МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №3
по дисциплине «Построение и анализ алгоритмов»
Тема: Коммивояжер(TCP)

Студент гр. 1304 _____ Мамин Р.А.
Преподаватель _____ Шевелева А.М.

Санкт-Петербург
2023

Цель работы.

Изучение алгоритмов на графах, нахождение гамильтонова цикла в взвешенных графах и реализация метода ветвей и границ.

Задание.

Дана карта городов в виде ассиметричного, неполного графа G = (V, E), где V(|V|=n) – это вершины графа, соответствующие городам; E(|E|=m) – это ребра между вершинами графа, соответствующие путям сообщения между этими городами.

Каждому ребру m_{ij} (переезд из города i в город j) можно сопоставить критерий выгодности маршрута (вес ребра) равный w_i (натуральное число [1, 1000]), m_{ij} =inf, если i=j.

Если маршрут включает в себя ребро m_{ij} , то x_{ij} =1, иначе x_{ij} =0.

Требуется найти минимальный маршрут (минимальный гамильтонов цикл):

Выполнение работы.

В ходе выполнения лабораторной работы был применен метод ветвей и границ, который использовал следующий алгоритм решения поставленной задачи:

1) Открытие текстового файла со входными параметрами и его правильное считывание в двумерный массив *path.*

2) Засечение времени выполнения задачи с помощью модуля *time*

3) Инициализация необходимых переменных (*visited*, *min_len* и *min_path*), в которых будет храниться ответ.

4) Рекурсивный поиск кратчайшего гамильтонова цикла методом ветвей и границ

5) Во время поиска вычисление нижней границы стоимости гамильтонова цикла для оставшихся вершин.

6) Запись ответов в соответствующие переменные

7) Вывод ответа в консоль в соответствующем формате

Для реализации данного алгоритма были написаны следующие функции:

Функция *lower_bound* - Вычисляет и возвращает нижнюю границу стоимости гамильтонова цикла для оставшихся вершин, используя жадный подход: для каждой непосещенной вершины, она выбирает минимальное ребро.

Параметры:

- *graph* - матрица смежности графа
- *v* - текущая вершина
- *visited* - список посещённых вершин

Функция *recursive_search* - Рекурсивно ищет кратчайший гамильтонов цикл, начиная с вершины v, обновляет минимальную длину и соответствующий путь при нахождении лучшего решения.

Параметры:

- *graph* - Матрица смежности, представляющая граф
- *v* - Текущая вершина, с которой начинается рекурсивный вызов.
- *visited* - Список булевых значений, отображающий посещенные вершины (True - посещена, False - не посещена).
- *curr_len* - Текущая длина пути.
- *min_len* - Список из одного элемента, содержащий минимальную длину гамильтонова цикла.
- *min_path* - Список из одного элемента, содержащий минимальный гамильтонов цикл.

Функция *read_graph_from_file* – Считывает матрицу смежности из текстового файла и заполняет двумерный массив *graph*, возвращая его после.

Параметры:

- *filename* – Имя файла

Исходный код предствлен в Приложение А Исходный код программы.

Тестирование.

Результаты тестирования представлены в табл. 1. Вложения Б.

Выводы.

