Національний технічний університет України «КПІ ім. Ігоря Сікорського»

Факультет Інформатики та Обчислювальної Техніки

Кафедра Автоматизованих Систем Обробки Інформації та Управління

Лабораторна робота № 7

з дисципліни «[Теорія алгоритмів](http://wiki.kpi.ua/index.php/%D0%A2%D0%B5%D0%BE%D1%80%D1%96%D1%8F%20%D0%B0%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC%D1%96%D0%B2_(20202020))»

на тему

«Хеш-таблицi»

Виконав:

студент гр. ІС-91

Хмелiнiн Андрiй

Викладач:

[ст.вик. Халус О. А.](http://rozklad.kpi.ua/Schedules/ViewSchedule.aspx?v=318f7459-ddf0-47ee-b882-5650e756cf2a)

Київ – 2020

import math

def Hash\_divide(hashTableLen, k): # метод деления

return k % hashTableLen

def Hash\_multiply(hashTableLen, k): # метод умножения

return int(hashTableLen \* ((k \* ((math.sqrt(5) - 1) / 2)) % 1))

def Hash\_searchD(HashTable, k): # поиск методом деления

for i in range(len(HashTable)):

j = Hash\_divide(len(HashTable), k)

if HashTable[j][i] == 0:

return 0

elif HashTable[j][i] == k:

return j

def Hash\_searchM(HashTable, k): # поиск методом умножения

for i in range(len(HashTable)):

j = Hash\_multiply(len(HashTable), k)

if HashTable[j][i] == 0:

return 0

elif HashTable[j][i] == k:

return j

with open("Inputs/input\_100000.txt") as f: # считывание файла

data = f.read().split()

N = int(data[0]) # количество чисел

M = int(data[1]) # количество сумм

A = [] # массив чисел

S = [] # массив сумм

HashTableD = [] # хеш таблица для divide method

HashTableM = [] # хеш таблица для multiply method

colisionsD = 0 # коллизии

colisionsM = 0 # коллизии

for i in range(2, len(data) - M): # заполнение массива чисел

A.append(int(data[i]))

for j in range(2 + N, len(data)): # заполнение массива сумм

S.append(int(data[j]))

for l in range(N\*3): # заполнение хеш таблицы нулями

HashTableD.append([0])

for l in range(N\*3): # заполнение хеш таблицы нулями

HashTableM.append([0])

for q1 in range(len(A)):

HashTableD[Hash\_divide(N\*3, A[q1])].insert(-1, A[q1])

if len(HashTableD[Hash\_divide(N\*3, A[q1])]) > 2: # проверка на коллизию

colisionsD += 1

for q2 in range(len(A)):

HashTableM[Hash\_multiply(N\*3, A[q2])].insert(-1, A[q2])

if len(HashTableM[Hash\_multiply(N\*3, A[q2])]) > 2: # проверка на коллизию

colisionsM += 1

### D ###

with open("Outputs/khmelinin\_output\_100000\_01.txt", 'w') as file: # запись в файл

file.write(str(colisionsD) + "\n")

for kk in range(len(S)):

for h in range(len(A)):

if Hash\_searchD(HashTableD, S[kk] - A[h]) != 0: # алгоритм поиска элемента в хеш таблице

for u in range(30):

if S[kk] - A[h] == HashTableD[Hash\_searchD(HashTableD, S[kk] - A[h])][u]: # если есть элемент сумма - элемент A[h],

file.write(str(A[h]) + " " + str(HashTableD[Hash\_searchD(HashTableD, S[kk] - A[h])][u]) + "\n") # то записываем элемент A[h] и его ключ

break

break

elif h == len(A)-1:

file.write("0 0" + "\n")

