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

ОТЧЕТ

по лабораторной работе №2
по дисциплине «Построение и анализ алгоритмов»
Тема: Кратчайшие пути в графах: коммивояжёр
Вариант: 1

Студентка гр. 3388	____________________	Беннер В.А.
Преподаватель	____________________	Жангиров Т.Р.

Санкт-Петербург
2025

Цель работы:

Реализовать алгоритм Литтла и АДО МОД для решения задачи коммивояжёра методом ветвей и границ и нахождения 2-приближения.

Задание:

Ветви и границы.

В волшебной стране Алгоритмии великий маг, Гамильтон, задумал невероятное путешествие, чтобы связать все города страны заклятием процветания. Для этого ему необходимо посетить каждый город ровно один раз, создавая тропу благополучия, и вернуться обратно в столицу, используя минимум своих чародейских сил. Вашей задачей является помощь в прокладывании маршрута с помощью древнего и могущественного алгоритма ветвей и границ.

Карта дорог Алгоритмии перед Гамильтоном представляет собой полный граф, где каждый город соединён магическими порталами с каждым другим. Стоимость использования портала из города в город занимает определённое количество маны, и Гамильтон стремится минимизировать общее потребление магической энергии для закрепления проклятия.

Входные данные:

Первая строка содержит одно целое число N — количество городов). Города нумеруются последовательными числами от 0 до N−1. Следующие N строк содержат по N чисел каждая, разделённых пробелами, формируя таким образом матрицу стоимостей M.Каждый элемент M i,j этой матрицы представляет собой затраты маны на перемещение из города i в город j.

Выходные данные:

Первая строка: Список из N целых чисел, разделённых пробелами, обозначающих оптимальный порядок городов в магическом маршруте Гамильтона. В начале идёт город 0, с которого начинается маршрут, затем последующие города до тех пор, пока все они не будут посещены.

Вторая строка: Число, указывающее на суммарное количество израсходованной маны для завершения пути.

2-приближение.

Разработайте программу, которая решает задачу коммивояжера при помощи 2-приближенного алгоритма. В данной постановке задачи нужно вернуться в исходную вершину после прохождения всех остальных вершин. При обходе остовного дерева (MST) необходимо идти по минимальному допустимому ребру из текущего. Каждая вершина в графе обозначается неотрицательным числом, начиная с 0, каждое ребро имеет неотрицательный вес. В графе нет рёбер из вершины в саму себя, в матрице весов на месте таких отсутствующих рёбер стоит значение -1.

Пример входных данных

2

-1 18.97 22.36 19.42 3.61

18.97 -1 35.61 38.01 17.0

22.36 35.61 -1 16.28 21.19

19.42 38.01 16.28 -1 21.02

3.61 17.0 21.19 21.02 -1

В первой строке указывается начальная вершина. Далее идёт матрица весов. В качестве выходных данных необходимо представить длину пути, полученного при помощи алгоритма. Следующей строкой необходимо представить путь, в котором перечислены вершины, по которым необходимо пройти от начальной вершины. Для приведённых в примере входных данных ответом будет:

91.92

2 3 0 4 1 2

Реализация

Ветви и границы:

Алгоритм Литтла представляет собой точный метод решения задачи коммивояжёра (TSP) с использованием стратегии ветвей и границ. Он гарантирует нахождение оптимального маршрута, проходящего через все города ровно один раз с минимальной общей стоимостью.

Основные этапы алгоритма

1. Инициализация

- Проверка базовых случаев (1 город, несвязный граф)
- Нормализация матрицы стоимостей (замена -1 на ∞)

2. Редукция матрицы

- Построчно: вычитание минимального элемента из каждой строки
- Поколоночно: вычитание минимального элемента из каждого столбца
- Вычисление нижней границы стоимости (сумма вычтенных минимумов)

3. Ветвление

- Выбор города с максимальным штрафом за невключение
- Разделение на подзадачи:
  - С включением конкретного ребра
  - Без включения этого ребра

4. Ограничение (отсечение)

- Проверка нижних границ для каждой подзадачи
- Отбрасывание подзадач с границей хуже текущего решения

5. Рекурсивное исследование

- Повторение процесса для перспективных подзадач
- Обновление лучшего найденного решения

6. Завершение

- Возврат оптимального маршрута и его стоимости

Временная сложность:

- Редукция матрицы $O(N^2)$ - Двойной проход по матрице
- Выбор ребра ветвления $O(N^2)$ - Вычисление штрафов для всех рёбер
- Рекурсивные вызовы $O(N!)$ - В худшем случае полный перебор
- С учётом отсечений $O(N^2 2^N)$ - На практике с оптимизациями

<u>Итог:</u>

- Худший случай $O(N!)$ - Полный перебор всех возможных перестановок
- Средний случай (с оптимизациями) $O(N^2 \cdot 2^N)$ - Эффективное ветвление с отсечениями
- Лучший случай $O(N^3)$ - Когда оптимальный маршрут находится быстро

Сложность относительно памяти:

- Матрица стоимостей $O(N^2)$ - Основная структура данных
- Текущий путь $O(N)$ - Хранение последовательности городов
- Стек вызовов $O(N)$ - Глубина рекурсии

<u>Описание приближённого алгоритма (алгоритм решения задачи коммивояжёра с использованием 2-приближенного метода на основе минимального остовного дерева (MST)):</u>

Общее описание:

Данный алгоритм решает задачу коммивояжёра (TSP) с использованием 2-приближенного подхода, основанного на построении минимального остовного дерева графа. Алгоритм гарантирует, что стоимость найденного пути не более чем в 2 раза превышает стоимость оптимального решения для метрического случая задачи.

