Федеральное государственное бюджетное образовательное учреждение высшего образования Нижегородский государственный технический университет им. Р.Е. Алексеева

Лабораторная работа №4
«Алгоритм Хаффмана»

Выполнил:
студент 2-го курса
кафедры «Прикладная математика»
Поляков Кирилл

2020г.

Оглавление

Введение

Целью данной работы является - написать программу, реализующую алгоритм Дейкстры на произвольном графе, представленном в виде матрицы смежности или списка смежности.

Файл FSPG.cpp

```
#include <iostream>
#include "Algorithms.h"
```

Функции

int **main** (int argc, char *argv[])

Функции

◆ main()

```
int main ( int       argc,
           char *    argv[]
         )
```

Входная точка программы

Аргументы

argc Количество аргументов командной строки

argv[] Аргументы командной строки

Возвращает

int

См. определение в файле **FSPG.cpp** строка **13**

FSPG.cpp

```cpp
#include <iostream>

#include "Algorithms.h"

int main(int argc, char* argv[])
{
    setlocale(LC_ALL, "rus");
    Algorithms::FindWay(argc, argv);
}
```

Файл Algorithms.h

```
#include <string>
#include <iostream>
#include <list>
#include "List.h"
#include "Structs.h"
```

Классы

class **Algorithms**

Класс Algorithms

`#include <Algorithms.h>`

Открытые статические члены

static void **FindWay** (int argc, char *argv[])

Закрытые статические члены

static int ** **ConvertToMatrix** (list< string > edges, int sizeMatrix)

static **List** * **ConvertToList** (list< string > edges, int sizeMatrix)

static **Data** **ReadFile** (string filename)

template<class T >

static **Way** **FindingWay** (T matrix, const int sizeMatrix, int start, int finish)

Подробное описание

Класс, который представляет основной функционал.

См. определение в файле **Algorithms.h** строка **13**

Методы

◆ ConvertToList()

List * Algorithms::ConvertToList (list< string > edges,
int sizeMatrix
)

static private

Функция, которая парсит входные данные и преобразует их в односвязанный список.

Аргументы

edges Строки, содержащие данные о ребрах и вершинах графа.

sizeMatrix Количество вершин графа.

Возвращает

List* представление графа в видк односвязвнного списка

См. определение в файле **Algorithms.cpp** строка **51**

◆ ConvertToMatrix()

```
int ** Algorithms::ConvertToMatrix ( list< string >  edges,
                                     int             sizeMatrix
                                   )
```

static private

Функция, которая парсит входные данные и преобразует их в матрицу.

Аргументы

edges	Строки, содержащие данные о ребрах и вершинах графа.
sizeMatrix	Количество вершин графа.

Возвращает

int** представление графа в матричном виде

См. определение в файле **Algorithms.cpp** строка **7**

◆ FindingWay()

```
template<class T >
Way Algorithms::FindingWay ( T           matrix,
                             const int   sizeMatrix,
                             int         start,
                             int         finish
                           )
```

static private

Функция, которая находит кратчайший путь от начальной вершины до конечной.

Аргументы

matrix	Данные о вершинах.
sizeMatrix	Количество вершин графа.
start	Номер вершины, от которой будет построен путь.
finish	Номер вершины, до которой будет построен путь.

Возвращает

Way

См. определение в файле **Algorithms.h** строка **63**

◆ FindWay()

```
void Algorithms::FindWay ( int      argc,
                           char *   argv[]
                         )
```

static

Основная функция, которая находит кратчайший путь от начальной вершины до конечной.

Аргументы

argc Количество элементов командной строки.

argv Аргументы командной строки.

См. определение в файле **Algorithms.cpp** строка **82**

◆ ReadFile()

```
Data Algorithms::ReadFile ( string  filename )
```

static private

Функция, которая считывает данные из файла и на их основании решает как хранить граф в памяти (matrix / link).

Аргументы

filename Имя файла, в котором хранятся данные.

Возвращает

Data

См. определение в файле **Algorithms.cpp** строка **138**

Объявления и описания членов классов находятся в файлах:

