Національний технічний університет України «КПІ ім. Ігоря Сікорського»

Факультет Інформатики та Обчислювальної Техніки

Кафедра Автоматизованих Систем Обробки Інформації та Управління

Лабораторна робота № 8

з дисципліни «Теорія алгоритмів»

на тему:

**" Хеш-функції”**

Виконала:

студентка гр. ІС-02

Павлущенко Ольга

Викладач:

Новікова П. А.

Київ – 2020

**Завдання**

В роботі необхідно реалізувати різні типи хеш-таблиць із використанням різних хеш-функцій для розв’язання наведеної вище задачі. При цьому потрібно порівняти ефективність різних підходів шляхом підрахунку кількості колізій для кожного типу хеш-функцій та хеш-таблиць.

Нижче наведений перелік типів хеш-таблиць та хеш-функцій:

1. Хеш-таблиця на основі ланцюгів (chained hash) із використанням хеш-функції за методом ділення.

2. Хеш-таблиця на основі ланцюгів (chained hash) із використанням хеш-функції за методом множення.

**Програмний код**

def main():

print("Enter name of file : ", end = '')

name = input() # назва файлу вхідних даних

\_input = GetInfo(name) #отримуємо зчитані дані

A = \_input[0] #масив елементів

S = \_input[1] #масив сум

list1 = CheckHashTable1(A,S) #отримуємо результат роботи програми

list2 = CheckHashTable2(A,S)

WriteToFile(list1) #записуємо результат роботи програми у файл

WriteToFile2(list2)

def DivisionHash(key, length): #функція для отримання значення за методом ділення

return key%length # остача від ділення

def MultiplicationHash(key, length): #функція для отримання значення за методом множення

return int(length \* ((key \* 0.6180339887) % 1))

def CheckHashTable1(A,S): #використовуючи метод ділення

result = [] #список результатів виконання програми

T = [[None]] \* 3 \* len(A) #хеш-таблиця

colisions = 0 # к-сті колізій

for i in A :

pos = DivisionHash(i, len(T)) #позиція в хеш-таблиці

if T[pos] == [None]: #якщо в комірці хеш-таблиці не записане значення

T[pos] = [i] #записуємо значення

elif(i != T[pos][0]): #якщо в комірці хеш-таблиці записане якесь значення

T[pos].append(i) #створюємо ланцюг

if(len(T[pos])>1): #якщо в хеш-таблиці є ланцюг

colisions += 1 #к-сть колізій збільшується на одну

result.append(colisions) #записуємо в масив-результат значення к-сті колізій

for sum in S:

for i in A: # елемент із заданого масиву

if(Search(T, sum - i)): #пошук в хеш-таблиці елемента, який утворив би задану суму

result.append([i,sum-i]) #додаємо у масив-результат

break

else: #якщо не знайдено другого елемента що утворює задану суму

result.append([0,0])

return result

def CheckHashTable2(A,S): #використовуючи метод множення

result = []

T = [[None]] \* 3 \* len(A) #хеш-таблиця

colisions = 0 #змінна для запису к-сті колізій

for i in A :

pos = MultiplicationHash(i, len(T)) #позиція в хеш-таблиці

if T[pos] == [None]:

T[pos] = [i]

elif(i != T[pos][0]):

T[pos].append(i)

if(len(T[pos])>1):

colisions += 1

result.append(colisions)

for sum in S:

for first in A:

if(Search(T, sum - first)):

result.append([first,sum-first])

break

else:

result.append([0,0])

return result

def Search(T, number): #функція для пошуку елемента у хеш-таблиці, який утворює задану суму

for j in T:

if(j[0] == number and j[0] != None):

return True

return False

def GetInfo(input\_file): #функція зчитування масиву із файлу

file = open(input\_file, "r")

arr = file.read().split('\n')

file.close()

numbers = arr[0].split(' ')

listA = [0] \* int(numbers[0])

listS = [0] \* int(numbers[1])

for i in range(1,len(listA)+1):

listA[i-1] = int(arr[i])

for i in range(len(listA)+1,len(arr)):

listS[i-len(listA)-1] = int(arr[i])

return [listA,listS]

def WriteToFile(result): #функція запису результатів у файл

file2 = open("IS0-2\_Pavlushchenko\_07\_outputDivision.txt", "w")

file2.write(str(result[0])+"\n") # к-сть колізій

for i in range(1,len(result)): #результат для кожної суми

file2.write(str(result[i][0]) + " " + str(result[i][1]) + "\n")

file2.close()

def WriteToFile2(result): #функція запису результатів у файл

file3 = open("IS-02\_Pavlushchenko\_07\_outputMultiplication.txt", "w")

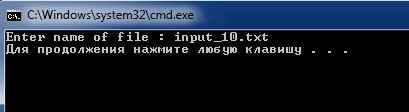
file3.write(str(result[0])+"\n")

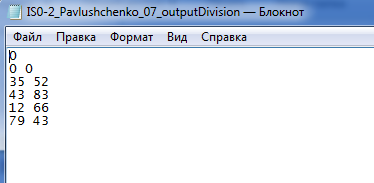
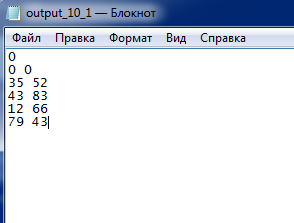
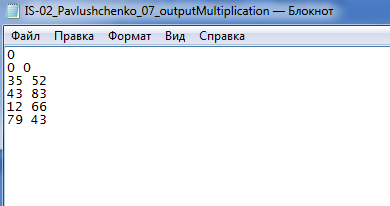
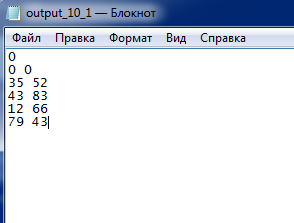
for i in range(1,len(result)):

file3.write(str(result[i][0]) + " " + str(result[i][1]) + "\n")

file3.close()

main()

**Результати**



**Висновок**

У ході даної лабораторної роботи було реалізовано різні типи хеш-таблиць із використанням різних хеш-функцій. Порівняно ефективність різних підходів шляхом підрахунку кількості колізій для кожного типу хеш-функцій та хеш-таблиць. Доступ до довільної позиції в масиві виконується за час О(1)