Algorytmy i struktury danych
Laboratorium 5

mgr inż. Andrii Shekhovtsov

11 grudnia 2023

1 Zasady oceniania

Ocena za laboratorium zależy od liczby dobrze zrobionych zadań, skala ocen jest pokazana w Tabeli 1. W niektórych przypadkach mogą być uwzględnione zadania rozwiązane częściowo.

Tabela 1: Skala ocen.

Liczba dobrze zrobionych zadań domowych	Ocena
1	3.0
2	3.5
3	4.0
4	5.0

Wykonane zadania proszę przesłać za pośrednictwem platformy moodle w postaci pliku tekstowego z kodem w Python (plik z rozszerzeniem **.py**). Proszę wrzucić kod ze wszystkich zadań do jednego pliku, rozdzielając poszczególne zadania za pomocą komentarzy.

UWAGA: Termin oddania zadania jest ustawiony w systemie moodle. W przypadku nie oddania zadania w terminie, uzyskana ocena będzie zmniejszana o 0,5 za każdy zaczęty tydzień opóźnienia.

UWAGA: W przypadku wysłania zadania w formie niezgodnej z opisem w instrukcji prowadzący zastrzega prawo do wystawienia oceny negatywnej za taką pracę. Przykład: wysłanie `.zip` lub `.pdf` tam, gdzie był wymagany plik tekstowy z rozszerzeniem `.py`.

2 Zadania do wykonania na zajęciach

1. Zaimplementować rekurencyjną funkcję obliczającą n-ty wyraz ciągu Fibonacciego (wzór poniżej). Funkcja ma używać mechanizm cache w celu przyśpieszenia obliczeń.

$$a_n = \begin{cases} 0, & n = 0 \\ 1, & n = 1 \\ a_{n-2} + a_{n-1}, & n > 1 \end{cases}$$

   Cache należy zaimplementować w postaci globalnie zadeklarowanego słownika - w ten sposób każde wywołanie funkcji będzie miało dostęp do tego słownika.

   Następnie, porównaj czas wykonania tej funkcji dla dużych liczb n z funkcją rekurencyjną i rekurencyjną implementowanymi w ramach poprzednich zajęć - jaka jest różnica?

2. Napisz prosty program z interfejsem tekstowym, który będzie imitował system logowania do pewnego systemu.

   Po uruchomieniu program ma wyświetlić menu składające się z trzech punktów:

   (a) Utwórz konto

   (b) Zaloguj się

   (c) Zakończ program

Użytkownik ma wprowadzić literę/cyfrę (zależy od implementacji) odpowiadającą opcji w menu. W przypadku wprowadzenie błędnej opcji użytkownik ma być o tym poinformowany i ponownie poproszony o dokonanie wyboru.

W przypadku wyboru **Utwórz konta** program ma poprosić użytkownika o login i hasło, a następnie zapisać parę login - hash hasła w słowniku reprezentującym bazę danych użytkowników.

W przypadku wyboru **Zaloguj się** program powinien poprosić użytkownika najpierw o podanie loginu, a potem hasła. Następie program powinien zweryfikować czy użytkownik z takim loginem istnieje i czy hash podanego przez niego hasła odpowiada temu przechowywanemu w bazie danych. Należy wyświetlić stosowny komunikat w zależności od wyniku weryfikacji.

Jako funkcje do obliczenia skrótu z hasła proszę użyć funkcje `djb2`, implementacja której jest podana poniżej. Skrót liczymy jako `h = djb2(key) % 997`.

```python
def djb2(key):
    h = 5381
    for c in key:
        h += h * 33 + ord(c)
    return h


# Przykład użycia
h = djb2('apple')
print(h)
# 244622176429
```

3. Zaimplementuj funkcje która utworzy listę wypełnioną wartościami `None`, która będzie traktowana jako tablica mieszająca przeznaczona do przechowywania n par klucz-wartość.

   Funkcja ma przyjmować argumenty n, który oznacza liczbę par do przechowywania, oraz argument m, który reguluje rozmiar "zakładki" tablicy mieszającej (dodatkowe wolne miejsce), a długość listy P jest wyznaczana jako liczba pierwsza większa lub równa iloczynowi $m \cdot N$, czyli: $P \geq m \cdot N$.

4. Zaimplementuj funkcję, która wypisze zawartość listy traktowanej jako tablica mieszająca. Zawartość listy musi być wyświetlona w dwóch kolumnach: w pierwszej muszą być wyświetlone indeksy, a w drugiej pary klucz-wartość lub wartość `None`.

```
Tablica mieszająca o długości 101
0   None
1   None
2   None
3   ('apple', 15)
4   None
...
```

5. Zaimplementuj funkcję, która będzie dodawała pary klucz-wartość do tablicy mieszającej. Funkcja ma przyjmować jako argumenty: przygotowaną przez poprzednie zadanie listę, oraz parę klucz-wartość którą należy wstawić do listy zgodnie z poniższym algorytmem:

   Funkcja wstaw(T, key, $value$)

   - Oblicz idx jako $idx = H(key)\ mod\ len(T)$
   - Jeżeli $T[idx]$ jest puste lub $T[idx][0] = key$
     - Zapisz parę (key, $value$) pod indeksem idx
     - Zakończ działanie funkcji
   - Dla wartości idx w przedziale $[0,\ len(T) - 1]$ wykonaj
     - Jeżeli $T[idx]$ jest puste lub $T[idx][0] = key$
       - Zapisz parę (key, $value$) pod indeksem idx

        * Zakończ działanie funkcji
    - Wyświetl komunikat o błędzie.
    - Zakończ działanie funkcji

3 Zadania do wykonania w domu

1. Pobrać plik tekstowy zawierający tekst książki „Alice's Adventures in Wonderland" (https://www.gutenberg.org/cache/epub/11/pg11.txt).

