МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Брянский государственный технический университет

Утверждаю
Ректор университета

_____О.Н. Федонин

«____»_____2020 г.

СТРУКТУРЫ И АЛГОРИТМЫ ОБРАБОТКИ ДАННЫХ

АЛГОРИТМЫ НА НАГРУЖЕННЫХ ГРАФАХ

Методические указания
к выполнению лабораторной работы №17
для студентов очной, очно-заочной и заочной форм обучения
по направлениям подготовки
230100 «Информатика и вычислительная техника»,
010500 «Математическое обеспечение и администрирование информационных систем»,
231000 «Программная инженерия»

Брянск 2020

УДК 006.91

Структуры и алгоритмы обработки данных. Алгоритмы на нагруженных графах [Электронный ресурс]: методические указания к выполнению лабораторной работы №17 для студентов очной, очно-заочной и заочной форм обучения по направлениям подготовки 230100 «Информатика и вычислительная техника», 010500 «Математическое обеспечение и администрирование информационных систем», 231000 «Программная инженерия». – Брянск: БГТУ, 2020.– 40 с.

Разработали:
канд. техн. наук, проф.
В.К. Гулаков
канд. техн. наук, доц.
А.О. Трубаков
канд. техн. наук, доц.
Е.О. Трубаков

Рекомендовано кафедрой «Информатика и программное обеспечение» БГТУ (протокол №1 от 13.09.20)

Научный редактор В.В. Конкин
Редактор издательства Л.Н. Мажугина
Компьютерный набор В.К. Гулаков

Темплан 2020 г., п. 204

Подписано в печать Формат 1/16
Усл.печ.л. 2,55 Уч.-изд.л. 2,55

Издательство Брянского государственного технического университета
241035, Брянск, бульвар им.50-летия Октября, 7, БГТУ, тел. 58-82-49
Лаборатория оперативной полиграфии БГТУ, ул. Институтская, 16

1. ЦЕЛЬ ЛАБОРАТОРНОЙ РАБОТЫ

Целью лабораторной работы является приобретение навыков работы с алгоритмами на нагруженных графах.

Продолжительность лабораторной работы − 4 часа.

2. ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Остовное дерево (стягивающее дерево, каркас) – это граф, в котором каждая пара вершин связана одной и только одной цепью.

Теорема: Граф *G* является связным тогда и только тогда, когда он содержит остовное дерево.

На рис. 1 представлен взвешеный (нагруженный) граф и некоторые из его остовных деревьев с весами.

Каркас связанного графа можно искать с помощью различных методов.

Для поиска произвольного каркаса используются: грубый метод, метод поиска в глубину, метод поиска в ширину.

Поиск каркаса минимального веса можно искать только в нагруженном графе.

Для поиска каркасов минимального веса применяются алгоритмы Прима и Краскала.

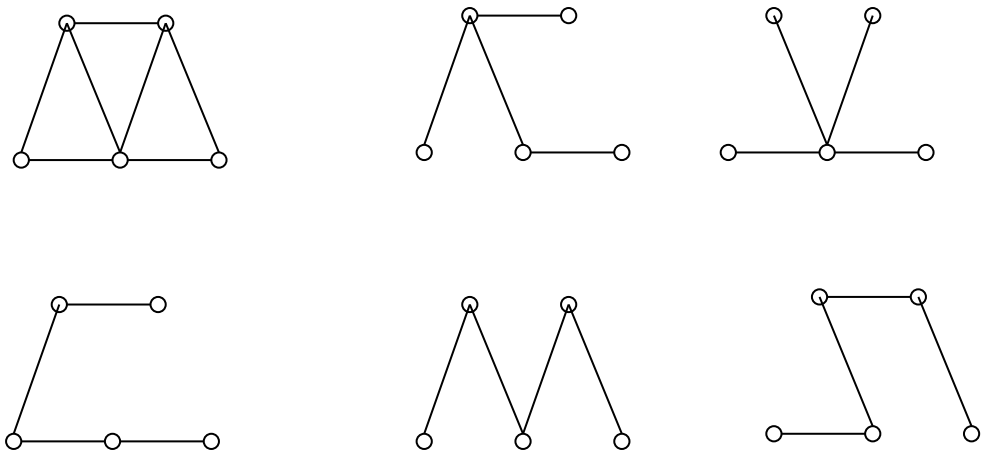


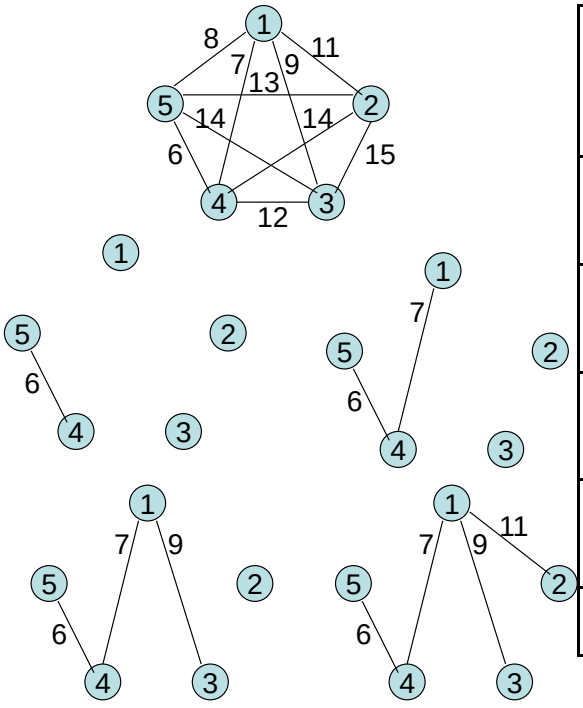
Рис.1. Взвешенный (нагруженный) граф и его некоторые остовные деревья

Отметим, что каждое дерево с n вершинами имеет n –1 ребро.

Для поиска каркаса минимального веса используются: метод Прима и метод Краскала.

Алгоритмы Прима и Краскала гарантируют без проведения проверок, что создана связная сеть без циклов.

- Алгоритм Прима заключается в следующем.
  - Вначале выбираем некоторую вершину *v*, остальные (*n - 1*) вершин графа отмечаются как невыбранные.
  - Определяются веса между выбранной вершиной *v* и остальными невыбранными вершинами.
  - Выбираем вершину с наименьшим весом до нее, фиксируем выбранные ребро и вес.
  - Выбранную вершину исключаем из перечня невыбранных, число не выбранных вершин уменьшаем на 1.
- Все эти шаги повторяем до тех пор, пока не будут выбраны всё вершины, т.е. (*n - 1*) раз. (Рис. 2)



Выбранная вершина	Невыбранные вершины и веса до них	Выбранная дуга и вес
5	4 1 2 3 6 8 13 14-	5->4: 6
4	1 3 2 7 12 14	4->1: 7
1	3 2 9 11	1->3: 9
1	2 11	1->2: 11
2	Выбраны все	

Рис. 2. Пример алгоритма Прима

Сложность алгоритма Прима

Способ представления приоритетной очереди и графа	Асимптотика
Массив d, списки смежности (матрица смежности)	O(V^2)
Бинарная пирамида, списки смежности	O((V+E) log V)=O(E log V)
Фибоначчиева пирамида, списки смежности	O(E+V log V)

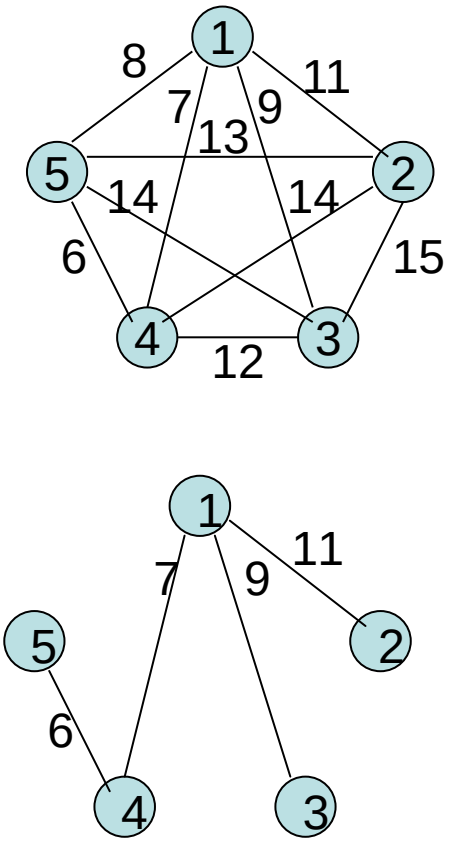
Общее время работы алгоритма Прима составляет
O(V * lg V + E * lg V) = O(E * lg V)

Алгоритм Краскала В отличие от алгоритма Прима, этот алгоритм не требует прохода по всем вершинам для нахождения ребра с минимальным весом. Вместо этого он использует 'жадный' подход. **Р**аботаем с вершинами, а не с ребрами G. Это дает нам n связных компонент. Будем увеличивать их размер по ребру за раз.
Число ребер, необходимое для остовного дерева: n-1.

Результат. Каркас с минимальным суммарным весом

- Алгоритм: Шаг 1. Начать с вполне несвязного графа G, содержащего n вершин.
- Шаг 2. Упорядочить ребра графа G в порядке неубывания их весов.
- Шаг 3. Начав с первого ребра в этом перечне, добавлять ребра в графе G, соблюдая условие: добавление не должно приводить к появлению цикла в G.
- Шаг 4. Повторять шаг 3 до тех пор, пока число ребер в G не станет равным n-1. Получившееся дерево является каркасом минимального веса.

- **Сложность алгоритма составляет O(E*lg E). E-число рёбер.**



Вес	Вершины		выборка
6	5	4	+
7	1	4	+
8	5	1	-
9	1	3	+
11	1	2	+
12	3	4	-
13	5	2	-
14	5	3	-
14	2	4	-
15	2	3	-

Рис. 3. Пример алгоритма Краскала

Другие алгоритмы поиска множества путей и кратчайших путей:

- алгоритм Форда-Беллмана, если нет контуров отрицательного веса, то он находит минимальный путь между двумя заданными вершинами
- алгоритм Дейкстры, если нет отрицательных ребер, то он позволяет найти пути из заданной вершины во все остальные
- алгоритм Флойда позволяет найти кратчайшие пути между всеми вершинами графа
- алгоритм Уоршала позволяет получить путевую матрицу, которая содержит все пути между различными узлами

Вторая задача на нагруженных графах – **поиск минимального потока от источника к стоку.**

Формулировка задачи: определить максимальный поток, протекающий от некоторой вершины S графа (источника) к некоторой вершине T (стоку).

