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: 4

|  |  |  |
| --- | --- | --- |
| Студент гр. 3388 |  | Трунов Б.Г. |
| Преподаватель |  | Жангиров Т.Р. |

Санкт-Петербург

2025

**Цель работы:**

Изучить принципы работы алгоритмов на графах. Решить с помощью них задачу Коммивояжёра.

**Задание:**

Решить задачу Коммивояжёра 2 различными способами. МВиГ: последовательный рост пути + использование для отсечения двух нижних

оценок веса оставшегося пути: 1) полусуммы весов двух легчайших рёбер по всем кускам; 2) веса МОД. Эвристика выбора дуги — не в глубину, а по антиприоритету *(S/k+L/N)(4N/(3N+k))*. Приближённый алгоритм: АВБГ "улучшенный". Замечание к варианту 4 И МВиГ, и АВБГ "улучшенный" начинать со стартовой вершины.

**Реализация**

### 1. Метод ветвей и границ (МВиГ)

#### Основная идея:

Метод ветвей и границ - это точный алгоритм, который систематически перебирает возможные решения, отсекая заведомо неперспективные ветви с помощью оценки нижней границы стоимости.

#### Шаги алгоритма:

1. **Инициализация**:
   * Начинаем с начальной вершины (по умолчанию 0)
   * Инициализируем лучший путь и его стоимость бесконечностью
   * Создаём очередь приоритетов для хранения частичных путей
2. **Основной цикл**:
   * Извлекаем частичный путь из очереди
   * Если путь содержит все вершины, проверяем возможность замкнуть цикл
   * Вычисляем нижнюю границу для текущего частичного пути
   * Если граница хуже текущего лучшего решения, отсекаем ветвь
   * Для каждой непосещённой вершины:
     + Вычисляем антиприоритет (эвристика для порядка рассмотрения вершин)
     + Добавляем новый частичный путь в очередь
3. **Вычисление нижней границы**:
   * Используется комбинация двух эвристик:
     + Полусумма двух минимальных рёбер в подграфе непосещённых вершин
     + Вес минимального остовного дерева для подграфа непосещённых вершин
   * Выбирается максимальное из этих двух значений
4. **Эвристика антиприоритета**:
   * Формула: *(S/k + L/N) \* (4\*N/(3\*N+k))*
   * Где:
     + *S* - сумма длин рёбер в текущем пути
     + *k* - длина текущего пути
     + *L* - длина ребра к следующей вершине
     + *N* - общее количество вершин

#### Особенности:

* Гарантирует нахождение оптимального решения
* Использует эвристики для ускорения работы

### 2. Улучшенный алгоритм ближайшего соседа с антиприоритетом

#### Основная идея:

Жадный алгоритм, который на каждом шаге выбирает следующую вершину не просто по минимальному расстоянию, а с учётом более сложной эвристики (антиприоритета).

#### Шаги алгоритма:

1. **Инициализация**:
   * Начинаем с начальной вершины
   * Инициализируем пустой путь и нулевую стоимость
2. **Основной цикл**:
   * Пока есть непосещённые вершины:
     + Для текущей вершины вычисляем антиприоритет для всех соседних непосещённых вершин
     + Выбираем вершину с наименьшим антиприоритетом
     + Добавляем её в путь и увеличиваем общую стоимость
   * Пытаемся замкнуть цикл, вернувшись в начальную вершину
3. **Формула антиприоритета**:
   * Та же, что и в методе ветвей и границ
   * Учитывает как историю пути *(S/k)*, так и локальную информацию *(L/N)*
   * Вводит коэффициент, зависящий от длины пути (антиприоритет)

#### Особенности:

* Работает значительно быстрее метода ветвей и границ
* Не гарантирует нахождение глобально оптимального решения
* Даёт хорошие результаты на практике благодаря взвешенной эвристике выбора

.

