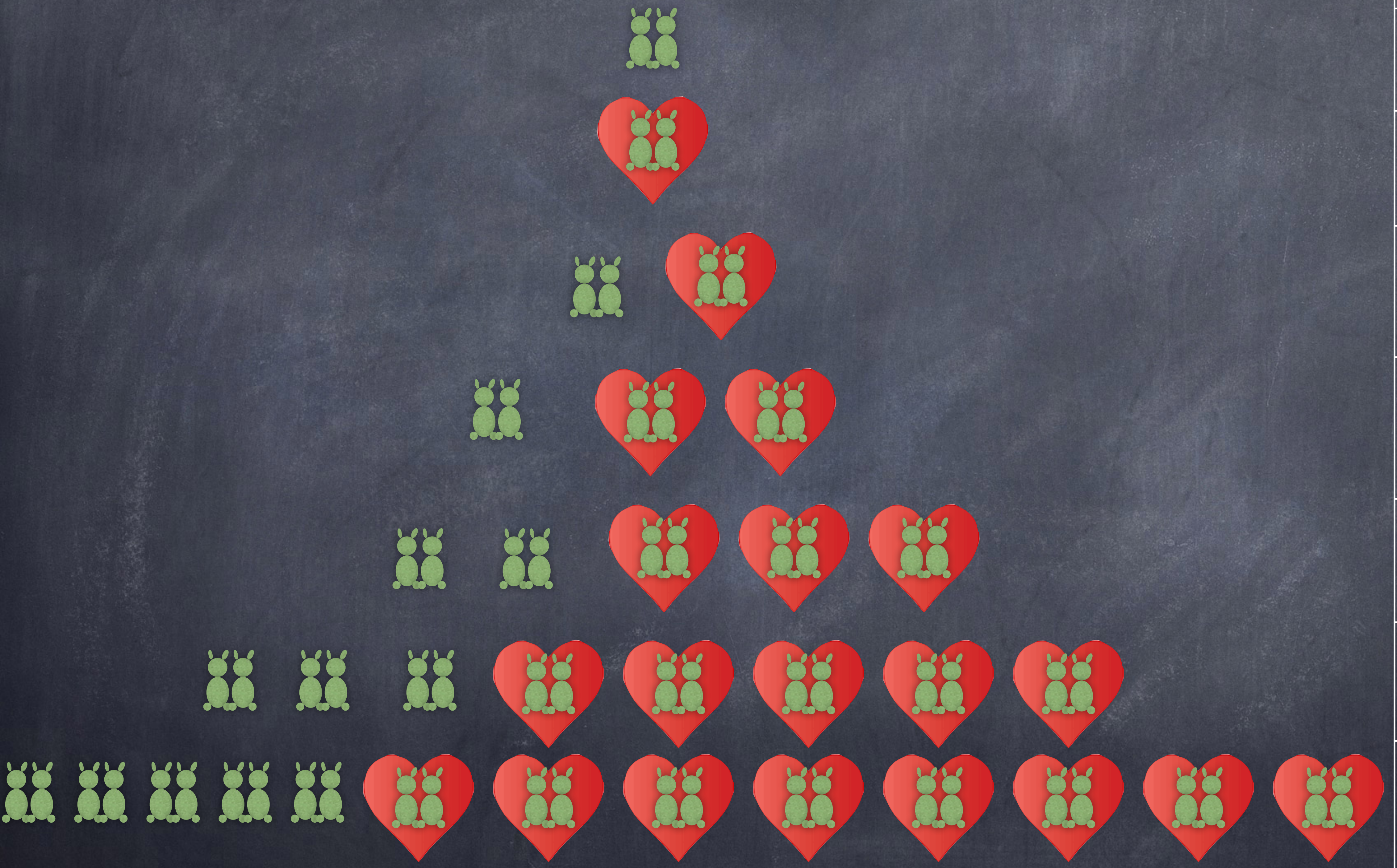
La sequenza è legata ai conigli (anche se alcuni dicono alle api).

Il quesito con cui Fibonacci si confronta è il seguente: siamo su un'isola in cui gli unici animali presenti sono una coppia di conigli i quali:

- impiegano un mese per raggiungere la maturità sessuale
- dal secondo mese, ogni mese, generano una nuova coppia di conigli, per ipotesi sempre un maschio e una femmina, che a loro volta impiegheranno un mese per raggiungere la maturità sessuale per poi generare una nuova coppia
- i conigli non muoiono mai!

