# ВВЕДЕНИЕ

Одной из наиболее интересных и важных задач теории графов является задача определения максимального потока, протекающего от некоторой вершины графа (источника) к некоторой конечной вершине (стоку). При этом каждой дуге графа приписана некоторая пропускная способность и эта пропускная способность определяет наибольшее значение потока, который может протекать по данной дуге. Эта задача и ее варианты могут возникать во многих практических приложениях, например при определении максимальной интенсивности транспортного потока между двумя пунктами на карте дорог, или для нахождения величины максимального потока информации между двумя узлами в сети.

# 1 Алгоритм нахождения величины максимального потока в сети

# 1.1 Метод Форда-Фалкерсона

Назовем сетью (flow network) ориентированный граф каждому ребру которого поставлено в соответствие число , называемое пропускной способностью (capacity) ребра.

Задача о максимальном потоке (maximum-flow problem) состоит в следующем: для данной сети G с истоком s и стоком t найти поток максимальной величины.

Потоком (flow) в сети G назовём функцию , удовлетворяющую трём свойствам:

1) ограничение, связанное с пропускной способностью: для всех из ;

2) кососимметричность: для всех из .

3) cохранение потока (flow conservation):

для всех из

Метод решения задачи о максимальном потоке (от s к t) был предложен Фордом и Фалкерсоном, и их «техника пометок» составляет основу других алгоритмов решения многочисленных задач, являющихся простыми обобщениями или расширениями указанной задачи [1].

Основная теорема метода Форда-Фалкерсона – теорема о максимальном потоке и минимальном разрезе.

Теорема. Пусть - по­ток в сети . Тогда следующие утверждения равносильны:

1. Поток максимален (является потоком максимальной величины) в сети .

2. Остаточная сеть не содержит дополняющих путей.

3. Для некоторого сечения сети выполнено равенство . В этом случае разрез является минимальным, то есть имеет минимально возможную пропускную способность.

Существует несколько алгоритмов, реализующих данный метод и отличающихся временем работы.

Общая схема алгоритма Форда-Фалкерсона: на каждом шаге мы выбираем произвольный дополняющий путь и увеличиваем поток , добавляя поток величины [2].

# 1.2 Схема определения величины максимального потока

Постановка задачи: дана транспортная сеть задача о максимальном потоке заключается в нахождении потока из истока в сток такого, что величина потока максимальна.

Учитывая основные положения метода Форда-Фалкерсона, для нахождения максимального потока можно применить описанный ниже алгоритм. Идея данного алгоритма состоит в нахождении сквозных путей с положительными потоками от источника к стоку.

Для ребра где используем запись для представления пропускных способностей в направлениях соответственно. Для удобства на схеме сети будем располагать на ребре ближе к узлу а - ближе к узлу. Рассмотрим ребро с начальной пропускной способностью В процессе выполнения алгоритма части этих пропускных способностей "забираются" потоками, проходящими через данное ребро, в результате каждое ребро будет иметь остаточную пропускную способность. Будем использовать запись для представления остаточных пропускных способностей. Сеть, где все ребра имеют остаточную пропускную способность, назовем *остаточной*.

Для произвольного узла , получающего поток от узла определим метку , где - величина потока, протекающего от узла к узлу . Алгоритм нахождения максимального потока предполагает выполнение следующих действий.

*Шаг 1.* Для всех ребер положим остаточную пропускную способность равной первоначальной пропускной способности, т.е. приравняем = Пометим узел 1 меткой Полагаем (текущая вершина - первая) и переходим ко второму шагу.

*Шаг 2.* Определяем множество как множество узлов , в которые можно перейти из узла по ребру с положительной остаточной пропускной способностью (т.е. для всех ). Если не пустое множество, выполняем третий шаг, в противном случае переходим к шагу 4.

*Шаг 3.* В множестве находим узел , такой, что для всех . Положим и пометим узел меткой. Если последней меткой помечен узел стока, сквозной путь найден, и мы переходим к пятому шагу. В противном случае полагаем и возвращаемся ко второму шагу.

*Шаг 4 (Откат назад).* Если , сквозной путь невозможен, и мы переходим к шагу 6. Если , находим помеченный узел , непосредственно предшествующий узлу i, и удаляем узел i из множества узлов, смежных с узлом . Полагаем и возвращаемся ко второму шагу.

