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: Кратчайшие пути в графе: коммивояжёр

Студент гр. 2384 _____ Лавренова Ю.Д.

Преподаватель _____ Шестопалов Р.П.

Санкт-Петербург

2024

Цель работы.

Реализовать методы ветвей и границ и приближенный для решения задачи коммивояжёра.

Задание.

Решить ЗК двумя методами в соответствии с вариантом:

1) Методом ВиГ.

2) Приближённым методом.

Дано: матрица весов графа, все веса неотрицательны; стартовая вершина.

Найти: путь коммивояжёра (последовательность вершин) и его стоимость.

Вариант 3:

МВиГ: последовательный рост пути + использование для отсечения двух нижних оценок ве́са оставшегося пути: 1) полусуммы весов двух легчайших рёбер по всем кускам; 2) веса МОД. Эвристика выбора дуги — поиск в глубину с учётом веса добавляемой дуги и нижней оценки веса остатка пути. Приближённый алгоритм: АМР.

Замечание к варианту 3. Начинать МВиГ со стартовой вершины

Выполнение работы.

В ходе решения задач реализованы функции для решения задачи коммивояжёра двумя методами.

void depth_search(std::vector<std::vector<int>> matrix, std::vector<std::vector<int>> pieces, int weigth, int &total_weight, std::vector<int> &result) — функция, реализующая рекурсивный алгоритм поиска в глубину в качестве эвристики выбора дуги в методе ветвей и границ. В начале формируются «куски» текущего решения и возможные пути из конечной вершины, далее используется функция find_next, которая находит возможные решения на данном этапе методом веса МОД и полусуммой легчайших ребер для определения границы. Далее функция проходится по всем следующим точкам, создаются новые «куски», увеличивается вес текущего решения. Далее, если путь закончился, то идет проверка на минимальность веса, обновляется результат и вес. Иначе вызывается поиск в глубину еще раз. Если вес не меньше, то ветка решения обрывается.

int min_weight(std::vector<std::vector<int>> pieces, std::vector<std::vector<int>> matrix, std::vector<std::vector<int>> may_pass) — функция определяет минимальный вес входящих и исходящих ребер в текущее решение, по все возможным путям решения. И выводит полусумму всех ребер.

int weight_MOD(std::vector<std::vector<int>> matrix, std::vector<std::vector<int>> pieces) — функция использует алгоритм Краскала для поиска минимального оставного дерева. В алгоритме Краскала весь единый список ребер упорядочивается по неубыванию весов ребра. Далее ребра перебираются от ребер с меньшим весом к большему, и очередное ребро добавляется к каркасу, если оно не образовывает цикла с ранее выбранными ребрами. В частности, первым всегда выбирается одно из ребер минимального веса в графе. По мере добавления новых ребер компоненты связности будут объединяться, пока не получится одна общая компонента связности. Пронумеровываются все компоненты связности, и для каждой вершины хранится номер ее компоненты связности, таким образом, в самом начале для

каждой вершины номер ее компоненты связности равен номеру самой вершины, а в конце у всех вершин будут одинаковые номера компоненты связности, которой они принадлежал. Так же прибавляются веса новых ребер — вес МОД увеличивается.

void AMP(std::vector<std::vector<int>> matrix, std::vector<int>& result_path, std::vector<std::vector<int>> pieces, int& total) — функция приближенного алгоритма модификации решения. Функция получаем решение: 1-2-3-...-N. Запускается цикл модификации решения. Под единичной модификацией подразумевается взятие некоторого города и перемещение его в другое место в цепочке. Запускается цикл пока модификации возможны и идет перебор решений. Если вес модифицированного решения получился меньше рекорда, то сохраняется новый текущий путь. Для поиска лучшего модифицированного решения используются методы отсечения границ как для МВиГ.

Пример работы программы:

Matrix size 6

IN 4 1 6 2 7

2 IN 4 10 1 8

2 9 IN 4 3 6

1 8 1 IN 7 4

5 2 7 4 IN 8

9 4 3 9 10 IN

Start 2

Результат:

Kommivoyazhor path1: 2 1 3 5 4 6 2

cost1 18

Kommivoyazhor path2: 1 2 6 3 5 4 1

cost2 23

Оценка алгоритмов:

Метод ветвей и границ: C^n, где C ~ 1,26, n — размер матрицы

Приближённый алгоритм: экспоненциальная.

Алгоритм Краскала: O(Mlog(M) + n*n)

Выводы.

Был изучен метод ветвей и границ для поиска лучшего решения задачи коммивояжера. Использована эвристика поиска в глубину. Реализован приближенный метод модификации решений.