Algorytmy i struktury danych - laboratorium

Lista nr 1

Janusz Szwabiński

Zad. 1 Stwórz klasę `Fraction` do reprezentacji ułamków zwykłych. Klasa powinna mieć zaimplementowane podstawowe działania arytmetyczne ($+,-,*,/$) oraz umożliwiać wypisanie ułamków na ekranie. Przykładowa sesja z użyciem tej klasy mogłaby wyglądać następująco:

```
>>> f1=Fraction(1,4)
>>> f2=Fraction(1,2)
>>> f3=f1+f2
>>> print(f3)
6/8
>>>
```

Zad. 2 Dodaj do klasy metody pozwalające na porównywanie ułamków.

Zad. 3 Dodaj do klasy metody `getNum` i `getDem` zwracajace odpowiednio wartość licznika i mianownika ułamka.

Zad. 4 Zmodyfikuj konstruktor klasy tak, aby sprawdzał, czy licznik i mianownik są liczbami całkowitymi i zgłaszał wyjątek, jeżeli nie są.

Zad. 5 Czy zaimplementowane przez Ciebie operatory porównań działają poprawnie w przypadku ułamka zainicjalizowanego z ujemnym mianownikiem? Jeżeli nie, popraw definicję klasy.

Zad. 6 Zmodyfikuj metodę `__str__` tak, aby ułamek był wypisywany na ekran w postaci nieskracalnej.

Zad. 7 Zmodyfikuj konstruktor klasy tak, aby ułamek był przechowywany w postaci nieskracalnej.

Algorytmy i struktury danych - laboratorium

Lista nr 2

Janusz Szwabiński

Zad. 1 Dla każdej grupy funkcji posortuj je od najmniejszej do największej złożoności asymptotycznej:

a) grupa 1:

$$\begin{aligned} f_1(n) &= n^{0.999999} \log n \\ f_2(n) &= 10000000n \\ f_3(n) &= 1.000001^n \\ f_4(n) &= n^2 \end{aligned}$$

b) grupa 2:

$$\begin{aligned} f_1(n) &= 2^{100n} \\ f_2(n) &= \binom{n}{2} \\ f_3(n) &= n\sqrt{n} \end{aligned}$$

c) grupa 3:

$$\begin{aligned} f_1(n) &= n^{\sqrt{n}} \\ f_2(n) &= 2^n \\ f_3(n) &= n^{10}2^{n/2} \\ f_4(n) &= \sum_{i=1}^{n}(i+1) \end{aligned}$$

W razie wątpliwości posiłkuj się wykresami funkcji.

Zad. 2 Trójka pitagorejska to trzy całkowite liczby dodatnie a, b i c spełniające równanie

$$a^2 + b^2 = c^2$$

Istnieje tylko jedna trójka taka, że

$$a + b + c = 1000$$

Znajdź abc.

Zad. 3 Podaj liczbę działań potrzebnych do rozwiązania poprzedniego zadania.

Algorytmy i struktury danych - laboratorium

Lista nr 3

Janusz Szwabiński

Zad. 1 Jeżeli prawdopodobieństwo pojedynczego sukcesu wynosi p, to prawdopodobieństwo osiągnięcia co najwyżej k sukcesów wyrazi się wzorem:

$$P(n,k) = \sum_{i=0}^{k} \binom{n}{i} p^i (1-p)^{n-i}$$

Napisz funkcję wyliczającą to prawdopodobieństwo. Nie może ona wymagać więcej niż $3k + \log n$ mnożeń.

Zad. 2 Ile potrzeba mnożeń, aby wyliczyć wartość wielomianu stopnia n o współczynnikach zawartych w liście `a`. Napisz funkcję realizującą Twój algorytm.

Zad. 3 Napisz program, który policzy, ile razy występuje każdy znak w pliku tekstowym podanym jako argument wywołania. Nie możesz przy tym używać wyrażenia warunkowego `if`.

Algorytmy i struktury danych - laboratorium

Lista nr 4

Janusz Szwabiński

Zad. 1 Zaimplementuj kolejkę przy użyciu pythonowych list w taki sposób, aby:

- koniec kolejki znajdował się na końcu listy,
- koniec kolejki znajdował się na początku listy.

Zad. 2 Zaprojektuj i przeprowadź eksperyment porównujący wydajność obu implementacji.

Zad. 3 Rozważ sytuację z życia wziętą, np.:

- auta w kolejce do myjni,
- kasy w supermarkecie,
- samoloty na pasie startowym,
- okienko w banku.