La domanda è questa: quante coppie di conigli ci saranno sull'isola al generico mese n?

Cerchiamo di capire...



Mese	Coppie
1	1
2	2
3	3
4	5
5	8
6	13
7	21

Perché i conigli

La serie di numeri che "nasce" dall'intimità dei conigli ha strane e curiose affinità in diversi contesti, a volte inaspettati.

La ritroviamo in:

matematica

economia

pittura

...

...

informatica

...

botanica

...

...

musica

...

Fibonacci nella matematica

- La sezione aurea è un numero particolare che ha caratteristiche estetiche e che indica il rapporto fra due lunghezze disuguali. Quando un oggetto ha queste proporzioni, statisticamente pare che ci piaccia di più.

- Per tale motivo viene usato nell'ambito dell'arte, del design, della architettura.

- E' un numero irrazionale le cui prime cifre sono:
1,61803398874989484820458683436563811772030917980576286213544862270526046281890244970720720418939113748475...

...	
$\frac{55}{34} =$	1,617 647
$\frac{89}{55} =$	1,618 182
$\frac{144}{89} =$	1,617 978
$\frac{233}{144} =$	1,618 056
$\frac{377}{233} =$	1,618 026
$\frac{610}{377} =$	1,618 037
$\frac{987}{610} =$	1,618 033
...	

- Se consideriamo il rapporto fra l'i-esimo numero della sequenza ed il suo precedente, al crescere di i, otteniamo proprio il numero "magico", la sezione aurea.

$$\frac{F_i}{F_{i-1}} \Rightarrow \text{Sez.Aurea}$$

Fibonacci nella botanica

- Quasi tutti i fiori hanno tre o cinque o otto o tredici o ventuno o trentaquattro o cinquantacinque o ottantanove petali
- I pistilli sulle corolle dei fiori spesso si dispongono secondo uno schema preciso formato da spirali il cui numero corrisponde ad uno della serie di Fibonacci. Di solito le spirali orientate in senso orario sono trentaquattro mentre quelle orientate in senso antiorario cinquantacinque (due numeri di Fibonacci); altre volte sono rispettivamente cinquantacinque e ottantanove, o ottantanove e centoquarantaquattro. Si tratta sempre di numeri di Fibonacci consecutivi.
- I numeri di Fibonacci sono presenti anche nel numero di infiorescenze di ortaggi come il Broccolo romanesco.
- Le foglie sono disposte sui rami in modo tale da non coprirsi l'una con l'altra per permettere a ciascuna di esse di ricevere la luce del sole. Se prendiamo come punto di partenza la prima foglia di un ramo e contiamo quante foglie ci sono fino a quella perfettamente allineata spesso questo numero è un numero di Fibonacci e anche il numero di giri in senso orario o antiorario che si compiono per raggiungere tale foglia allineata dovrebbe essere un numero di Fibonacci.

Fibonacci in economia

I numeri di Fibonacci sono utilizzati anche in economia nell'Analisi tecnica per le previsioni dell'andamento dei titoli in borsa, secondo la teoria delle onde di Elliott.

Teorema di Pitagora

Nel 1948 il matematico Charles Raine ha scoperto che è possibile utilizzare i numeri di Fibonacci per generare terne pitagoriche.

- Siano A,B,C,D quattro numeri qualsiasi consecutivi nella successione di Fibonacci
- Sia T1 Il prodotto degli estremi
- Sia T2 il doppio del prodotto dei medi
- Sia T3 la somma dei quadrati dei medi
- T1, T2 e T3 formano una terna pitagorica
- Inoltre T3 è esso stesso un numero di Fibonacci!

L'enunciato formale

L'i-esimo numero di Fibonacci si ottiene sommando i due precedenti, tranne per il primo ed il secondo che sono pari a 1

