додатки треба доробити, наприклад бібліотека SymPy потребує збільшення кількості платформ які її підтримують.

Список використаних джерел

1. *Leigh C. Becker Ordinary Differential Equations: Concepts, Methods, and Models. – Christian Brothers University Memphis, TN 2012–2013 Edition*
2. *http://hypertextbook.com/eworld/packages/*
3. *Гой Т. П. Диференціальні рівняння : навчальний посібник / Т. П. Гой, О. В. Махней. – Івано-Франківськ : Сімик, 2012. – 352 с*
4. *Wolfram Mathematica. URL: https://www.wolfram.com/mathematica/*
5. *Maplesoft Maple. URL: https://de.maplesoft.com/products/maple/*
6. *Python SymPy. URL: https://docs.sympy.org/latest/index.html*
7. *https://www.tutorialspoint.com/sympy/sympy_quick_guide.htm*

УДК 004.01

Бєляєв О.Д., здобувач кафедри прикладної математики
Ліваковський В.К., здобувач кафедри інформаційних технологій

ЕМПІРИЧНІ ОЦІНКИ СКЛАДНОСТІ ДЕЯКИХ АЛГОРИТМІВ В PYTHON

Донецький національний університет імені Василя Стуса, м. Вінниця

З теоретичної точки зору основним методом оцінки ефективності того чи іншого алгоритму за часовою та просторовою характеристиками є асимптотичний аналіз складності алгоритмів. Математичною основою аналізу складності алгоритмів є *O*-символіка Ландау [1].

Асимптотична складність алгоритму обчислюється відносно набору вхідних даних: найкращого випадку (найменша кількість операцій для реалізації алгоритму для конкретного екземпляра задачі – набору вхідних даних), найгіршого випадку (найбільша кількість необхідних дій) та середнього випадку (проміжний варіант).

В даній роботі для прикладу досліджений елементарний алгоритм лінійного пошуку [1] – алгоритм, який зазвичай з методичної точки зору розглядається для демонстрації методології асимптотичного аналізу складності алгоритмів.

Реалізація найгіршого випадку алгоритму лінійного (послідовного) пошуку має оцінку *O(n)*; середній випадок також має лінійну оцінку, а конкретно

$T(n) = \frac{n+1}{2}$ – у випадку, якщо шуканий елемент гарантовано наявний в наборі

$T(n) = \frac{n+2}{2}$ – у випадку, якщо шуканий елемент може бути наявним в наборі, а може бути і відсутнім.

де n – кількість елементів в наборі, в якому ми шукаємо значення x.

На лістингу 1 представлена елементарна програмна реалізація задачі послідовного пошуку у випадку гарантованої наявності значення x в наборі (array). Розмірність масиву даних обрана 10^8 елементів. Реалізується 10 заходів по 10^6 експериментів (number_of_experiments) в кожному.

Лістінг 1.

```
from random import randint

def DeviationMathExpectation(value, exp_value):
    deviation = int(abs(1 - value/expected_value) * 10)
    return deviation if deviation < 5 else 5

N = 10**8
expected_value = (N+1)/2
array = [0] * N
x = 1
number_of_experiments = 10**6
for k in range(10):
    results = [0] * 6
    for i in range(number_of_experiments):
        rand_index = randint(0, N-1)
        array[rand_index] = x
        for i, value in enumerate(array):
            if value == x:
                break
        array[rand_index] = 0'
        index = DeviationMathExpectation(i+1, expected_value)
        results[index] += 1
```

В програмі аналізується віддаленість в кожній групі експериментів реальної кількості порівнянь задля знаходження позиції шуканого елемента. Зазначимо, що наведений код має більше методичний характер, оскільки в контексті даної задачі рядки власне пошуку елемента *for i, value in enumerate(array): if value == x: break* могли б просто бути відкинутими, оскільки шукана позиція i (відповідна кількість порівнянь для оцінки $i+1$) просто дорівнює значенню rand_index.

В представленій таблиці 1 наведені значення результатів експериментів – відносного відхилення від математичного очікування – параметра expected_value.

Таблиця 1.