Шаги алгоритма:

1. Инициализация и подготовка данных:

- Матрица стоимостей преобразуется: значения -1 (отсутствие ребра) заменяются на math.inf
- Граф представляется в виде матрицы смежности с весами рёбер

2. Построение минимального остовного дерева (алгоритм Прима):

- Используется приоритетная очередь (min-heap) для выбора рёбер с минимальным весом
- Начинается построение из заданной стартовой вершин
- На каждом шаге добавляется ребро минимального веса, соединяющее уже посещённые вершины с непосещёнными
- Результат - список смежности, представляющий MST

3. Обход MST с поиском в глубину (DFS):

- Обход начинается со стартовой вершины
- На каждом шаге выбирается непосещённая вершина, соединённая ребром с минимальным весом
- Вершины добавляются в путь в порядке их посещения
- В конце пути добавляется стартовая вершина для замыкания цикла

4. Вычисление стоимости пути:

- Суммируются веса всех рёбер в полученном пути
- Учитывается вес ребра возврата в стартовую вершину

5. Форматирование вывода:

- Стоимость пути выводится с двумя знаками после запятой
- Путь выводится как последовательность вершин, включая возврат в начало

Особенности

- Использует жадную стратегию при построении MST
- Обеспечивает полиномиальную сложность $O(n^2 \log n)$
- Гарантирует 2-приближение для метрического TSP
- Всегда возвращает допустимый гамильтонов цикл
- Корректно обрабатывает отсутствующие рёбра (значения -1 в матрице)

Оценка сложности алгоритма:

Пошаговый анализ сложности:

1. Построение матрицы смежности:
    - Замена -1 на ∞: $O(n^2)$
    - Создание копии матрицы: $O(n^2)$
2. Алгоритм Прима для построения MST:
    - Инициализация структур данных: $O(n)$
    - Основной цикл (выполняется для всех n вершин)
        - Извлечение из кучи: $O(\log n)$ для каждой из n операций → $O(n \log n)$
        - Проверка всех рёбер для каждой вершины: $O(n)$ для каждой из n вершин → $O(n^2)$
    - Общая сложность: $O(n^2)$
3. Обход DFS MST:
    - Посещение всех вершин: $O(n)$
    - Сортировка соседей для каждой вершины:
        - В худшем случае $O(d \log d)$, где d - степень вершины
        - Для полного графа $d = n-1$ → $O(n \log n)$ для каждой из n вершин → $O(n^2 \log n)$
    - Общая сложность: $O(n^2 \log n)$
4. Вычисление стоимости пути:
    - Проход по всем рёбрам пути: $O(n)$
5. Форматирование вывода:
    - Преобразование пути в строку: $O(n)$

Итоговая сложность:

- Доминирующая операция - обход DFS с сортировкой соседей: $O(n^2 \log n)$
- Общая временная сложность: $O(n^2 \log n)$

Пространственная сложность:

- Хранение матрицы смежности: $O(n^2)$
- MST: $O(n)$
- Структуры данных для алгоритма Прима: $O(n)$

- Путь: O(n)
- **Общая пространственная сложность: O(n²)**

Оптимальность:

Алгоритм обеспечивает 2-приближение для метрической задачи коммивояжёра, то есть стоимость найденного пути не более чем в 2 раза превышает стоимость оптимального пути.

Тестирование

Таблица 1. Тестирование.

Входные данные	Выходные данные Литтла	MST-приближени
inf 37 27 8 inf 34 12 11 inf	Минимальный путь: [(0, 2), (1, 0), (2, 1)] Длина пути: 46.0	Маршрут: [0,2,1,0] Длина: 46.0
inf 3 4 1 1 inf 3 4 9 2 inf 4 8 9 2 inf	Маршрут: [0,3,2,1,0] Длина: 6.0	Маршрут: [0,1,2,3,0] Длина: 10.0
inf 27 16 12 29 28 inf 23 36 10 12 18 inf 38 22 49 23 18 inf 10 16 1 26 36 inf	Маршрут: [0,3,4,1,2,0] Длина: 58.0	Маршрут: [0,3,4,1,2,0] Длина: 62.0
inf 12 29 22 13 24 12 inf 19 3 25 6 29 19 inf 21 23 28 22 3 21 inf 4 5 13 25 23 4 inf 16 24 6 28 5 16 inf	Маршрут: [0,4,3,1,5,2,0] Длина: 76.0	Маршрут: [0,1,3,4,2,5,0] Длина: 85.0
inf 10 15 20 10 inf 35 25 15 35 inf 30 20 25 30 inf	Маршрут: [0,1,3,2,0] Длина: 80.0	Маршрут: [0,1,3,2,0] Длина: 80.0
inf 14 6 31 32 14 inf 49 30 44 6 49 inf 43 38 31 30 43 inf 12	Маршрут: [0,2,1,3,4,0] Длина: 87.0	Маршрут: [0,2,3,4,1,0] Длина: 103.0

32 44 38 12 inf		

Вывод

В ходе лабораторной работы была написана программа с использованием алгоритма Литтла и алгоритма решения задачи коммивояжёра с использованием 2-приближенного метода на основе минимального остовного дерева (MST). На основании тестирования, можно сказать, что первый алгоритм более точный, нежели жадный алгоритм.