$$F_1 = 1$$

$$F_2 = 1$$

$$F_i = F_{i-1} + F_{i-2} \qquad \text{per } i > 2$$

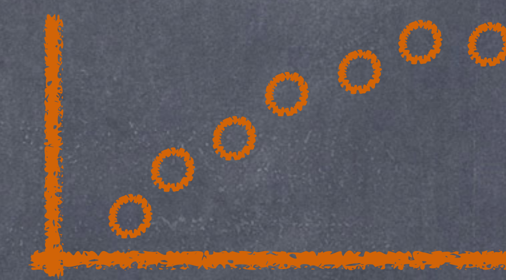
La complessità

- "funziona" non è abbastanza
- alla ricerca dell'ottimo

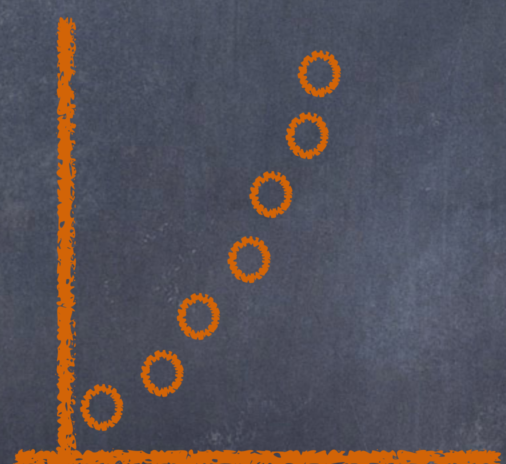

Un po' di matematica

- lineare

- logaritmica



- polinomiale



- esponenziale

Nello studio della complessità di un algoritmo è importante identificare come cresce la complessità al crescere della dimensione dell'input

Tipi di complessità qualche numero

	Dimensione problema										
	1	2	4	6	8	10	12	14	16	18	20
Logaritmo	0,000	1,000	2,000	2,585	3,000	3,322	3,585	3,807	4,000	4,170	4,322
Lineare	2	4	8	12	16	20	24	28	32	36	40
Polinomiale	1	4	16	36	64	100	144	196	256	324	400
Esponenziale	2	4	16	64	256	1.024	4.096	16.384	65.536	262.144	1.048.576

La potenza del due

Tutti cercavano di rallegrare il re, ma nessuno vi riusciva. Un giorno si presentò al palazzo un brahmano, Lahur Sessa, che, per rallegrare il re, gli propose un gioco che aveva inventato: il gioco degli scacchi. Il re si appassionò a questo gioco. Il re fu finalmente felice, e chiese a Lahur Sessa quale ricompensa egli volesse: ricchezze, un palazzo, una provincia o qualunque altra cosa. Il monaco rifiutò, ma il re insistette per giorni, finché alla fine Lahur Sessa, guardando la scacchiera, gli disse: «Tu mi darai un chicco di grano per la prima casa, due per la seconda, quattro per la terza, otto per la quarta e così via». Il re rise di questa richiesta, meravigliato del fatto che il brahmano potesse chiedere qualunque cosa e invece si accontentasse di pochi chicchi di grano. Il giorno dopo i matematici di corte andarono dal re e lo informarono che per adempiere alla richiesta del monaco non sarebbero bastati i raccolti di tutto il regno per ottocento anni. In questo modo, Lahur Sessa insegnò al re che una richiesta apparentemente modesta può nascondere un costo enorme. In effetti, facendo i calcoli, il brahmano chiese 18.446.744.073.709.551.615 (18 trilioni 446 biliardi 744 bilioni 73 miliardi 709 milioni 551mila 615) chicchi di grano. In ogni caso, il re capì, il brahmano ritirò la richiesta e divenne il governatore di una delle province del regno. Una fonte accreditata ne La variante di Lüneburg di Paolo Maurensig riporta invece l'uccisione del monaco.

per curiosità, in un kg di riso ci sono circa 5400 chicchi

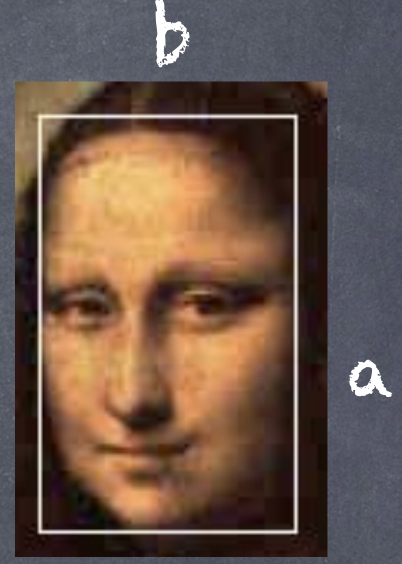
Piega, piega, piega...

Altra impressionante progressione esponenziale che si può "toccare con le mani" è un foglio di carta (spessore 0.1 mm) piegato a metà. Per molti anni si è creduto che non si potesse piegare un foglio a metà più di 8 volte (provate...), finché una studentessa liceale californiana, Britney Gallivan, è riuscita a ripiegare a metà una striscia di carta per ben 12 volte. Non sembra molto, direte voi. Ha avuto bisogno di una striscia lunga 1200 metri.

