В.В. Мендель, А.Е. Поличка

Информатика: элективные курсы для учащихся 8–11 классов общеобразовательных учреждений

ВЫПУСК 9

Методическое пособие по информатике для учащихся и учителей

Публикуемые в настоящем сборнике методические материалы разработаны для реализации программ дополнительного образования по информатике для участников очной сессии Хабаровской краевой заочной физико-математической школы.

Хабаровск: КГБОУ ДОД «Хабаровский краевой центр развития творчества детей и юношества», 2014. – 58с. Тираж 70 экз.

СОДЕРЖАНИЕ

В.В. Мендель, к.ф-м.н.,
проректор по учебной работе ФГБОУ ВПО ДВГГУ

БАЗОВЫЕ АЛГОРИТМЫ НА ГРАФАХ

1. ПРЕДВАРИТЕЛЬНЫЕ СВЕДЕНИЯ О ГРАФАХ

1.1 Графы. Пути, циклы, связанность

Определение 1.1. *Графом (будем обозначать его буквой Г) называется рисунок, состоящий их нескольких точек (вершин графа) и ребер – отрезков или дуг, соединяющих некоторые вершины.*

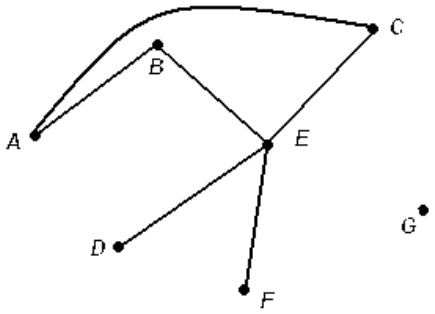


Рисунок 1. *Граф с семью вершинами*

На рисунке 1 показан пример графа.

Ребрами соединены не все вершины графа. Вершины, из которых не выходит ни одного ребра, называют изолированными.

С помощью графов удобно и наглядно изображается информация о разных объектах и отношениях между ними. Рассмотрим пример.

Пример 1. В розыгрыше финальной части турнира участвуют семь команд: шесть команд, набравших наибольшее количество очков в предварительной части турнира и команда – победитель прошлого года. Сначала играют друг с другом первые шесть команд, затем три команды, одержавшие победы и команда, победитель прошлого года, играют друг с другом. Два победителя этого тура встречаются в финале.

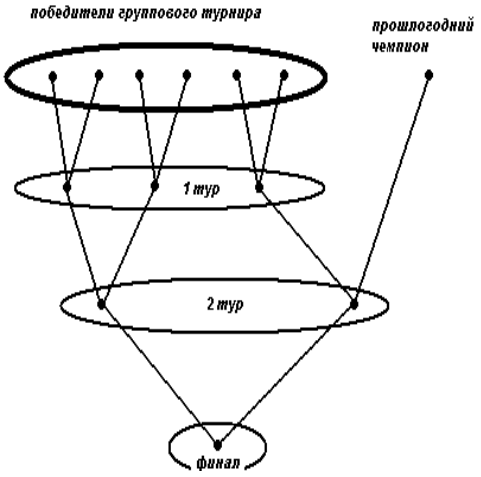


Рисунок 2. *Граф к примеру 1*

Понять о чем идет речь в этом тексте нелегко. Попробуем представить его в виде наглядной схемы (смотри рисунок 2) и порядок организации финальной части розыгрыша станет очевидным.

Обратимся теперь к истории появления графов. Впервые в рассмотрение их ввел великий математик *Леонард Эйлер*. В популярной литературе часто упоминается его *задача о Кенигсбергских мостах*. Смысл задачи таков: в городе Кенигсберге (ныне это город Калининград) протекает река Прегель. Сам город расположен на берегах этой реки и ее островах. Естественно, что в городе построены мосты, связывающие все его районы (смотри рисунок 3). Во время прогулки по городу Эйлер захотел пройти по всем мостам, причем по каждому только один раз. Однако ему это не удалось. Вернувшись домой, ученый составил схему, изобразив участки суши точками, а мосты отрезками (рисунок 4). Это и был первый граф.

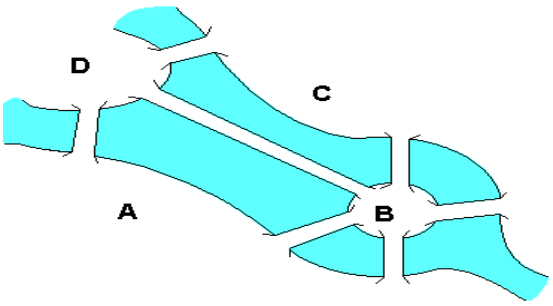


Рисунок 3. *Схема мостов в Кенигсберге*

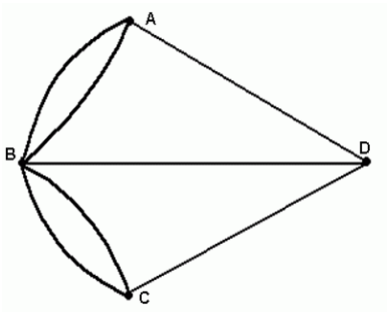


Рисунок 4. *Граф к рисунку 3*

Определение 1.2. Степенью вершины графа называется число выходящих из него ребер.

Степени вершин A, C и D на рисунке 4 равны трем, а степень вершины B равна 5.

В графе на рисунке 1 степень вершины G равна нулю, так как из нее не выходит ни одно ребро.

Для степени вершины принято следующее обозначение: deg(A).

Рассмотрим некоторый граф *Г*. Обозначим количество его вершин буквой *p*, а количество ребер буквой *q*. Сформулируем в виде теорем простейшие свойства степени вершины.

Теорема 1.3. Сумма степеней всех вершин графа Г равна удвоенному количеству его ребер (2q).

Определение 1.4. Граф называют простым, если две вершины соединяет

не более одного ребра, в противном случае, граф называют мультиграфом.

На рисунке 1 изображен простой граф, а на рисунке 4 мультиграф.

Теорема 1.5. *В простом графе найдется не менее двух вершин с одинаковыми степенями.*

Теорема 1.6. *В простом графе с p вершинами число ребер не больше* $\frac{p(p-1)}{2}$: $\frac{p(p-1)}{2} \geq q$.

Определение 1.7. Граф называется полным, если каждая его вершина соединена со всеми другими вершинами графа.

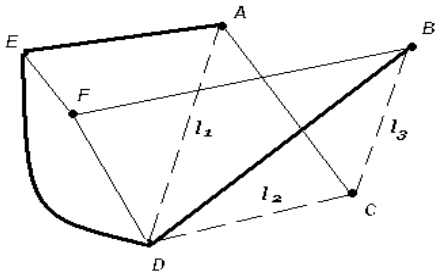


Рисунок 5. *Пути в графе*

Выпуклый многоугольник с построенными всеми его диагоналями является полным графом. В полном графе с *p* вершинами $\frac{p(p-1)}{2}$ ребер.

Еще одно важное понятие, относящееся к графам – связность. Для того чтобы ввести его, дадим несколько определений. Начнем с понятия путь.

Определение 1.8. Путем (из вершины A в вершину B) в графе называется последовательная цепочка смежных ребер, которая начинается в вершине A и заканчивается в вершине B. Путь может проходить через ребро только один раз.

Запись пути в графе зависит от того, как определены его ребра. Если ребра определены с помощью вершин, то записывают последовательность тех вершин, через которые проходит путь. Если же ребра имеют собственные названия, то выписывается последовательность из этих названий.

Рассмотрим примеры. На рисунке 5 жирными линиями показан путь из точки *A* в точку *B*, который мы можем записать так: (*AEDB*). Другой путь из *A* в *B* показан пунктирными линиями, его можно записать как ($l_1 l_2 l_3$).

Заметим, что может существовать несколько путей из одной точки в другую. Рассмотрим теперь специфический случай, когда начало и конец пути совпадают.

Определение 1.9. Циклом называется путь, у которого начало и конец совпадают.

На рисунке 5 циклами являются следующие пути: (*AEFDA*), (*EFBCADE*).

Определение 1.10. Пути и циклы называются простыми, если через каждую свою вершину они проходят только один раз.

Все приведенные в примерах пути и циклы являются простыми.

Определение 1.11. Граф называется связным, если любые две его вершины можно соединить хотя бы одним путем. В противном случае граф называется несвязным.

Граф на рисунке 1 несвязен, так как нет ни одного пути, соединяющего точку G с остальными вершинами, графы на рисунках 2, 4, 5 - связные.

Существуют задачи, в которых предлагается обвести ту или иную фигуру, не отрывая карандаш от бумаги. При этом запрещается проводить карандаш по одной линии несколько раз. Понятно, что аналогичное задание может быть дано и относительно некоторого графа. Далеко не все графы можно построить описанным выше способом. Те, для которых это требование выполняется, называются *уникурсальными* или *эйлеровыми*. Эти графы непосредственно связаны с задачей о Кенигсбергских мостах и впервые описаны Леонардом Эйлером. Сам Эйлер доказал следующую теорему.