### M ###

with open("Outputs/khmelinin\_output\_100000\_02.txt", 'w') as file: # запись в файл

file.write(str(colisionsM) + "\n")

for kk in range(len(S)):

for h in range(len(A)):

if Hash\_searchM(HashTableM, S[kk] - A[h]) != 0: # алгоритм поиска элемента в хеш таблице

for u in range(30):

if S[kk] - A[h] == HashTableM[Hash\_searchM(HashTableM, S[kk] - A[h])][u]: # если есть элемент сумма - элемент A[h],

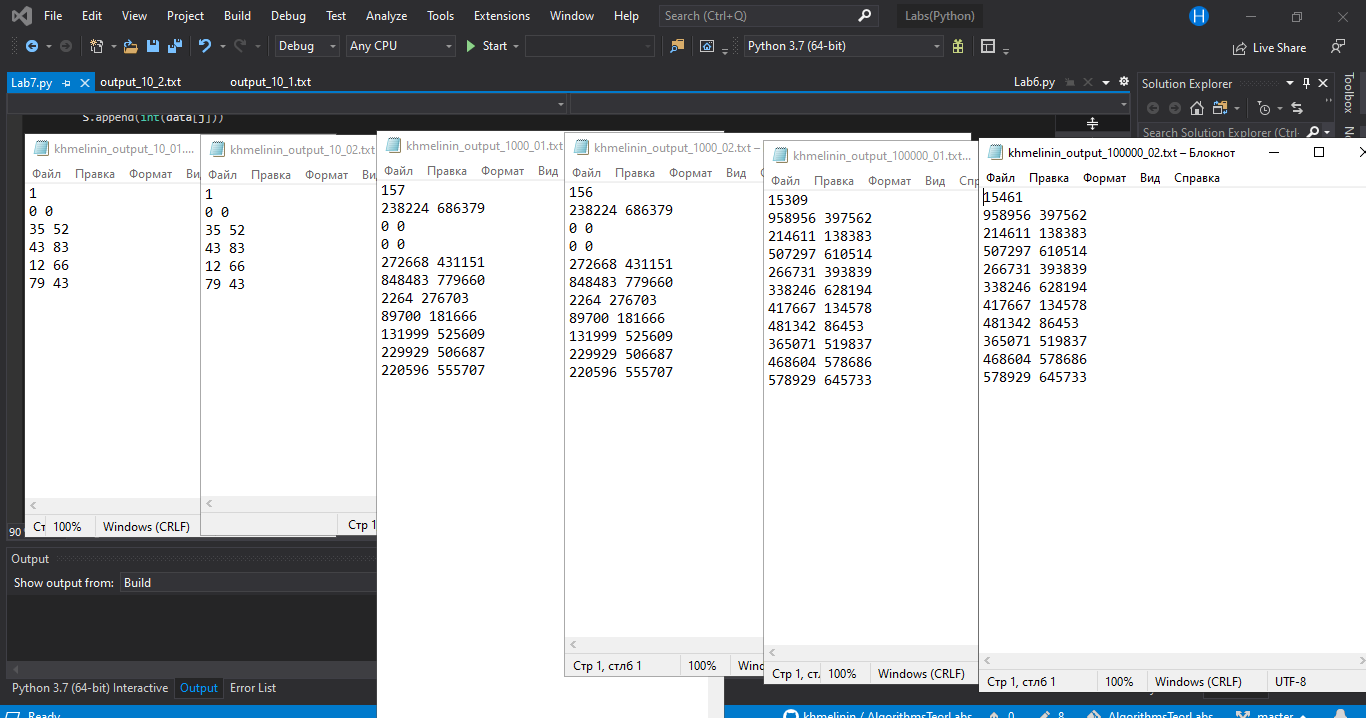
file.write(str(A[h]) + " " + str(HashTableM[Hash\_searchM(HashTableM, S[kk] - A[h])][u]) + "\n") # то записываем элемент A[h] и его ключ

break

break

elif h == len(A)-1:

file.write("0 0" + "\n")



Висновок:

Я реалізував різні типи хеш-таблиць із використанням хеш-функцій методу дiлення та множення для розв’язання наведеної вище задачі. На бiльших значеннях метод множення працює краще.