Induzione, ricorsione ed esponenziazione veloce

Giorgio Audrito

Online, 8 aprile 2021

Due semplici problemi di riferimento

Esponenziazione

Dati b ed n, calcola b^n.

Fibonacci

Calcola l'n-esimo numero di Fibonacci F_n, dove:

- $F_0 = F_1 = 1$
- $F_{n+1} = F_n + F_{n-1}$

Ricorsione e induzione

Fibonacci

Calcola l'n-esimo numero di Fibonacci F_n, dove:

- $F_0 = F_1 = 1$
- $F_{n+1} = F_n + F_{n-1}$

Definizione per ricorsione:
definisco in un colpo solo un insieme di cose (in questo caso gli F_n),
descrivendo come si ottengono da altre cose dell'insieme (F_{n+1} da F_n e F_{n-1}),
e dando alcuni elementi base (F_0, F_1).

Come posso sapere se la definizione ha senso? Tramite *induzione*.

Induzione

Insieme parzialmente ordinato

Sia $X = \{x, y, z, \ldots\}$ un insieme (finito o infinito) di cose qualunque.
X è parzialmente ordinato da una relazione $<$ tra coppie di elementi di X, se:

- $x < x$ non è mai vero
- $x < y$ e $y < z$ implica che $x < z$
- non ci sono sequenze discendenti infinite: $x > y > z > \ldots$

Parzialmente: può essere che x non sia minore di y e nemmeno y minore di x

Esempi

- Numeri con ordine classico: $2 < 5$ (anche totale)
- Stringhe o tuple con ordine lessicografico: (anche totale)
  `"ciao"` $<$ `"mondo"`, $(2, 4) < (4, 2)$
- Tuple con ordine per componente: $(1, 1) < (4, 2)$
- Insiemi con $\subset$: $\{1, 3, 7\} \subset \{3, 7\}$

Induzione

Dimostrazione per induzione

Sia X insieme parzialmente ordinato, $P(x)$ proprietà dei suoi elementi. Se:

- $P(x)$ vale in un insieme finito $Y \subset X$ (caso base), e
- per ogni $x \in X \setminus Y$, posso dimostrare $P(x)$ assumendo $P(y)$ per ogni $y < x$ (*ipotesi induttiva*)

allora $P(x)$ vale per ogni $x \in X$.

Ricorsione

ricorsione = definizione per induzione:

Fibonacci

Calcola l'n-esimo numero di Fibonacci F_n, dove:

- $F_0 = F_1 = 1$
- $F_{n+1} = F_n + F_{n-1}$

$X = \mathbb{N}$, $Y = \{0, 1\}$, ordine $<$ numerico

Esponenziazione

Calcola b^n, dove:

- $b^0 = 1$
- $b^{n+1} = b \cdot b^n$

$X = \mathbb{N}$, $Y = \{0\}$, ordine $<$ numerico

Soluzione ricorsiva

Fibonacci

Calcola l'n-esimo numero di Fibonacci F_n, dove:

- $F_0 = F_1 = 1$
- $F_{n+1} = F_n + F_{n-1}$

Esponenziazione

Calcola b^n, dove:

- $b^0 = 1$
- $b^{n+1} = b \cdot b^n$

```
int fib(int n) {
    if (n <= 1) return 1;
    return fib(n-1) + fib(n-2);
}
```

$$T(n) = \mathcal{O}(1) + T(n-1) + T(n-2)$$

$$\Rightarrow T(n) \in \mathcal{O}(F_n)$$

```
int exp(int b, int n) {
    if (n == 0) return 1;
    return b * exp(b, n-1);
}
```

$$T(n) = \mathcal{O}(1) + T(n-1)$$

$$\Rightarrow T(n) \in \mathcal{O}(n)$$

Ma si può fare meglio!