Теорема 1.12 (Эйлера). Связный граф уникурсален тогда и только тогда, когда степени всех его вершин четны или у него ровно две вершины нечетной степени.

Используя доказанную теорему, можно убедиться в том, что задача о Кенигсбергских мостах не имеет решения.

Контрольные вопросы

1.1. Что называется графом?

1.2. Какой граф называется простым?

1.3. Что называется путём?

1.4. Какой путь является циклом?

1.5. Какой граф называется полным?

1.6. Какой граф называется связным?

1.7. Какой граф называется уникурсальным (эйлеровым)?

1.8. Что является условием уникурсальности?

1.2 Деревья. Планарные графы. Раскраски

Важным частным случаем графа является дерево. Это название связано с типичным видом некоторых представителей данного класса. Дадим определение.

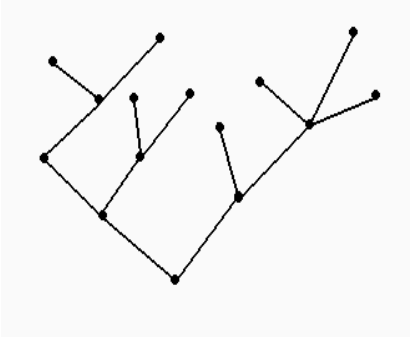


Рисунок 6. *Дерево (похожее на куст)*

Определение 2.1. Деревом называется связный граф, не содержащий циклов. Несвязный граф, не имеющий циклов, называют лесом. (В лесу, как известно, не меньше двух деревьев.)

На рисунках 6 и 7 приведены примеры деревьев, далеко не все они напоминают по форме дерево. Тем не менее, можно так деформировать граф на рисунке 7, что он станет полностью похож на граф с рисунка 6. Когда же граф является деревом? Ответ на вопрос сформулируем ниже, а сейчас укажем одно интересное свойство деревьев.

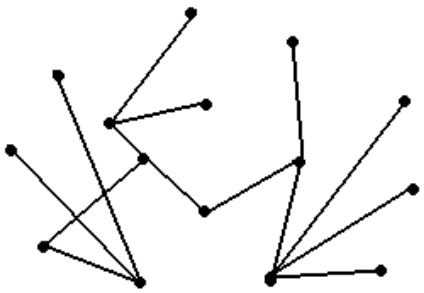


Рисунок 7. *Дерево (не похожее на дерево)*

Теорема 2.2. В любом дереве существует хотя бы одна вершина степени единица. (Такие вершины называют «висячими».)

Доказательство. Используем метод «от противного». Предположим, что в графе нет вершин степени единица. Тогда все вершины имеют степень не меньше двух. В подобных графах обязательно существую циклы. Это противоречит тому, что исходный граф – дерево.

Теперь сформулируем теорему, являющуюся признаком дерева.

Теорема 2.3. Связный граф является деревом тогда и только тогда, когда количество его вершин на единицу превосходит количество ребер:

$$p - q = 1.$$

Одна из важных прикладных задач, относящихся к графам, это отыскание *минимального остова* графа. Поясним, о чем идет речь. Пусть имеется связный граф Г, необходимо найти такой его подграф, который связен, содержит все вершины исходного графа, и при удалении любого ребра становится несвязным (в этом и заключено условие минимальности). Нетрудно догадаться, что такой минимальный остов является деревом.

Исторически планарные графы связаны с одной знаменитой задачей.

Задача о домиках и колодцах. В некоторой деревне есть три колодца. Трое жителей, живущие в трех стоящих рядом домиках перессорились, и решили так протоптать тропинки от своих домов к каждому из трех колодцев, чтобы они не пересекались. Удастся ли им выполнить свой план?

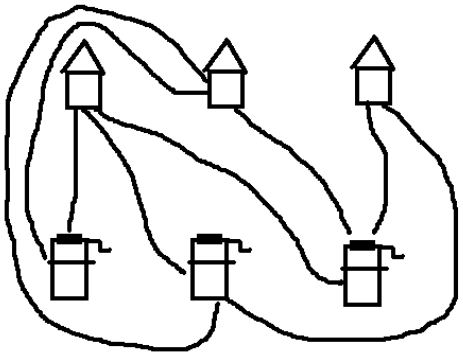


Рисунок 8. *Домики и колодцы*

Попробуем решить эту задачу. Проведем тропинки так, как это показано на рисунке 8.

Как видно, нам удалось провести только восемь тропинок, а девятая должна пересечься хотя бы с одной. Можно доказать (мы не будем приводить строгое доказательство), что эта задача не имеет решения. Дело в том, что по мере проведения тропинок из двух первых домиков, будет получаться некоторый замкнутый контур, внутри которого будет стоять один из колодцев, при этом третий домик будет находиться снаружи от этого контура. Для того чтобы соединить этот домик с колодцем, обязательно потребуется пересечь новой тропинкой одну из уже проложенных.

Можно предложить еще одну задачу.

Задача о пяти хуторах. Пять хуторов расположены в вершинах правильного пятиугольника. Нужно проложить дороги

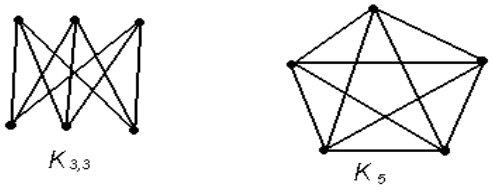


Рисунок 9. *Графы к задачам о домиках и колодцах и о пяти хуторах*

от каждого из них к остальным так, чтобы не было перекрестков.

Эта задача также не может быть решена.

На рисунке 9 построены графы $K_{3,3}$ и K_5, соответствующие задаче о домиках и колодцах и задаче о пяти хуторах. Оказывается, что проблема укладки графа на плоскости тесно связана с этими типами графов.

Перейдем теперь к более строгим формулировкам.

Определение 2.4. Граф называется планарным, если его можно изобразить на плоскости без самопересечений. Такое изображение называют плоской укладкой графа.

На рисунке 10 изображен полный граф K_4 и его плоская укладка.

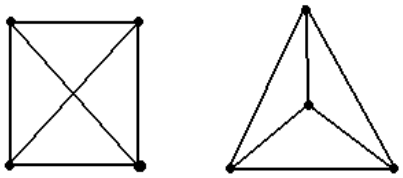


Рисунок 10. *Граф и его плоская укладка*

Далее нам понадобится понятие *гомеоморфизма*. Это очень сложное математическое понятие, но в отношении к графам оно формулируется довольно просто. Мы не будем давать строго определения, а рассмотрим это свойство на примерах.

Будем говорить, что два графа *гомеоморфны*, если один из них получен из другого, путем добавления новых вершин на уже имеющиеся ребра.

На рисунке 11 показаны два гомеоморфных графа. Граф справа получен из первого графа добавлением четырех вершин. На практике бывает довольно трудно увидеть, что два графа гомеоморфны.

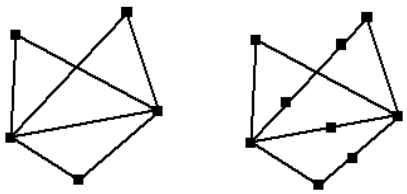


Рисунок 11. *Гомеоморфные графы*

Теперь сформулируем условие планарности графов.

Теорема 2.5. Граф планарен тогда и только тогда, когда он не содержит подграфов, гомеоморфных $K_{3,3}$ и K_5.

Эта общая теорема ставит окончательную точку в рассмотренных выше задачах о хуторах и о домиках и колодцах.

Замечание 2.6. Доказать что граф планарен и построить его плоскую укладку – это две независимых задачи. Заметим, что если граф не имеет пяти вершин, степень которых не меньше четырех, то он наверняка не имеет подграфа, гомеоморфного K_5. Если в графе нет шести вершин, степень которых не меньше трех, то он гарантированно не имеет подграфа, гомеоморфного $K_{3,3}$. Однако построить плоскую укладку графа с достаточно большим количеством вершин бывает непросто.

Рассмотрим теперь еще одну классическую задачу теории графов.

Задача о четырех красках. На политической карте мира нарисовано несколько государств. Карту нужно раскрасить так, что бы две страны, имеющие общую границу, были покрашены в разные цвета.

В классическом варианте предполагалось, что карту можно раскрасить четырьмя цветами. Покажем, как эта задача связана с графами. Обозначим каждую страну на карте точкой, вершины, отвечающие странам, имеющим общую границу, соединим ребрами. Теперь задачу о раскрашивании можно сформулировать так: раскрасить вершины планарного графа так, чтобы любые две смежные были покрашены в разные цвета. Эта задача может быть решена для графов с малым количеством вершин. Если же число вершин достаточно велико, то гипотеза четырех красок оказывается неверной. (Этот факт установлен с помощью мощных компьютеров.)

Вместе с тем, довольно простыми средствами была доказана теорема о пяти красках.

Теорема 2.7. Планарный граф можно раскрасить пятью красками так, что любые смежные вершины будут окрашены в разные цвета.