Postaw pytanie badawcze. Wykorzystując liniowe struktury danych zaprojektuj i przeprowadź symulację, która udzieli na nie odpowiedzi. Pamiętaj o określeniu wszystkich uproszczeń swojego modelu.

Zad. 4 Napisz program, który sprawdzi poprawność składni dokumentu HTML pod kątem brakujących znaczników zamykających.

Zad. 5 Dodaj brakujące metody do klasy `UnorderedList` prezentowanej na wykładzie.

Zad. 6 Zaimplementuj stos przy pomocy listy jednokierunkowej.

Zad. 7 Zaimplementuj kolejkę dwustronną przy pomocy listy jednokierunkowej.

Zad. 8 Zaprojektuj i przeprowadź eksperyment porównujący wydajność listy jednokierunkowej i listy wbudowanej w Pythona.

Algorytmy i struktury danych - laboratorium

Lista nr 5

Janusz Szwabiński

Zad. 1 Jednym z ważniejszych zagadnień metod numerycznych jest rozwiązanie układu równań liniowych,

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n &= b_2 \\ &\vdots \\ a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n &= b_n \end{aligned}$$

ze względu na niewiadome x_1, x_2, ..., x_n.

Tego typu równania można rozwiązać przy pomocy funkcji `solve` z modułu `scipy.linalg`, np.

```
>>> import numpy as np
>>> a = np.array([[3, 2, 0], [1, -1, 0], [0, 5, 1]])
>>> b = np.array([2, 4, -1])
>>> from scipy import linalg
>>> x = linalg.solve(a, b)
>>> x
array([ 2., -2., 9.])
>>> np.dot(a, x) == b #sprawdzenie!!!
array([ True, True, True], dtype=bool)
```

Przeprowadź analizę eksperymentalną złożoności obliczeniowej funkcji `solve`. Rozmiarem danych wejściowych jest liczba niewiadomych w równaniu.

Zad. 2 Napisz program rozwiązujący zagadnienie wieży z Hanoi. Użyj trzech stosów do przechowywania krążków.

Zad. 3 Korzystając z modułu `turtle`, napisz program, który narysuje krzywą Hilberta (`https://pl.wikipedia.org/wiki/Krzywa_Hilberta`).

Zad. 4 Podobnie, napisz program, który narysuje krzywą Kocha (`https://pl.wikipedia.org/wiki/Krzywa_Kocha`).

Algorytmy i struktury danych - laboratorium

Lista nr 6

Janusz Szwabiński

Zad. 1 Stwórz własną klasę implementującą binarne drzewa przeszukiwań. Zadbaj o poprawne przetwarzanie powtarzających się kluczy.

Zad. 2 Zaimplementuj kopiec binarny. Korzystając z tego kopca napisz funkcję sortującą listę elementów w czasie $O(n \log n)$. Przeprowadź analizę eksperymentalną czasu wykonania algorytmu.

Zad. 3 Zaimplementuj kopiec binarny o ograniczonej wielkości n. Innymi słowy, stwórz strukturę przechowującą n najważniejszych (największych) wartości.

Zad. 4 Napisz funkcję, która na wejściu przyjmuje drzewo wyprowadzenia jakiegoś wyrażenia matematycznego, a na wyjściu zwraca pochodną tego wyrażenia względem podanej zmiennej.

Algorytmy i struktury danych - laboratorium

Lista nr 7

Janusz Szwabiński

Zad. 1 Zaimplementuj własną klasę `Graph` o własnościach podanych na wykładzie.

Zad. 2 Dodaj do powyższej klasy metodę generującą reprezentację grafu w języku `dot`. Użyj programu `graphviz` (lub jego wersji online: `http://www.webgraphviz.com/`) do przedstawienia wyniku na rysunku.

Zad. 3 Rozbuduj klasę o metody przeszukiwania w głąb i wszerz.

Zad. 4 Zmodyfikuj metodę przeszukiwania w głąb tak, aby sortowała ona graf topologicznie.

Zad. 5 Korzystając z przeszukiwania wszerz, stwórz algorytm wyliczający najkrótsze ścieżki od dowolnego węzła grafu do wszystkich pozostałych.

Zad. 6 Korzystając z grafów, napisz program rozwiązujący zagadnienie misjonarzy i kanibalów (`https://en.wikipedia.org/wiki/Missionaries_and_cannibals_problem`).

Zad. 7 Napisz program, który znajdzie sposób na odmierzenie dwóch litrów wody przy użyciu dwóch kanistrów o objętości 3l i 4l.