Каждой дуге (граф ориентированный) (i,j) приписана некоторая пропускная способность C(i,j), определяющая

максимальное значение потока, который может протекать по данной дуге.

Задачи о потоках в сетях, относятся как к линейному программированию, так и к разделу дискретной математики - комбинаторике.

Важнейшим свойством многих потоковых задач является их абсолютная целочисленность: при целочисленных исходных данных можно найти целочисленный оптимальный план.

К области приложений потоковых методов относятся

- комбинаторные задачи:
  - о допустимых назначениях на должности (иначе, о представителях подмножеств);
  - о назначениях на должности с максимальной суммарной эффективностью;
  - о назначениях, оптимальных по минимаксу;
  - о минимальных множествах ребер или вершин сети, удаление которых нарушает ее связность.
- каноническая область приложений:
  - задачи о потоках в сетях (электрических, гидравлических и т.д.);
  - составлении расписаний;
  - нахождении оптимальных планов перевозок;
- и другие.

Одним из фундаментальных фактов теории потоков в сетях является классическая теорема о максимальном потоке и минимальном разрезе.

Разрезом называют множество дуг, удаление которых из сети приводит к "разрыву" всех путей, ведущих из s в t.

Пропускная способность разреза - это суммарная пропускная способность дуг, его составляющих.

Разрез с минимальной пропускной способностью называют минимальным разрезом.

Теорема (Форд и Фалкерсон). Величина каждого потока из s в t не превосходит пропускной способности минимального разреза, разделяющего s и t, причем существует поток, достигающий этого значения.

Теорема устанавливает эквивалентность задач нахождения максимального потока и минимального разреза, однако не определяет метода их поиска.

Грубый подход определения максимального потока

Логика поиска :

генерации всех возможных подмножеств дуг;

для каждого подмножества дуг проверяем, является ли оно разрезом.

если является, то вычисляем его пропускную способность и сравниваем ее с минимальным значением.

при положительном результате сравнения запоминаем разрез и изменяем значение минимума.

Очевидно, что даже при наличии различных отсечений в переборе метод применим только для небольших сетей. Однако, как найти максимальный поток, то есть его распределение по дугам, по-прежнему открытый вопрос.

Алгоритм Форда и Фалкерсона

Суть алгоритма- последовательное (итерационное) построение максимального потока путем поиска на каждом шаге пути (последовательности дуг), поток по которому можно увеличить. При этом узлы (вершины графа) специальным образом помечаются.

Алгоритм базируется на "Технике меток" Форда и Фалкерсона:

Суть "Техники меток" Форда и Фалкерсона:

1. На каждой итерации вершины сети могут находиться в одном из трёх состояний:

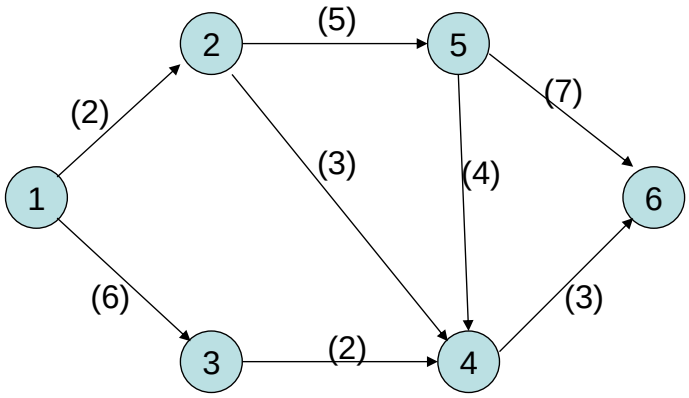
Вершине присвоена метка и она просмотрена;

Вершине присвоена метка и она не просмотрена, т.е. не все смежные с ней вершины обработаны;

Вершина не имеет метки.

2. На каждой итерации мы выбираем помеченную, но не просмотренную вершину V и пытаемся найти вершину U, смежную с V, которую можно пометить. Помеченные вершины образуют множество вершин, достижимых из вершины источника. Если среди этих вершин окажется вершина сток, то найдена цепочка, увеличивающая поток. При неизменности этого множества поток увеличить нельзя.

Пример построения максимального потока в сети



Источник – вершина 1

Сток - вершина 6

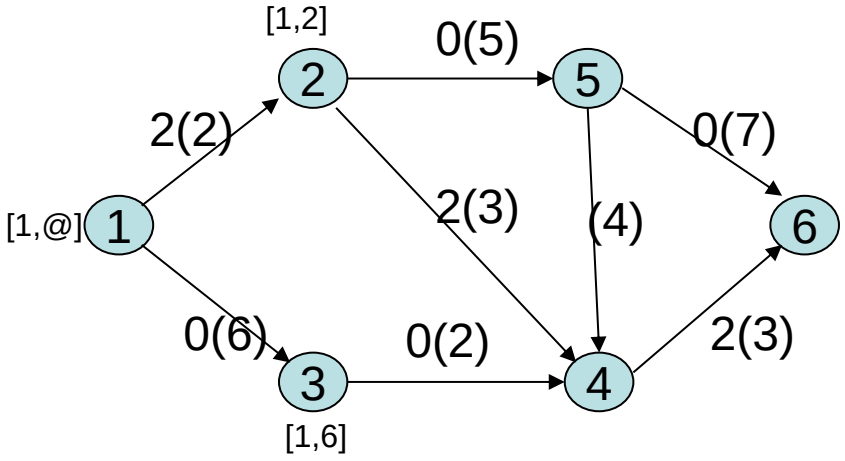
Максимальный поток между этими вершинами – F

Начальное значение F=0

Структурой данных для описания F является матрица C, в которой определены пропускные способности дуг.

Рис. 4. Пример построения максимального потока в сети

Первая итерация.



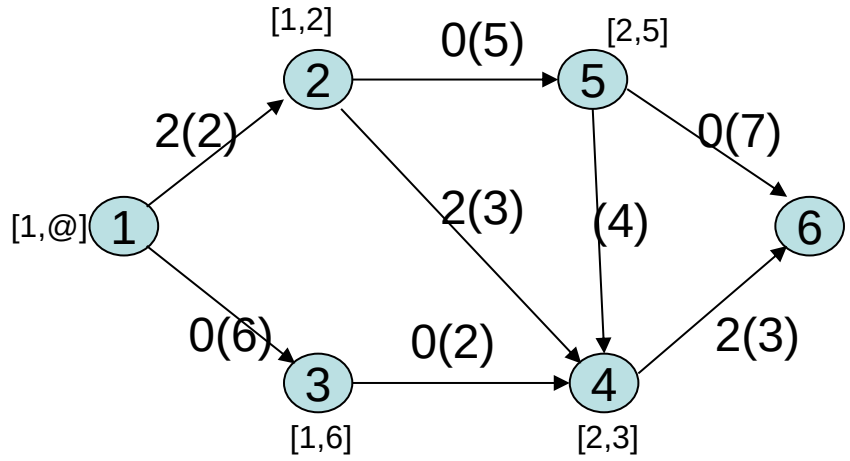
Вершине 1 присваиваем метку [1,@].

Первый шаг.

Рассмотрим дуги, началом которых является вершина 1 - дуги (1,2) и (1,3). Вершины 2 и 3 не помечены, поэтому присваиваем им метки для 2-й - [1,2], 3-й - [1,6].

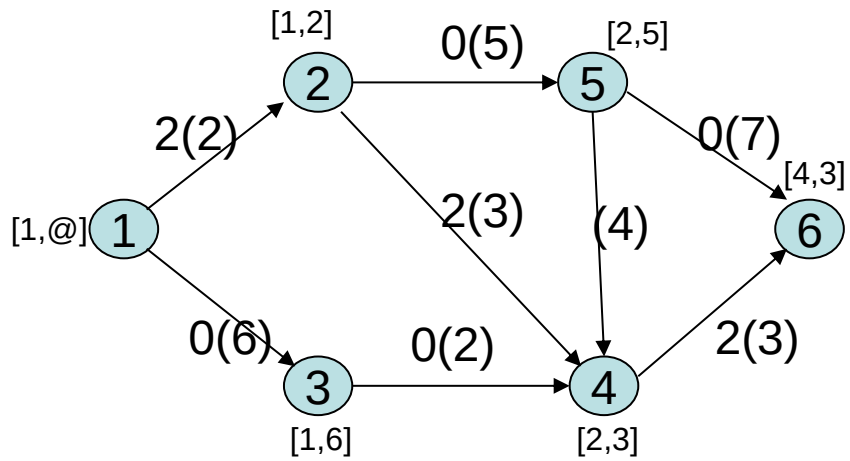
Первая цифра - номер вершины, из которой идет поток, вторая цифра - численное значение потока, который можно передать по этой дуге.

Второй шаг



Выберем помеченную, но не просмотренную вершину. Первой в соответствующей структуре данных написана вершина 2.
Рассмотрим дуги, для которых она является началом - дуги (2,4) и (2,5). Вершины 4 и 5 не помечены. Присвоим им метки - [2,3] и [2,5]. Итак, на втором шаге вершина 2 просмотрена, вершины 3, 4, 5 помечены, по не просмотрены, остальные вершины не помечены.

Третий шаг.



Выбираем из помеченных вершину 3. Рассмотрим дугу (3,4). Вершина 4 уже помечена.

Переходим к следующей вершине - четвертой, соответствующая дуга -(4,6). Вершина 6 не помечена. Присваиваем ей метку [4,3].

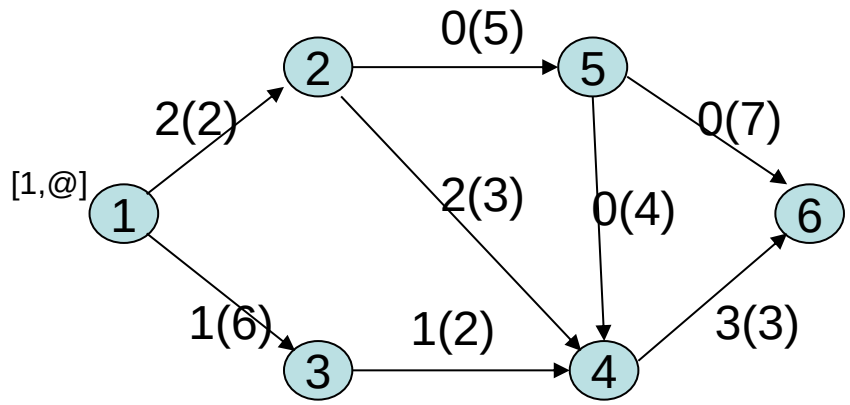
Мы достигли вершины-стока, тем самым найдя путь (последовательность дуг), поток по которым можно увеличить. Информация об этом пути содержат метки вершин. В данном случае путь или увеличивающаяся цепочка $1 \rightarrow 2 \rightarrow 4 \rightarrow 6$. Максимально возможный поток, который можно передать по дугам этого пути, определяется второй цифрой метки вершина стока, то есть 2.

