Министерство образования и науки Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»

Кафедра математической
кибернетики и компьютерных наук

РЕШЕНИЕ ЗАДАЧИ КИТАЙСКОГО ПОЧТАЛЬОНА ДЛЯ СЛУЧАЯ НЕОРИЕНТИРОВАННОГО И ОРИЕНТИРОВАННОГО ГРАФОВ

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

Студентки 4 курса 411 группы
направления 02.03.02 — Фундаментальная информатика и информационные технологии
факультета КНиИТ
Борушевской Жанны Владимировны

Научный руководитель
доцент, к. ф.-м. н. _____ А. С. Иванова

Заведующий кафедрой
к.ф.-м.н. _____ С. В. Миронов

Саратов 2016

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Задача о нахождении кратчайшего пути для коммивояжера, который должен посетить несколько городов, давно известна. Гораздо менее известной является задача китайского почтальона, который должен обойти каждую улицу в городе и разнести письма.

Существует много вариантов постановки условия для этой задачи, но особый интерес представляет случай, когда все дороги ориентированы, и почтальон должен вернуться в точку отправления.

Целью данной выпускной квалификационной работы является описание и реализация различных алгоритмов решения задачи китайского почтальона для взвешенного ориентированного и неориентированного графа, а также последующее сравнение эффективности и способы оптимизации этих алгоритмов на графах с различными характеристиками.

Данная работа состоит из введения, трех глав, заключения, списка использованных источников, четырех приложений. Содержит 7 рисунков, 2 таблицы, 10 использованных источников. В введении кратко сформулирована предметная область и цели исследования.

Первая глава включает в себя описание основных понятий теории графов [1–10], описывается критерий существования эйлерового цикла. Данная глава содержит теорему существования эйлерового цикла и доказательство достаточности и необходимости, также приведены следствия данной теоремы. Описывается неформальная задача китайского почтальона и данный алгоритм в терминах теории графов. Во второй главе подробно рассматривается алгоритм Куна и Венгерский алгоритм.

В третьей главе описывается практическая реализация алгоритма. Описан алгоритм нахождения эйлерового цикла. Подробно рассмотрены фрагменты кода. Рассмотрена работа алгоритма на примере тестовых дорог. Приведено сравнение эффективности работы в таблицах и на диаграмме для графов, содержащих различное количество вершин и ребер.

В заключении приведены основные выводы, в приложениях содержится полный код разработанных программ.

1 Задача китайского почтальона

1.1 Неформальная постановка задачи

В классической формулировке задачи главным действующим лицом является почтальон, которому необходимо забрать корреспонденцию в почтовом отделении, разнести почту по всем улицам города, и затем вновь вернуться в почтовое отделение. Задача состоит в том, чтобы отыскать наиболее оптимальный путь для почтальона. Сформулированная выше задача имеет много потенциальных приложений. Например, оптимальный обход улиц снегоуборочной техникой, а также решение других проблемы, связанных с маршрутизацией.

Стоит заметить, что только в случае ориентированного и неориентированного графа задача китайского почтальона имеет полиномиальное решение. В случае смешанного графа задача является NP-полной.

1.2 Задача, сформулированная в терминах теории графов

Сформулируем данную задачу в терминах теории графов. Из неформальной постановки задачи следует, что существует сильно связный ориентированный граф. Предполагается, что в нем не существует параллельных ребер и ребер отрицательного веса.

По условию, ребрам ориентированного графа сопоставлены положительные веса. Требуется найти цикл минимального веса, проходящий через каждое ребро графа , по крайней мере, один раз. Очевидно, что если содержит эйлеров цикл, то любой такой цикл будет оптимальным, так как каждое ребро проходится по один раз и вес этого цикла равен , где — вес j-ого ребра. В этом случае достаточно найти эйлеров цикл, используя алгоритм эйлерого цикла, который будет описан далее. [6] В случае неэйлерова графа в первую очередь нужно решить следующую подзадачу. В неэйлеровом графе необходимо найти минимальное дополнение графа до эйлерова, где минимальное дополнение графа до эйлерова — это рёбра и числа k повторений каждого ребра, такие, что при дублировании каждого ребра из этого подмножества k раз будет получен граф, в котором все вершины будут сбалансированы, т. е. эйлеров граф. После того, как будет найдено минимально дополнение графа до эйлерова, следует применить алгоритм поиска эйлерова цикла. Таким образом, будет получено решение задачи китайского почтальона.

2 Реализация задачи китайского почтальона

2.1 Алгоритм поиска эйлерова цикла

Для решения поставленной задачи используется следующий алгоритм поиска эйлерова цикла в эйлеровом графе. Перед выполнением алгоритма стоит выполнить проверку на эйлеровость следующим образом: пройти по всем вершинам графа и проверить равенство входящих и исходящих степеней каждой вершины.