2. ПРЕДСТАВЛЕНИЕ ИНФОРМАЦИИ О ГРАФАХ В ПАМЯТИ КОМПЬЮТЕРА

2.1 Матрица смежности

Наиболее удобный из простых способов хранения информации о графе - *матрица смежности*. Идея следующая: для графа с p вершинами задается таблица размерами pxp, в каждую ячейку которой записаны числа 0 или 1 по следующему простому правилу:

Если вершины графа с номерами i и j соединены ребром - в ячейку таблицы с адресом i, j записывают 1, в остальных случаях - 0.

Недостаток этого способа – большая трудоемкость подготовки входного файла.

2.2 Список смежности

Список смежности – способ, наиболее удобный для подготовки входного файла.

Список представляет текст, каждая строка которого начинается числом – номером вершины, после которого перечисляются номера всех вершин, смежных текущей.

Такой способ, как отмечалось выше – самый компактный.

Обычно, для удобства работы с входным файлом, в первой строке текстового файла, задающего список, указывают число вершин в графе, а в каждой строке, после первого числа (номера текущей вершины) указывают степень этой вершины (количество выходящих из нее ребер).

2.3 Список ребер

В ряде прикладных задач удобно представлять информацию о графе, перечисляя все его ребра. Например, это можно сделать, задав двумерный массив длины q, в первой строке которого записаны номера начальных, а во второй - номера конечных вершин каждого ребра. Таким образом, в i – том столбце будет задан номер начальной и конечной вершины i – того ребра.

2.4. Конвертирование данных: переход от одного способа представления графа к другому

На практике удобно использовать разные способы представления графа. Например, изначально пользователь во входном файле представляет структуру графа списком смежности, затем, в памяти компьютера, используется матрица смежности, и, наконец, выходной файл после решения задачи представляется в виде списка ребер.

Рассмотрим механизмы конвертирования данных.

1. Матрица смежности в список смежности

1.1 создаем и открываем для записи текстовый файл.

1.2 во внешнем цикле последовательно просматриваются строки матрицы, при начале просмотра каждой новой строки в файл с новой строчки записывают номер этой строки.

1.3 для просмотра каждой строки матрицы открывается цикл. В нем проверяется условие на равенство 1 значения в текущей ячейке строки. Если условие выполнено, то в файл через пробел записывается номер столбца, которому принадлежит текущая ячейка (он равен текущему значению индекса внутреннего цикла).

2. Список смежности в матрицу смежности

2.1 в разделе описания переменных создается двумерный массив, размеры которого равны pxp.

2.2 при запуске программы этот массив инициализируется – заполняется нулями.

2.3 открывается для чтения файл, содержащий список смежности.

2.4 внешний цикл по очереди переходит к каждой строке списка.

2.5 внутренний цикл просматривает текущую строку списка и заменяет на единицу значения массива – матрицы смежности с адресами a[i, j], где i – первое число строки (просматриваемая вершина), j – номер вершины, смежной с просматриваемой.

3. Список ребер в матрицу смежности

3.1 и 3.2 аналогично предыдущему случаю.

3.3 просматриваются в цикле от 1 до q столбцы массива, задающего список ребер. Элементы матрицы смежности a [I, j] приравниваются единице. Где I и j- значения, стоящие в верхней и нижней ячейках текущего столбца.

Упражнения 1–2

Опишите алгоритм, преобразующий матрицу смежности и список смежности в список ребер графа.

2.5 Дополнительная информация: метки, вес

Помимо информации о смежности или инцидентности вершин и ребер графа бывает полезно учитывать некоторые дополнительные данные. Эти данные могут иметь как внешний (связанный с постановкой задачи) так и внутренний характер (данные появляются и используются только внутри программы, но не являются конечным результатом ее работы).

Примером данных, относящихся к первому типу, относится вес ребра или вершины. Вес это неотрицательное число, сопоставленное с каждым ребром или каждой вершиной графа.

Если требуется задать вес ребра, то эту информацию легко включить в матрицу смежности (вместо единицы писать вес ребра) или в список ребер (ввести третью строку в матрицу, задающую список ребер и поместить в нее информацию о весе).

В случае, если используется список смежности, то можно в строке, задающей список вершин, смежных с первой по списку, записывать после каждой вершины ее вес.

На практике наиболее подходит представление графа с взвешенными ребрами в виде списка ребер.

3. АЛГОРИТМЫ ПОИСКА И ИХ ПРИМЕНЕНИЕ

3.1 Остов графа

Остов графа – дерево, содержащее все его вершины, причем ребрами остова могут быть только ребра исходного графа.

Нахождение остова относится к базовым задачам теории алгоритмов. Известны различные способы построения остова, причем эти способы позволяют получать дополнительную информацию о графе.

В этом пункте мы рассмотрим алгоритмы поиска «в глубину» и «в ширину», позволяющие дополнительно выделять компоненты связности графа и расстояние между ребрами. Начнем с алгоритма поиска «в глубину».

3.2 Поиск остова в глубину

Само название алгоритма говорит за себя: нужно так исследовать граф, чтобы уйти как можно дальше от начальной вершины («корня» - остовного дерева).

Идея алгоритма:

Выбирается одна «корневая» вершина, которая получает метку, из этой вершины выполняется переход в любую смежную с ней непомеченную вершину, которая также помечается. Далее на каждом очередном шаге предпринимается попытка перейти из текущей помеченной вершины смежную с ней непомеченную. Если оказывается, что у текущей вершины нет ни одной смежной непомеченной вершины, то выполняется переход в предыдущую просмотренную вершину. Процесс выполняется до тех пор, пока не будут помечены все вершины. Для того чтобы вовремя остановить работу алгоритма, необходимо вести подсчет помеченных вершин. Когда счетчик станет равен числу вершин графа, тогда процесс останавливается.

Обратимся к представлению информации о графе и результате – остовном дереве. Входной файл может быть представлен списком смежности, который перерабатывается в матрицу смежности при запуске программы. Кроме этого

потребуется одномерный массив длины p для хранения меток вершин графа, которым изначально присваивается значение «не помечена». Для записи ребер остова можно применить двумерный массив из двух строк и p-1 столбца, в который будут заноситься номера вершин очередного выбранного ребра.

3.3 Поиск остова в ширину

Поиск в ширину позволяет строить ярусное дерево, причем пути в полученном остовном дереве, соединяющие корень с другими вершинами графа будут кратчайшими.

Идея алгоритма:

На начальном этапе все вершины графа имеют одинаковые метки (не просмотрены), например: -1. Выбранная корневая вершина получает метку 0. Далее рассматриваются вершины, смежные с корнем, они получают метку 1. Затем рассматриваются все непомеченные вершины, смежные вершинам первого яруса, имеющим метку 1. Они получают метку 2. И так далее, пока не будут помечены все вершины графа.

Для ввода данных и вывода результатов можно использовать те же структуры данных, что и в предыдущем случае (с учетом того, что метки вершин теперь могут принимать не два, а несколько значений).

3.4 Выделение компонент связности

В предыдущих задачах мы предполагали, что граф является связным: любые две его вершины соединены путем. Теперь рассмотрим несвязный граф. Такой граф распадается на несколько компонент связности: вершины, принадлежащие к одной компоненте, могут соединяться путем, а вершины из разных компонент не могут.

При решении прикладных задач целесообразно разложить граф на компоненты и затем исследовать каждую компоненту самостоятельно, это может значительно ускорить вычисления.

Опишем коротко, как алгоритм поиска в глубину модернизировать в алгоритм выделения компонент связности.

Для этого нужно несколько изменить формат массива меток. Далее процесс работает по такому принципу:

Произвольная вершина получает метку 1, в результате поиска все связанные с ней вершины также получают метку 1. С некоторого момента других вершин добавить нельзя. Тогда выбирается любая вершина с меткой 0, ей присваивается метка 2 и снова производится поиск. Процесс с увеличением номера метки продолжается до тех пор, пока все вершины не окажутся помеченными.

3.5 Вычисление расстояния между вершинами, диаметр, радиус, эксцентриситет и центры графа

Собственно расстояние между двумя выбранными вершинами легко получить, используя поиск «в ширину». В качестве корневой берется одна выбранная вершина, номер яруса второй выбранной вершины, присвоенный ей по итогам поиска, и есть искомое расстояние.

Диаметром связного графа называют наибольшее расстояние между двумя его вершинами.

Эксцентриситетом вершины графа называют наибольшее расстояние от этой вершины до остальных.

Радиус графа равен наименьшему эксцентриситету из всех его вершин. Все вершины, эксцентриситеты которых равны радиусу графа, называют его *центрами*.

Внимание: в полном графе радиус равен 1, все его вершины – центры графа.

Нетрудно понять, что, умея вычислять расстояния между вершинами графа (для этого используют поиск «в ширину») можно найти диаметр, радиус и центры графа.

4. МИНИМАЛЬНЫЙ ОСТОВ ВЗВЕШЕННОГО ГРАФА

4.1 Что такое минимальный остов?

Рассмотрим следующую задачу: в районе решено модернизировать электрические сети. Проектировщик на плане соединил все населенные пункты района напрямую линиями электропередачи. Заказчик решил сэкономить и попросил оставить минимально число участков ЛЭП, причем сделать это так, чтобы суммарная длина оставшихся участков была минимальна.