- **Algorithms.h**
- **Algorithms.cpp**

Algorithms.h

```
#pragma once
#include <string>
#include <iostream>
#include <list>
#include "List.h"
#include "Structs.h"

using namespace std;

static class  Algorithms
{
    static int** ConvertToMatrix(list<string> edges, int sizeMatrix);
    static List* ConvertToList(list<string> edges, int sizeMatrix);

    static Data ReadFile(string filename);

    template <class T>
    static Way FindingWay(T matrix, const int sizeMatrix, int start, int finish);

public:
    static void FindWay(int argc, char* argv[]);
};

template <class T>
Way Algorithms::FindingWay(T matrix, const int sizeMatrix, int start, int finish)
{
    Way way;

    if (start <= 0 || start > sizeMatrix)
    {
        cout << "Error. The start point was entered incorrectly." << endl;
        return way;
    }

    if (finish <= 0 || finish > sizeMatrix)
    {
        cout << "Error. The finish point was entered incorrectly." << endl;
        return way;
    }

    int beg = start - 1;
    int* distMin = new int[sizeMatrix];
    int* vertex = new int[sizeMatrix];
    int minIndex, min;

    for (int i = 0; i < sizeMatrix; i++)
    {
        distMin[i] = INT_MAX;
        vertex[i] = 1;
    }
    distMin[beg] = 0;

    do
    {
        min = minIndex = INT_MAX;
        for (int i = 0; i < sizeMatrix; i++)
        {
            if ((vertex[i] == 1) && (distMin[i] < min))
            {
                min = distMin[i];
                minIndex = i;
            }
        }

        if (minIndex != INT_MAX)
        {
            for (int i = 0; i < sizeMatrix; i++)
            {
                if (matrix[minIndex][i] > 0)
                {
                    int temp = min + matrix[minIndex][i];
                    if (temp < distMin[i])
                        distMin[i] = temp;
```

```
                }
            }
            vertex[minIndex] = 0;
        }
    } while (minIndex < INT_MAX);

    int* ver = new int[sizeMatrix];
    int end = finish - 1;
    ver[0] = finish;
    int ind = 1;
    int weight = distMin[end];

    way.length = distMin[end];

    while (end != beg)
    {
        for (int i = 0; i < sizeMatrix; i++)
            if (matrix[i][end] != 0)
            {
                int temp = weight - matrix[i][end];
                if (temp == distMin[i])
                {
                    weight = temp;
                    end = i;
                    ver[ind++] = i + 1;
                }
            }
    }

    way.count = ind;
    way.way = ver;
    way.start = start;
    way.finish = finish;

    return way;
}
```

Файл List.h

```
#include "Structs.h"
```

Классы

class **List**

List.h

```
#pragma once
#include "Structs.h"

class List
{
    Node* head = nullptr;
    Node* last = nullptr;
    int count = 0;
public:
    void Add(int indexVertex, int weight);
    void Print(int index);
    int Count();

    int& operator[] (const int i);
};

```

Класс List

`#include <List.h>`

Открытые члены

void	**Add** (int indexVertex, int weight)
void	**Print** (int index)
int	**Count** ()
int &	**operator[]** (const int i)

Закрытые данные

Node *	**head** = nullptr
Node *	**last** = nullptr
int	**count** = 0

Подробное описание

Класс, реализующий односвязанный список.

См. определение в файле **List.h** строка **7**

Методы

◆ Add()

```
void List::Add ( int  indexVertex,
                 int  weight
               )
```

Функция, добавляющая новый элемент в список.

Аргументы

indexVertex Номер вершины.

weight Вес до вершины.

См. определение в файле **List.cpp** строка **7**

◆ Count()

int List::Count ( )

Функция, которая возвращает количество элементов списка.

Возвращает

int Возвращает количество элементов списка.

См. определение в файле **List.cpp** строка **42**

◆ operator[]()

int & List::operator[] (const int i)

Перегруженный оператор [].

Аргументы

i Индекс выбранного элемента в списке.

Возвращает

int Возвращает выбранный элемент списка (0, если ничего не найдено).

См. определение в файле **List.cpp** строка **47**

◆ Print()

void List::Print (int index = -1)

Функция, которая выводит значение списка.

Аргументы

index Номер вершины.

См. определение в файле **List.cpp** строка **27**

Данные класса

◆ count

int List::count = 0 private

См. определение в файле **List.h** строка **11**

◆ head

Node* List::head = nullptr `private`

См. определение в файле **List.h** строка **9**

◆ last

Node* List::last = nullptr `private`

См. определение в файле **List.h** строка **10**

Объявления и описания членов классов находятся в файлах:

- **List.h**
- **List.cpp**

List.cpp

```cpp
#include "List.h"
#include <iostream>
#include <assert.h>

using namespace std;

void List::Add(int indexVertex, int weight)
{
    count++;
    if (head)
    {
        Node* temp = new Node();
        temp->indexVertex = indexVertex;
        temp->weight = weight;
        last->next = temp;
        last = temp;
    }
    else
    {
        head = new Node();
        head->indexVertex = indexVertex;
        head->weight = weight;
        last = head;
    }
}

void List::Print(int index = -1)
{
    Node* temp = head;
    if (index >= 0)
        cout << "Vertex " << index + 1 << ": ";
    while (temp)
    {
        cout << temp->indexVertex + 1;
        if (temp->next)
            cout << " -> ";
        temp = temp->next;
    }
    cout << endl;
}

int List::Count()
{
    return count;
}

int& List::operator[](const int i)
{
    assert(i >= 0);
    int notFind = 0;

    if (i == 0)
        return notFind;

    Node* node = head;
    while (node)
    {
        if (node->indexVertex == i)
            return node->weight;
        node = node->next;
    }

    return notFind;

}
```

Файл Structs.h

```
#include <list>
#include <string>
```

Классы

struct	**Way**
struct	**Data**
struct	**Node**

Функции

void	**DeleteSpaces** (string &str)

Функции

◆ DeleteSpaces()

void DeleteSpaces (string & str)

Функция удаления первых пробелов из переданной строки.

