Kalkulacja "Silnej" oraz "Słabej" liczby na podstawie nazwiska w języku Go

Łukasz Minasiewicz, 286143

16 kwietnia 2024

1 Warunki Zadania

Zadanie polega na wyznaczeniu dwóch liczb zależnych od imienia i nazwiska. W moim przypadku wpływ na wynik miały litery "lukmin".
Silna Liczba: To jest najmniejsza liczba, której silnia zawiera w sobie kody ASCII wszystkich liter wyżej wymienionego tekstu zależnego od imienia i nazwiska.
Słaba Liczba: Rekurencyjnie obliczamy 30. element ciągu Fibonacciego. Niech n_i to będzie liczba wywołań funkcji Fib(i) wymagana do wyliczenia Fib(30) rekurencyjnie. Słaba liczba to takie i, dla którego n_i jest najbardziej zbliżone do Silnej liczby.

2 Proces Myślowy

2.1 Silna Liczba

Proces myślowy wyznaczania Silnej liczby był dosyć prosty. Z pomocą modułu "math/big", który umożliwia pracę na dowolnie wielkich liczbach, byłem w stanie obliczyć teoretycznie dowolnie wielką silnię. Napisałem więc funkcję, która oblicza $n!$ i zwraca wynik zamieniony na string.
Aby wyznaczyć Silną liczbę, wystarczy obliczać silnię coraz większego n i za każdym razem sprawdzać, czy kod ASCII każdej litery napisu jest podnapisem danego $n!$.

2.2 Słaba Liczba

Aby wyznaczyć Słabą liczbę, w przeciwieństwie do polecenia, które nakazuje obliczenie ciągu fibonacciego za pomocą funkcji rekurencyjnej aby obliczyć liczbę wywołań funkcji z każdym argumentem z osobna, postąpiłem nieco inaczej. Funkcję napisałem iteracyjnie, co gwarantuje lepszą złożoność obliczeniową, a liczbę wywołań funkcji rekurencyjnej wyznaczyłem wzorem matematycznym, którego opisuje poniższa sekcja.

2.2.1 Wzór na liczbę wywołań rekurencyjnej funkcji fibonacciego dla każdego argumentu

Zakładając $n, a \in \mathbf{N}$, funkcję $W(n, a)$, oznaczającą liczbę wywołań funkcji Fib z argumentem a w celu wyliczenia $Fib(n)$, opisuje się w ten sposób:

$$W(n,a) = \left\{ \begin{array}{ll} 0, & \text{dla } a > n \\ 1, & \text{dla } a = n \\ Fib(n+1-a), & \text{dla } a \in <1; n-1> \\ Fib(n-1), & \text{dla } a = 0 \end{array} \right\}$$

To znaczy, że na przykład licząc $Fib(7)$ ($n = 7$), wykonamy $Fib(7)$ tylko raz, Wykonamy $Fib(6)$ do $Fib(1)$ tyle razy, ile wynoszą kolejne wyrazy ciągu Fibonacciego (Dla $a = 6$ i $a = 1$ liczba wykonań będzie wynosiła odpowiednio $Fib(2)$ i $Fib(7)$), a $Fib(0)$ wykonamy tyle samo razy, co wykonaliśmy $Fib(2)$.

Wzór ten można udowodnić w ten sposób, że funkcja Fib(a) wykonuje się tyle razy, ile wykonuje się funkcja $Fib(a+1)$ oraz $Fib(a+2)$ razem wzięte. Jest tak dlatego, że dla każdego $n \in \mathbf{N}$, $n > 1$ funkcja $Fib(n)$ wykonuje po jednym razie funkcje $Fib(n-1)$ oraz $Fib(n-2)$.

Wyjątek dla $a = n$ bierze się z tego, że to użytkownik a nie funkcja z wyższym argumentem wywołuje tą funkcję, a z założeń wynika już, że użytkownik wywołuje tą funkcję raz.

Wyjątek dla $a = 0$ bierze się z tego, że funkcja $Fib(1)$ ma z definicji stałą wartość 1, nie wywołuje więc ona siebie z mniejszymi argumentami. $Fib(0)$ więc musi być wykonana tyle samo razy ile została wykonana funkcja $Fib(2)$.

3 Wyniki

liczba	Silna	Słaba
lukmin	509	22

Tabela 1: Silna i Słaba liczba obliczona na podstawie ciągu znaków "lukmin".

4 Analiza Czasowa

Przeprowadzona została analiza czasowa funkcji Fibonacciego rekurencyjnej oraz iteracyjnej w celu porównania ich efektywności w kontekście argumentu silnej liczby (509) oraz w celu estymacji jak długo zajmie jej wyliczenie w przypadku implementacji rekurencyjnej (w przypadku iteracyjnej nie ma potrzeby estymacji ze względu na jej złożoność czasową).

Idea za estymacją ile czasu zajmie wyliczenie funkcji rekurencyjnej Fibonacciego dla dowolnego argumentu była następująca: Jeśli dla $Fib(n)$ parametr n był mniejszy lub równy 40, to po prostu wykonywana była funkcja i mierzony czas wykonania. Dla większych argumentów ze względu na ich nakład czasowy użyta była inna procedura:

Obliczyć czas wykonania funkcji dla pierwszych 40 argumentów i zobaczyć jak szybko czas wykonania wzrasta wzraz ze wzrostem parametru n. Okazało się, że współczynnik wzrostu czasu wykonania pośród tych pierwszych 40 argumentów oscylował wokół tzw. Złotej Proporcji, czyli $\varphi = \frac{1+\sqrt{5}}{2} \approx 1.618033988$.

W końcu, oszacować nakład czasowy wykonania funkcji $Fib(n), n > 40$ przez pomnożenie czasu wykonania $Fib(40)$ i pomnożenie przez φ^{n-40}.

Otrzymane wyniki przedstawia Tabela 2 poniżej.

metoda	**przybliżenie czasu wykonania**
iteracyjna	297 nanosekund
rekurencyjna	$1.49 * 10^{91}$ lat

Tabela 2: Przybliżenie czasu wymaganego na policzenie $Fib(509)$ iteracyjnie oraz rekurencyjnie.

5 Podsumowanie

Niniejsze sprawozdanie opisało warunki zadania i proces myślowy za obliczeniem dwóch liczb oraz przeprowadziło analizę czasową ich wykonania iteracyjnie oraz rekurencyjnie.