Soluzione iterativa

Fibonacci

Calcola l'n-esimo numero di Fibonacci F_n, dove:

- $F_0 = F_1 = 1$
- $F_{n+1} = F_n + F_{n-1}$

Esponenziazione

Calcola b^n, dove:

- $b^0 = 1$
- $b^{n+1} = b \cdot b^n$

```
int fib(int n) {
    int F[n];
    F[0] = F[1] = 1;
    for (int i=2; i<=n; ++i)
        F[i] = F[i-1] + F[i-2];
    return F[n];
}
```

$$T(n) \in \mathcal{O}(n)$$

$$S(n) \in \mathcal{O}(n)$$

```
int exp(int b, int n) {
    int E[n];
    E[0] = 1;
    for (int i=1; i<=n; ++i)
        E[i] = b * E[i-1];
    return E[n];
}
```

$$T(n) \in \mathcal{O}(n)$$

$$S(n) \in \mathcal{O}(n)$$

Ma si può fare meglio!

Soluzione compressa

Fibonacci

Calcola l'n-esimo numero di Fibonacci F_n, dove:

- $F_0 = F_1 = 1$
- $F_{n+1} = F_n + F_{n-1}$

Esponenziazione

Calcola b^n, dove:

- $b^0 = 1$
- $b^{n+1} = b \cdot b^n$

```
int fib(int n) {
    int F[3];
    F[0] = F[1] = 1;
    for (int i=2; i<=n; ++i)
        F[i%3] = F[(i-1)%3] + F[(i-2)%3];
    return F[n%3];
}
```

$$T(n) \in \mathcal{O}(n)$$

$$S(n) \in \mathcal{O}(1)$$

```
int exp(int b, int n) {
    int E[2];
    E[0] = 1;
    for (int i=1; i<=n; ++i)
        E[i%2]=b*E[(i-1)%2];
    return E[n%2];
}
```

$$T(n) \in \mathcal{O}(n)$$

$$S(n) \in \mathcal{O}(1)$$

Ma si può fare meglio!

Esponenziazione veloce

Dati b ed n, calcola b^n.

Una ricorsione più efficiente

- $b^0 = 1$
- $b^n = \left(b^{\lfloor \frac{n}{2} \rfloor}\right)^2 \cdot b^{n \bmod 2}$

$T(n) = \mathcal{O}(1) + T(n/2)$

$\Rightarrow S(n), T(n) \in \mathcal{O}(\log n)$

```
int exp(int b, int n) {
    if (n == 0) return 1;
    int r = exp(b, n/2);
    r = r * r;
    if (n%2 == 1) r = r * b;
    return r;
}
```

Possiamo fare qualcosa di simile con Fibonacci?

Fibonacci veloce

Calcola F_n, dove:

- $F_0 = F_1 = 1$
- $F_{n+1} = F_n + F_{n-1}$

$\mapsto$

Calcola $G_n = (F_n, F_{n-1})$, dove:

- $G_1 = (F_1, F_0) = (1, 1)$
- $G_{n+1} = G_n \times \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}$

$$\Rightarrow (F_{n+1}, F_n) = (F_n, F_{n-1}) \times \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}$$

$$\Rightarrow \begin{cases} F_{n+1} & = F_n \cdot 1 + F_{n-1} \cdot 1 \\ F_n & = F_n \cdot 1 + F_{n-1} \cdot 0 \end{cases}$$

funziona!

Fibonacci veloce

Dato n, calcola $(F_n, F_{n-1}) = (1,1) \times \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^{n-1}$.

Esponenziazione di matrici

- $\begin{pmatrix} a & b \\ c & d \end{pmatrix}^0 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$
- $\begin{pmatrix} a & b \\ c & d \end{pmatrix}^n = \left[\begin{pmatrix} a & b \\ c & d \end{pmatrix}^{\lfloor \frac{n}{2} \rfloor}\right]^2 \cdot \begin{pmatrix} a & b \\ c & d \end{pmatrix}^{n \bmod 2}$

$$S(n), T(n) \in \mathcal{O}(\log n)$$

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \times \begin{pmatrix} e & f \\ g & h \end{pmatrix} = \begin{pmatrix} ae + bg & af + bh \\ ce + dg & cf + dh \end{pmatrix}$$