В результате решения получится связный граф без циклов (дерево), суммарная длина ребер которого (их вес) будет наименьшим. Вот такое основное дерево и принято называть *минимальным остовом* взвешенного графа.

Уточним определение. Рассматривается взвешенный связный граф. *Минимальный остов* – остовное дерево искомого графа, суммарный вес всех ребер которого является наименьшим.

Замечание: у графа может быть несколько разных минимальных остовов. Например, если вес всех ребер графа одинаковый, то любое его остовное дерево является минимальным.

Известно несколько алгоритмов построения минимального остова и их модификаций. Но, как правило, все они являются модификациями двух наиболее популярных: «жадного» алгоритма и алгоритма «ближайшего соседа». Ниже приводятся идеи, на которых основаны эти алгоритмы.

4.2 Алгоритм «ближайшего соседа»

Идея алгоритма близка идее поиска «в ширину». Прежде всего, ребра графа нужно отсортировать в порядке возрастания их веса. Все вершины на начальном этапе получают метку 0. Очевидно, что самое «легкое» ребро должно принадлежать остову, включаем его в остов и присваиваем его вершинам метку 1. Теперь будем рассматривать с начала отсортированный список ребер, пока не встретим ребро, метки вершин которого равны 0 и 1. Это

ребро включаем в остов, обе его вершины получают метку 1. Вновь продолжаем просмотр списка с начала и повторяем процедуру, пока не будет выбрано p-1 ребро.

4.3 Жадный алгоритм

«Жадный» алгоритм руководствуется правилом «разумного жадины». Он выбирает из отсортированного списка ребер только такие, которые удовлетворяют условию: при их добавлении к будущему минимальному остову не должны появляться циклы. При этом список ребер просматривается всего только один раз. Этот метод кажется довольно быстрым, но есть две проблемы:? Первая как проверить, что не получается цикл, вторая – в процессе генерации будущий минимальный остов может состоять из нескольких компонент связности, которые, на конечном этапе, должны объединиться в одно дерево.

Рассмотрим один из способов решить обе проблемы. Опять будем считать, что на начальном этапе вершины графа имеют нулевые метки. Выбираем самое легкое ребро и присваиваем его вершинам метку 1. Далее, просматривая по порядку остальные ребра, руководствуемся следующим:

- если обе вершины текущего ребра имеют метки 0, то это ребро образует новую компоненту, поэтому счетчик компонент увеличивается на 1 и значение счетчика присваиваются меткам обеих вершин текущего ребра,

- если одна вершина текущего ребра имеет метку 0, а другая метку, не равную 0, то ребро включают в остов, а вершина с «нулевой» меткой получает метку, равную метке второй вершины этого ребра,

- если вершины текущего ребра имеют различные, не равные нулю, метки, то ребро включается в остов, оно склеивает две компоненты, поэтому требуется переписать метки вершин объединяемых компонент: например, заменить номера меток одной компоненты на номера меток второй компоненты,

- если оказывается, что обе вершины текущего ребра имеют одинаковые ненулевые метки, тогда, при добавлении такого ребра будет получаться цикл, поэтому такое ребро не включается в остов.

Естественно, что просмотр списка ребер проводится до тех пор, пока не будет выбрано p-1 ребро.

5. ЦИКЛЫ

5.1 Поиск цикла в графе

Найти цикл в графе – сама по себе интересная задача. Будем рассматривать граф, в котором степень каждой вершины не меньше двух (то есть из каждой вершины выходит не менее двух ребер). На начальном этапе положим метки всех вершин равными 0. Выбираем одну из вершин, присваиваем ей метку 1, из этой вершины выходит как минимум два ребра – переходим по этому ребру в смежную вершину и присваиваем ей метку 1. Продолжаем движение, запоминая порядок пройденных вершин. В какой-то момент алгоритм приведет нас в одну из уже посещенных вершин (она имеет метку 1) запомним ее. Это будет означать, что замкнулся цикл. Теперь будем возвращаться в обратном направлении по списку посещенных вершин, до тех пор, пока не попадем в запомненную вершину, этот список и укажет нам искомый цикл.

5.2 Построение эйлерового цикла в графе

Напомним, что граф называют *эйлеровым циклом*, если существует цикл, проходящий через все ребра этого графа. Поиск такого цикла – весьма занятная задача, имеющая много приложений.

Необходимыми и достаточными условиями того, что граф является эйлеровым являются требования связности и четности всех его вершин (напомним, что вершину называют четной, если из нее выходит четное число ребер).

Алгоритм построения эйлерова цикла основан на последовательном применении алгоритма поиска цикла, описанного выше. После каждого такта работы алгоритма поиска цикла ребра найденного цикла изымаются из структуры графа. Это происходит до тех пор, пока степени всех вершин не станут равными нулю. После этого запускается процедура склеивания циклов,

суть которой в том, что два цикла, имеющие общую вершину (а в силу связности они есть всегда) объединяются в один очевидным способом.

5.3 Методы поиска гамильтонова цикла

Не менее знаменитым, чем эйлеров, является *гамильтонов цикл*. Это простой цикл, проходящий через все вершины графа. К сожалению, простого алгоритма построения такого цикла не известно, более того, есть веские причины полагать, что достаточно быстрого алгоритма решения такой задачи не существует. Кроме этого, не известно необходимое и достаточное условие существования гамильтонова цикла. Достаточное условие выглядит так: *в графе обязательно найдется гамильтонов цикл, если каждая его вершина смежна не менее чем половине оставшихся*.

Тем не менее, существует способ проверки наличия гамильтонова цикла, основанный на переборе всевозможных комбинаций его вершин, т.е. на перестановках. Имеется в виду следующее: берется перестановка всех вершин графа и проверяется, существует ли цикл, вершины которого идут в указанном порядке. Устроить проверку того, что выбранный цикл принадлежит данному графу нетрудно, проблема в том, что количество перестановок для p вершин равно p!. Если рассматривать граф с несколькими сотнями вершин, то поиск гамильтонова цикла займет гигантское время. Тем не менее, для графа с небольшим (один – два десятка) числом вершин вполне можно исследовать даже на персональном компьютере.

Заметим также, что существуют алгоритмы поиска гамильтонова цикла, основанные на поиске в глубину. Они не всегда позволяют отыскать нужный цикл, но весьма эффективны в случае, когда граф имеет небольшое количество ребер.

6. ДРУГИЕ АЛГОРИТМЫ

В этом пункте мы рассмотрим постановку еще нескольких базовых задач.

6.1 Задача о кратчайшем взвешенном пути

Суть задачи почти очевидна: имеется взвешенный связный граф, в котором выделены две вершины. Нужно найти кратчайший взвешенный путь из одной вершины в другую (имеется в виду такой путь, суммарный вес ребер которого – наименьший).

6.2 Задача о назначениях

Эту задачу часто называют задачей о женихах и невестах. Имеется два множества – женихов и невест. Некоторые пары, образованные мужчинами и женщинами симпатизируют друг другу и, в принципе, могут образовать семью. Допускается, что мужчина или женщина может симпатизировать нескольким лицам другого пола. Задача заключается в том, чтобы составить как можно больше попарно не пересекающихся пар симпатизирующих друг другу, чтобы сыграть как можно больше свадеб.

Алгоритм решения этой задачи основан на оптимизации: рассматривается какое-нибудь сочетание пар, которое пытаются улучшить (увеличить число пар).

6.3 Задача о раскраске вершин графа

Еще одна знаменитая задача: требуется раскрасить все вершины графа, используя как можно меньше цветов, так, чтобы вершины, соединенные общим ребром, были разных цветов. Решение этой задачи основано на широко применяемом в программировании *методе рекурсии*.

Если рассматривается *планарный* граф (такой, что его можно нарисовать на плоскости так, что ребра попарно не пересекаются), то алгоритм решения основывается на доказательстве *теоремы о пяти красках*.

6.4 Поиск максимального независимого парасочетания

Для произвольного графа нужно выбрать наибольшее количество ребер, ни какие два из которых не имеют общих вершин. Заметим, что, по сути, ребро и есть *сочетание пар*, так как оно соединяет пару вершин!

7. ЗАДАЧИ ДЛЯ САМОСТОЯТЕЛЬНОГО РЕШЕНИЯ

В этом параграфе приведены задания, которые предлагается выполнить на лабораторном практикуме. Максимальное количество баллов, в которые оценивается каждое задание, можно узнать у преподавателя.

1. Составить программу, которая преобразует входной файл, представляющий структуру графа в одном виде, в другой (внутренний) вид: список смежности – в матрицу смежности, список смежности в список ребер и т.п.

2. Решить первую задачу, добавив модуль, проверяющий корректность структуры графа.

3. Составить программу, реализующую поиск «в ширину» / «в глубину» для различных случаев представления структуры графа.

4. Составить программу, сортирующую заданный набор взвешенных ребер в порядке возрастания их веса.

5. Составить программу, реализующую построение минимального остова взвешенного графа с помощью «жадного» алгоритма / алгоритма «ближайшего соседа».

