Golang - zadanie projektowe 1

Oskar Gawryszewski (285789)

kwiecień 2024

1 Opis problemu

Zadanie polegało na obliczeniu dwóch liczb zależnych od mojego imienia i nazwiska. Pierwszym krokiem było wygenerowanie nicku złożonego z 3 liter imienia i 3 liter nazwiska. Następnie ten nick został zamieniony na odpowiednie kody ASCII, a następnie na liczbę całkowitą. Kolejnym krokiem było obliczenie takiej wartości n, dla której silnia n zawierała w sobie wszystkie kody ASCII z naszego nicku. Ta wartość n została nazwana "Silną Liczbą". Następnie należało obliczyć 30-ty element ciągu Fibonacciego i zliczyć liczbę wywołań dla każdego argumentu podczas obliczeń. W ten sposób znaleziono "Słabą Liczbę", czyli argument funkcji Fibonacciego, dla którego liczba wywołań była najbliższa wartości Silnej Liczby.

2 Szukanie silnej liczby

Poszukiwania silnej liczby rozpoczynam od przekształcenia mojego imienia i nazwiska na kod ASCII. Funkcja: `gen_ascii_for_name(name string, surname string)` jako argumenty przyjmuje imię i nazwisko. Pierwszym krokiem jest połączenie trzech pierwszych liter imienia i nazwiska. Następnie funkcja przechodzi przez każdy znak nicku i zamienia go na kod ASCII, dodając kolejne elementy do listy. Dla danych wejściowych **Oskar Gawryszewski** otrzmuję wynik:

```
oskgaw
111 [111]
115 [111 115]
107 [111 115 107]
103 [111 115 107 103]
97 [111 115 107 103 97]
119 [111 115 107 103 97 119]
[111 115 107 103 97 119]
```

Następnie iteracyjnie przechodzę przez kolejne wartości silni n! (używając funkcji MulRange z biblioteki math/big), sprawdzając przy każdej iteracji, czy nie zawiera ona ciągu liczb odpowiadającemu kodom ASCII mojego nicku. Finalnie, dla zadanego nicku wynikiem jest 261!, więc moja silna liczba to 261.

3 Szukanie słabej liczby

W celu znalezienia słabej liczby, rozpoczynam od wywołania rekurencyjnej funkncji szukającej wartości dla 30 znaku ciągu fibonacciego. Tworzę przy okazji `var cache = make(map[int]int)`, która jest mapą używaną do przechowywania informacji o częstotliwości wywołań funkcji Fibonacciego dla różnych argumentów.

Następnie w funkcji `fibFrequencyClosestToStrongNumber()` zmienna cache jest używana do znalezienia argumentu funkcji Fibonacciego, który jest najbliższy wartości silnej liczby, obliczonej wcześniej. Przeszukiwana jest mapa cache, aby znaleźć argument, którego wartość jest najbliższa wartości silnej liczby, co pozwala na określenie słabej liczby.

Dla moich danych wynikiem jest **18**.

4 Wnioski

Dla danych wejściowych **Oskar Gawryszewski** wynikiem silnej liczby jest wartość 261, a wynikiem słabej liczby jest wartość 18.

5 Kosmicznie wielkie liczby

Swoje rozważania rozpocząłem od zrobienia pomiaru ile czasu zajęłoby wykonanie funkcji `fibFrequency()` dla wartości n = 40 i n = 39. Załóżmy, że czas wykonania funkcji `fibFrequency(40)` wynosi około 3.685080708 sekundy. W oparciu o to, możemy oszacować przybliżony czas wykonania funkcji `fib(261)`. Wiemy, że czas Złożoność obliczeniowa funkcji Fibonacciego wynosi $O(2^n)$, więc każdy kolejny wyraz, będzie się wykonywać średnio dwa razy dłużej. Możemy więc stwierdzić, że `fibFrequency(261)`, będzie się wykonywać o 2^{231} razy dłużej niż `fibFrequency(40)`. Daje nam to $3.685080708 + 2^{221}$, co wynosi około $3.3699933e + 66$. zobrazowania tej liczby, możemy zamienić ją na ilość lat, jakie by upłynęły:

$$\text{rok} \approx 31536000 \text{ sekund}$$

$$\frac{3.3699933 \times 10^{66}}{31536000} \approx 1.0680558 \times 10^{59} \text{ lat}$$

.

