I numeri di Fibonacci

Studio dell’espansione di una popolazione di conigli su un’isola deserta.

Ipotesi:

1. una coppia di conigli genera una coppia di coniglietti ogni anno
2. i conigli si riproducono a partire dal secondo anno di vita
3. i conigli sono immortali

I numeri di Fibonacci

F_n = # coppie di conigli all’anno n

- $F_1 = 1$ si parte con una coppia
- $F_2 = 1$ la coppia è troppo giovane per riprodursi
- $F_3 = 2$ nasce la seconda coppia di conigli
- $F_4 = 3$ nasce un’altra coppia di conigli
- $F_5 = 5$ nascono i primi nipoti della coppia iniziale
- …

I numeri di Fibonacci

In generale, all’anno n sono presenti tutte le coppie di conigli dell’anno precedente (F_{n-1})

+

tutte le coppie di coniglietti nate nell’ultimo anno, che sono tante quante le coppie fertili dell’anno precedente, cioè tante quante le coppie presenti due anni prima (F_{n-2}).

I numeri di Fibonacci

Relazione di ricorrenza: *Relazione di Fibonacci*

$$F_n = 1 \qquad n = 1,2$$
$$F_n = F_{n-1} + F_{n-2} \qquad n > 2$$

Problema algoritmico: come calcolare F_n?

Formula di Binet

$$F_n = \frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^n - \frac{1}{\sqrt{5}}\left(\frac{1-\sqrt{5}}{2}\right)^n$$

Asintoticamente: $F_n \sim 0.45\ (1.618)^n$

Un algoritmo numerico

```
fib1(n)
    if (n ≤ 2) return 1;
    else return
```

$$\frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^n - \frac{1}{\sqrt{5}}\left(\frac{1-\sqrt{5}}{2}\right)^n$$

Svantaggi:

- l'algoritmo lavora con numeri reali, rappresentati nei calcolatori con precisione limitata
- corriamo il rischio di fornire risposte non corrette a causa degli errori di arrotondamento.

Un algoritmo ricorsivo

```
fib2(n)
    if (n ≤ 2) return 1;
    else return fib2(n-1) + fib2(n-2)
```

Svantaggi:

- Codice elegante ma molto dispendioso: calcola ripetutamente la soluzione dello stesso sottoproblema.
- Costo in tempo esponenziale:

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

➔ relazione "tipo Fibonacci"

➔ $T(n) \geq F_n \sim 0.45\ (1.618)^n$

Osservazioni

- Questo esempio evidenzia uno dei peggiori svantaggi della ricorsione.
- Spesso eleganza e semplicità di una formulazione ricorsiva possono andare a scapito dell’efficienza:
    - è possibile ripetere inutilmente gli stessi calcoli, provocando una disastrosa complessità superpolinomiale.

Un algoritmo iterativo

Idea: risolviamo ogni sottoproblema una sola volta, memorizziamo la soluzione e la utilizziamo nel seguito invece di ricalcolarla.

Sia `F` un array di n interi:

```
fib3(n)
   Fib[0] = 0;
   Fib[1] = 1;
   for (i = 2; i ≤ n; i++) {
      Fib[i] = Fib[i-1] + Fib[i-2];
   }
   return Fib[n];
```

Costo: $\Theta(n)$
Svantaggio: richiede spazio di memoria $\Theta(n)$

Un algoritmo più efficiente in spazio

```
fib4(n)
   if (n ≤ 2) return 1;
   a = 1;
   b = 1;
   for (i = 3; i ≤ n; i++) {
       c = a + b;
       a = b;
       b = c;
   }
   return b
```

Costo: tempo $\Theta(n)$ e spazio $\Theta(1)$

Un algoritmo più efficiente in tempo

- È possibile calcolare $\mathtt{F}_{\mathtt{n}}$ in tempo $\Theta(\log n)$ e spazio $\Theta(1)$ (si vedano i riferimenti bibliografici).

Riferimenti bibliografici

- Camil Demetrescu, Irene Finocchi, Giuseppe F. Italiano.
  Algoritmi e strutture dati.
  McGraw-Hill 2004.

- Fabrizio Luccio.
  La struttura degli Algoritmi.
  Boringhieri 1982.