Поток в сети стал равным 2.

Вторая итерация.

Вершине 1 присваиваем метку [1,@].

Первый шаг

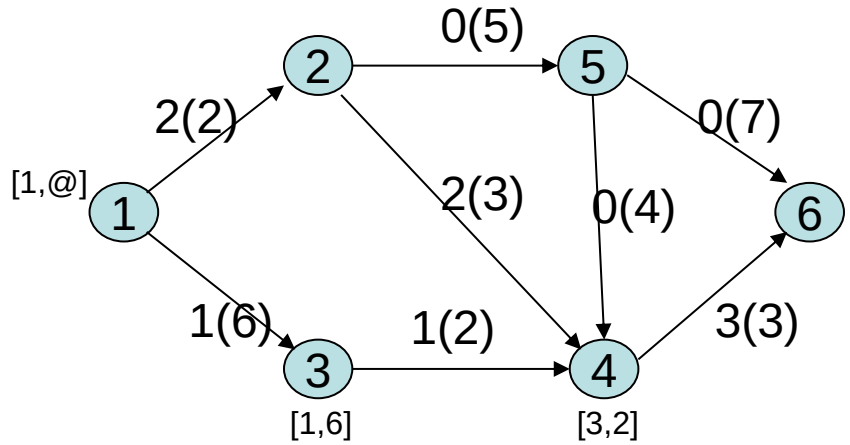


Рассмотрим дуги, началом которых является помеченная вершина 1 это дуги (1,2) и (1,3), Вершина 2 не может быть помечена, так как пропускная способность дуги (1,2) исчерпана.

Вершине 3 присваиваем метку [1,6].

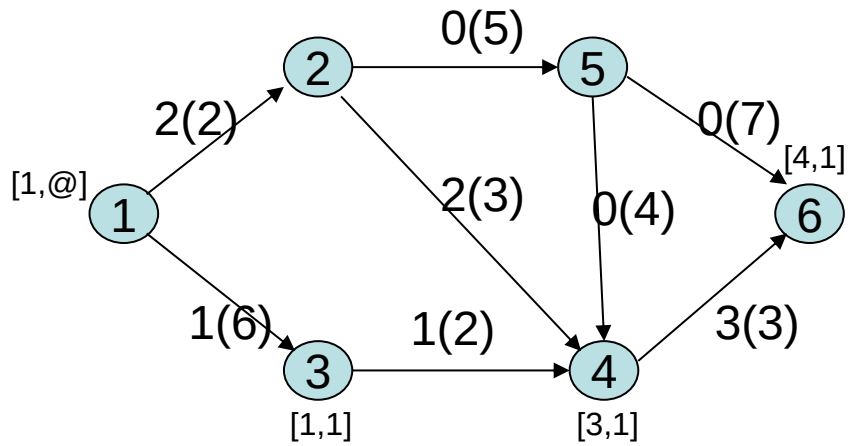
Второй шаг

Выберем помеченную, но не просмотренную вершину. Это вершина 3.



Повторяем действия. В результате вершина 4 получает метку [3,2].

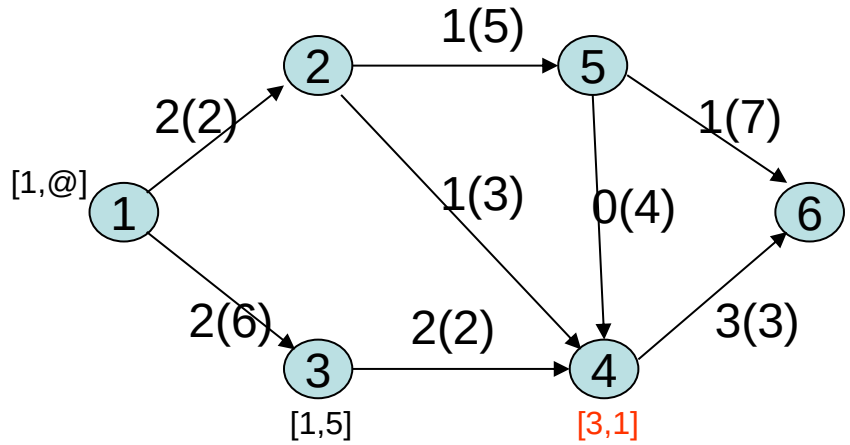
Третий шаг



Выбираем вершину 4. Только она помечена и не просмотрена. Вершине 6 присваиваем метку [4,1]. Почему только одна единица потока? На предыдущей итерации израсходованы две единицы пропускной способности данной дуги, осталась только одна.

Вершина - сток достигнута. Найдена увеличивающая поток цепочка, это 1→3→4→6, по которой можно "протащить" единичный поток. Результирующий поток в сети F равен 3.

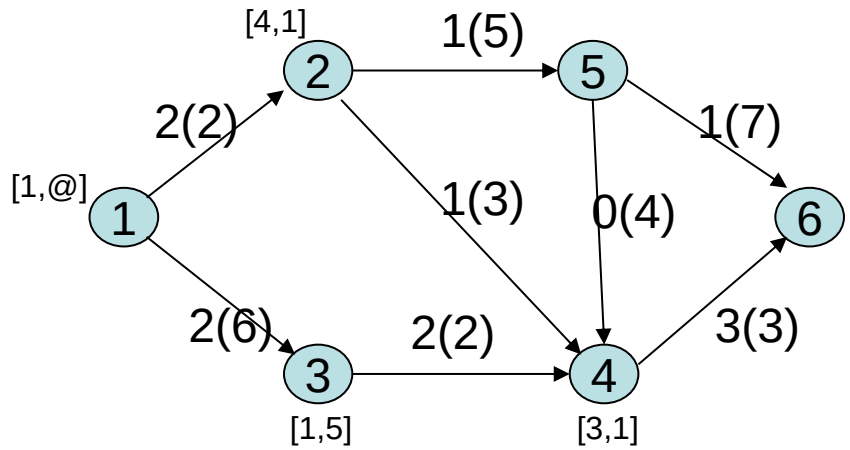
<u>Третья итерация</u>



Вершине 1 присваиваем метку [1,@]

Первый шаг. Результат - метка [1.5] у вершины 3. (5-оставшаяся часть пропускной способности).

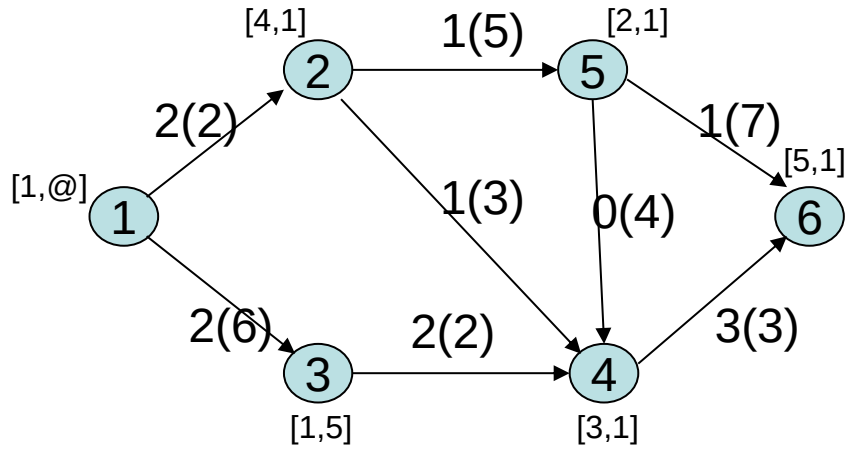
Второй шаг - метка [3.1] у вершины 4.
Третий шаг



Пропускная способность дуги(4,6) израсходована полностью. Однако есть *обратная* дуга (2,4), по которой передается *поток, не равный нулю* ("изюминка" метода). Попробуем перераспределить поток. Нам необходимо передать из вершины 4 поток, равный единице (зафиксирован в метке вершины). Задержим единицу потока в вершине 2, то есть вернем единицу потока из вершины 4 в вершину 2. Эту особенность зафиксируем в метке вершины 2 - [4,1]. Тогда единицу потока из вершины 4 мы передадим по сети вместо той, которая задержана в вершине 2, а единицу потока из вершины 2 попытаемся "протолкнуть" по сети, используя другие дуги. И так,

вершина 4 просмотрена, вершина 2 помечена, вершины 5 и 6 не помечены.

Четвертый и пятый шаги

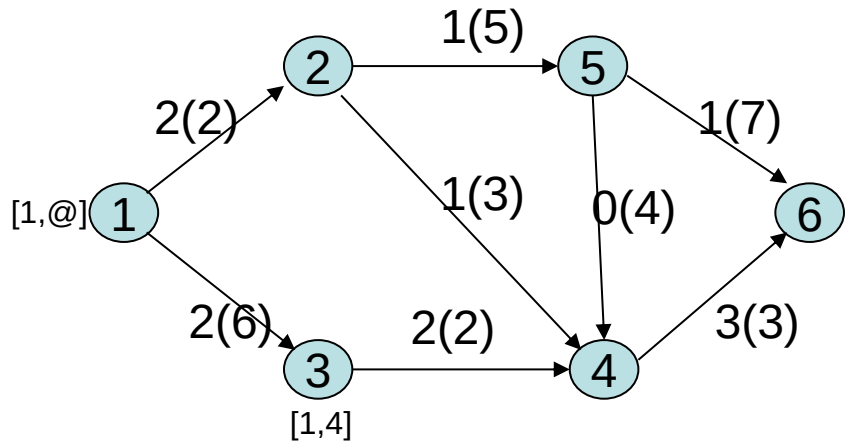


Четвертый и пятый шаги очевидны. Передаем единицу потока из вершины 2 в вершину 6 через вершину 5.