*Шаг 5 (Определение остаточной сети).* Обозначим через множество узлов, через которые проходит -й найденный сквозной путь от узла источника (узел 1) до узла стока (узел ). Тогда максимальный поток, проходящий по этому пути, вычисляется как

Пропускные способности ребер, составляющих сквозной путь, уменьшаются на величину в направлении движения потока и увеличиваются на эту же величину в противоположном направлении. Таким образом, для ребра входящего в сквозной путь, текущие остаточные стоимости изменятся следующим образом:

* , если поток идет от узла к узлу ,
* если поток идет от узла к узлу .

Далее восстанавливаем все узлы, удаленные на шаге 4. полагаем и возвращаемся ко второму шагу для поиска нового сквозного пути.

*Шаг 6 (Решение).* При найденных сквозных путях максимальный потоквычисляется по формуле .

# 1.3 Пример пошаговой реализации алгоритма

Найдем максимальный поток в сети, изображенной на рисунке 1.1. На рисунке 1.2 предлагается графическая иллюстрация выполнения алгоритма.

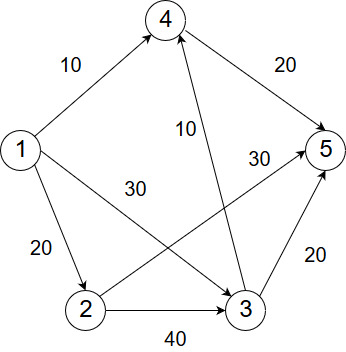


Рисунок 1.1 – Ориентированный граф

*Итерация 1.*

Положим остаточные пропускные способности всех ребер равными первоначальным пропускным способностям .

*Шаг 1.* Помечаем узел 1 меткой . Полагаем

*Шаг 2.*  (множество не пустое).

*Шаг 3.* , поскольку = . Назначаем и помечаем узел 3 меткой [30, 1]. Полагаем и возвращаемся к шагу 2.

*Шаг 2.* .

*Шаг 3.*  и . Помечаем узел 5 меткой [20,3]. Получен сквозной путь. Переходим к шагу 5.

*Шаг 5.* Сквозной путь определяем по меткам, начиная с узла 5 и заканчивая узлом 1: (5) (3) (1). Таким образом, . Вычисляем остаточные пропускные способности вдоль пути N1:



Рисунок 1.2 – Графическое представление задачи и пошаговое выполнение алгоритма

*Итерация 2.*

*Шаг 1.* Назначаем a1 = и помечаем узел 1 меткой [,-]. Полагаем i = 1.

*Шаг 2.* S1 = [2, 3, 4].

*Шаг 3.* k = 2, назначаем a2 = c12 = max {20, 10, 10} = 20 и помечаем узел 2

меткой [20, 1]. Полагаем i =2 и возвращаемся к шагу 2.

*Шаг 2.* S2 = [3, 5].

*Шаг 3.* k = 3, назначаем a3 = c23 = 40. Помечаем узел 3 меткой [40, 2]. Полагаем i =3 и возвращаемся к шагу 2.

*Шаг 2.*S3 = [4] (отметим, что c35 = 0, поэтому узел 5 не включается в S3).

*Шаг 3.* k = 4, назначаем a4 = c34 = 10 и помечаем узел 4 меткой [10, 3]. Полагаем i = 4 и возвращаемся к шагу 2.

*Шаг 2.* S4 = [5] (поскольку узлы 1 и 3 уже помечены, они не включаются в

S4).

*Шаг 3.* k = 5 и a5 = c45 = 20. Помечаем узел 5 меткой [20, 4]. Получаем

сквозной путь. Переходим к шагу 5.

*Шаг 5.* N2 = {1, 2, 3, 4, 5} и f2 = min {, 20, 40, 10, 20} = 10.

Вычисляем остаточные пропускные способности вдоль пути N2:

(c12, c21) = (20 - 10, 0 + 10) = (10, 10),

(c23, c32) = (40 - 10, 0 + 10) = (30, 10),

(c34, c43) = (10 - 10, 5 + 10) = (0, 15),

(c45, c54) = (20 - 10, 0 + 10) = (10, 10).

*Итерация 3.*

*Шаг 1.* Назначаем a1 = и помечаем узел 1 меткой [,-]. Полагаем i = 1.

*Шаг 2.* S1 = [2, 3, 4].

*Шаг 3.* k = 2, назначаем a2 = c12 = max {10, 10, 10} = 10 и помечаем узел 2

меткой [10, 1]. Полагаем i = 2 и возвращаемся к шагу 2.

