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

ОТЧЕТ

по лабораторной работе №2 (Вариант 1)
по дисциплине «Построение и анализ алгоритмов»
Тема: Задача коммивояжёра.

Студент гр. 3388 _____ Лутфулин Д.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

Выполнение работы.

Метод ветвей и границ:

Для решения задачи коммивояжёра был реализован алгоритм Литтла. Данный алгоритм находит кратчайший гамильтонов цикл в полном графе с помощью метода ветвей и границ, выполняя бинарное ветвление на каждом шаге:

левая ветвь включает в себя данное ребро, а правая – исключает. Перед ветвлением происходит редукция матрицы – из каждой строки вычитается минимальный элемент, затем то же самое происходит и со столбцами. Всё, что было вычтено добавляется к graph.d – нижней границе решения. Далее среди нулей (а они есть в каждой строке и в каждом столбце после редукции) эвристически выбирается наилучший: ветвление происходит по тому ребру, которое будет увеличивать d как можно сильнее, если это ребро исключить.

Если нижняя граница d текущего решения превышает стоимость наилучшего найденного решения, то ветвления на данном этапе не происходит.

2-приближение:

Для нахождения 2-приближения используется алгоритм двойного обхода минимального остовного дерева.

Для начала находим МОД графа с помощью алгоритма Прима.

В алгоритме Прима сначала все вершины кроме стартовой отмечаются как непосещённые. В мин-кучу записываются все рёбра, инцидентные ей. Далее в цикле из мин-кучи достаются минимальное рёбро и, если вершина, в которую оно ведёт, ещё не посещена, то она отмечается посещённой и ребро добавляется в МОД. Так продолжается, пока не посетим все вершины.

Далее выполняется жадный обход МОД: в путь добавляется непосещённая вершина, в которую ведёт самое короткое ребро из текущей. Если в МОД нет таких рёбер, то берётся ребро из предыдущей.

Оценка сложности.

Метод ветвей и границ:

В худшем случае алгоритму придётся перебрать все возможные случаи, которых O(n!) и каждый этап решения просчитывается за $O(n^2)$. Однако, первое решение находится уже за $O(n^3)$: так как нужно обойти n вершин и в каждой происходит редукция за $O(n^2)$, построение включающей ветви за $O(n^2)$ и

исключающей за $O(n^2)$. А так как ребро для каждого ветвления подбирается так, чтобы нижняя граница решения при исключающем ребро решении стала как можно больше, вероятность того, что исключающую ветвь не нужно будет просматривать становится выше.

2-приближение:

Сложность алгоритма Прима: $O(n^2\log(n))$ (n раз нужно пройтись по n рёбрам из выбранной вершины и добавить в кучу за $O(\log(n))$)

Сложность обхода МОД: Каждая вершина посещается за O(n) и выполняется сортировка по соседним рёбрам.
Соседние рёбра – подмассив всех рёбер. Но в МОД всего (n-1)*2 рёбер (граф двунаправленный). Сложность сортировки такого массива была бы n*log(n) если бы граф оказался звездой. Но сложность сортировок подмассивов меньше, чем сложность сортировки самого массива, поэтому все сортировки в алгоритме выполняются не более чем за O(n*log(n)). Итоговая сложность: O(n) + O(n*log(n)) = O(n*log(n))

Таким образом, сложность алгоритма равна $O(n^2\log(n))$ + O(n*log(n)) = $O(n^2\log(n))$.

Вывод.

Были разработаны и проанализированы алгоритмы для решения задачи коммивояжёра. В качестве точного решения – алгоритм Литтла, в качестве 2-приближения - АДО МОД.