6. Составить программу, находящую цикл в графе, все вершины которого – четные.

7. Составьте программу, склеивающую два цикла графа, имеющих общую вершину.

8. Составьте программу, находящую эйлеров путь в графе, которая предварительно проверяет, что все вершины имеют четную степень.

9. Составьте программу, выделяющую компоненты связности графа.

10. Составьте программу, проверяющую, что заданный цикл принадлежит данному графу.

11. Составьте программу, генерирующую все перестановки для p первых натуральных чисел.

12. Составьте программу, определяющую, имеется ли в данном графе гамильтонов цикл.

8. СПРАВОЧНЫЙ МАТЕРИАЛ. ФАЙЛЫ, МАССИВЫ, ЦИКЛЫ И УСЛОВИЯ

8.1 Работа с файлами в Turbo Pascal

В реальной ситуации, как правило, компьютерная программа работает с входными и выходными данными, которые, хранятся в двух файлах: входном и выходном. Довольно часто, например – при работе с текстовыми редакторами, складывается впечатление, что входной и выходной файлы совпадают. На самом деле это не совсем так. Дело в том, что программы, написанные на универсальных языках программирования (Pascal, Basic, C) открывают файл только для чтения или только для записи. Для многократных манипуляций с одним и тем же файлом создается временный файл, который, по окончанию работы программы, сохраняется вместо первоначального.

Таким образом, сколько ни будь серьезное программирование требует умения работать с файлами.

Рассмотрим, как осуществляются манипуляции с файлами в TurboPascal. Ограничимся текстовыми файлами.

Первое, что нужно сделать – создать в заголовочной части программы специальную файловую переменную и указать ее тип. Файловая переменная для текстового файла задается так:

```
……
var       f: text;
…… .
```

Затем, уже в основном тексте программы, с этой файловой переменной нужно связать конкретный файл:

```
begin
……
assign(f, “путь к файлу и его имя”);
……
end.
```

Путь к файлу и его имя могут выглядеть, например, так: *“c:\work\input.txt”* – диск *c*, папка *work*, имя файла *input.txt*.

Третий шаг – открытие файла, связанного с переменной *f*, для манипуляций: чтения или записи. Открытие файла для *чтения* из него данных осуществляется оператором *reset(f)*, а открытие файла для *записи* осуществляется командой *rewrite(f)* (если файла с данным именем и расширением в указанной папке нет, то он будет создан).

Чтение данных из файла производится в переменную заданного типа. Очень важно, что при чтении из текстового файла программа сама интерпретирует данные в соответствии с типом переменной. Так, если данные считываются в переменную числового типа, программа игнорирует весь текст, не являющийся записью числа, и, естественно, читает только числовые данные. Если же считывание информации производится в переменную символьного типа (например – *string*), то даже числовые данные интерпретируются как текст.

Чтение данных из файла в переменную производится оператором *read(f, name_var) (readln(f, name_var))*, здесь *f* – файловая переменная, связанная с файлом, *name_var* – имя переменной, в которую считываются данные.

Листинг 1. Открытие текстового файла и чтение из него данных

```
program openfiles1;
uses crt;
var f: text;  a,b,c: real;
begin
  assign(f, 'c:\work\inp.txt');
  reset(f);
  readln(f, a);
  readln(f, b);
  readln(f, c);
  write(a,” “,b,” “,c);
  close(f); end.
```

Ниже приведен пример простой программы, создающей текстовый файл с именем “proba.txt” в папке work на диске C и открывающей его для записи. Затем в этот файл записывается приветствие “hello world”, и файл закрывается.

Листинг 2. Создание текстового файла и запись в него текста

```
program openfiles;
uses crt;
var f: text;
begin
  assign(f, 'c:\work\proba.txt');
  rewrite(f);
  write(f, 'hello world');
  close(f); end.
```

8.2 Числовые и символьные массивы

Рассмотрим один из наиболее популярных типов данных, используемых при решении задач по программированию – массивы.

В TurboPascal этот тип данных объявляется следующим образом:

…

Var a: array [1..100] of real; {одномерный массив из 100 действительных чисел} b: array [1..3][1..6] of integer; {двумерный целочисленный массив} c: array [1..20] of string; {одномерный массив строк}

…

Важное замечание: в части программы, содержащей описание переменных, процедур и функций, указывается размерность массива и тип его элементов, однако значения этих элементов там не задаются. Фактически, там происходит резервирование памяти для элементов массива. Поэтому, перед тем, как использовать некоторый элемент массива, нужно присвоить ему значение (выполнить инициализацию). Ниже приведен пример, когда всем элементам массива присваивается значение ноль.

…

```
For i:=1 to 100 do
a[i]:=0; …
```

8.3 Циклы

Известны различные приемы организации циклов, но сами циклы можно разделить на два основных цикла: цикл, в котором повторение производится заранее известное количество раз и цикл, количество повторений в котором определяется (вычисляется) логическим условием.

В первом случае цикл реализуется оператором for … to … do:

```
for i:=1 to N do
begin
оператор 1;
……
оператор k;
end;
```

Операторы, расположенные между логическими скобками

begin …. end;

повторяются N раз.

Рассмотрим теперь циклы второго типа. Их можно разделить на два типа.

В *первом типе* циклов логическое условие проверяется до выполнения группы операторов (если верно – производятся вычисления). Это оператор с предусловием. В TurboPascal эта схема реализуется циклом while:

```
while usl=%t do
begin
оператор 1;
……
оператор k;
end;
```

Во *втором типе* циклов сначала выполняется группа операторов, потом

проверяется условие (постусловие). Если оно оказывается ложным, то выполняется повтор вычислений. В языке TurboPascal для реализации таких циклов предусмотрена конструкция repeat … until:

```
repeat
оператор 1;
……
оператор k;
until usl=%f ;
```

8.4 Условный оператор

В любом языке программирования имеются операторы или конструкции, предназначенные для проверки логических (принимающих истинные или ложные значения) выражений. Основная конструкция для проверки условий в языке TurboPascal – конструкция if … then … , которая также может содержать ключевые слова else. Такие конструкции могут принимать как простой вид:

```
if usl=%t then оператор1 else оператор2 ,
```

так и более сложный:

```
if usl=%t then
  begin
группа операторов №1
 else
группа операторов №2
 end;
```

Заметим, что в группах операторов внутри условного оператора может повторно встречаться конструкция if … then.

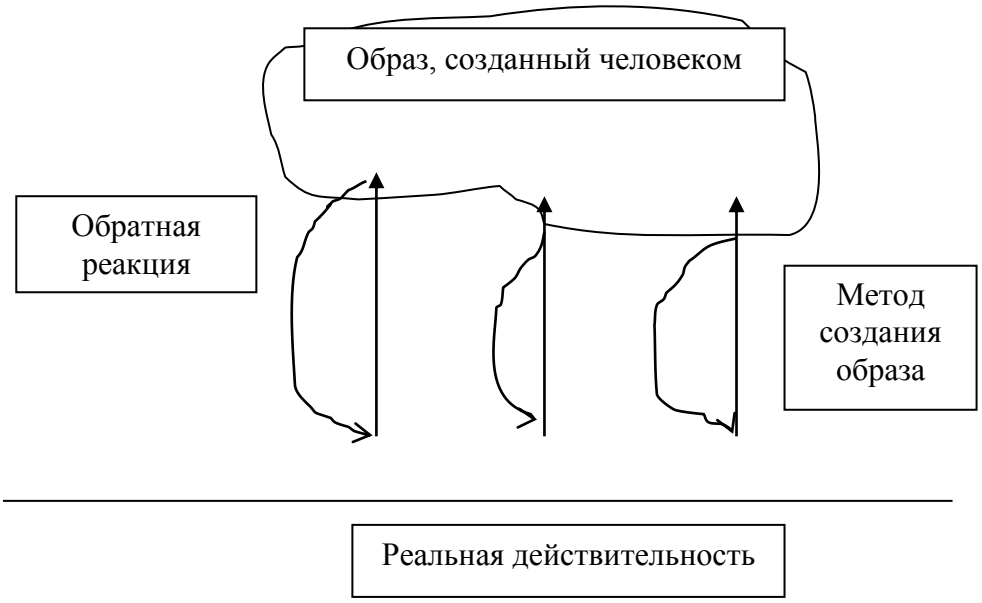
А.Е. Поличка, д.п.н., к.ф.-м.н,
профессор кафедры математики
и информационных технологий
ФГБОУ ВПО ДВГГУ

ФОРМАЛИЗАЦИЯ И МОДЕЛИРОВАНИЕ: ИСПОЛЬЗОВАНИЕ КОМПЬЮТЕРА

Не то, что мыслим мы – природа,
Не слепок, не бездушный лик,
В ней есть душа, в ней есть свобода,
В ней есть любовь, в ней есть язык...
Ф.И. Тютчев

1. ФОРМАЛИЗАЦИЯ

Рассмотрим один из подходов общения с реальной действительностью. Изобразим этот процесс в виде схемы:



На ней изображен процесс общения человека с реальной действительностью. Для начала реализации своих действий человек должен принять решение об этих действиях. Для этого он может создать образ реальной действительности и спланировать эти действия. Одним из методов для этого является использование человеком некоторого языка для создания образа. Язык состоит из алфавита, синтаксиса и семантики.