Вершина - сток достигнута, найдена цепочка 1→3→4→2→5→6, по которой можно передать поток, равный единице. При этом по прямым дугам поток увеличивается на единицу, по обратным - уменьшается.

Суммарный поток в сети - 4 единицы

<u>Четвёртая итерация</u>



Вершине 1 присваиваем метку [1,@].

Первый шаг. Помечаем вершину 3 - [1,4].

Второй шаг. Рассматриваем помеченную, по не просмотренную вершину 3. Одна дуга - (3.4).

Вершину 4 пометить не можем - пропускная способность дуги исчерпана. Помеченных вершин больше нет, и вершина-сток не достигнута. Увеличивающую поток цепочку построить не можем.

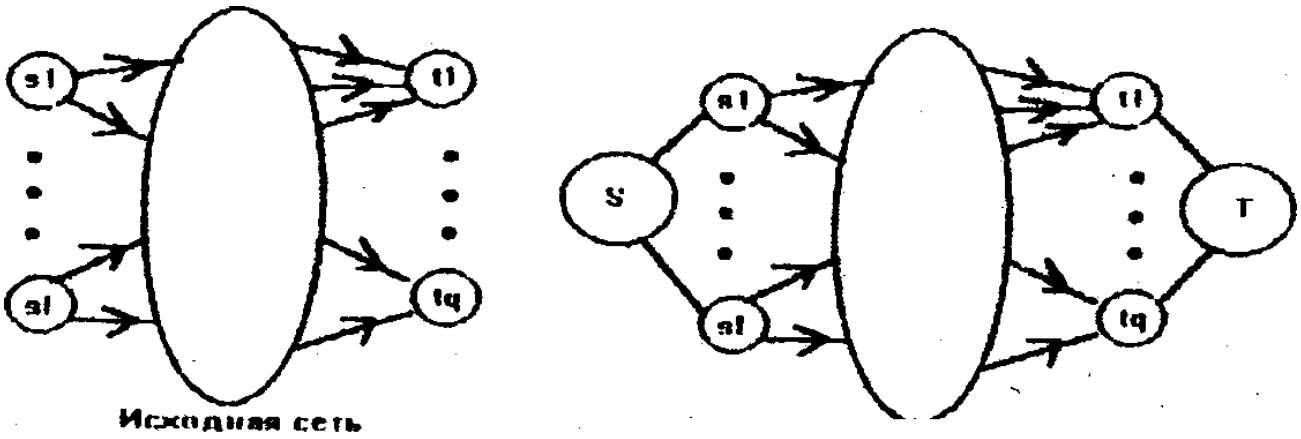
Найден максимальный поток в сети. Можно заканчивать работу
Задачи для самостоятельной работы:

- Рассмотреть предыдущий пример если изменится нумерация узлов
- Рассмотреть предыдущий пример если изменится пропускная способность некоторых рёбер
- Рассмотреть алгоритм на примере другого графа

Варианты задачи о максимальном потоке

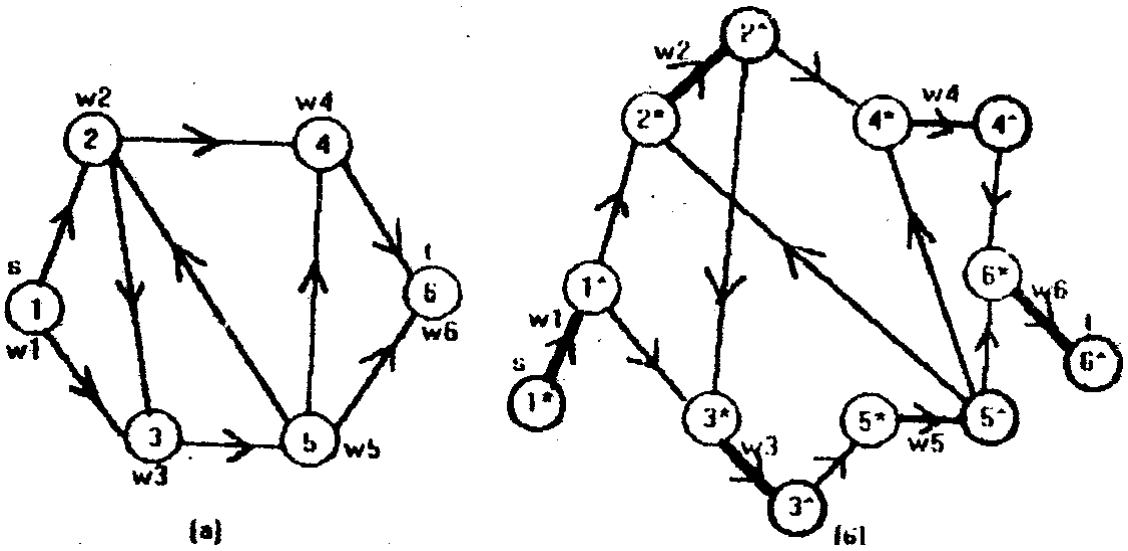
Существуют различные постановки потоковых задач, которые можно свести к решению методом Форда и Фалкерсона.

Сети с многими источниками и стоками.



необходимо ввести две новые вершины - главный источник S и главный сток T. Соединим главный источник S со всеми источниками сети s1,s2. ..., s1 дугами (S,s1), (S,s2), ..., (S,sl), считая пропускную способность каждой из них неограниченной, а стоки исходной сети t1, t2,..., tq с главным стоком T дугами (t1,T), (t2,T), ...,(tq.T) также с неограниченной пропускной способностью
Известно что максимальный поток в расширенной сети соответствует максимальному потоку в исходной сети.

Сети с пропускными способностями дуг и вершин.

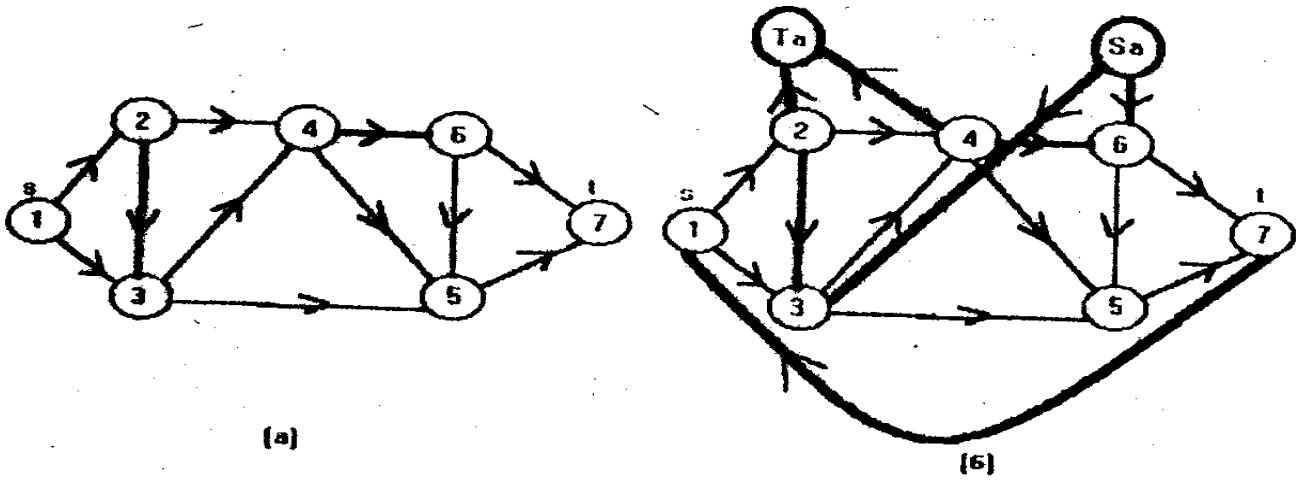


В сети дуги и вершины имеют пропускные способности. Для вершин они обозначены wi Требуется найти максимальный поток между вершинами s и t. Определим сеть G* так, чтобы каждая вершина j сети G преобразовывалась в две вершины j* и j^ сети G*. Каждой дуге типа (i,j) сети G ставится в соответствие дуга (i^, j*) из G*, а дуге типа (j,k) - дуга (j^,k*).

Кроме того, введем дуги (j*,j^) (выделены "жирными" линиями на рис. с пропускной способностью wj, то есть равной пропускной способности вершины j сети G. Так как полный поток, входящий в вершину j*, обязательно должен протекать по дуге (j*,j^) с пропускной способностью wj, т.е. максимальный поток в сети G с пропускными способностями дуг и вершин равен максимальному потоку в сети G*, имеющей только пропускные способности дуг. Отметим, что если минимальный разрез в G* не содержит дуг типа (j*,j^), то пропускные способности вершин G пассивны, в противном случае, имеются насыщенные потоком вершины. Схема применения алгоритма Форда - Фалкерсона для сетей этого типа определена

Сеть с ограничением пропускной способности дуг сверху и снизу.

Для сетей этого типа определены нижние границы потока R[i,j] для каждой дуги (i,j), то есть поток по дуге должен превышать или совпадать с величиной R[i,j]. На рис. (а) "жирными" линиями выделены дуги, для которых значение R[i,j] больше нуля. Требуется найти максимальный поток между вершинами s и t.



Введем искусственные источник Sa и сток Ta. Определим дуги (i,j) с ненулевым значением R[i,j]. Введем дополнительные дуги (Sa, j) и (i,Ta) с пропускными способностями R[i,j] и с нижними границами, равными нулю. Уменьшим пропускную способность дуги (i,j) до значения C[i,j]-R[i,j], а R[i,j] - до нуля. Кроме того, введем дугу (t,s) с пропускной способностью C[t,s]=R[t,s]=0.

После этих преобразований найдем максимальный поток F между вершинами Sa и Ta.

Если значение максимального потока между вершинами Sa и Ta равно суммарному значению нижних границ потоков R[i,j] исходной сети G, то есть, если все дуги выходящие из Sa и входящие в Ta, насыщены, то в сети G существует допустимый поток со значением F, иначе (если поток не равен сумме нижних границ) в сети G не существует никакого допустимого потока.