Вход: Ориентированный/неориентированный взвешенный эйлеров граф.

Выход: Цикл, проходящий через каждое ребро графа.

— Выбрать произвольную вершину u. Добавить вершину u в список результата.
— Выбрать произвольное ребро (u, v).
— Отметить ребро (u, v), как пройденное.
— Добавить вершину v в список результата.
— Выбрать произвольное ребро (v, w).
— Повторять пункты 2 - 5 до тех пор, пока все ребра не будут помечены как пройденные.

2.2 Описание разработанного приложения. Файл ChinesePostman.cpp

В `ChinesePostman.cpp` прописываются основные условия вывода кода и происходят основные вычисления. Описываются функции `optimalCost()` и `printEuler()`, подробнее рассмотренные в разделе `ChinesePostman.h`

Происходит поиск нечетных вершин:

Листинг 1. фрагмент кода файла ChinesePostman.cpp отвечающий за поиск нечетных вершин

```
1 void ChinesePostman::findOddVertices() {
2   for (int i=0; i<nVertices; i++) {
3     if (degree[i] % 2 != 0) {
4       odd.push_back(i);
5     }
6   }
7 }
```

Находим очереди минимального приоритета.

Листинг 2. фрагмент кода файла ChinesePostman.cpp отвечающий за нахождение очереди минимального приоритета

```
1 bool ChinesePostman::compareVertexPair(VertexPair &lt, VertexPair &rt)
```

```
{
  return cost[ lt . i ][ lt . j ] > cost[ rt . i ][ rt . j ];
}
```

2.3 Описание разработанного приложения. Файл ChinesePostman.h

В заголовочном файле `ChinesePostman.h` присутствуют функции `optimalCost()` и `printEuler()`, описанные в `ChinesePostman.cpp`.

Функция `optimalCost()` отвечает за оптимальный подсчет стоимости решения задачи китайского почтальона.

Листинг 3. фрагмент кода файла ChinesePostman.cpp описывающий optimalCost()

```
double ChinesePostman::optimalCost()
{
  if (!solveRan) {
    cout << "You must run solve() before using this function" << endl;
    return -1;
  }

  // add costs of all edges
  double totalCost = 0;
  for(int i = 0; i < nVertices; i++)
    for(int j = i+1; j < nVertices; j++)
      totalCost += edges[i][j]*cost[i][j];

  return totalCost;
}
```

Функция `printEuler()` отвечает за вывод результата алгоритма.

Листинг 4. фрагмент кода файла ChinesePostman.cpp описывающий printEuler()

```
inline bool operator<(const VertexPair &lt, const VertexPair &rt)
double ChinesePostman::optimalCost()
void ChinesePostman::printEuler () {

  if (!solveRan) {
    cout << "You must run solve() before using this function" << endl;
    return;
  }

  vector<vector<int>> tempEdges;
  tempEdges = edges;
```

```
12
13      int level = 0;
14      int start = 0;
15
16      vector<vector<int>> tours;
17      bool edgesExist = false;
18
19      do {
20      edgesExist = false;
21      tours.push_back(generateTour( start , tempEdges));
22      for (int i = 0; i < nVertices; i++) {
23        for (int j = 0; j < nVertices; j++) {
24          if (tempEdges[i][j] > 0 && (i != j)) {
25            edgesExist = true;
26            break;
27          }
28        }
29      }
30      for (int i = 0; i < tours[level].size(); i++) {
31        if (degree[tours[level][i]] > 0) {
32          start = tours[level][i];
33          break;
34        }
35      }
36      level++;
37      } while(edgesExist);
38
39      int size = tours.size();
40      for (int i = 0; i < size - 1; i++) {
41          int num = tours[i+1][0];
42      int tour_i_size = tours[i].size();
43          for (int j = 0; j < tour_i_size; j++) {
44              if (tours[i][j] == num) {
45                  auto it = tours[0].begin()+i+j+1;
46                  tours[0].insert(it, tours[i+1].begin()+1, tours[i+1].end());
47          break;
48              }
49          }
50  }
```

Ниже представлен фрагмент кода, отвечающий за нахождение минимального приоритета.

Листинг 5. Фрагмент кода файла ChinesePostman.h, отвечающий за нахождение минимального приоритета

```
inline bool operator<(const VertexPair &lt, const VertexPair &rt)
{
  return lt.cost[ lt.i ][ lt.j ] > lt.cost[ rt.i ][ rt.j ];
}
```

Создается класс `ChinesePostman`.

