Algorytmy i struktury danych
Laboratorium 3

mgr inż. Bartłomiej Kizielewicz, mgr inż. Andrii Shekhovtsov

5 grudnia 2023

1 Zasady oceniania

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

Tabela 1: Skala ocen.

Liczba zrobionych zadań domowych	Ocena
1	3.0
2	4.0
3	4.5
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. Zadania oddawane później niż miesiąc po terminie ustawionym na moodle są oddawane i rozliczane w trybie indywidualnym na zajęciach lub po umówieniu się z prowadzącym.

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 wzór na obliczenie n-tego wyrazu ciągu Fibonacciego (wzór poniżej) na dwa sposoby: **iteracyjnie** i **rekurencyjnie**. W komentarzach proszę podać która implementacja jest lepsza i dlaczego. Która implementacja jest bardziej czytelna? W celach badawczych można również spróbować zmierzyć czas wykonania obu funkcji dla dużych wartości n.

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

2. Zaimplementować **rekurencyjny** algorytm wyszukiwania binarnego w posortowanej liście wartości. W komentarzach proszę podać która implementacja (iteracyjna lub rekurencyjna) tego algorytmu jest lepsza i pod jakim względem? Która z dwóch implementacji algorytmu jest bardziej czytelna?

3. Zaimplementować **rekurencyjny** algorytm sortowania przez scalanie (merge sort). Przetestować jego działanie i oszacować złożoność obliczeniową tego algorytmu. Krótko (1-2 zdania) opisać dlaczego ten algorytm ma taką złożoność obliczeniową.

4. Zaimplementować **rekurencyjny** algorytm rozwiązujący problem wieży Hanoi. Algorytm ma wypisywać kolejne kroki konieczne do przeniesienia N krążków z 1 na 3 słupek. Przykład działania niżej. Jaką złożoność obliczeniową ma ten algorytm (Jaką liczbę kroków musimy wykonać żeby przenieść N krążków)? Uzasadnij odpowiedź.

   Przykład:

```
>>> hanoi(3)
Move (1) from 1 to 3
Move (2) from 1 to 2
Move (1) from 3 to 2
Move (3) from 1 to 3
Move (1) from 2 to 1
Move (2) from 2 to 3
Move (1) from 1 to 3
```

3 Zadania do wykonania w domu

1. Zaimplementować funkcję obliczającą silnię podanej jako argument funkcji liczby n. Funkcja powinna używać rekurencji do obliczenia silni. Rekurencyjnie wzór na silnie możemy zdefiniować jako:

$$n! = \begin{cases} 1 & gdy\ n = 0 \\ n \cdot (n-1)! & inaczej \end{cases} \tag{2}$$

2. Zaimplementować funkcje który obliczy i wypisze kolejne elementy sekwencji Collatza (wzór poniżej) używając rekurencji. Algorytm ma wystartować od podawanej jako argument funkcji liczby c_n i działać do momentu aż osiągniemy wartości $c_{n+1} = 1$.

$$c_{n+1} = \begin{cases} \frac{1}{2}c_n & gdy\ c_n\ jest\ parzysta \\ 3c_n + 1 & gdy\ c_n\ jest\ nieparzysta \end{cases} \tag{3}$$

Przykład działania funkcji:

```
>>> collatz(5)
5
16
8
4
2
1
```

Podpowiedź: Należy wypisać liczbę będącą argumentem funkcji na początku funkcji, a potem rekurencyjnie wywoływać tą samą funkcję z odpowiednio zmodyfikowanym argumentem (zgodnie ze wzorem).

3. Zaimplementuj funkcje obliczającą największy wspólny dzielnik dwóch liczb a i b za pomocą rekurencji. Liczby a i b powinny być argumentami funkcji. Rekurencyjny wzór na obliczenia NWD można przedstawić następująco:

$$NWD(a,b) = \begin{cases} a & gdy\ b = 0 \\ NWD(b, a\ mod\ b) & inaczej \end{cases} \tag{4}$$

4. Zaimplementować **rekurencyjny** algorytm szybkiego sortowania (quicksort) i przetestować jego działanie.