Предположим, что максимальный поток в сети G* от Sa к Ta равен сумме значений всех нижних границ дуг сети G. Вычтем поток, равный R[i,j] из потока на дугах (i,Ta) и (Sa,j). Чтобы не изменить поток, идущий по дугам исходной сети, добавим поток, равный R[i,j] единицам, к потоку на дуге (i,j). Почему именно так? Вычитая поток по дуге (i,Ta), мы тем самым задерживаем поток, равный R[i,j], в вершине i, и должны передать его дальше. Вычитание потока R[i,j] из потока по дуге (Sa,j) соответствует необходимости "протаскивания" этого потока в вершину j. Таким образом, чтобы не изменить поток по дугам исходной сети, увеличим поток по дуге (i,j) на значение R[i,j]. Выполнив это преобразование для всех дуг, выходящих из Sa и входящих в Ta, мы перераспределим поток d сети G* таким образом, что в вершинах Sa

и Та отпадет необходимость (потоки на дугах, связанных с этими вершинами, равны нулю). Если поток по дуге (t,s) будет равен значению максимальною потока F сети G*, то, исключив введенные дуги и вершины, мы получаем исходную сеть G, в которой циркулирует поток со значением F. При этом значение потока на всех дугах (i,j) с ненулевой нижней границей R[i,j] принадлежит интервалу (R[i,j],C[i,j]).

Максимальный поток между каждой парой вершин неориентированной сети

Использование алгоритма Форда - Фалкерсона для каждой пары вершин решает проблему. Это N*(N-1)/2 "отдельных" задач.

Однако существует более эффективный алгоритм Гомори и Ху.

Алгоритм Эдмондса — Карпа решает задачу нахождения максимального потока в транспортной сети. Алгоритм представляет собой частный случай метода Форда — Фалкерсона и работает за время $O(VE^2)$. Впервые был опубликован в 1970 году советским учёным Е. А. Диницом. Позже, в 1972 году, был независимо открыт Эдмондсом и Карпом.

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.

Обнуляем все потоки. Остаточная сеть изначально совпадает с исходной сетью.

В остаточной сети находим *кратчайший* путь из источника в сток. Если такого пути нет, останавливаемся.

Пускаем через найденный путь (он называется **увеличивающим путём** или **увеличивающей цепью**) максимально возможный поток:

На найденном пути в остаточной сети ищем ребро с минимальной пропускной способностью c_{min}.

Для каждого ребра на найденном пути увеличиваем поток на c_{min}, а в противоположном ему - уменьшаем на c_{min}.

Модифицируем остаточную сеть. Для всех рёбер на найденном пути, а также для противоположных им рёбер, вычисляем новую пропускную способность. Если она стала ненулевой, добавляем ребро к остаточной сети, а если обнулилась, стираем его.

Возвращаемся на шаг 2.

- Чтобы найти кратчайший путь в графе, используем поиск в ширину:
- **Сложность**
- В процессе работы алгоритм Эдмондса — Карпа будет находить каждый дополняющий путь за время $O(V + E)$. Можно доказать, что общее число увеличений потока, выполняемое данным алгоритмом, составляет $O(VE)$. Таким образом, сложность алгоритма Эдмондса — Карпа равна $O(VE^2)$.

– Улучшенной версией алгоритма Эдмондса-Карпа является алгоритм Диница, требующий $O(|V|^2 |E|)$ операций.

– Назовём **вспомогательной бесконтурной сетью** графа G с источником s его подграф, содержащий только такие рёбра (u, v), для которых минимальное расстояние от s до v на единицу больше минимального расстояния от s до u.

– Строим минимальную бесконтурную сеть остаточного графа.

Пока в сети есть путь из s в t, выполнить следующие шаги:
Находим кратчайший путь из s в t. Если его нет, выйти из цикла.
На найденном пути в остаточной сети ищем ребро с минимальной пропускной способностью c_{min}.
Для каждого ребра на найденном пути увеличиваем поток на c_{min}, а в противоположном ему - уменьшаем на c_{min}.
Удаляем все рёбра, достигшие насыщения.
Удаляем все тупики (то есть вершины, кроме стока, откуда не выходит рёбер, и вершины, кроме источника, куда рёбер не входит) вместе со всеми инцидентными им рёбрами.
Повторяем предыдущий шаг, пока есть что удалять.
Если найденный поток ненулевой, добавляем его к общему потоку и возвращаемся на шаг 1.

Алгоритмы нахождения максимального потока
n — число вершин, m — число рёбер, U — наибольшая величина максимальной пропускной способности сети.

- Алгоритм Форда — Фалкерсона (1956) — .
- Алгоритм Эдмондса — Карпа, кратчайших увеличивающихся цепей (1969) — .
- Алгоритм Диница (1970) — .

- Алгоритм Эдмондса — Карпа, локально-максимального увеличения (1972) — .
- Алгоритм Диница 2 (1973) — .
- Алгоритм Карзанова (1974) — .
- Алгоритм Черкаского (1977) — .
- Алгоритм Малхотры — Кумара — Махешвари (1977) — .
- Алгоритм Галила (1980) — .
- Алгоритм Галила — Наамада (1980) — .
- Алгоритм Слейтора — Тарьяна (1983) — .
- Алгоритм Габоу (1985) — .
- Алгоритм Голдберга — Тарьяна (1988) — .
- Алгоритм Ахьюа — Орлина (1989) — .
- Алгоритм Ахьюа — Орлина — Тарьяна (1989) — .
- Алгоритм Кинга — Рао — Тарьяна 1 (1992) — .
- Алгоритм Кинга — Рао — Тарьяна 2 (1994) — .
- Алгоритм Черияна — Хейджрапа — Мехлхорна (1996) — .
- Алгоритм Голдберга — Рао (1998) — .
- Алгоритм Кёлнера — Мондры — Спилмана — Тена (2010) — .
- Алгоритм Орлина 1 (2012) — .
- Алгоритм Орлина 2 (2012) — .

3. ТЕХНИКА БЕЗОПАСНОСТИ ПРИ ВЫПОЛНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ

Меры безопасности при работе с электротехническими устройствами соответствуют мерам безопасности, принимаемым при эксплуатации установок с напряжением до 1000 В и разработанным в соответствии с «Правилами техники безопасности при эксплуатации электроустановок потребителей», утвержденными Главгосэнерго-надзором 21 декабря 1984 г.

Студенту не разрешается приступать к работе, если замечены какие-либо неисправности в лабораторном оборудовании.

Студент не должен прикасаться к токоведущим элементам электрооборудования, освещения и электропроводке, открывать двери электрошкафов и корпусов системных блоков, мониторов.

Студенту запрещается прикасаться к неизолированным или поврежденным проводам и электрическим устройствам, наступать

на переносные электрические провода, лежащие на полу, самостоятельно ремонтировать электрооборудование и инструмент.

Обо всех замеченных неисправностях электрооборудования студент должен немедленно поставить в известность преподавателя или лаборанта.

При выполнении всех работ необходимо соблюдать осторожность и помнить, что только человек, относящийся серьезно к своей безопасности, может быть застрахован от несчастного случая.

4. ЗАДАЧИ ДЛЯ САМОСТОЯТЕЛЬНОГО ВЫПОЛНЕНИЯ

1. В космическом центре проектируется система организации связи между спутниками, действующая в реальном масштабе времени и расчитанная на обслуживание не менее 1000 спутников. Каждый спутник может иметь либо не иметь как радиостанцию, так и приемник радиосигналов. Сигналы с одного спутника могут передаваться на другие через произвольное число промежуточных спутников. В тот момент, когда два спутника появляются в зоне радиовидимости и или выходят из такой зоны, на Землю поступает соответствующая информация. Промоделировать текущую обработку информации со спутников. Обеспечить оперативный ответ на запрос о способах взаимной связи двух заданных спутников с минимальным числом промежуточных звеньев.

2. Пусть G= (V,E) - неориентированный связный граф. Предположим, что имеется только одна вершина, не имеющая заходящих в нее ребер, и одна вершина, не имеющая исходящих из нее ребер. Разработать алгоритм и программу для отыскания разреза между этими вершинами, который имеет наименьшее число ребер.

3. Имеются расписания вылетов самолетов в ряде аэропортов. Требуется по начальному и конечному пунктам предложить маршрут с возможными пересадками, оптимальной по одному из следующих критериев:

а) минимальное число пересадок;

б) наименьшее время в пути, включая время ожидания в аэропортах.

При выборе маршрута считать, что пересадка допустима, если интервал времени между прилетом самолета и последующим вылетом составляет не менее двух часов.

4. Проект научно-технической программы задан с помощью ориентированного графа. Вершина графа соответствует отдельному исследованию, а дуги показывают очередность выполнения исследований (каждое исследование может начаться не ранее окончания предшествующих исследований). Продолжительность каждого исследования известна. Задан конечный срок выполнения проекта. требуется определить максимальный интервал времени для проведения каждого исследования, чтобы проект мог быть завершен к назначенному сроку.

5. Информация о некотором изделии задана с помощью ориентированного графа. Вершина графа соответствует отдельной операции, а дуги показывают очередность выполнения операций (каждая операция может начаться не ранее окончания предшествующих операций). Продолжительность каждой операции известна. Имеется один станок для выполнения каждого типа операций. Если станок освобождается, то производится, то производится та из готовых к выполнению операций, для которой наибольшая суммарная трудоемкость на путях, начинающихся от соответствующей вершины, максимальна. Определить время изготовления изделия. Выдать для каждого станка информацию о последовательности операций, выполненных на данном станке.

6. Имеются расписания вылетов самолетов в ряде аэропортов. Требуется по начальному и конечному пунктам предложить маршрут с возможными пересадками, оптимальный по одному из следующих критериев:

а) наименьшая суммарная стоимость билетов;

б) минимальное суммарное расстояние.

При выборе маршрута считать, что пересадка допустима, если интервал времени между прилетом самолета и последующем вылетом составляет не менее 2 часов.

7. Проект научно-технической программы задан с помощью ориентированного графа. Вершина графа соответствует отдельному исследованию, а дуги показывают очередность выполнения исследований (каждое исследование может начаться не ранее окончания предшествующих исследований). Продолжительность

каждого исследования известна. Требуется определить минимально возможное время выполнения программы.

8. Имеется информация о торговых связях предпринимателей в виде множества пар (A, B) , где A – продавец, а B – покупатель. Покупатели сами являются продавцами и перепродают закупленный товар в соответствии со своими связями с наценкой 20%. Задан список предпринимателей, получивших некоторый товар. Определить такой способ получения этого товара указанным предпринимателем, чтобы наценка была минимальной.

9. Имеется сеть автомобильных дорог. Известны расстояния всех участков дорог. Некоторые участки аварийно опасны. Требуется найти путь из пункта A в пункт B, имеющий наименьшую суммарную протяженность аварийно опасных участков.