Według współczesnych ustaleń wiek Wszechświata wynosi około 13,82 mld lat, więc można zobaczyć, że jest to naprawdę ogromna liczba.

Funkcja Ackermanna to jedna z najbardziej znanych funkcji rekurencyjnych. Jest to bardzo prosta, ale bardzo szybko rosnąca funkcja. Oto jej definicja:

$$A(m,n) = \begin{cases} n+1 & \text{jeśli } m = 0 \\ A(m-1,1) & \text{jeśli } m > 0 \text{ i } n = 0 \\ A(m-1, A(m,n-1)) & \text{jeśli } m > 0 \text{ i } n > 0 \end{cases}$$

Szybkość wzrostu funkcji fibonacciego w porównaniu z szybkością wzrostu funkcji Ackermanna jest znikła. Więc skoro obliczenie funkcji fibonacciego dla mojej silnej liczby zajmuje $\approx 1.0680558 \times 10^{59}$ lat, to obliczenie funkcji Ackermanna, podstawiając pod argumenty moją silną i słabą liczbę zajęłoby niewyobrażalnie dłużej. Wydaję mi się, że nie jesteśmy w stanie zapisać tak ogromnej liczby, jaką byłby ten czas.

Dla małych wartości m, takich jak 1, 2 lub 3, funkcja Ackermanna rośnie względnie wolno w stosunku do n, co najwyżej wykładniczo. Jednak dla $m \geq 4$, rośnie znacznie szybciej; nawet $A(4,2)$ wynosi około $2.00353 \times 10^{19728}$, a rozwinięcie dziesiętne $A(4,3)$ jest bardzo duże, około $2.12004 \times 10^{(6.03123 \times 10^{19727})}$.

Ciekawym aspektem jest to, że jedyną operacją arytmetyczną, jaką kiedykolwiek wykonuje, jest dodawanie 1. Jego szybki wzrost opiera się wyłącznie na zagnieżdżonej funkcji rekurencyjnej. Oznacza to również, że czas jego działania jest przynajmniej proporcjonalny do jego wyniku, co oznacza, że jest również ogromny. W rzeczywistości, w większości przypadków czas działania jest znacznie większy niż wynik.

Dla parametrów $m = 3$; $n = 5$ i $m = 3$, $n = 6$ wyniki wynoszą następująco (2.25275ms) i (9.63875ms). Dla parametru $m = 4$ i $n = 1$ czas wykonania wyniósł już (11.702888042s). Dla parametrów $m = 4$ i $n = 2$ czas obliczenia wynosi już zbyt długo. Zauważmy, że dla $m = 3$ i $n = 1$ wynik wynosi (417ns). Możemy założyć, że z każdym (m+1) czas wykonywania wzrasta o (11.702888042s / 417ns). Daje nam to informacje, że wraz ze wrostem m wynik będzie wykonywać się o 28072328.88 razy dłużej. Dla wzrostu n (9.63875ms / 2.25275ms) możemy obliczyć, że będzie to wzrstać o 4.279268895.

Na podstawie tych informacji możemy oszacować czas wykonania obliczenia Ackermanna dla $m = 261$ i $n = 18$. Wykorzystując proporcje, możemy przyjąć, że z każdym wzrostem m czas wykonania wzrasta o 28072328.88 razy, a z każdym wzrostem n czas wzrasta o 4.279268895 razy.

Dla $m = 4$ i $n = 1$, czas wykonania wynosił $11.702888042s$. Wykorzystując proporcje, możemy obliczyć przybliżony czas dla $m = 261$ i $n = 18$:

$$11.702888042s \times 28072328.88^{(261-4)} \times 4.279268895^{(18-1)}$$

Teraz przeliczmy to:

$$11.702888042 \times 28072328.88^{257} \times 4.279268895^{17}$$

Wynik tej operacji będzie ogromny.