Листинг 6. класс ChinesePostman

```
class ChinesePostman
{
public:
  // Конструкторы
  ChinesePostman();
  ChinesePostman(std::string filename);

  // Решение проблемы китайского почтальона
  void solve();
  double optimalCost();
  //оптимальная стоимость решения китайского почтальона
  void printEuler();//Вывод результатов. Сначала исполняется solve()

private:
  // Описываем частные функции (Private Member Functions)
  void addEdge(std::string lab, int u, int v, double c);
  void findLeastCostPaths();
  //Алгоритм Флойда-Уоршелла (Floyd-Warshall Algorithm)
  void findOddVertices();
  void makeEulerian();
// Добавление пути, для создания Эйлерова графа (Graph Eulerian)
  std::vector<int>
  generateTour(int start, std::vector<std::vector<int>> &tempEdges);

  // Объявление переменных - членов класса
  bool solveRan;                    // has the user run solve() yet?
  int nVertices;// количество вершин
  std::vector<int> degree;// степени вершин
  std::vector<int> odd;// нечетные вершины
  std::vector<std::vector<int>> edges;
//матрица смежности, для подсчета ребер между вершинами
  std::vector<std::vector<std::string>> label;
```

```
33 //метки(для каждой пары вершин). Lable является идентификатором края.
34   std :: vector<std :: vector<double>> cost;//стоимость "дешевых"ребр
35   std :: vector<std :: vector<std :: string >> cheapestLabel;
36 // метки "дешевых"краев
37   std :: vector<std :: vector<bool>> defined;
38 //выевляет определена ли стоимость пути между вершинами
39   std :: vector<std :: vector<int>> path;// остов графа
40   double basicCost;//общая стоимость прохождения каждого ребра один раз
41
42   bool compareVertexPair( VertexPair  &lt,  VertexPair  &rt);
43 };
```

2.4 Описание разработанного приложения. Файл Driver.cpp

В данном заголовочном файле происходит считывание графа из текстового файла.

Листинг 7. фрагмент Driver.cpp

```
1 #include "ChinesePostman.h"
2 int main()
3 {
4   ChinesePostman cp("test1.txt");
5   cp.solve();
6   cp.printEuler();
7   return 0;
8 }
```

3 Эффективность работы алгоритма решения задачи китайского почтальона

На рисунке 1 приведен пример неориентированного графа.

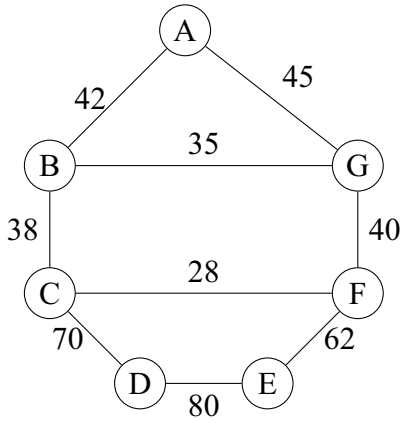

Рисунок 1 – Пример произвольного графа

Путь китайского почтальона для данного графа следующий:

$A - B - G - B - C - D - E - F - C - F - G - A$.

На рисунке 2 приведены графики зависимости времени работы алгоритма в миллисекундах от количества ребер и вершин. График построен в логарифмической шкале.

Как можно заметить при достижении 500 вершин и 1000 ребер заметно сильное увеличение времени работы алгоритма. Так же стоит заметить, что время работы алгоритма на ориентированном графе незначительно выше. Наглядно можно увидеть на рисунке 2.