10. Имеется сеть автомобильных дорог, связывающих населенные пункты некоторого района. Известны длина участков дорог между пунктами. В M–пунктах производят пряжу, а в N – перерабатывают пряжу на фабриках. Решено закрыть K ($K<N$) фабрик переработки, наиболее удаленных от источников сырья. Требуется:

а) получить список пунктов, где закрываются фабрики;

б) для каждой из оставшихся фабрик выдать S ($S<M$) ближайших пунктов, где производят пряжу.

11. Имеется сеть железных дорог, связывающая различные города. Города принадлежат разным суверенным государствам. Каждое государство взимает значительную пошлину за въезд в его пределы. В некоторых городах берут дополнительный налог. Требуется найти такой путь из пункта A в пункт B, чтобы:

а) не встречались города, взимающие налог;

б) число пересечений государственных границ было минимальным.

12. В файле записаны предложения по обмену жилплощадью в пределах Брянска. Имеются варианты размена одной квартиры на две других либо на квартиру и комнату. Требуется по заявке клиента предложить способы обмена. Предусмотреть возможность нахождения обменов, в которых участвуют более двух сторон.

13. Написать программу, которая по матрице смежности и двум узлам графа вычисляет:

– число путей данной длины между данными двумя узлами;

– длину кратчайшего пути между данными двумя узлами.

14. Написать программу, получающую следующую входную информацию об электронной схеме:

а) N – число проводов в сети;

б) величина тока, поступающая по первому проводу и выходящая по N-ому проводу;

в) сопротивление каждого провода (со 2-ого до N–1-го);

г) множество упорядоченных пар (I, J), показывающего, что провод I соединен с проводом J и что ток идет через провод I к проводу J.

Программа должна вычислить по законам Ома и Кирхгофа величину тока, протекающего через каждый провод (со 2-го до N–1-ого). Закон Кирхгофа утверждает, что сумма величин токов, входящих в узел, равна сумме величин токов, выходящих из узла. По закону Ома, если между двумя узлами существует два пути, то сумма произведений величин токов на каждом проводе на величины сопротивлений этого провода по всем проводам первого пути равна аналогичному произведению по второму пути.

15. Стоком простого орграфа G= (V,E) называется вершина с $|V|$–1 заходящими в нее ребрами и не имеющая ребер, из нее исходящих. Найдите алгоритм, который при заданной матрице смежности орграфа G определяет за O ($|V|$) битовых проверок, содержит ли орграф G сток.

16. Разработать программу поиска минимального остовного дерева с помощью алгоритмов Прима и Краскала. Оценить эффективность программ на произвольных графах.

17. Рассмотрим транспортную сеть, включающую мосты. Будем предполагать, что при превышении, грузоподъемности некоторого моста последний разрушается. Определить максимальный вес груза. который может быть транспортирован в рассматриваемой сети из пункта A в пункт B без превышения грузоподъемности находящихся на маршруте движения транспорта мостов. Указание. Если поставить в соответствие мостам дуги определенного графа и задать длины дуг равными соответствующим ограничениям на нагрузку моста, то задача сводится к поиску такою пути между вершинами A и B, в котором длина кратчайшей дуги максимальна.

18. Фирма, являющаяся основным производителям погрузочно-

разгрузочного оборудования, проектирует высокоскоростную систему перевозки пассажиров между шестью различными пунктами аэропорта Шереметьево. Значение каждого элемента матрицы равно расстоянию между соответствующей парой пунктов, вычисленному по кратчайшему пути, соединяющему их.

Инструкция по технике безопасности не позволяет, чтобы пути высокоскоростной системы пересекались. Найти длину пути минимальной стоимости, имеющего форму цикла, проходящего через все пункты. Стоимость одной единицы расстояния постоянна и равна 10000 рублей.

	1	2	3	4	•5	б
1		870	910	796	860	712
2	870		1100	560	430	630
3	900	1100		270	600	250
4	796	560	270		300	850
5	850	430	600	300		370
6	712	630	180	850	378	

19. Составление расписания движения транспортных средств. Из пункта *s* в пункт *t* транспортной сети необходимо перегнать транспортное средство за время, не больше заданного, при этом желательно существенно снизить затраты на эту операцию. Между промежуточными пунктами возможна платная перевозка попутного груза, при этом расходы на передвижение снижаются и даже могут стать отрицательными. Указание. Каждому пункту сопоставим вершину графа, а каждой паре пунктов – дугу (x,y), если существует попутный груз из x в y. Каждой дуге будут соответствовать два числа: длина $C[x,y]$ и вес $C[x,y]$. Длина дуги (x,y) равна времени перехода из x в y. а вес дуги – расходам, соотнесенным к соответствующему перегону. Очевидно, что при наличии отрицательного цикла величина кратчайшей повесу (s,t) – цепи может быть сколько угодно малой. Но ограничение на время передвижения позволяет найти оптимальное расписание движения транспорта. Искомое оптимальное расписание соответствует допустимой кратчайшей (s,t) цепи. Эта задача теории графов – экспоненциальное время решения и общем случае [20], но может быть решена за полиномиальное время при равенстве длин дуг.

20. Анализ пропускной способности коммуникационной сети. Коммуникационная сеть моделируется сетью (ориентированной или

неориентированной). Каждой дуге (ребру) сопоставляется число, равное пропускной способности соответствующей коммуникации (линия связи, транспортная магистраль, участок водопровода и т. п.). Необходимо определить пропускную способность между двумя выделенными вершинами. Указание. Решение задачи о максимальном потоке является решением поставленной прикладной задачи.

21. Построение кратчайшей коммуникационной сети. На территории размещены пункты, которые надо связать коммуникационной сетью минимальной стоимости. Предположим, что в результате изысканий определены возможные трассы для прокладки коммуникаций и оценена стоимость их создания для каждой трассы. Сопоставим каждому пункту сети вершину графа $G=(X,U)$. Две вершины x, y будут инцидентны некоторому ребру u, если соответствующие пункты соединены возможной трассой коммуникации. Ребру и графа $G=(X,U)$ сопоставим число $c(u)$, равное стоимости строительства соответствующей коммуникации. Указание. Оптимальная структура коммуникационной сети будет соответ-ствовать кратчайшему каркасу во взвешенном графе $G=(X,U)$.

*22. Коммивояжер. Есть N городов и известна стоимость проезда из всех городов во все остальные. Начиная с некоторого города, необходимо объехать все города и вернуться назад так, чтобы стоимость проезда была минимальной.

Исходные данные программы:

- $50 \le N \le 200$;
- даны координаты N точек на плоскости - этого города;
- стоимость проезда из города A в город $Б$ – это расстояние между точками A и $Б$.

23. Задан граф из N вершин, число C и две вершины графа $v1$ и $v2$. Требуется определить, существует ли в заданном графе путь из вершины $v1$ в вершину $v2$ длиной в C ребер (путь может иметь самопересечения как по ребрам, так и по вершинам) и найти минимум функции $|X–C|$, где X – длина некоторого пути из $v1$ в $v2$.

Исходные данные расположены в ASCII – файле, имя которого вводится с клавиатуры. Первая строка исходного файла содержит N –число вершин в графе, далее расположены строки матрицы $N*N$ из нулей и единиц, элемент матрицы (i,j) содержит единицу, если в

графе есть ребро и нуль, если нет ребра. Строка N+2 содержит номера вершин $v1$ и $v2$. Строка N+3 содержит десятичную запись числа C.

Технические ограничения:

• N – натуральное число N<10;

• C – натуральное число C< 10^50;

• исходные данные корректны и их проверка не требуется;

• граф может содержать ребра, идущие из вершин в самих себя.

24. Вы победили в соревновании, организованном Канадскими авиалиниями. Приз – бесплатное путешествие по Канаде. Путешествие начинается с самого западного города, в который летают самолеты, проходит с запада на восток, пока не достигнет самого восточного города, куда летают самолеты. Затем путешествие продолжается обратно с востока на запад, пока не достигнет начального города. Ни один из городов нельзя посещать более одного раза за исключением начального города, который надо посетить ровно дважды (в начале и в конце путешествия). Вам также нельзя пользоваться авиалиниями других компаний или другими способами передвижения. Задача состоит в следующем: дан список городов и список прямых рейсов между парами городов: найти маршрут, включающий максимальное количество городов и удовлетворяющий вышеназванным условиям.

Несколько наборов входных данных помещены в ASCII – файле. Каждый набор содержит:

• в первой строке – количество городов N, в которые летают самолеты, и количество прямых рейсов V, N – положительное целое число, меньшее или равное 100. V – положительное целое число;

• в каждой из следующих N строк – название города, в который летают самолеты. Названия упорядочены с запада на восток, то есть i-й по порядку город находится восточнее j-го тогда и только тогда, когда $i>j$ (не существует городов на одном меридиане). Название каждого города – строка, состоящая из <= 15 цифр и/или латинских букв:

• в каждой из следующих V строк – названия двух городов из списка городов, разделенные пробелом. Если пара ГОРОД1 ГОРОД2 содержится в строке, что означает, что есть прямой рейс из ГОРОД1 в ГОРОД2, а также прямой рейс из ГОРОД2 а ГОРОД1. Входные данных корректны, и их проверка не требуется.

Решения, полученные для каждого набора входных данных, должны быть последовательно записаны в выходной ASCII-файл следующим образом: в первой строке – общее количество городов из входного файла; в следующей строке – число *M*, равное количеству различных городов, посещаемых на маршруте; в следующих *M*+1 строках – названия городов по одному в строке в порядке их посещения.

Обратите внимание, что исходный город должен быть также выведен последним. Если для набора входных данных не существует решения, то для этого набора в выходном файле должны быть только две записи: общее количество городов и сообщение «NO SOLUTION» (нет решения).

*25. На остановке останавливаются автобусы одного или нескольких маршрутов. Человек пришел на автобусную остановку в 12:00 и находился на ней до 12:59. За это время он записал времена прибытия автобусов. Эти времена являются исходными данными.

Автобусы одного маршрута прибывают с равномерным интервалом (через одинаковые промежутки времени) с 12:00 до 12:59. Время задается в минутах целыми числами с 12:00 до 12:59.

В указанный период останавливались по крайний мере два автобуса каждого маршрута.

Количество маршрутов в тестовых примерах ≤ 17.

Несколько автобусных маршрутов могут иметь одинаковое время прибытия и (или) одинаковые интервалы.

Напишите программу, которая определяет наименьшее количество автобусных маршрутов, проходящих через данную остановку, и определяет график движения автобусов по этим маршрутам.