#	0%..10%	10%..20%	20%..30%	30%..40%	40%..50%	> 50%
1)	9,95	10,05	10,07	10,01	9,97	49,96
2)	9,99	10,05	10,04	9,99	9,98	49,96
3)	10,01	10,00	9,97	9,98	9,94	50,11
4)	10,05	10,01	10,04	9,96	9,97	49,96
5)	9,96	9,99	10,00	10,05	10,02	49,98
6)	10,00	9,97	10,01	10,01	9,98	50,03
7)	9,99	9,96	10,05	10,02	9,97	50,01
8)	9,96	10,02	10,03	10,07	9,99	49,94
9)	10,02	10,03	10,01	9,96	10,00	49,98
10)	9,99	9,99	10,04	9,94	10,00	50,04

Аналізуючи дані таблиці 1, ми спостерігаємо в принципі очікувану картину. Відхилення від теоретично обчисленої складності середнього випадку с кожним збільшенням відхилення на 10% охоплює приблизно 10% випадків закінчення експерименту. При цьому відхилення більше ніж в половину спостерігається приблизно в 50% відсотків випадків.

Більш цікавим випадком дослідження є та ситуація, коли ми заздалегідь не знаємо – чи наявний елемент *x* в масиві чи ні.

Заповнення масиву array будемо моделювати

```
left, right = 0, N
array = tuple(randint(left, right) for i in range(N))
```

тобто розмах нашої вибірки співпадає розмірністю вхідного масиву даних.

В зазначеній серії експериментів отримаємо аналогічну таблицю результатів

Таблиця 2.

#	0%..10%	10%..20%	20%..30%	30%..40%	40%..50%	> 50%	false
1)	7,0	8,0	7,0	5,0	5,0	35,0	33,0
2)	9,0	7,0	3,0	6,0	8,0	33,0	34,0
3)	5,0	7,0	5,0	5,0	6,0	27,0	45,0
4)	6,0	5,0	8,0	5,0	3,0	30,0	43,0
5)	2,0	5,0	5,0	5,0	4,0	37,0	42,0
6)	5,0	8,0	8,0	4,0	8,0	32,0	35,0
7)	5,0	7,0	4,0	7,0	1,0	41,0	35,0
8)	6,0	3,0	7,0	6,0	2,0	41,0	35,0
9)	5,0	11,0	4,0	4,0	6,0	23,0	47,0
10)	6,0	6,0	5,0	5,0	10,0	38,0	30,0

З даних, представлених в таблиці 2, можемо спостерігати, що в реальних випадках, коли ми задаємо випадковий набір чисел, на відміну від

теоретичних конструкцій, ми завжди залежимо від розмаху вибірки та розмірності масиву.

Справедливою є математична оцінка відсотка попадань в найгірший випадок роботи алгоритму – випадок, коли значення *x* відсутнє в наборі при великих обсягах даних

$$\lim_{N\to\infty}\left(1-\frac{1}{N}\right)^{n}=e^{-\frac{n}{N}},$$

де N – розмах вибірки, n – розмірність масиву даних.

Тобто, в прикладних реалізаціях наших оцінок складності ми маємо робити поправку та той спосіб, в який ми формуємо наші дані, бо неврахування математичної постановки задачі і реальної програмної реалізації може вносити серйозну похибку в наші комп'ютерні обчислення при збереженні формальної коректності нашого алгоритму.

Список використаних джерел

1. *Кормен Т.Г., Лейзерсон Ч.Е., Рівест Р.Л., Стайн К. Вступ до алгоритмів. – К.: К.І.С., 2019. –1288 с.*

УДК 519.67, 623.44

Варер Б.Ю., магістрант
Крикун І.Г., к.ф.-м.н., доцент,
доцент кафедри прикладної
математики

ІДЕНТИФІКАЦІЯ ВОГНЕПАЛЬНОЇ ЗБРОЇ ЗА АКУСТИЧНИМИ СИГНАЛАМИ РОБОТИ ЇЇ МЕХАНІЗМІВ

Донецький Національний університет імені Василя Стуса, м. Вінниця

Вступ. Криміналістичне розслідування широко використовує методи відео- та аудіофіксації в своїй роботі. Станом на 2008 рік основним способом дослідження та співставленні звуків пострілу, особливостей голосу, інтонацій, тощо, був органолептичний метод – тобто, на слух [1]. За час, що минув, якість засобів фіксації звуку та відео значно зросла, що дало можливість отримувати набагато якісніші дані зі звукового середовища події правопорушення. Дане дослідження присвячене встановленню ефективного метода ідентифікації типу операції, що здійснюється зі зброєю методом машинного навчання.