*Шаг 2.* S2 = [3, 5].

*Шаг 3.* k = 3 и a3 = c23 = 30. Помечаем узел 3 меткой [30, 2]. Полагаем i = 3

и возвращаемся к шагу 2.

*Шаг 2.*S3 пусто, поскольку c34 = c35 = 0. Переходим к шагу 4.

*Шаг 4.* Метка [30, 2] узла 3 показывает номер предшествующего узла r = 2. На этой итерации узел 3 в дальнейшем во внимание не принимается, его

метку вычеркиваем. Полагаем i = r = 2 и возвращаемся к шагу 2.

*Шаг 2.* S2 = [5] (поскольку узел 3 удален из возможного сквозного пути).

*Шаг 3.* k = 5 и a5 = c25 = 30. помечаем узел меткой [30, 2]. Получаем

сквозной путь. Переходим к шагу 5.

*Шаг 5.* N3 = {1, 2, 5} и f3 = min {, 10, 30} = 10. Вычисляем остаточные пропускные способности вдоль пути N3:

(c12, c21) = (10 - 10, 10 + 10) = (0, 20), (c25, c52) = (30 - 10, 0 + 10) = (20, 10).

*Итерация 4.* На этой итерации получен путь N4 = {1, 3, 2, 5} с f4 = 10.

*Итерация 5.* На этой итерации получен путь N5 = {1, 4, 5} с f5 = 10.

*Итерация 6.* Новые сквозные пути невозможны, поскольку все ребра,

исходящие из узла 1, имеют нулевые остаточные пропускные способности.

Переходим к шагу 6 для определения решения.

*Шаг 6.* Максимальный объем потока в сети равен F = f1 + f2 + … +f5 = 20

+ 10 + 10 + 10 + 10 = 60 единиц [3].

# 2 Программная реализация алгоритма Форда-Фалкерсона

Ниже приведен программный код на языке Lisp для решения задачи поиска максимального потока через сеть. Основная функция ford-falkerson является рекурсивной функцией, осуществляющей схему алгоритма, описанного в первой главе работы. Common Lisp удобен для программирования рекурсии, что в нашем случае сыграло ключевую роль при выборе языка для решения данной задачи, поскольку алгоритм Форда-Фалкерсона подразумевает выполнение похожих действий на каждой итерации.

Код на Common Lisp:

На рисунке 2.1 представлено окно с результатом работы программы для графа, изображенного на рисунке 1.1. Дуги графа заданы в текстовом файле с именем «1.txt». Ответ, полученный при запуске программы для первой вершины в качестве истока и пятой вершины в качестве стока, совпадает с вычисленным в ходе пошагового разбора алгоритма.

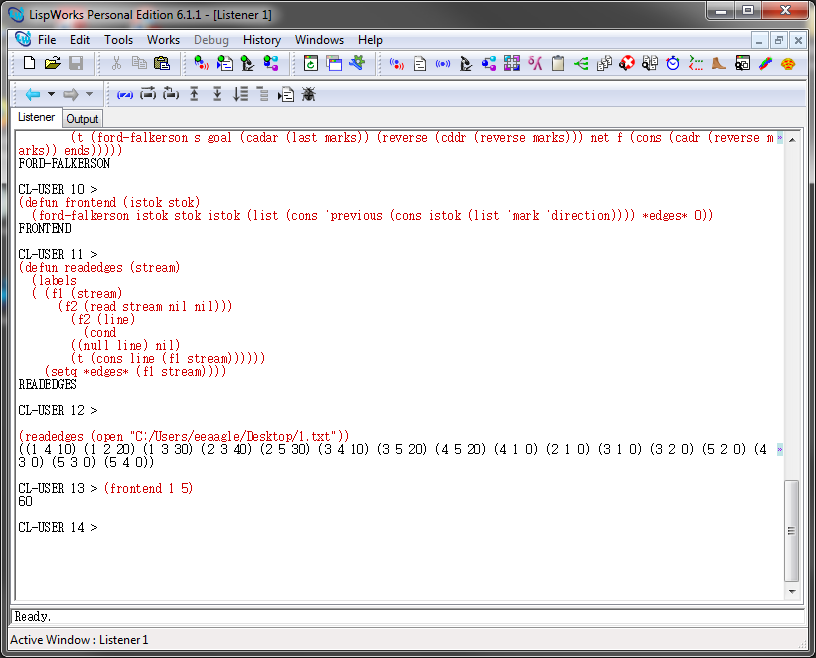


Рисунок 2.1 – Окно работы программы в LispWorks

Величина максимального потока из в для графа, изображенного на рисунке 2.2, равна 19. Проверим работу программы для данного графа. Для этого сохраним в файл «2.txt» список дуг, описывающий структуру графа. Содержимое файла представлено на рисунке 2.3.