Исходные данные расположены во входном файле с именем INPUT.TXT, который содержит число N ($N \leq 300$) – количество прибывших автобусов, записанных человеком. Затем следует строка с временами прибытия автобусов (в минутах). Пример:

17
0 3 5 13 13 15 21 26 27 29 37 39 39 45 51 52 53

Вывести в выходной файл с именем OUTPUT.TXT график движения автобусов по маршрутам. Каждая строка файла содержит данные для одного маршрута. Маршрут определяется временем прибытия первого автобуса с интервалом времени движения,

заданным в минутах. Порядок расположения маршрутов не важен. В случае нескольких возможных решений, выдать только один из возможных вариантов ответа. Для нашего примера: 0 13, 3 12, 5 8

*26. (автор Ю.Корженевич). На острове Новой Демократии каждый из ее жителей организовал партию, которую сам и возглавил. К всеобщему удивлению, даже в самой малочисленной партии оказалось не менее двух человек. К сожалению, финансовые трудности не позволили создать парламент, куда вошли бы, как предполагалось по конституции острова, президенты всех партий.

Посовещавшись, островитяне решили, что будет достаточно, если в парламенте будет хотя бы один член каждой партии.

Помогите островитянам организовать такой, как можно более малочисленный парламент, в котором будут представлены члены всех партий.

Исходные данные: каждая партия и ее президент имеют один и тот же порядковый номер от 1 до N ($4 \le N \le 150$). Вам даны списки всех N партий острова Новой Демократии. Вывести предлагаемый вами парламент в виде списка номеров его членов.

Например, для четырех партий:

Президенты	Члены партий
1	2, 3, 4
2	3
3	1, 4, 2
4	2

Список членов парламента: 2 состоит из одного члена).

Примечание. Получите список одного, как можно более малочисленного, на ваш взгляд, парламента.

27. Туристическая группа из тридцати человек должна быть размещена в гостинице. Каждый номер гостиницы имеет две двуспальные кровати. Администратор гостиницы желает распределить гостей в возможно меньшем числе номеров таким образом, чтобы в одном номере не находились лица противоположного пола, не связанные родственными узами. Допускаются сочетания соседей по номеру типа муж – жена, отец – дочь, брат – сестра. Каким образом администратору гостиницы решить стоящую перед ним задачу?

28. ООН субсидирует программы сотрудничества городов – побратимов: согласно этим программам, города разделяются на

пары для осуществления между ними обмена в области культуры и образования.

	1	2	3	4	5	6	7	8	9	10
1	**0**	**80**	**70**	**70**	**60**	**45**	**90**	**110**	**85**	**155**
2		**0**	**75**	**95**	**90**	**80**	**90**	**160**	**70**	**45**
3			**0**	**65**	**70**	**60**	**100**	**80**	**80**	**55**
4				**0**	**80**	**80**	**70**	**170**	**200**	**250**
5					**0**	**110**	**170**	**190**	**270**	**300**
6						**0**	**100**	**150**	**110**	**200**
7							**0**	**75**	**95**	**100**
8								**0**	**90**	**100**
9									**0**	**50**
10										**0**

В текущем году привлечено для этих целей 10 новых городов. Каким образом разделить эти 10 городов на пары так, чтобы минимизировалось суммарное расстояние между городами – побратимами. Оценки расстояний между городами представлены в таблице.

29. Пираты. У капитана шхуны N пиратов. Ему необходимо разбить их на две команды для абордажа так, чтобы в каждой команде не было ни одной пары пиратов, испытывающих друг к другу антипатию. Помогите капитану.

Входные данные (файл INPUT.TXT):

- $N \le 100$ – количество пиратов (первая строка);
- A – матрица связей, $A(i,j) = 1$ – пираты с номерами i и j не любят друг друга, $A(i,j) = 0$ – любят.

Выходные данные (файл OUTPUT.TXT):

- сообщение «NOT», если разбивку сделать нельзя;
- один из вариантой разбивки (каждая команда в отдельной строке), если разбивку сделать можно.

30. Обмен. В некотором городе N ($M \le 50$) домов. В городе возникла эпидемия обменов – все жители жаждут обменять свои квартиры. Про любые два дома можно сказать, находятся они їî соседству или нет. Требуется указать все возможные способы обмена квартир, при котором жители дома, жившие по соседству, и после обмена проживаю по соседству. Обмен описывается парой <старый номер дома, новый номер дома>.

Входные данные (файл INPUT. TXT):

• N – количество домов (первая строка):

• *A* – матрица связей: 1 элемент – соответствующие дома соседние: 0 – нет.

Выходные данные (файл OUTPUT.TXT):

• сообщение «NOT», если вариантов обмена не;

• количество вариантов обмена (в отдельной строке), если обмен возможен.

31. Сумасшедший диктатор. У диктатора *N* подготовленных боевиков. Опыт их использования против инакомыслящих показал, что наиболее эффективно они действуют при работе парами. Психологическое тестирование определило совместимость боевиков. Требуется составить для диктатора наибольшее количество пар, пригодных для ведения действий.

Входные данные (файл INPUT.TXT):

• $N \le 100$ – количество боевиков (первая строка):

• *A* – матрица связей (1 – допускается совместная работа боевиков, 0 – нет).

Выходные данные (файл OUTPUT.TXT) – один вариант решения:

• количество пар (первая строка)*

• каждая пара в отдельной строке.

32. Задача о китайском почтальоне. Ребрам графа *G* приписаны положительные веса. Найти цикл, проходящий через каждое ребро графа *G* по крайней мере, один раз и такой, что для него общий вес минимален. Указание. Очевидно, что если *G* эйлеров цикл, то любой такой цикл будет оптимальным, так как каждое ребро проходится один раз и вес этого цикла равен сумме весов ребер. Выделим вершины графа, имеющие нечетное число ребер, инцидентных им. С помощью алгоритма Флойда найдем кратчайшие расстояния между этими вершинами. Сформируем двудольный граф – *X* и *Y* совпадают с выделенным множеством вершин, веса ребер – кратчайшее расстояние по Флойду. Используя технику чередующихся цепочек, найдем наибольшее паросочетание минимального веса. Остаюсь найти эйлеров цикл в графе *G*, вершины с нечетным числом ребер, а их четное число, разбиваются на пары в соответствии с найденным паросочетанием.

33. Уличная гонка. На рис. 3(а) изображен пример плана улиц для гонки. Вы видите точки, помеченные числами от 0 до *N* (где *N* = 9), а также стрелки, соединяющие их. Точка 0 является стартовой:

точка N – финишной. Стрелками представлены улицы с односторонним движением. Участники гонки передвигаются от точки к точке по улицам только в направлении стрелок. В каждой точке участник может выбрать любую из исходящих стрелок.

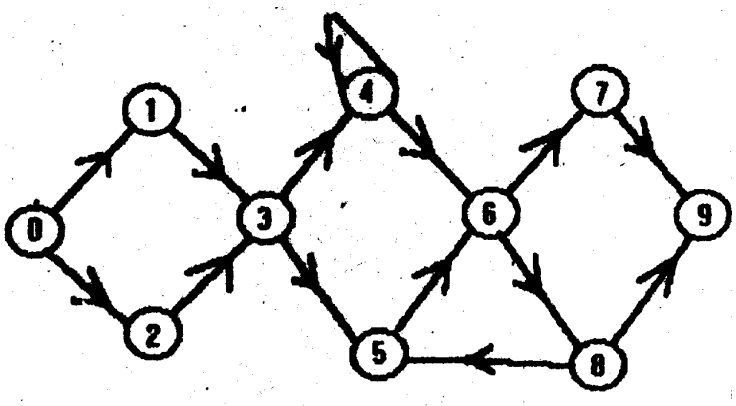
Назовем план улиц хорошим, если он обладает следующими свойствами: каждая точка плана может быть достигнута со старта; финиш может быть достигнут из любой точки плана; у финиша нет исходящих стрелок.

Для достижения финиша участник не обязан пройти через все точки. Однако некоторые точки невозможно обойти. Назовем их неизбежными. В примере такими точками являются точки 0, 3, 6, 9. Для заданного хорошего плана Ваша программа должна определить множество неизбежных точек (за исключением старта и (финиша), которые посещают все участники (подзадача А).

Предположим, что гонка проводится за два последовательных дня. Для этой цели план следует разбить на два хороших плана, по одному на каждый день. В первый день стартом является точка 0, а финишем служит некая «точка разбиения». Во второй день старт находится в этой точке разбиения, а финиш находится в точке N. Для заданного хорошего плана Ваша программа должна определить множество всех возможных точек разбиения (подзадача В).

INPUT.TXT
1 2 - 2
3 - 2
5 4 - 2
6 4 - 2
6 - 2
7 8 - 2
9 - 2
5 9 - 2
- 1
OUTPUT.TXT
2 3 6
1 3

(б)



(а)

Рис. 3

Точка S является точкой разбиения для хорошего плана C, если

S отличается от старта и финиша *C* и план разбивается ею на два хороших плана без общих стрелок и с единственной общей точкой *S*. В примере точкой разбиения является точка 3.

Входные данные (рис.3(б)). В файле INPUT.TXT описан хороший план, содержащий не более 50 точек и не более 100 стрелок. В файле имеется *N*+1 строка. Первые *N* строк содержат конечные точки стрелок, исходящие соответственно из точек от 0 до *N*–1. Каждая из этих строк заканчивается числом –2. В последней строке содержится число –1.

Выходные данные (рис. 3(б)). Ваша программа должна записать две строки в файл OUT PUT. TXT. В первой строке находится количество неизбежных точек в заданном плане, после чего в той же строке – номера этих точек в любом порядке (подзадача А). Во второй строке – количество точек разбиения в заданном плане, за которым следуют номера этих точек в любом порядке (подзадача В).

34. Транспортная компания находится в городе *A*, имеет один самолет и обслуживает с помощью авиарейсов семь городов. Расстояния между городами приведены в таблице.

