МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний аерокосмічний університет ім. М.Є. Жуковського

«Харківський авіаційний інститут»

Факультет радіоелектроніки, комп’ютерних систем та інфокомунікацій

Кафедра комп’ютерних систем, мереж і кібербезпеки (503)

Лабораторна робота № *5*

|  |  |
| --- | --- |
|  | *Дослідження організації даних методом розстановки* |
|  | (назва лабораторної роботи) |
| з дисципліни | *Моделі та структури даних* |
|  | (шифр)  ХАІ**.**503**.**525a**.**03О**.**123-Комп'ютерна інженерія**,** ПЗ №9629619 |

|  |  |  |
| --- | --- | --- |
| Виконав студент гр. | 525а | *Литвиненко А.В.* |
| 22.11.2022 | (№ групи) | (П.І.Б.) |
| (підпис, дата) |  |  |
| Перевірив | канд. техн. наук, доцент | |
|  |  | *А. В. Шостак* |
| (підпис, дата) |  | (П.І.Б.) |

Харків – 2022

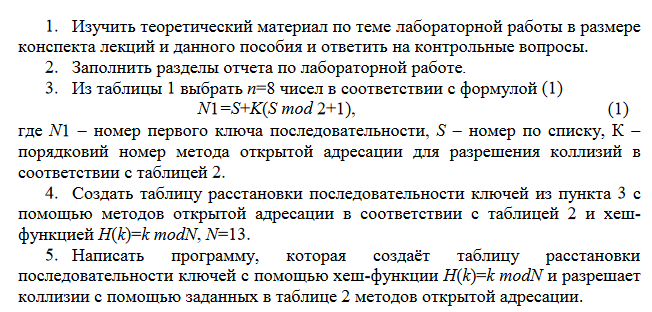
**Тема роботи:** дослідження організації даних методом розстановки.

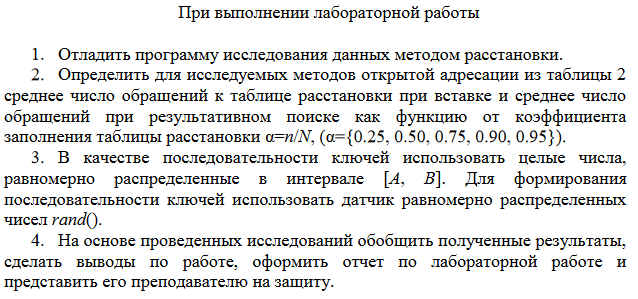
**Варіант 5**

**Задача 1**

**Частина 1**. Постановка завдання

**Умова:**

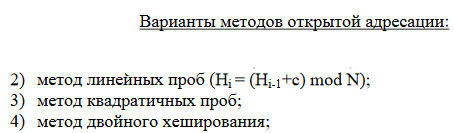




**Умова з додатка:**







**Частина 2**. Схема класу

На основі постановки завдання розроблена схема, представлений на рисунку 1.

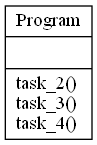


Рисунок 1 - Схема класу перетворення

**Частина 4**. Текст програми

Відповідно до розробленого алгоритму в середовищі Microsoft Visual Studio була написана програма, яка наведена нижче.

Main.py

import math

from pprint import pprint

# File: main.py

# Author: Lytvynenko Andrii 525a

# Date: 27.11.2022

# Інтерва розподілу чисел [A,B]

Interval = [0,1000]

# Розмірність таблиці

N = 83

# Рівномірно розподілені числа

Table1\_Array = [23, 20, 90, 25, 60, 99, 1, 90, 25, 29]

# Для методу лінійних проб

N1 = 5 + 2\*(5%2+1) # 9

# Для квадратичних проб

N2 = 5 + 3\*(5%2+1) # 11

# Метод подвійного хешування

N3 = 5 + 4\*(5%2+1) # 13

L = N3 - 3

# N1 = S+K(S % 2 + 1)

# N1 - номер першого ключа последовательности

# S - номер по списку

# K - порядковий номер метода відкритої адресації для дозволу колізії в табл 2

# Варіанти методів відкритої адресації

# 2) метод лінійниз проб (H.i = (H.i-1 + c) % N)

# 3) метод квадратичних проб

# 4) метод подвійного хешування

"""

Значенняя з презентації

Table1\_Array = [1,0,13,26,39]

N = 13

c = 2

"""

def H(k: int) -> int:

return k % N

def H1(k: int) -> int:

return k % L

def task\_2():

# 2) метод лінійниз проб (H.i = (H.i-1 + c) % N)

"""

H0 = H(k)

Hi = (H0 + i) % N, i = [1, ..., N-1]

OR

Hi = (H(i-1) + c) % N

c - константа, така що c та N взаємно прості

i - індекс повторного хешування

"""

N = N1

O\_list = []

kolizion = []

d = {}

tlength = int(math.sqrt(max(Table1\_Array)))+2

i = 0

for k in range(tlength):

d[i] = 0

i += 1

# Вставка

for k in Table1\_Array:

hX = H(k)

count = 1

while hX in kolizion:

hX = H(hX) + 2

count += 1

if count > tlength:

break

kolizion.append(hX)

d[hX] = k

O\_list.append(count)

print("H = ", kolizion)

print("O = ", O\_list)

pprint(d)

print("Si = ", sum(O\_list) / len(O\_list))

print("Alpha = ", round(len(O\_list)/N, 4))