   Następnie, należy wczytać ten plik do programu oraz zliczyć liczbę występowań poszczególnych słów w pliku. Na końcu program powinien wyświetlić top 10 słów występujących w tekście. Zadanie należy wykonać bez użycia klasy `Counter`.

2. Zadanie jest tłumaczeniem Problemu 14 ze strony Project Euler https://projecteuler.net/problem=14.

   Dla zbioru dodatnich liczb całkowitych zdefiniowano następującą sekwencję iteracyjną:

   $$n \to n/2 \qquad (n \text{ is even})$$
   $$n \to 3n+1 \quad (n \text{ is odd})$$

   Używając tych reguł i zaczynając od 13, wygenerujemy następujący ciąg:

   $$13 \to 40 \to 20 \to 10 \to 5 \to 16 \to 8 \to 4 \to 2 \to 1$$

   Widać, że uzyskany ciąg (zaczynający się od 13 i kończący się na 1) zawiera 10 wartości. Mimo że nie istnieje formalnego dowodu na to, uważa się że niezależnie od liczby początkowej ciąg ten zakończy się na 1.

   Która liczba poniżej 1 miliona generuje najdłuższy ciąg?

   UWAGA: W trakcie obliczeń wyrazy ciągu mogą przekraczać jeden milion.

   Komentarz do zadania: *Należy napisać kod, odpowiadający na postawione w problemie pytanie. Po założeniu konta na wyżej wspomnianej stronie można sprawdzić czy obliczona przez nasz kod odpowiedź jest dobra czy nie. Podejście do rozwiązania „wprost" będzie liczyło odpowiedź przez jakiś czas. Można jednak zastosować sprytniejszego podejścia, dodając słownik który będzie pamiętał długości ciągów które są generowane przez te lub inne liczby. W ten sposób nie będziemy musieli po raz kolejny obliczać ciągów długości których były już przez nas zliczane.*

3. Napisz funkcję która znajdzie kolizję dla podanego jako argument hash. Jako funkcja hash należy użyć funkcję używaną w zadaniach do zrobienia na zajęciach (`h = djb2(key) % 997`).

   Funkcja powinna działać mniej-więcej zgodnie z następującym algorytmem:

   - Wygeneruje losową kombinacje liter alfabetu angielskiego o zadanej długości.
   - Oblicza hash z wygenerowanej kombinacji.
   - Powtarzaj poprzednie dwa kroki dopóki nie zostanie znaleziona kombinacja z której uzyskujemy taki sam hash jak podany jako argument.
   - Zwróć znalezioną kombinacje liter.

   Przykładowe działanie takiej funkcji:

```
1 >>> find_collision(42)
2 'abklmwboe'
```

   Przy prostej weryfikacji hasła za pomocą hash możemy właśnie w taki sposób obejść weryfikacje. W przypadku, gdy znamy hash hasła ofiary i używaną funkcję hash, możemy odszukać kolizje i zalogować się poprawnie hasłem, które tak naprawdę nie jest hasłem. Przykładowo, jeżeli dla obu wierszy „tajne haslo" i „ableibmwlboe" wynik obliczeń funckji hash to 198, to możemy użyć oba napisy w celu zalogowania się na konto bo oba będą równe zapisanemu w bazie danych hashu hasła. W przypadku współczesnych systemów jednak są używane bardzo skomplikowane funkcje hash i inne zabezpieczenia, w związku z czym przeprowadzenie takiego typu ataków jest bardzo utrudnione.

4. Zaimplementować funkcję, która sprawdzi czy w liście pełniącej rolę tablicy mieszającej istnieje wartość przypisana to danego klucza. Funkcja ma przyjmować jako argumenty listę oraz klucz, wartość dla którego należy znaleźć zgodnie z algorytmem:

   Funkcja pobierz(T, key)

   - Oblicz idx jako $idx = H(key)\ mod\ len(T)$
   - Jeżeli $T[idx]$ nie jest puste i $T[idx][0] = key$
     - Zakończ działanie funkcji i **zwróć** $T[idx][1]$, będące $value$ przypisanym do klucza key
   - *Komentarz:* Rozwiązywanie kolizji
   - Dla wartości idx w przedziale $[0,\ len(T) - 1]$ wykonaj
     - Jeżeli $T[idx]$ jest puste
       - Zakończ działanie funkcji i **zwróć None**
     - Jeżeli $T[idx]$ nie jest puste i $T[idx][0] = key$
       - Zakończ działanie funkcji i **zwróć** $T[idx][1]$, będące $value$ przypisanym do klucza key
   - Wyświetl komunikat o błędzie.
   - Zakończ działanie funkcji