Ripasso di Go e complessità asintotica

30 settembre 2025

Esercizio 1. Scrivete un programma che, ricevuto in input un numero intero $n > 0$, genera in maniera pseudo-casuale n caratteri scegliendoli uniformemente tra `a,b,...,y,z,0,1,...,9` e li stampa su un file uno per linea (quindi separati tra loro dal carattere a capo \n).

Esercizio 2. Considerate un semplice programma che legge una serie di caratteri da un file `input`, separati dal carattere a capo (\n), li concatena in un'unica stringa (separati da spazi), e stampa la stringa ottenuta su un nuovo file `output`. Scrivete due versioni del programma: la prima legge un carattere alla volta con un ciclo `for`, e li concatena uno ad uno a una stringa inizialmente vuota `s` usando l'operatore +. La seconda legge tutti i caratteri prima di concatenarli in un'unica stringa usando la funzione `strings.Join`. Paragonate il tempo di esecuzione dei due programmi per un numero crescente di caratteri in input (1000, 10000, 100000 e 1000000) usando il pacchetto `time`. Sapete giustificare la differenza nei tempi di esecuzione in termini di complessità asintotica? Qual è la complessità di ciascuna delle due versioni dell'algoritmo?

Esercizio 3. Sia A una slice di n interi, ordinati in ordine crescente. Qual è il costo asintotico di cancellare il primo elemento di A, ottenendo quindi una slice di lunghezza $n - 1$, se si vuole che la slice risultante sia ancora ordinata? E se non si richiede che il risultato sia ordinato?

Esercizio 4. Scrivete un programma che legge un intero $n > 0$ da stdin, genera in modo casuale una slice di grandi interi con almeno 50 cifre l'uno e lo stampa. Notate che numeri così grandi non possono essere salvati come int né come int64: usate il package Go `math/big` (`https://pkg.go.dev/math/big#pkg-overview`).