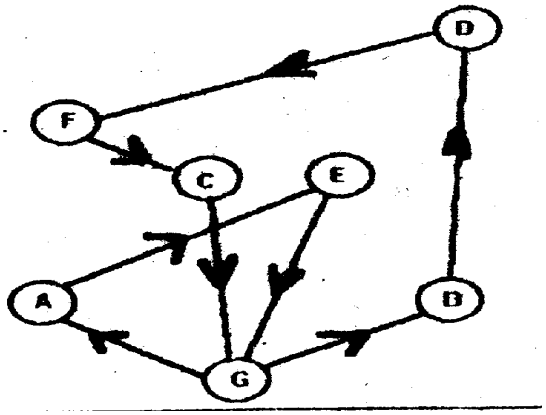


Рис. 4

	A	B	C	D	E	P	G
A	**0**	**56**	**43**	**71**	**35**	**41**	**36**
B	**56**	**0**	**54**	**58**	**36**	**79**	**31**
C	**43**	**54**	**0**	**30**	**20**	**31**	**58**
D	**71**	**58**	**30**	**0**	**38**	**59**	**75**
E	**35**	**36**	**20**	**38**	**0**	**44**	**70**
F	**41'**	**79**	**31**	**59**	**44**	**0**	**72**
G	**36**	**31**	**58**	**75**	**70**	**72**	**0**

Каждое утро компания получает заявки на обслуживание, которые она должна выполнить в этот день. Минимизировать затраты компании при следующих условиях: авиарейсы начинаются в городе A и заканчиваются в городе *A*; за один рейс компания может выполнить только одну заявку.

Написать программу, которая определяет маршрут перелетов начинающийся в городе *A* и заканчивающийся в городе *A*, имеющий минимальное значение суммарного расстояния и обслуживающий все поступившие заявки. Входные данные. Количество заявок – N ($1 \le N \le 20$). В следующих N строках – описание заявки. Пример – рис. 4.

5
F C
G B
B D
A E
G A.

Выходные данные. В первой строке – минимальное суммарное расстояние. Во второй строке – маршрут перелетов. Для нашего примера:

348
A E G B D F C G A

35. Составить программу поиска остовного дерева грубым методом и по алгоритму Прима – Краскала. Оценить эффективность на различных примерах.

36. В банке имеется несколько счетов различных предприятий, имеющих задолженности друг перед другом. Остаток на счетах предприятия должен быть неотрицательным. Общая задолженность конкретного предприятия больше чем остаток на его счете. Банк может выдавать целевые беспроцентные кредиты для проведения взаимозачета с условием обязательного и полного возврата средств после проведения операций.

Необходимо решить проблему взаимозачетов, так чтобы:

1) общая сумма задолженностей предприятий была минимальной;

2) количество перемещений средств между счетами (проводок) было минимальным;

3) общая сумма целевых кредитов была минимальной. Результат должен быть представлен в виде отчета.

37. Составление расписания для параллельных процессоров. Три одинаковых процессора могут выполнять *m* заданий. Каждое задание может быть выполнено на любом процессоре, и, если задание загружено в процессор, оно находится в нем до полного завершения (т.е. задания не могут прерываться и разделяться между двумя и более процессорами). При i=1,…,m задание i требует для выполнения времени t_i. Для любого линейного порядка заданий следующее задание из списка выполняется на первом освободившемся процессоре. Задания могут быть перечислены в любом порядке. Определите такую очередность заданий, при которой все задания выполняются в кратчайшее время.

5. ПРИМЕР РЕШЕНИЯ ЗАДАЧ

Имеется сеть междугородних автомобильных дорог. Известны расстояния всех участков дорог. Требуется перечислить все пути, ведущие из пункта *A* в пункт *B*, не проходящие дважды через один и тот же пункт.

```
	#include <stdio.h>
#include <conio.h>

const int max=11;

int versh;   // количество пунктов
int gr[max]; // текущий путь
int a,b,j;

struct set
{
	int arr[max];
	int n;
} m; // множество вершин, входящих в путь

int matr[max][max];  // матрица смежности
```

```
// ввод матрицы смежности
void WwodMatr(int matr[][max])
{
	int i,j;
	textbackground(BLACK);
	textcolor(WHITE);
	clrscr();
	cprintf("Введите количество пунктов: ");
	scanf("%d",&versh);
	for(i=1;i<=versh;i++)
		for(j=1;j<=versh;j++)
		{
			matr[i][j]=0;
			matr[j][i]=0;
		}
	do
	{
		cprintf("Введите связи в виде пары вершин (99-конец): ");
		scanf("%d",&i);
		if(i!=99) scanf("%d",&j);
		if((i>0 && i<=versh)&&(j>0 && j<=versh))
		{
			matr[i][j]=1;
			matr[j][i]=1; // матрица смежности симметрична
		}
		else if(i!=99) cprintf("ОШИБКА ввода!");
	} while(i != 99);
	cprintf("Ввод закончен");
	getch();
}

void Wiwod(int matr[][max])
{
	int j;
	window(46,2,75,22); // окно вывода результатов
	textbackground(CYAN);
	clrscr();
	textcolor(LIGHTGREEN);
```

```
	cprintf("  ");
	for(int i=1;i<=versh;i++) cprintf("%2d",i); // номера столбцов
матрицы
	cprintf("\n\r");
	for(i=1;i<=versh;i++)
	{
		textcolor(LIGHTGREEN);
		cprintf("%2d",i); // номера строк матрицы
		textcolor(WHITE);
		for(j=1;j<=versh;j++) printf("%2d",matr[i][j]);
		cprintf("\n\r");
	}
}

int isin(int a)
{
	for(int i=0;i<m.n;i++)
		if(m.arr[i]==a)   return 1;
	return 0;
}

// поиск всех путей в графе
void PoiskPut(int t)
{
	gr[j]=t; // добавление в путь текущей вершины
	m.arr[m.n++]=t;  // коррекция множества вершин пути
	j++;
	if(t==b)
	{		cprintf("\n\rНайден путь: ");
		for(int i=1;i<=j-1;i++) // вывод пути
			cprintf("%d ",gr[i]);
		getch();
	}
	else
		for(int i=1;i<=versh;i++)
			if(!isin(i) && matr[t][i]) // поиск в глубину: выбор
продолжения пути без цикла
				PoiskPut(i);
```

```
    // здесь оказываемся после нахождения очередного пути или в случае попадания в тупик
    m.n--; // исключение из множества вершин пути последней вершины
    j--;    // возврат в предыдущую вершину
}

void main(void)
{
    int l=1;
    char ch;
    clrscr();
    WwodMatr(matr); // ввод матрицы смежности
    while(l)
    {
        Wiwod(matr);
        cprintf("Введите исходный пункт A: ");
        scanf("%d",&a);
        cprintf("\r\nВведите конечный пункт B: ");
        scanf("%d",&b);
        for(int i=1;i<=max;i++)
            m.arr[i]=0; // инициализация множества вершин пути
        j=1;
        PoiskPut(a); // перечисление всех путей
        cprintf("\n\rПутей больше нет!");
        cprintf("\n\rПовторить поиск y/n ? ");
        ch=getche();
        if(ch=='n') l=0; // для выхода из цикла
    }
}
```

6. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Какие формы представления графа в ЭВМ Вы знаете? Перечислите их.
2. Что из себя представляет матрица инциденций?
3. Что из себя представляет матрица смежности?
4. Что такое остовное дерево?
5. Какие алгоритмы используются для поиска остовного дерева?

6. Какие алгоритмы используются для поиска остовного дерева минимального веса?
7. Какую структуру данных использует алгоритм поиска в ширину?
8. Какую задачу решает алгоритм Форда–Белмана?
9. Какую задачу решает алгоритм Дейкстры?
10. Какую задачу решает алгоритм Флойда?
11. Какую задачу решает алгоритм Уоршала?
12. Что такое минимальный разрез в потоковой задаче?
13. В чем суть теоремы Форда–Фалкерсона?
14. Приведите разновидности потоковых задач.

СПИСОК РЕКОМЕНДУЕМОЙ ЛИТЕРАТУРЫ

1. Кнут, Д. Искусство программирования для ЭВМ Т.1, Основные алгоритмы / Д. Кнут – М.: Диалектика Вильямс 2020. – 760 с. ISBN 978-5-907144-23-1

2. Вирт, Н. Алгоритмы и структуры данных / Н. Вирт – М.: ДМК Пресс, 2010 – 272 с.: ил. ISBN 978 5 94074 584 6

3. Кормен, Т. Алгоритмы: построение и анализ / Т. Кормен, Ч. Лейзерсон, Р. Ривест. – М: Издательский дом «Вильямс», 2005. –1296 с. ISBN 5-8459-0857-4(рус)

4. Ахо, А. В. Структуры данных и алгоритмы / А. В. Ахо, Д. Э. Хопкрофт, Д. Д. Ульман. – М.: Вильямс, 2003. – 382 с. ISBN 10:0-201-00023-7

5. Седжвик, Р. Фундаментальные алгоритмы на C++. Анализ. Структуры данных. Сортировка. Поиск: пер. с англ. /Р. Седжвик. – К.: ДиаСофт, 2001. – 688 с. ISBN 5-93772-054-7

6. Хэзфилд, Р. Искусство программирования на С. Фундаментальные алгоритмы, структуры данных и примеры приложений. Энциклопедия программиста: пер. с англ. / Р. Хэзфилд, Л. Кирби [и др.]. – К.: ДиаСофт, 2001. – 736 с. ISBN 966-7393-82-8, 0-672-31896-2

7. Хусаинов, Б.С. Структуры и алгоритмы обработки данных. Примеры на языке Си (+ CD): учеб. Пособие / Б.С. Хусаинов. – М.: Финансы и статистика, 2004. – 464 с. ISBN: 5-279-02775-8

Структуры и алгоритмы обработки данных. Алгоритмы на нагруженных графах [Электронный ресурс]: методические указания к выполнению лабораторной работы №17 для студентов очной, очно-заочной и заочной форм обучения по направлениям подготовки 230100 «Информатика и вычислительная техника», 010500 «Математическое обеспечение и администрирование информационных систем», 231000 «Программная инженерия».

ВАСИЛИЙ КОНСТАНТИНОВИЧ ГУЛАКОВ
АНДРЕЙ ОЛЕГОВИЧ ТРУБАКОВ
ЕВГЕНИЙ ОЛЕГОВИЧ ТРУБАКОВ

Научный редактор В.В. Конкин
Редактор издательства Л.Н. Мажугина
Компьютерный набор В.К. Гулаков

Темплан 2020 г., п. 209

Подписано в печать Формат 1/16
Усл.печ.л. 1,51 Уч.-изд.л. 1,51

Издательство Брянского государственного технического университета
241035, Брянск, бульвар им.50-летия Октября, 7, БГТУ, тел. 58-82-49
Лаборатория оперативной полиграфии БГТУ, ул. Институтская, 16