Был реализован алгоритм нахождения кратчайшего гамильтонова пути в графе, а также реализован метод ветвей и границ, который для полного графа ранга 20 выполняет поиск минимального пути в среднем примерно за половину миллисекунды. Такой быстрой скорости алгоритма удалось получить благодаря использованию эффективного метода оценок, а также отсечению заведомо неверных вариантов. Стоит отметить, что при построении алгоритма был применен метод математического модерлирования, когда задача перенеслась на граф и свелась к более легкой задачи — задачи нахождение гамильтонова цкила во взвешенном ассиметричном графе.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.py

```
import sys
from time import perf_counter


def lower_bound(graph, v, visited):
    """
    Вычисляет нижнюю границу стоимости
гамильтонова цикла для оставшихся вершин
    :param graph: матрица смежности графа
    :param v: текущая вершина
    :param visited: список посещённых вершин
    :return: нижняя граница стоимости
гамильтонова цикла для оставшихся вершин
    """
    bound = 0
    for i in range(len(graph)):
        if not visited[i]:
            min_edge = sys.maxsize
            for j in range(len(graph)):
                if i != j and graph[i][j] <
min_edge:
                    min_edge = graph[i][j]
            bound += min_edge
    return bound


def recursive_search(graph, v, visited,
curr_len, min_len, min_path):
    """
    Рекурсивно ищет кратчайший гамильтонов цикл,
начиная с вершины v,
    и обновляет минимальную длину и
соответствующий путь при
    нахождении лучшего решения.
    :param graph: Матрица смежности,
представляющая граф
    :param v: Текущая вершина, с которой
начинается рекурсивный вызов.
    :param visited: Список булевых значений,
отображающий посещенные вершины (True -
посещена, False - не посещена).
    :param curr_len: Текущая длина пути.
    :param min_len: Список из одного элемента,
содержащий минимальную длину гамильтонова цикла.
    :param min_path: Список из одного элемента,
содержащий минимальный гамильтонов цикл.
    :return: None
    """
    visited[v] = True

    all_visited = all(visited)
```

```
    if all_visited:
        if graph[v][0] != sys.maxsize:
            curr_len += graph[v][0]
            if curr_len < min_len[0]:
                min_len[0] = curr_len
                min_path[0] = [i for i, visited
in enumerate(visited) if
                               visited] + [0]
        visited[v] = False
        return

    for neighbor in range(len(graph)):
        if neighbor == v:
            continue
        if not visited[neighbor] and
graph[v][neighbor] != sys.maxsize:
            bound = lower_bound(graph, neighbor,
visited) + curr_len + \
                    graph[v][neighbor]
            if bound < min_len[0]:
                recursive_search(graph,
neighbor, visited, curr_len +
graph[v][neighbor],
                    min_len, min_path)
    visited[v] = False


def shortest_hamiltonian_cycle(graph):
    """
    Ищет кратчайший гамильтонов цикл для
заданного графа, представленного матрицей
смежности.
    :param graph: Матрица смежности
    :return: Минимальная длина пути и сам путь
    """
    visited = [False] * len(graph)
    min_len = [sys.maxsize]
    min_path = [None]
    recursive_search(graph, 0, visited, 0,
min_len, min_path)
    return min_len[0], min_path[0]


def read_graph_from_file(filename):
    """
    Считывает матрицу смежности графа из
текстового файла и возвращает ее в виде списка
списков.
    :param filename: Имя файла
    :return: Минимальная длина пути и сам путь
    """
    file = open(filename, 'r')
    size = int(file.readline())
    graph = []
    for line in file.readlines():
        row = [int(x) if x != "inf" and x != '-'
else sys.maxsize for x in
               line.strip().split()]
```

```
            graph.append(row)
        return graph


filename = "input.txt"
graph = read_graph_from_file(filename)

start_time = perf_counter()
length, path = shortest_hamiltonian_cycle(graph)
elapsed_time = (perf_counter() - start_time) *
1000  # Время в миллисекундах
if path is not None:
    ans_path = list(map(lambda x: x + 1, path))
    print(f"{ans_path}, {length},
{elapsed_time:.5f}mc")
else:
    print("Нет существует гамильтонова цикла")
```

ПРИЛОЖЕНИЕ Б

ТЕСТИРОВАНИЕ ПРОГРАММЫ

Таблица 1 – Результаты тестирования

№ п	Входные данные	Выходные данные	Комментарий
1	inf 1 - 1 1 - 1 inf 1 - 1 - - - inf 1 1 - - - 1 inf 1 - - - - 1 inf 1 - - 1 - 1 inf	Не сущ. гамильтонова цикла	Программа работает успешно
2	inf 1 - 1 1 - - inf - - 1 - - 1 inf 1 1 - 1 - 1 inf 1 - - 1 - - inf 1 - 1 - - 1 inf	Не сущ. гамильтонова цикла	Программа работает успешно
3	inf 1 - 1 1 - 1 inf 1 - - - - 1 inf 1 1 - 1 - 1 inf 1 - - - - - inf 1 - 1 1 1 1 inf	[4, 1, 2, 3, 5, 6, 4] 6 1.0061264038085938mc	Программа работает успешно
4	inf 1 1 1 1 1 1 inf 1 1 1 1 1 1 inf 1 1 1 1 1 1 inf 1 1 1 1 1 1 inf 1 1 1 1 1 1 inf	[4, 1, 2, 3, 5, 6, 4] 6 0.0mc	Программа работает успешно
5	inf 2 2 2 2 2 2 inf 2 2 2 2 2 2 inf 2 2 2 1 2 2 inf 2 2 2 2 2 2 inf 2 2 2 2 2 2 inf	[4, 1, 2, 3, 5, 6, 4] 11 0.0mc	Программа работает успешно