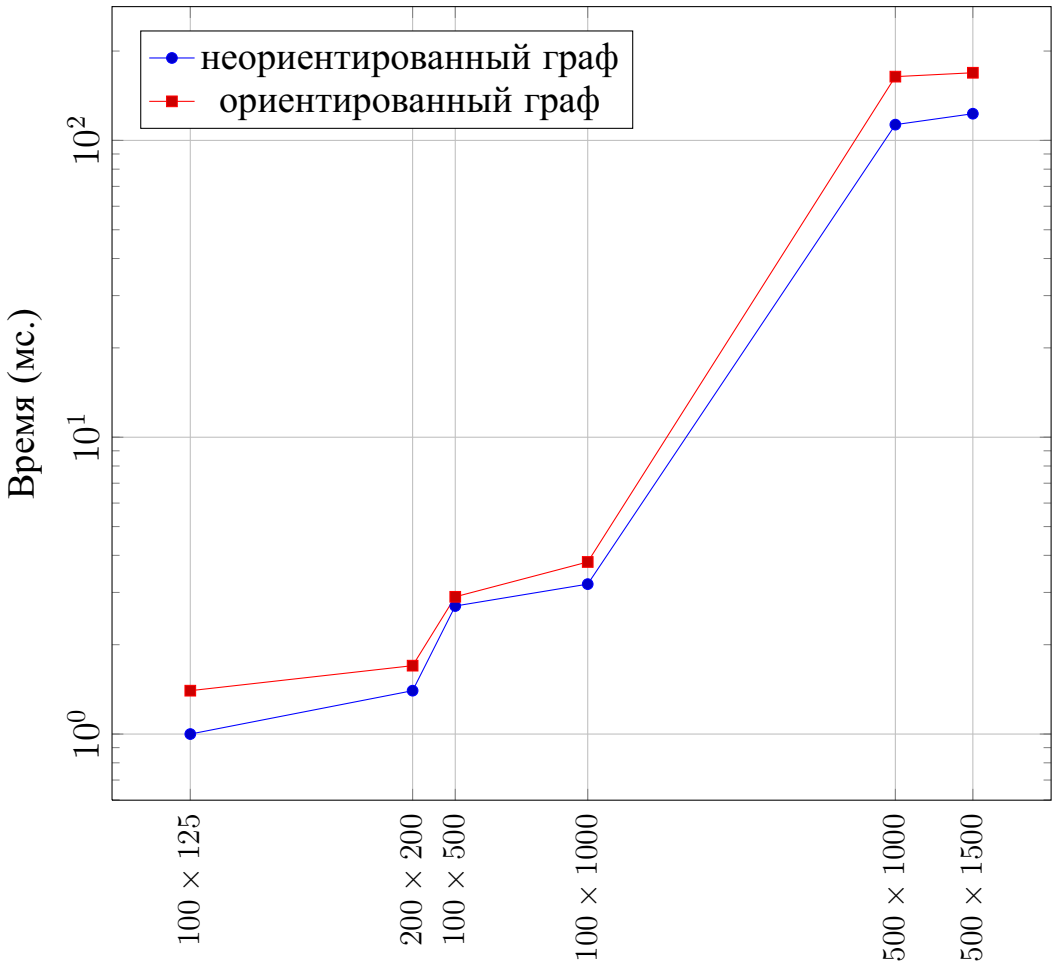

Рисунок 2 – Диаграмма зависимости времени работы алгоритма китайского почтальона от количества вершин и ребер графа.

ЗАКЛЮЧЕНИЕ

В данной дипломной работе были выполнены все поставленные задачи. Были реализованы алгоритмы решения задачи китайского почтальона, и проведено их тестирование, а также приведена сравнительная характеристика этих алгоритмов.

В ходе дипломной работы был реализован алгоритм китайского почтальона на языке $C++$ в среде Microsoft Visual Studio 2015 в ОС Windows 10, на процессоре Intel(R) Core(TM) $i7-4510U$ 2.60 GHz. На практике разработаны приложения реализующие алгоритм на ориентированном и неориентированном графе. Произведено сравнение времени работы данных алгоритмов на нескольких примерах.

Следует заметить, что алгоритм китайского почтальона может применяться для различных задач маршрутизации, как например для разработки кратчайшего маршрута снегоуборочной машины.

Стоит заметить, что только в случае ориентированного и неориентированного графа задача китайского почтальона имеет полиномиальное решение.

На диаграмме видно, что поиск кратчайшего пути на ориентированном графе занимает больше времени. Так же стоит отметить: при достижении 500 вершин и 1000 ребер заметно сильное увеличение времени работы алгоритма. Наглядно можно увидеть на рисунке 2.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 *Басакер, Р.* Конечные графы и сети / Р. Басакер, Т. Сали. — Москва: Наука, 1974.

2 *Кристофидес, Н.* Теория графов. Алгоритмический подход. / Н. Кристофидес. — Издательство «Мир», 1978.

3 *Берж, К.* Теория графов и ее приложения / К. Берж. — Москва: ИЗДАТЕЛЬСТВО ИНОСТРАННОЙ ЛИТЕРАТУРЫ, 1962.

4 *Новиков, Ф.* Дискретная математика для программистов. / Ф. Новиков. — Издательство «Питер», 2000.

5 *Ловас, Л. Н.* Прикладные задачи теории графов / Л. Н. Ловас. — Москва: Мир, 1998.

6 *Jensen, J.* Digraphs. Theory, Algorithms and Applications / J. Jensen. — 2007.

7 *Асанов, М.* Дискретная математика: Графы, матроиды, алгоритмы / М. Асанов, В. Баранский, В. Расин. — СПБ: Издательство «Лань», 2010.

8 *Kuhn, H. W.* The Hungarian Method for the assignment problem / H. W. Kuhn. — 1955. — Pp. 83–97.

9 *Munkres, J.* Algorithms for the Assignment and Transportation Problems / J. Munkres. — 1957. — Pp. 32–38.

10 *Кормен, Т.* Алгоритмы. Построение и анализ / Т. Кормен. — Издательство «Вильямс», 2010.