Под *алфавитом* понимают набор специальных символов, элементов и составных частей, принятых для рассматриваемого направления человеческого

знания или практики.

Синтаксис – это набор правил для создания комбинаций составляющих алфавита, своего рода слов, предложений и групп представлений на основе выбранного алфавита.

Семантика – объяснение смысла построенных слов, предложений и сочетаний элементов, составляющих алфавит.

Например, художник выражает свой взгляд на мир, свои чувства в своих картинах. А они создаются по правилам искусства живописи. Если другой человек, глядя на эту картину художника, испытывает чувства, подобные тем, которые испытывал создатель картины, то такую картину можно назвать классической. Аналогичный пример можно привести и про творчество поэта. Подобные творения человека на языке искусства называются художественными образами.

Если же для определения своих действий человек использует язык науки или производства, создаваемый им образ называется научной или производственной моделями.

Если модель создана с помощью языка математики — это *математическая модель*. Рассмотрим типовую задачу, показывающую необходимость изобретения специального языка для описания решения поставленной проблемы. Серия таких задач рассмотрена в пособии А. Г. Кушниренко и др [1990]. Ярким представителем этой серии задач является известная задача о «ханойской башне». К этой серии относится и старая русская шуточная задача о волке, козе и капусте.

ТЗ. «Ханойская башня»: «Волк-коза-капуста». На одном берегу у крестьянина находятся волк, коза и капуста. Вместе нельзя оставлять волка и козу, козу и капусту. Требуется их перевести на другой берег. В лодку помещаются только крестьянин и либо волк, либо коза, либо капуста.

Требование: привести не менее трех вариантов записи ответа.

По сути дела надо придумать некоторые языки записи ответа. На этом пути важным является понятие «Формализация».

Формализация – это уточнение содержания изучаемых предметов, которое давало бы право оперировать ими с помощью математических и логических методов [Пекелис В., 1990, стр. 393].

Жёсткие рамки формальных взаимоотношений, выделяющие «жёсткое существо дела», – вот то главное, что наиболее характерно для метода формализации, где основа основ – формальная система, представляющая собой совокупность четырёх элементов [Пекелис В., 1990, стр. 397, Концепции современного естествознания, 2000. Формализация. Язык науки]:

1. Исходящий алфавит – это то, что формализуется.
2. Система синтаксических правил – она предназначена чтобы из алфавита строить синтаксически правильные конструкции.
3. Система аксиом – она позволяет осуществлять эти правила.
4. Система семантических (смысловых) правил – она наполняет синтаксические конструкции смыслом.

Самый распространённый из формальных языков – язык математики.

Пример формальной системы. Рассмотрим язык графов. Основу языка составляют такие элементы как вершины, дуги и ребра. В качестве элементов синтаксиса приведем алгоритм Фалкерсона упорядочения дуг графа [Сборник задач и упражнений ... Под общ. ред. А.В. Кузнецова, 1995.].

Основу языка графов образуют вершины, ребра и дуги. Будем понимать под *графом* некоторое множество точек плоскости и множество линий их соединяющих все эти точки или некоторые из них. Точки множества называются *вершинами.* Линии, их соединяющие называются *ребрами*. Если указаны начальная и конечная вершины, то эта линии называется *дугами*. Граф, состоящий из дуг, называется *ориентированным* или *орграфом.* Граф, состоящий из ребер, называется *неориентированным.* Граф, состоящий из ребер

и дуг, называется *смешанным*. Вершины графа можно располагать по-разному. Полученные изображения описывают один и тот же граф, если между его составляющими можно установить взаимно однозначное соответствие. Такие изображения будем называть *изоморфными.*

При большом числе элементов рисунок графа теряет свою наглядность. Здесь может помочь другой язык, который будет применен для описания графов — язык матриц. *Матрицей* будем называть прямоугольную таблицу действительных чисел. Обозначение матрицы имеет вид

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ & & \dots & \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix}.$$

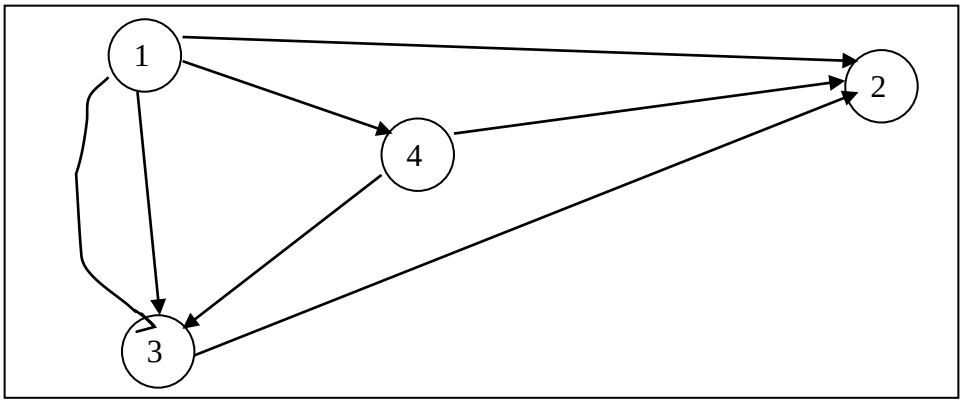
Матрицы будем обозначать большими латинскими буквами. Числа a_{ij} называются элементами, стоящими на пересечении i – той строки и j – того столбца.

Введем следующие матрицы, описывающие графы.

Матрица смежности вершин орграфа — это квадратная матрица n-го порядка (n-вершин). Строки и столбцы этой матрицы соответствуют вершинам графа. Элементы p_{ij} матрицы равны числу дуг, направленных из i-той вершины в j-тую вершину. Аналогично вводятся матрицы смежности дуг орграфа, смежности ребер неориентированного графа, инциденций орграфа (строки соответствуют вершинам, а столбцы — дугам орграфа).

ТЗ «Построение по графу матрицы смежности вершин»

Для заданного орграфа составить матрицу смежности вершин



Согласно определения матрицы смежности вершин орграфа имеем

Вершины	1	2	3	4
1	0	1	2	1
2	0	0	0	0
3	0	1	0	0
4	0	1	1	0

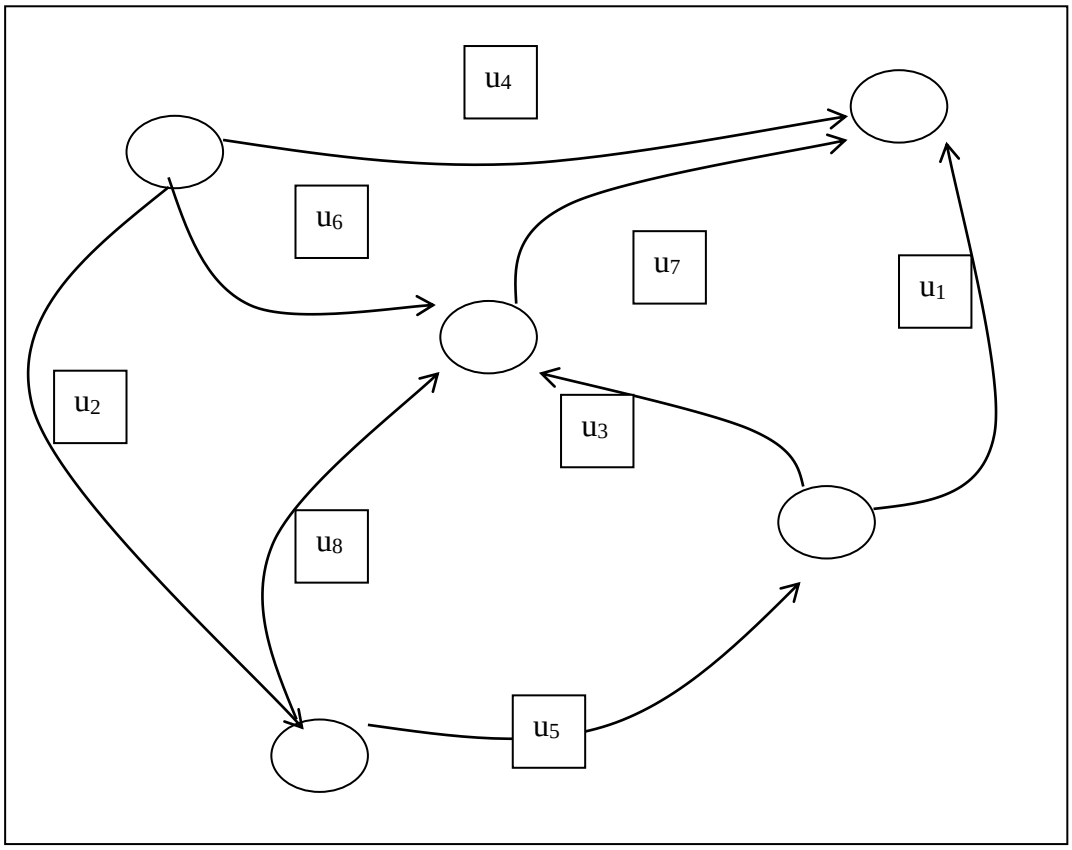
Теперь можно по заданной матрице изобразить граф.