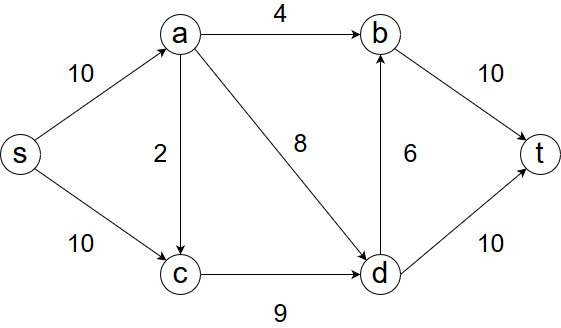


Рисунок 2.2 – Ориентированный граф

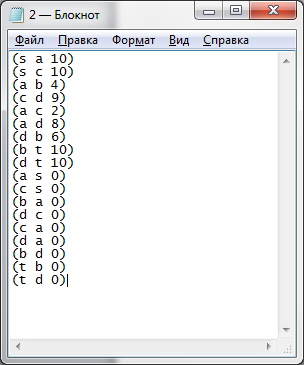


Рисунок 2.3 – Список дуг графа в файле «2.txt»

Результат вычисления величины максимального потока из вершины в вершину представлен на рисунке 2.4.

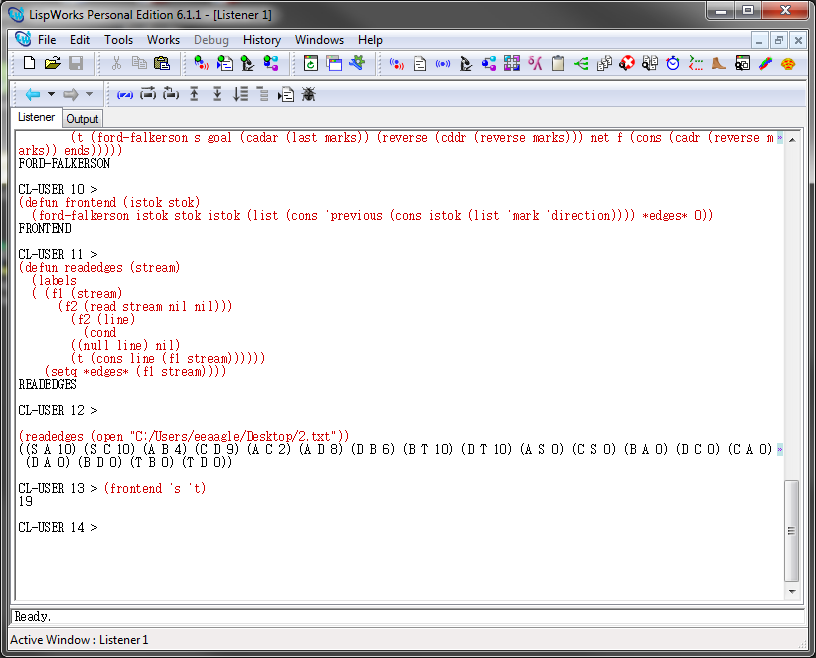


Рисунок 2.4 – Результат работы программы

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Кристофидес, Н. Теория графов. Алгоритмический подход, — М.:Мир, 1978, стр.310.

2 Кормен, Т. , Лейзерсон, Ч., Ривест, Р. Алгоритмы: построение и анализ, 2-е, — М.:Вильямс, 2005, стр.553.

3. Таха, Хэмди, А. Введение в исследование операций, 6-е, — М.:Вильямс, 2001, стр.245.

4 [LispWorks](https://www.google.ru/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=0ahUKEwiKy8rd1OnXAhVoMZoKHRuADDUQFggoMAA&url=http%3A%2F%2Fwww.lispworks.com%2F&usg=AOvVaw0qlrOs26KenEU3d4guSl2p) [Электронный ресурс]: URL: www.lispworks.com/