**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра САПР**

ОТЧЁТ

**по курсовой работе**

**по дисциплине «Алгоритмы и структуры данных»**

**Тема: «Потоки в сетях»**

**Вариант 2.**

|  |  |  |
| --- | --- | --- |
| Студент гр. 9309 |  | Корягин Е.А. |
| Преподаватель |  | Тутуева А.В. |

Санкт-Петербург

2021

## Постановка задачи.

**Задание:**

Входные данные: текстовый файл со строками в формате V1, V1, P, где V1, V2 направленная дуга транспортной сети, а P – ее пропускная способность. Исток всегда обозначен как S, сток – как T

**Найти:** максимальный поток в сети используя алгоритм Эдмондса-Карпа.

## Описание работы программы

Для реализации понадобятся классы «Graph» и «List» , обеспечивающие необходимый функционал. Алгоритм реализован так: сначала мы считываем из файла все уникальные символы как графы. Затем создается экземпляр класса «Graph», в который вносятся пропускные способности, которые считываются из файла в виде чисел. Теперь применяется сам алгоритм Эдмондса-Карпа, который находит максимальный поток в транспортной сети. Результат выводится на экран.

## Описание алгоритма

Алгоритм Эдмондса-Карпа — это вариант алгоритма Форда-Фалкерсона, при котором на каждом шаге выбирают кратчайший дополняющий путь из s в t в остаточной сети (полагая, что каждое ребро имеет единичную длину). Кратчайший путь находится поиском в ширину.

Сложность алгоритма Эдмондса-Карпа равна O (|V|⋅|E|^2).

template<class data\_T>

class List

|  |  |  |
| --- | --- | --- |
| Поля | Описание | |
| Node \*head; | Указатель на начало списка | |
| Node \*tail; | Указатель на конец списка | |
| size\_t size; | Переменная, хранящая размер списка | |
| Методы | Описание | Оценка вр. сложности |
| void push\_back(data\_T) | Добавление в конец списка | O(1) |
| void push\_front(data\_T) | Добавление в начало списка | O(1) |
| void pop\_back() | Удаление последнего элемента | O(1) |
| void pop\_front() | Удаление первого элемента | O(1) |
| void insert(size\_t, data\_T) | Добавление элемента по индексу | O(n) |
| data\_T at(const int) | Получение элемента по индексу | O(n) |
| void remove(size\_t) | Удаление элемента по индексу | O(n) |
| size\_t GetSize() | Получение размера списка | O(1) |
| void print\_to\_console() | Вывод элементов в консоль через разделитель | O(n) |
| bool isEmpty() | Проверка на пустоту списка | O(1) |
| void clear() | Удаление всех элементов списка | O(n) |
| void reverse | Изменение порядка элементов в списке на обратный | O(n) |
| void set(size\_t, data\_T) | Замена элемента по индексу | O(n) |
| data\_T top() | Получение первого элемента в списке | O(1) |
| int search(data\_T info) | Поиск индекса элемента по информации | O(n) |

class Node

|  |  |
| --- | --- |
| Поля | Описание |
| Node \*next; | Указатель на следующий элемент списка |
| Node \*prev; | Указатель на предыдущий элемент списка |
| data\_T data; | Информация, хранящаяся в элементах списка |

class Graph

|  |  |  |
| --- | --- | --- |
| Поля | Описание | |
| int dimension; | Размерность графа | |
| int\*\* matrix; | Двумерный массив (для хранения пропускной способности) | |
| Методы | Описание | Оценка временной сложности |
| void SetValue(int, int, int); | Установление значения по индексу | O(1) |
| int GetValue(int, int ); | Получение значения по индексу | O(1) |
| List<int>\* BFS(Graph& ); | Поиск в ширину | O(n^2) |

Функции:

|  |  |  |
| --- | --- | --- |
| Функция | Описание | O |
| void SearchUniqueVertexes(string&, int&, ifstream&) | Поиск уникальных (неповторяющихся) вершин | O(n^2) |
| void SearchThroughput(int, Graph\*, char\*, ifstream&) | Поиск пропускных способностей и внесение их в двумерный массив | O(n^2) |
| void BubbleSort(int, char\*) | Сортировка массива | O(n^2) |
| unsigned EdmondsKarp(int, Graph&) | Алгоритм Эдмондса-Карпа | O(v^3) |
| void Process(ifstream&, int&); | Процесс работы программы | O(v^3) |

## Описание реализованных unit-тестов.

1. SearchUniqueVertexesTest - Проверка поиска уникальных вершин

2. EdmondsKarpTest\_1 - Проверка работы программы на текстовом файле «test1.txt»

3. EdmondsKarpTest\_2 - Проверка работы программы на текстовом файле «test2.txt»

4. GetValueTest - Проверка метода GetValue для «Graph»

5. SetValueTest - Проверка метода SetValue для «Graph»

6. BFSTest – Проверка метода BFS

Остальные проверяют

## Результат выполнения всех unit-тестов

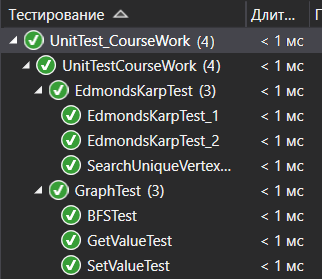


Рисунок 1. Unit-тесты

## Примеры работы программы