ТЗ. «Изображение графа по заданной матрице смежности вершин»

Вершины	1	2	3	4
1	2	1	0	3
2	1	0	2	1
3	0	2	1	1
4	3	1	1	0

ТЗ «Упорядочение дуг орграфа графическим способом»

Дуги орграфа

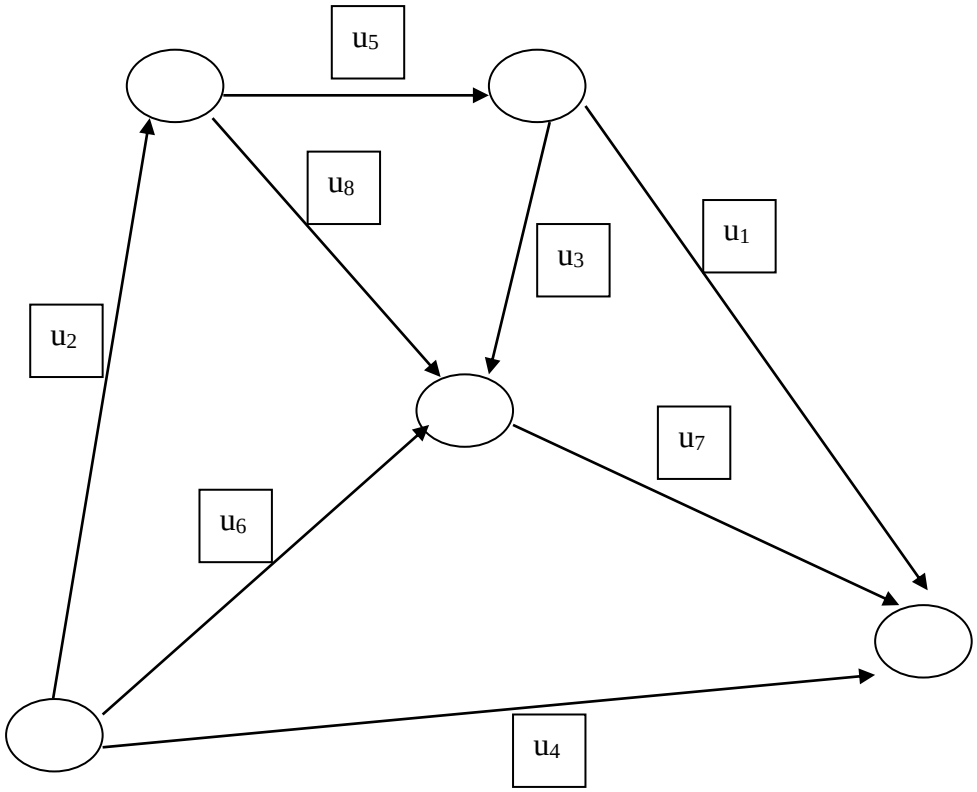


Алгоритм Фалкерсона: 1) найти дуги, не имеющие непосредственно предшествующих (они образуют первую группу); 2) вычеркнуть найденные

дуги; после этого появится по крайней мере одна дуга, не имеющая непосредственно предшествующей (в графе без дуг первой группы). Такие дуги составят вторую группу. Надо повторять этот шаг, пока все дуги не будут разбиты на группы.

В данном примере к первой группе относятся дуги u_2, u_4, u_6. После их вычеркивания непосредственно предшествующих не будут иметь дуги u_5 и u_8 (это дуги второй группы). К третьей группе отойдут дуги u_1 и u_3 и к четвертой группе отойдет дуга u_7.

Изобразим изоморфный граф, последовательно изображая дуги сначала первой, затем второй, третьей и четвертой групп.



Пример формализации – преобразования описательной модели в математическую. Естественные языки служат для создания описательных информационных моделей. С помощью формальных языков строятся формальные информационные модели (математические, логические и др.). Процесс построения информационных моделей с помощью формальных языков

называется формализацией. Одним из наиболее широко распространенных формальных языков является математический.

2. МОДЕЛИРОВАНИЕ КАК МЕТОД ПОЗНАНИЯ

Замещение одного объекта другим с целью получения информации о важнейших свойствах объекта-оригинала с помощью объекта – модели называется моделированием. Таким образом, моделирование может быть определено как представление объекта моделью для получения информации об объекте путем проведения экспериментов с его моделью. Теория замещения одних объектов (оригиналов) другими объектами (моделями) и исследования свойств объектов на их моделях называется теорией моделирования.

Моделирование применимо для тех процессов и систем, которые невозможно постигнуть путем непосредственного наблюдения. На этом пути модели классифицируют на материальные и идеальные. Первые — природные объекты, подчиняющиеся естественным законам. Вторые — идеальные образования, функционирующие по законам логики, отражающей мир.

При рассмотрении моделирования применим программно-целевой подход. Здесь для моделирования, прежде всего, необходимо четко определить цель моделирования.

Моделирование используется в тех случаях, когда сам объект либо труднодоступен, либо его прямое изучение экономически невыгодно и т.д. Различают ряд видов моделирования:

1. Предметное моделирование, при котором модель воспроизводит геометрические, физические, динамические или функциональные характеристики объекта. Например, модель моста, плотины, модель крыла самолета и т.д.

2. Аналоговое моделирование, при котором модель и оригинал описываются единым математическим соотношением. Примером могут

служить электрические модели, используемые для изучения механических, гидродинамических и акустических явлений.

3. Знаковое моделирование, при котором в роли моделей выступают схемы, чертежи, формулы. Роль знаковых моделей особенно возросла с расширением масштабов применения ЭВМ при построении знаковых моделей.

4. Со знаковым тесно связано мысленное моделирование, при котором модели приобретают мысленно наглядный характер. Примером может в данном случае служить модель атома, предложенная в свое время Бором.

5. Наконец, особым видом моделирования является включение в эксперимент не самого объекта, а его модели, в силу чего последний приобретает характер модельного эксперимента. Этот вид моделирования свидетельствует о том, что нет жесткой грани между методами эмпирического и теоретического познания.

С моделированием органически связана идеализация – мысленное конструирование понятий, теорий об объектах, не существующих и не осуществимых в действительности, но таких, для которых существует близкий прообраз или аналог в реальном мире. Примерами построенных этим методом идеальных объектов являются геометрические понятия точки, линии, плоскости и т.д. С подобного рода идеальными объектами оперируют все науки - идеальный газ, абсолютно черное тело, общественно-экономическая формация, государство и т.д.

При моделировании важен процесс создания условий для вариативного выбора способов решения задач, разработка нестандартных подходов. На этом пути интересна типовая задача «Способы измерений» [Пак Н.И., 1994].

ТЗ «Способы измерений»

Предложите несколько способов измерения (не менее семи): 1) высоты дерева; 2) высоты дома; 3) толщины бумажного листа; 4) диаметра мяча; 5) объема булыжника; 6) площади поверхности куриного яйца; 7) диагонали

кирпича; 8) ширины реки; 9) скорости течения реки; 10) скорости движения проезжающего мимо автомобиля.

3. ФОРМАЛЬНАЯ И НЕФОРМАЛЬНАЯ ПОСТАНОВКИ ЗАДАЧ

Продемонстрируем неформальную и формальную постановку задачи на примере построения сетевого графика [Сборник задач и упражнений .... Под общ. ред. А.В. Кузнецова, 1995; Власюк Л.И., Поличка А.Е., 1999].

ТЗ «Построение сетевого графика»

Язык теории графов позволяет решать оптимизационные задачи в сетевом планировании и управлении сложными комплексами взаимосвязанных работ. Этот язык позволяет более эффективно выбирать нужный вариант в множестве переборов всех вариантов решения. Для этой цели разработаны специальные методы, основанные на использовании сетевых графиков, являющихся графической моделью комплекса работ или производственного процесса.

Под сетевым графиком будем понимать ориентированный граф, в котором любые две вершины можно связать последовательностью дуг графа и в котором нет таких последовательностей дуг, у которых начало и конец совпадают (нет контуров). Дуги сетевого графика моделируют работы, а вершины — события (окончания работ).

Перед построением сетевого графика составляют список всех работ, входящих в комплекс. При этом необходимо четко представлять конечный результат (событие) каждой работы, знать продолжительность выполнения работы, а также предшествующие и последующие за ней работы. Сетевой график строится с соблюдением установленных правил.

Сетевое планирование – совокупность методов и приемов для планирования комплекса взаимосвязанных работ.

Сетевой график (сетевая модель, или просто сеть), представляет собой информационно-динамическую модель, в которой изображаются взаимосвязи и

результаты всех работ, необходимых для достижения конечной цели разработки.

В основе сетевого моделирования лежит изображение планируемого комплекса работ в виде графа.

