Esercizi di Informatica Generale

corso di Informatica Generale
Ivano Salvo – Sapienza Università di Roma
Anno Accademico 2023-24

In questa dispensa sono raccolti esercizi usati negli anni accademici precedenti come argomento di esercitazioni oppure come esercizi d'esame.

Sono segnalati con uno o più simboli ★ gli esercizi che possono risultare impegnativi o che necessitano di un po' di inventiva per essere risolti. Sono segnalati con il simbolo ♣ esercizi che sono stati problemi "*famosi*" e che spesso vengono presentati in libri di algoritmi o programmazione.

1 Problemi Elementari su Numeri Naturali

1.1 Quadrato come somma di dispari

Assumendo di avere un esecutore capace di eseguire solo somme e confronti tra valori interi, scrivere un algoritmo iterativo che calcola il quadrato di un numero naturale n come la somma dei primi n numeri dispari.

Scrivere gli invarianti di ciclo e discutere la terminazione del vostro algoritmo. Calcolare la complessità.

Dimostrare per induzione che effettivamente $n^2 = \sum_{i=0}^{n-1} 2i + 1$.

1.2 Calcolo della radice quadrata♣

Avendo a disposizione un esecutore capace solo di fare somme e moltiplicazioni, dare un algoritmo iterativo per calcolare la radice quadrata intera di un numero naturale n, cioè un numero naturale r tale che $r^2 \leq n < (r+1)^2$.

Scrivere gli invarianti di ciclo e discutere la terminazione e la complessità.

Osservazione: L'algoritmo "naturale" costa $\Theta(\sqrt{n})$, ma è possibile trovare un algoritmo di complessità $\Theta(\log n)$.

1.3 Calcolo del logaritmo intero

Avendo a disposizione un esecutore capace solo di fare somme e moltiplicazioni, dare un algoritmo che presi in input due interi $n > 0$ e $b > 1$, calcola $\lfloor \log_b n \rfloor$, cioè un numero naturale k tale che $b^k \leq n < b^{k+1}$.

Scrivere gli invarianti di ciclo, discutere la terminazione e calcolare la complessità.

1.4 Divisione intera

Avendo a disposizione un esecutore capace solo di fare somme e sottrazioni, dare un algoritmo che presi in input due interi $m \geq 0$ e $n > 0$, calcola la divisione intera tra m ed n, cioè trova due numeri q e r tali che:

$$m = qn + r, \quad 0 \leq r < n$$

Progettare una funzione dal prototipo:

```
q, r = div(m, n)
```

Scrivere gli invarianti di ciclo, discutere la terminazione e calcolare la complessità.

1.5 MCD della maestra

Conoscerete sicuramente questa filastrocca per definire il massimo comun divisore:

> *il massimo comun divisore di due numeri naturali $m, n > 0$, è il prodotto di tutti i fattori primi comuni di m ed n presi con il loro minimo esponente.*

Scrivere una funzione `M = mcd(m, n)` che calcola il massimo comun divisore di m ed n calcolando la scomposizione in fattori primi e facendo la produttoria di quelli comuni.

OSSERVAZIONE: Non è necessario memorizzare i fattori primi trovati.

1.6 Numeri perfetti, abbondanti e difettivi (homework 1, 2012)

Un numero naturale n si dice *perfetto* se è uguale alla somma di tutti i suoi divisori propri. Ad esempio, $28 = 1 + 2 + 4 + 7 + 14$ è perfetto.

Un naturale n si dice *abbondante* se la somma di tutti i suoi divisori propri è maggiore di n. Ad esempio $24 < 1 + 2 + 3 + 4 + 6 + 8 + 12 = 36$ è abbondante.

Viceversa, n si dice *difettivo* se la somma di tutti i suoi divisori propri è minore di n. Ad esempio $27 > 1 + 3 + 9 = 13$ è difettivo.

Scrivere una funzione `p = isPerfect(n)` che torna -1 se n è difettivo, 0 se n è perfetto e 1 se n è abbondante.

1.7 Moltiplicazione Egiziana

Avendo a disposizione un esecutore capace solo di fare somme, dare un algoritmo che presi in input due interi $m, n > 0$, calcola il prodotto $m \cdot n$.

L'algoritmo "naturale" itera n volte la somma di m e quindi necessita di $\Theta(n)$ somme. Dare un algoritmo che risolve il problema con sole $\Theta(\log n)$ somme, ispirato dalle seguenti equazioni:

$$\begin{aligned} m \cdot 2n &= (m + m) \cdot n \\ m \cdot (2n + 1) &= m + m \cdot 2n \\ m \cdot 0 &= 0 \end{aligned}$$

Giustificare la correttezza e la complessità dell'algoritmo usando asserzioni logiche.

1.8 Esponenziale Egiziano (I esonero, 25/3/22)

Come nel caso precedente, è possibile calcolare l'esponenziale usando le seguenti equazioni:

$$\begin{aligned} m^{2n} &= (m \cdot m)^n \\ m^{2n+1} &= m \cdot m^{2n} \\ m^0 &= 1 \end{aligned}$$

Anche in questo caso, il numero di moltiplicazioni richieste è $\Theta(\log n)$, ma qual è una stima più adeguata della complessità di questo algoritmo su una comune architettura di calcolatore contemporaneo? Perchè?