**Описание функций и структур:**

* *generate\_matrix(N, min\_weight, max\_weight)* → генерирует случайную матрицу весов *N×N*.
* *save\_matrix(matrix, filename)* → сохраняет матрицу в файл (∞ заменяется на -1).
* *read\_matrix\_from\_file(filename)* → загружает матрицу из файла (-1 → ∞).
* *read\_input()* → считывает матрицу с клавиатуры.
* *calculate\_mst\_weight(vertices, matrix)* → вычисляет вес минимального остовного дерева (алгоритм Прима).
* *get\_two\_min\_edges(vertices, matrix)* → находит полусумму двух минимальных рёбер.
* *calculate\_antipriority(path, next\_vertex, M, N)* → эвристика для выбора следующей вершины.
* *mvag(N, M, start)* → метод ветвей и границ.
* *improved\_avnn(N, M, start)* → улучшенный жадный алгоритм. Выбирает вершины не просто по минимальному весу, а с учётом истории пути.

**Оценка сложности алгоритмов:**

1. Метод ветвей и границ (МВиГ)

Временная сложность: *O(N!)*

* В худшем случае потребуется перебрать все возможные перестановки вершин *O(N!)*
* Вычисление нижней границы: *O(N²)* для *MST*
* Сортировка вершин по антиприоритету: *O(N log N)*
* Вычисление антиприоритета: *O(N)*

Пространственная сложность: *O(N²)*

* Хранение матрицы смежности: *O(N²)*
* Хранение списка путей: *O(N)*
* Хранение множества непосещенных вершин: *O(N)*

2. Улучшенный алгоритм ближайшего соседа (АВБС)

Временная сложность: *O(N²)*

* Основной цикл: *O(N)* итераций
* На каждой итерации:
  + Перебор непосещенных вершин: *O(N)*
  + Вычисление приоритета: *O(N)*
  + Выбор лучшей вершины: *O(N)*

Пространственная сложность: *O(N²)*

* Хранение матрицы смежности: *O(N²)*
* Хранение пути: *O(N)*
* Хранение множества непосещенных вершин: *O(N)*
* Хранение приоритетов: *O(N)*

**Тестирование**

Таблица 1. Тестирование.

|  |  |  |
| --- | --- | --- |
| Входные данные | Выходные данные МВиГ | Выходные данные АВБС |
| 3  -1 40.67 37.83  2.11 -1 58.32  92.73 32.47 -1 | Путь: 0 2 1  Стоимость: 72.41  Время выполнения: 0.001 секунд | Путь: 0 2 1  Стоимость: 72.41  Время выполнения: 0.000 секунд |
| 5  -1 51.23 7.19 37.41 50.69  83.97 -1 77.8 21.55 13.81  39.72 47.6 -1 72.6 3.31  66.87 98.72 94.68 -1 72.91  13.29 20.23 15.37 71.47 -1 | Путь: 0 2 4 1 3  Стоимость: 119.15  Время выполнения: 0.005 секунд | Путь: 0 2 4 1 3  Стоимость: 119.15  Время выполнения: 0.001 секунд |
| 8  -1 83.17 79.2 39.64 15.95 88.48 68.01 95.32  40.33 -1 42.69 30.34 77.94 16.36 12.96 7.93  39.89 44.08 -1 42.96 99.87 87.02 27.33 64.81  90.37 65.04 79.95 -1 12.01 22.94 82.39 98.12  65.65 14.37 32.62 35.96 -1 1.32 35.89 20.29  72.78 64.62 43.56 86.67 97.33 -1 33.07 33.04  13.97 46.26 55.36 48.17 55.38 74.68 -1 81.07  26.52 39.66 88.28 89.23 40.34 57.05 10.38 -1 | Путь: 0 3 4 5 2 1 7 6  Стоимость: 172.89  Время выполнения: 1.010 секунд | Путь: 0 4 5 7 6 1 3 2  Стоимость: 257.13  Время выполнения: 0.002 секунд |

**Вывод**

В ходе лабораторной работы была написана программа с использованием метода ветвей и границ и улучшенного алгоритма ближайшего соседа. На основании тестирования, можно сказать, что первый алгоритм более точный, нежели жадный алгоритм, но второй на порядки быстрее.