Аргументы

str[out] строка, из которой будут удалены пробелы

См. определение в файле **Structs.h** строка **44**

Structs.h

```cpp
#pragma once
#include <list>
#include <string>

using namespace std;

struct Way
{
    int length = 0;
    int* way = nullptr;
    int count = 0;
    int start = 0;
    int finish = 0;
};

struct Data
{
    int sizeGraph = 0;
    list<string> values;
    string typeConvert;
};

struct Node
{
    int indexVertex;
    int weight;
    Node* next;
};

void DeleteSpaces(string& str)
{
    size_t strBegin = str.find_first_not_of(' ');
    size_t strEnd = str.find_last_not_of(' ');
    str.erase(strEnd + 1, str.size() - strEnd);
    str.erase(0, strBegin);
}
```

Структура Way

`#include <Structs.h>`

Открытые атрибуты

int	**length** = 0 Длина пути Подробнее...
int *	**way** = nullptr Сам путь Подробнее...
int	**count** = 0 Количесвто шагов Подробнее...
int	**start** = 0 Вершина старта Подробнее...
int	**finish** = 0 Вершина финиша Подробнее...

Подробное описание

Структкра, которая хранит данные о найденном пути.

См. определение в файле **Structs.h** строка **10**

Данные класса

◆ count

int Way::count = 0

Количесвто шагов

См. определение в файле **Structs.h** строка **17**

◆ finish

int Way::finish = 0

Вершина финиша

См. определение в файле **Structs.h** строка **21**

◆ length

int Way::length = 0

Длина пути

См. определение в файле **Structs.h** строка **13**

◆ start

int Way::start = 0

Вершина старта

См. определение в файле **Structs.h** строка **19**

◆ way

int* Way::way = nullptr

Сам путь

См. определение в файле **Structs.h** строка **15**

Объявления и описания членов структуры находятся в файле:

- **Structs.h**

Структура Node

`#include <Structs.h>`

Открытые атрибуты

int	**indexVertex** Индекс вершины Подробнее...
int	**weight** Вес до вершины Подробнее...
Node *	**next** Указатель на следующую вершины Подробнее...

Подробное описание

Структкра, которая хранит данные о вершинах графа.

См. определение в файле **Structs.h** строка **40**

Данные класса

◆ indexVertex

int Node::indexVertex

Индекс вершины

См. определение в файле **Structs.h** строка **43**

◆ next

Node* Node::next

Указатель на следующую вершины

См. определение в файле **Structs.h** строка **47**

◆ weight

int Node::weight

Вес до вершины

См. определение в файле **Structs.h** строка **45**

Объявления и описания членов структуры находятся в файле:

- **Structs.h**

Структура Data

`#include <Structs.h>`

Открытые атрибуты

int	**sizeGraph** = 0 Количество вершин графа Подробнее...
list< string >	**values** Данные о вершинах Подробнее...
string	**typeConvert** Тип хранения графа в памяти (matrix/list) Подробнее...

Подробное описание

Структкра, которая хранит данные о графе.

См. определение в файле **Structs.h** строка **27**

Данные класса

◆ sizeGraph

int Data::sizeGraph = 0

Количество вершин графа

См. определение в файле **Structs.h** строка **30**

◆ typeConvert

string Data::typeConvert

Тип хранения графа в памяти (matrix/list)

См. определение в файле **Structs.h** строка **34**

◆ values

list<string> Data::values

Данные о вершинах

См. определение в файле **Structs.h** строка **32**

Объявления и описания членов структуры находятся в файле:

- **Structs.h**

Тестирование

```
0 7 6 0
0 0 4 5
0 0 0 9
0 0 0 0

Type of graph representation in memory: matrix.
Start: 1   Finish: 3
Way length: 6.
Count of steps: 2.
Way: 1 -> 3
```

```
Vertex 1: 2 -> 3
Vertex 2: 3 -> 4
Vertex 3: 4
Vertex 4:

Type of graph representation in memory: link.
Start: 1   Finish: 3
Way length: 6.
Count of steps: 2.
Way: 1 -> 3
```

```
4 link vertex_1 -> (2,7) -> (3, 6)
vertex_2 -> (3, 4) -> (4, 5)
vertex_3 -> (4, 9)
```