Go zadanie 1 - opracowanie

Kajetan Wiśniewski

Kwiecień 2024

1 Wstęp

Treść zadania:

Wygeneruj swój nick z 3 liter imienia i 3 liter nazwiska, następnie zamień go na ASCII i znajdź taką liczbę, której silnia zawiera w sobie wszystkie wartości numeryczne kodów ASCII z Twojego nicku. Będzie to Silna Liczba. Następnie wyznacz wartość 30 elementu Ciągu Fibonacciego, i oblicz liczbę wywołań każdego argumentu podczas tych obliczeń. Znajdź liczbę wywołań najbardziej zbliżoną do Silnej Liczby. Słaba liczba to argument dla którego wykonuje się ta liczba wywołań. Jeżeli nie rozumiesz o co chodzi, przeczytaj wersję długą.

Zadanie podzieliłem na dwa pliki. Jeden plik functions.go zawiera funkcje niezbędne do wykonania zadania, oraz plik main.go który zawiera funkcję main wywołującą potrzebne do obliczenia silnej i słabej liczby funkcje. Zawinięte jest to w package main. w folderze znajduje się również plik go.mod.

2 functions.go

funkcja factorial wylicza silnie. Ponieważ mamy do czynienia z dużymi liczbami, to funkcja operuje na typie big int i za pomocą operatorów z tego package'u.

```
func factorial(x *big.Int) *big.Int {
   n := big.NewInt(1)
   if x.Cmp(big.NewInt(0)) == 0 {
      return n
   }
   return n.Mul(x, factorial(n.Sub(x, n)))
}
```

funkcja **generateNickname** bierze string jako input oraz zwraca dwie wartości, string i error. Dopasowana jest tak aby akceptować tylko input określony w zadaniu i zwracać 6-znakowy nickname.

```
func generateNickname(input string) (string, error) {

    parts := strings.Split(input, " ")
    if len(parts) != 2 {
        return "", fmt.Errorf("input powinien zawierac imie i nazwisko
            rozdzielone spacja")
    }
    name := strings.ToLower(parts[0])
    surname := strings.ToLower(parts[1])
    var nickname string
    if len(name) >= 3 && len(surname) >= 3 {
        nickname = name[:3] + surname[:3]
    } else {
        return "", fmt.Errorf("imie i nazwisko, oba, powinny miec
            przynajmniej 3 znaki dlugosci")
    }
    return nickname, nil
}
```

funkcja **nicknameToASCII** to prosta funkcja konwertująca wygenerowany nickname na 6-elementową tablicę zawierającą kody ASCII dla danego nickname'u.

funkcja **containsSubNumber** to prosta funkcją zamieniająca liczby typu big int oraz int na string i sprawdzająca czy w dużej liczbie mieści się podciąg.

funkcja **containsAllDigits** to funkcja boolowska wykorzystująca containsSubNumber sprawdzająca czy dany wynik silni zawiera wszystkie kody ASCII z 6-elementowej tablicy.

funkcja **fibonacciWithCalls** to funkcja sprawdzająca kolejne wyrazy ciągu fibonnaciego wykorzystującą typ map[int]int do zapisywania liczby wywołań, gdzie kluczem jest kolejny wyraz ciągu, a wartością liczba wywołań (czyli kolejne wyrazy ciągu fibonnaciego). callCount[n-2] odpowiada na przesunięcie wyrazów które występuje bez tego warunku (np. wyraz 20 miał wartość wyrazu 18tego).

```
    func fibonacciWithCalls(n int, callCount map[int]int) int {
   if n < 0 {
      callCount[n] = 1
      return 1
   }
   callCount[n-2]++
   return fibonacciWithCalls(n-1, callCount) + fibonacciWithCalls(n-2,
       callCount)
}
```

funkcja **abs** to po prostu funkcja zwracająca wartosc absolutną z wartości int i zwracająca też wartość int. math.Abs zwraca typ float64.

3 main.go

Krok 1

Pobranie od użytkownicka imienia oraz nazwiska, a następnie wygenerowanie nicku i przekonwertowanie go na ASCII. ta część odbywa się w nieskończonej pętli, aż do momentu gdy użytkownik wpisze poprawnie swoje imię i nazwisko. w przeciwnym razie wyskoczy błąd i zostanie poproszony o ponowne podanie danych.

```
 func main() {

var generatedNickname string
var err error
var nicknameInASCII [6]int

for {
   fmt.Println("wpisz swoje imie i nazwisko rozdzielone spacja aby
       wygenerowac nickname:")
   reader := bufio.NewReader(os.Stdin)
   input, _ := reader.ReadString('\n')
   input = strings.TrimSpace(input)
   generatedNickname, err = generateNickname(input)
   if err == nil {
      nicknameInASCII = nicknameToASCII(generatedNickname)
      break
   }
   fmt.Println("Error:", err)
}
```

Krok 2

Wyliczenie silnej liczby. Wykorzystywane są dwa iteratory, ponieważ później w programie mocno gmatwały się typy big int, int64 (wbudowana konwersja z package'u big jest z typu big int na int64) oraz int. Prosciej było zastosować drugi iterator i pomimo, że oba mają dokładnie te samą wartość liczbową to ich użycie w programie różni się. n, big.int używane jest do wyliczania silni, natomiast strongNumber do porównywania słabych liczb.

```
 n := big.NewInt(1)
strongNumber := 1
for {
   factorialResult := factorial(n)
   if containsAllDigits(factorialResult, nicknameInASCII) {
      break
   }
   n.Add(n, big.NewInt(1))
   strongNumber++
}
```

Krok 3

Wyliczenie słabej liczby i prezentacja wyników. Do wyliczenia słabej liczby używana jest map callCount zbierajcy wyniki funkcji fibonacciWithCalls. w pętli porównywane są różnice kolejnych wartości z tego map z silną liczbą aby wyliczyć najmniejszą różnicę, a zatem słabą liczbę dla danego użytkownika

```
    callCount := make(map[int]int)
   for i := 30; i >= 1; i-- {
      fibonacciWithCalls(i, callCount)
   }
   minDiff := strongNumber
   weakNumber := 0
   weakNumber2 := 0
   for i := 0; i <= 30; i++ {
      diff := abs(callCount[i]+1 - strongNumber)
      if diff < minDiff {
         minDiff = diff
         weakNumber = i
         weakNumber2 = callCount[i] + 1
      }
   }
   fmt.Printf("Silna liczba dla %s to %d a Slaba liczba to %d [dla %d
       wywolan]:\n", generatedNickname, strongNumber, weakNumber,
       weakNumber2)
}
```

```
    go run .
    wpisz swoje imie i nazwisko rozdzielone spacja aby wygenerowac
        nickname:
    Kajetan Wisniewski
    Silna liczba dla kajwis to 297 a Slaba liczba to 18 [dla 233 wywolan]
```

Gdyby chcieć wyliczyć fib(297) prawdopodobnie zajęło by to sporo czasu.. nie wiem czy bym się kiedykolwiek doczekał, mimo, że mam całkiem mocny komputer. Tymbardziej utwierdza mnie w tym w wniosku sprawdzenie wartości ciągu aż do 300 ze strony Liczby Fibonnaciego. Gdyby wsadzić te wartości jak 297 i 18 do funkcji ackermanna to zakładam, że czarne dziury zdążyłyby wyzionąć ducha a ta dalej by się liczyła.