# Пошук

Oi = O\_list

print("Sf = ", sum(Oi) / len(Oi))

def task\_3():

# 3) метод квадратичних проб

"""

H0 = H(k)

Hi = (H0 + i\*\*2) % N, i > 0

OR

Hi = (H0 + c1\*i + c2\*i) % N

i - індекс повторного хешування

"""

N = N2

O\_list = []

kolizion = []

d = {}

i = 0

for k in range(int(math.sqrt(max(Table1\_Array)))+3):

d[i] = 0

i += 1

# Вставка

I = 1

for k in Table1\_Array:

hX = H(k)

count = 1

while hX in kolizion:

hX = (H(hX) + 1 \*\* 2) % N

count += 1

kolizion.append(hX)

d[hX] = k

O\_list.append(count)

print("H = ", sorted(kolizion))

print("O = ", O\_list)

# print(d)

pprint(d)

print("Si = ", sum(O\_list) / len(O\_list))

print("Alpha = ", round(len(O\_list)/N, 4))

# Пошук

Oi = O\_list

print("Sf = ", sum(Oi) / len(Oi))

def task\_4():

"""

Використовується 2 функції:

H(k) = q(k) % N - исходня хеш-функція

H1(k) = q(k) % L - додаткова хеш фукнція

N та L вибираються взаємно простими

Метод подвійного хешування:

H0 = H(k) = q(k) % N

Hi = ( H(k) + i\*H1(k) ) % N, i>0

i - індекс повторного хешування

"""

# 4) метод подвійного хешування

N = N3

# example data

O\_list = []

kolizion = []

d = {}

i = 0

for k in range(int(math.sqrt(max(Table1\_Array)))+3):

d[i] = 0

i += 1

# Вставка

I = 1

kol\_repeat = False

for k in Table1\_Array:

hX = H(k)

count = 1

while hX in kolizion:

if H(hX) not in kolizion:

hX = H(hX)

else:

hX = (H(k) + I\*H1(k)) % N

count += 1

I += 1

if hX in kolizion:

kol\_repeat = True

break

if not kol\_repeat:

kolizion.append(hX)

d[hX] = k

O\_list.append(count)

kol\_repeat = False

print("H = ", sorted(kolizion))

print("O = ", O\_list)

# print(d)

pprint(d)

print("Si = ", sum(O\_list) / len(O\_list))

print("Alpha = ", round(len(O\_list)/N, 4))

# Пошук

Oi = O\_list

print("Sf = ", sum(Oi) / len(Oi))

def print\_decorator(text, func):

print(text)

func()

print()

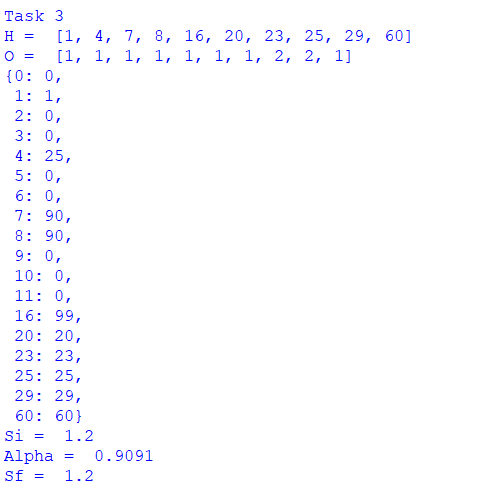
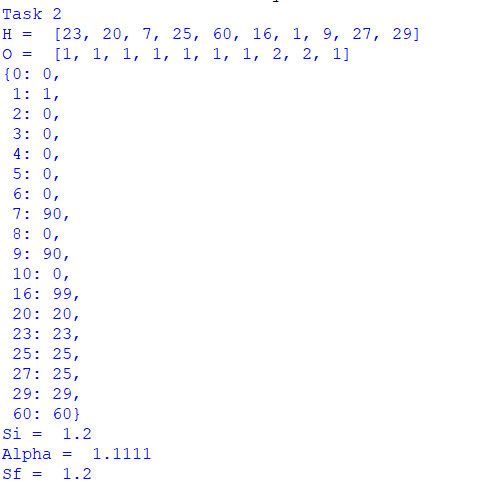
print\_decorator("Task 2", task\_2)

print\_decorator("Task 3", task\_3)

print\_decorator("Task 4", task\_4)

**Частина 5**. Тестування

**Скриншот тестування:**



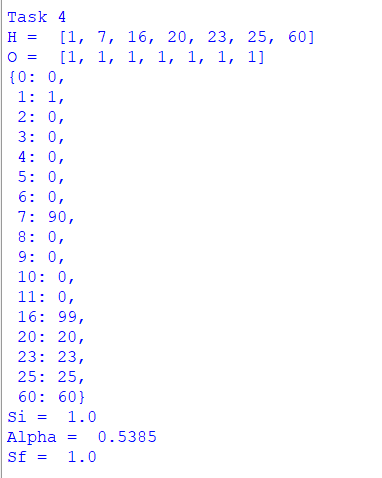


Рисунок 2 – скришот тестування

**Висновки**

Під час цієї лабораторної роботи я вивчив матеріал лабораторної роботи у розмірі конспекту і додаткових матеріалів з інтернеті та виконав поставлене завдання.