Pieghe	Altezza
10	1024 strati= 2 risme
11	4 risme
23	1 km
30	Fuori atmosfera
42	luna
51	Sole
103	100 miliardi di anni luce

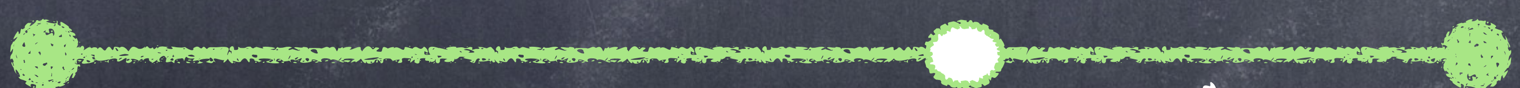
Una soluzione "aurea"

Quando il rapporto fra due grandezze è pari a questo numero, bhe... il risultato sembra più bello e/o funzionare meglio



- La sezione aurea, indicata con ϕ, è il rapporto fra due grandezze diseguali $a>b$, in cui a è medio proporzionale tra b e $(a+b)$

$$(a+b) : a = a : b$$



$$(a+b)/a = a/b = \phi$$

$\Rightarrow$ $\phi=[1+\mathrm{rad}(5)]/2$ $\quad$ $\phi'=[1-\mathrm{rad}(5)]/2$

Fibonacci1:pseudo codice

Keplero (nel 1700), scoprì che il rapporto fra l'ennesimo numero della serie di Fibonacci ed il suo predecessore, al crescere di n si avvicina sempre di più al numero aureo 'ϕ'

$$\lim_{n\to\infty} \frac{F_{n+1}}{F_n} = \phi$$

da cui si può ricavare che Fn:

$F_n = (\phi^n - \phi'^n)/\text{radice}(5)$

$\boxed{\phi = [1+\text{rad}(5)]/2}$

$\boxed{\phi' = [1-\text{rad}(5)]/2}$

algoritmo fibonacci1(intero n) → intero

return $(\phi^n - \phi'^n)$/radice(5)

Fibonacci 1: how much?

- Fibonacci1 ha costo unitario, ma... pur essendo molto efficiente, non è corretto.
- Infatti un computer reale non può gestire numeri irrazionali e quindi è costretto ad operare degli arrotondamenti che falsano il risultato

$$\phi \approx 1.618 \text{ e } \hat{\phi} \approx -0.618$$

n	`fibonacci1`(n)	arrotondamento	F_n
3	1.99992	2	2
16	986.698	987	987
18	2583.1	2583	2584

Fibonacci2

- Proviamo a cambiare strategia e, partendo dalla definizione intrinsecamente ricorsiva dell'algoritmo...

```
algoritmo fibonacci2(intero n) → intero
    if (n ≤ 2) then return 1
    else return fibonacci2(n-1) +
                         fibonacci2(n-2)
```

Fibonacci 2: how much?

- fibonacci2 è corretto per definizione, ma non è efficiente: infatti la sua esecuzione richiede un numero di linee di codice esponenziale rispetto all'indice da cercare
- In pratica la sua complessità segue la complessità dell'attività dei conigli che, come visto, è esponenziale

  esempio: per calcolare il 45 numero della successione richiede l'esecuzione di 3.404.709.508 linee di codice

  per il 100-esimo, con il computer più potente attuale servirebbero... 8000 anni

Fibonacci3

- il problema di fibonacci2 risiede nel suo ricalcolare ripetutamente la soluzione dello stesso sottoproblema.

- Proviamo a riutilizzare le sottosoluzioni ottenute senza ricalcolarle più e più volte:

```
algoritmo fibonacci3(intero n) → intero
    sia Fib un array di n interi
    Fib[1] ← Fib[2] ← 1
    for i = 3 to n do
        Fib[i] ← Fib[i-1] + Fib[i-2]
    return Fib[n]
```

Fibonacci 3: how much?

