**Міністерство освіти і науки України**

**Національний технічний університет України «Київський політехнічний**

**інститут імені Ігоря Сікорського"**

**Факультет інформатики та обчислювальної техніки**

**Кафедра інформатики та програмної інженерії**

**Звіт**

з лабораторної роботи № 8 з дисципліни

«Алгоритми та структури даних 2. Структури даних»

**«Жадібні алгоритми»**

**Виконав** ІП-11Трикош Іван Володимирович

**Перевірила** Халус Олена Андріївна

Київ 2022

**Код програми**

from math import sqrt

def algorithm(A):

'''Жадібний алгоритм для вирішення задачі комівояжера'''

numberEdges = len(A) # Кількість вершин

numberedge = 0 # Вершина, яка розглядається

Edges = [numberedge] # Вже розглянуті вершини

Length = 0 # Довжина маршруту

for i in range(numberEdges): # Проходимо по всіх вершинах

e = float("inf") # Найкоротше ребро

temp = numberedge # Вершина, для якої знаходимо найкоротшу відстань до наступної вершини

for j in range(numberEdges): # Проходимо по вершинах

if temp != j and not j in Edges and A[temp][j] < e: # Якщо це не розглянута вершина і відстань між цією вершиною та тією, яка розглядається є меншою за попередню, то

e = A[temp][j] # Зберігаємо поточне ребро

numberedge = j # Зберігаємо вершину

if e != float("inf"): # Якщо найкоротше ребро != infinity, то

Length += e # Додаємо ребро до маршруту

Edges += [numberedge] # Додаємо вершину до вже розглянутих

Edges.append(Edges[0]) # Додаємо першу вершину в кінець для завершення побудови маршруту

Length += A[Edges[0]][numberedge] # Додаємо ребро між першою та останньою вершинами

return Length, Edges # Повертаємо довжину маршруту та порядок вершин

def readfile():

'''Функція для зчитування і запису вмісту файлу у список'''

filename = "input.txt"

edges = []

with open(filename) as file:

edges = file.readlines()

edges = edges[1:]

for i in range(len(edges)):

edges[i] = edges[i].split()

edges[i][0], edges[i][1] = int(edges[i][0]), int(edges[i][1])

return edges

def matr(edges):

'''Функція для перетворення списку вершин з координатами в матрицю ваг'''

graph = [[0 for i in range(len(edges))] for i in range(len(edges))]

for i in range(len(edges)):

for j in range(len(edges)):

if i != j:

graph[i][j] = sqrt((edges[i][0] - edges[j][0]) \*\* 2 + (edges[i][1] - edges[j][1]) \*\* 2)

return graph

def writefile(Length, Edges):

'''Функція для запису результатів у файл'''

filename = "output.txt"

with open(filename, "w") as file:

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

for i in Edges:

file.write(str(i) + " ")

return

def main():

edges = readfile()

matrix = matr(edges)

Length, Edges = algorithm(matrix)

writefile(Length, Edges)

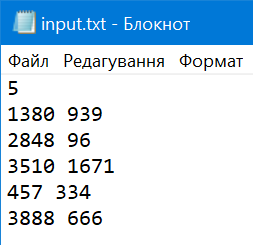
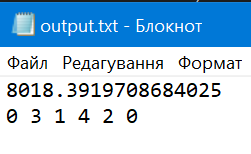
return

if \_\_name\_\_ == "\_\_main\_\_":

main()

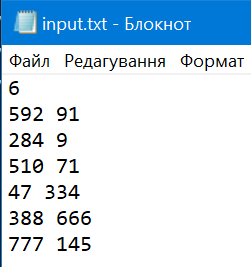
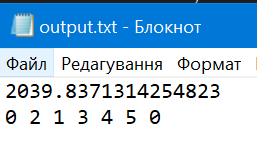
**Приклади роботи програми**

*Приклад 1.*

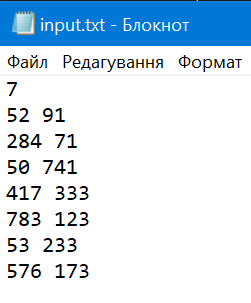
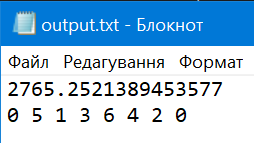
Найоптимальніший розв'язок: 6175.384376171725, тому алгоритм для цих даних працює коректно.

*Приклад 2.*

Найоптимальніший розв'язок: 1589.333510164241, тому алгоритм для цих даних працює коректно.

*Приклад 3.*

Найоптимальніший розв'язок: 2151.37781148064, тому алгоритм для цих даних працює коректно.