- Si può facilmente dimostrare che le linee di codice eseguite in Fibonacci3 per il calcolo dell'n-esimo numero della successione sono proporzionali ad n, in particolare al suo doppio
- Fibonacci3 risulta quindi essere ben 38 milioni di volte più veloce di Fibonacci2 per il calcolo del 45-esimo numero della successione

Fibonacci4

- Fibonacci3 sembra un ottimo risultato: nella realtà delle cose, però non dobbiamo prestare attenzione solo alla complessità temporale di un algoritmo (quanto ci impiega), ma anche a quella spaziale (quanto occupa).
- Per complessità spaziale, Fibonacci3 non è ottimale: usa un array di dimensione proporzionale all'indice n.
- Proviamo a togliere tale vincolo

```
algoritmo fibonacci4(intero n) → intero
    a ← b ← 1
    for i = 3 to n do
        c ← a+b
        a ← b
        b ← c
    return b
```

Fibonacci 4: how much?

Fibonacci4 è corretto ed anche efficiente in termini temporali (ordine di n) e spaziali

Buono, ma... si può fare di meglio?????

SI!

Fibonacci6

Spostiamoci nel "mondo" delle matrici

Si può dimostrare che:

$$\begin{vmatrix} 1 & 1 \\ 1 & 0 \end{vmatrix}^{n} = \overbrace{\begin{vmatrix} 1 & 1 \\ 1 & 0 \end{vmatrix} \times \dots \times \begin{vmatrix} 1 & 1 \\ 1 & 0 \end{vmatrix}}^{n \text{ volte}} = \begin{vmatrix} F_{n+1} & F_{n} \\ F_{n} & F_{n-1} \end{vmatrix}$$

algoritmo fibonacci6(intero n) -> intero

$M \leftarrow \begin{vmatrix} 1 & 0 \\ 0 & 1 \end{vmatrix}$

for (i=1 to n-1) do

$M \leftarrow M * \begin{vmatrix} 1 & 1 \\ 1 & 0 \end{vmatrix}$

return M[1][1]

Fibonacci 5: how much?

- Si può dimostrare che Fibonacci5 ha anche esso un costo lineare
- Stesso risultato di Fibonacci4...
- Siamo però sulla buona strada

Ancora matematica

- Possiamo calcolare la n-esima potenza elevando al quadrato la $\lfloor n/2 \rfloor$ - esima potenza
- Se n è dispari eseguiamo una ulteriore moltiplicazione
- Esempio: se devo calcolare 3^8:

$$3^8=(3^4)^2 = [(3^2)^2]^2 = [(3\cdot3)^2]^2 = [(9)^2]^2 = [(9\cdot9)]^2 = [81]^2 = 81\cdot81 = 6561$$

3 prodotti invece di 8

- Esempio: se devo calcolare 3^7:

$$3^7=3\cdot(3^3)^2 = 3\cdot(3\cdot(3)^2)^2 = 3\cdot(3\cdot(3\cdot3))^2=3\cdot(3\cdot9)^2=3\cdot(27)^2=3\cdot(27\cdot27)=3\cdot(729) = 2187$$

4 prodotti invece di 7

Fibonacci6

```
algoritmo fibonacci6 (intero n) -> intero
 A<- |1 0|
     |0 1|

 M<-potenzaDiMatrice(A,n-1)
 return M[1][1];

funzione potenzaDiMatrice(matrice A, intero n) -> matrice

  if (k<=1) then M<- |1 0|
  else               |0 1|
    potenzaDiMatrice(A,int(k/2))
    M<-M*M
  if (k è dispari) then
    M<-M*A
  return M
```

Fibonacci6: how much?

Si può dimostrare che Fibonacci6 ha un costo logaritmo rispetto ad n.

Quindi Fibonacci6 è esponenzialmente più ottimale di Fibonacci5 che già ci sembrava buono

Riepilogando

Algoritmo	Costo	Simul. per n=45 (l.o.c)
Fibonacci1	costo unitario, ma non corretto su macchine reali	1
Fibonacci2	costo esponenziale	3.404.709.508
Fibonacci3	costo lineare, ma con spreco di memoria	~90
Fibonacci4	costo lineare	~360
Fibonacci5	costo lineare	~360
Fibonacci6	costo logaritmico	~8

Quindi…

- Non dobbiamo accontentarci di “ma tanto funziona”
- Esistono algoritmi più o meno efficienti
- Esistono problemi più o meno complicati
- L’informatica, o meglio la computer Science, si occupa ANCHE di studiare i problemi per capire quanto sono complicati e, se possibile, trovare l’algoritmo migliore per affrontarli
- la risposta alle domande sulla complessità dei problemi ha impatti nella vita di tutti i giorni: navigazione satellitare, sicurezza delle password, efficienza di internet, finanza, economia, fotografia, ecc.