Другими словами, *сетевой график* – это ориентированный граф без циклов и петель, ребра которого имеют одну или несколько числовых характеристик.

Ребрами изображаются на графе работы, а вершинами графа – события.

Работа – это протяженный во времени процесс, необходимый для свершения события и требующий затрат ресурсов. Действия, приводящие к достижению определенных результатов (событий). Работой следует считать и возможное *ожидание*, то есть пассивный процесс, не требующий затрат труда и материальных ресурсов, но требующий затрат времени.

Кроме работ *действительных* т.е. требующих затрат труда, времени и материальных ресурсов, существуют так называемые фиктивные работы (зависимости). *Фиктивная работа* (зависимость) – условная зависимость между событиями, которая вводится обычно для удобства изображения сети. Фиктивная работа не связана с затратами труда, времени и ресурсов.

Итак, работа бывает действительная, фиктивная и ожидание.

Событие – это результат произведенной работ, момент завершения процесса, отображающий отдельный этап выполнения проекта.

Событие, за которым непосредственно начинается данная работа (работы), называется *начальным* для данной работы, оно обозначается символом i.

Событие, которому непосредственно предшествует данная работа (работы), называется *конечным* для данной работы; оно обозначается символом j.

Событие, располагающееся в сети непосредственно перед данным событием, так, что между ними нет никаких промежуточных событий, называется *предшествующим.* Событие, располагающееся в сети непосредственно после данного события так, что между ними нет никаких промежуточных событий, называется *последующим.*

Первоначальное событие в сети, не имеющее предшествующих ему событий и отражающее начало выполнения всего комплекса работ, включенных в данную сеть, называется *исходным событием.* Событие, которое не имеет последующих событий и отражает конечную цель комплекса работ, включенных в данную сеть, называется *завершающим событием.*

Правила построения сетевых графиков.

Каждую стрелку в сетевом графике по возможности рисуют так, чтобы её конец находился правее начала, по возможности горизонтально.

1. Для удобства сетевой график строят без лишних пересечений стрелок. (При составлении чернового варианта не следует увлекаться внешним видом сети).

2. Следят за тем, чтобы во все вершины сети, кроме той, которая соответствует исходному событию, входила по меньшей мере одна стрелка, так как все события, кроме исходного, имеют предшествующую работу.

3. Следят за тем, чтобы из всех вершин сети, кроме той, которая соответствует завершающему событию, выходили стрелки, так как все события, кроме завершающего, имеют последующую работу.

4. Следят за тем, чтобы в сетевом графике не образовывалось циклов.

5. Если одно событие служит началом для двух или более работ, после завершения которых начинается выполнение следующей работы, то вводится штриховая стрелка (условная зависимость) и дополнительное событие со своим номер.

6. Если какие-то работы могут начаться до полного завершения предыдущей работы, то её следует разбить на части и считать каждую из них самостоятельной.

7. На сетевом графике следует чётко отражать последовательность выполнения отдельных работ и их взаимосвязи. В помощь вводятся штриховые стрелки (условные зависимости) и дополнительные вершины (события).

Сетевое планирование и управление осуществляется в два последовательно проводимых этапа:

1. составление исходного плана;
2. оперативный контроль за ходом выполнения работ.

Первый этап сетевого планирования содержит в себе следующий перечень работ, проводимых в основном последовательно:

а) определение структуры разработки, назначение руководителей и ответственных исполнителей;

б) выявление и описание ответственными исполнителями всех событий и работ, необходимых для выполнения задания;

в) определение времени выполнения каждой работы;

г) составление («сшивание») первичных сетей ответственными исполнителями, частных и сводных сетей и т.д.;

(Мы эти 4 пункта будем называть просто построением сетевого графика, а время выполнения работ будем считать известным)

д) определение в сети критического пути и резервов времени;

е) анализ сетей и их оптимизация.

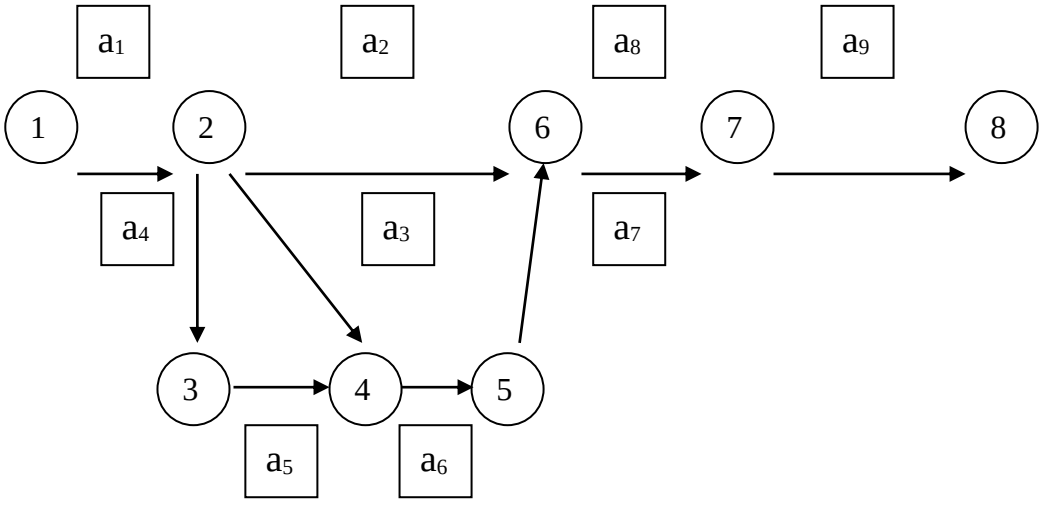
Оптимизацией сетей заниматься не будем, а ограничимся их анализом и экономическими выводами о сложившейся ситуации. Постоим вариант сетевого графика по организации на промышленной выставке зала для демонстрации образцов продукции, выпускаемых производственным объединением.

Таблица. Рассмотрим перечень работ в виде таблицы

Содержание работы	Исходная работа	Опирается на работу
Отбор образцов продукции для выставки	a_1	—
Изготовление информационных и рекламных материалов, указателей, надписей и т.п.	a_2	a_1
Изготовление стендов и другого оборудования для установки образцов в демонстрационном зале	a_3	a_1
Доставка образцов в демонстрационный зал	a_4	a_1
Доставка в демонстрационный зал стендов и другого оборудования	a_5	a_4
Монтаж стендов и другого оборудования	a_6	a_5
Установка образцов продукции на стендах	a_7	a_3 ,a_6
Оформление залов и стендов указателями, надписями, рекламными и информационными материалами	a_8	a_2 ,a_7
Репетиция открытия выставки	a_9	a_8

Работы обозначены в порядке их следования. С учетом технологической зависимости работ друг от друга устанавливается их последовательная связь. Именно для каждой работы указывается на какие работы она «опирается».

В начале отмечаем, что работа a_1 не опирается ни на какую работу, поэтому она изобразится дугой, выходящей из события I, означающего исходный момент, с которого начинается выполнение рассматриваемого комплекса работ. На работу a_1 опираются работы a_2, a_3 и a_4, поэтому дуги, соответствующие этим работам, на сетевом графике будут следовать непосредственно за дугой a_1 от события 2, означающего момент окончания работы a_1 и начало работ a_2, a_3 и a_4.. На работу a_4 опирается работа a_5 , а на нее – работа a_6 . Это отображено на сетевом графике следующими друг за другом дугами a_5 и a_6 . Работа a_7 опирается на работы a_3 и a_6 , поэтому дуга a_7 исходит из события 5, означающего момент, к которому завершены обе эти работы. Аналогичная ситуация имеет место и для работы a_8 , исходящей из события 6, которое означает факт выполнения работ a_2 и a_7. Дуга a_9 соответствует последней работе, а конечное ее событие 8 означает момент завершения работ всего рассматриваемого комплекса.



Задание. Выбрать комплекс работ (подготовка к занятиям, подготовка к контрольной работе и экзамену, организация праздника, семейного торжества, ремонта квартиры, покупки нового оборудования и т.д.). Построить сетевой график для выбранного комплекса работ.

4. ОСНОВНЫЕ ТИПЫ ИНФОРМАЦИОННЫХ МОДЕЛЕЙ (ТАБЛИЧНЫЕ, ИЕРАРХИЧЕСКИЕ, СЕТЕВЫЕ)

Табличные модели. Одним из наиболее часто используемых типов информационных моделей является таблица, которая состоит из строк и столбцов.

ТЗ. «Табличная информационная модель»

Построим, например, табличную информационную модель, отражающую стоимость отдельных устройств компьютера. Пусть в первом столбце таблицы содержится перечень объектов (устройств), входящих в состав компьютера, а во втором – их цена.

Наименование устройства	Цена, у.е.
Системная плата	80
Процессор Celeron (800 МГц)	60
Память 64 Мб	8
Жесткий диск 40 Гб	130
Дисковод 3,5"	14
Видеоплата 8 Мб	30
Монитор 15"	180
Звуковая карта 16 бит	30
Дисковод CD-ROM 50	40
Корпус	25
Клавиатура	10
Мышь	5

Задание. Выбрать предметную область и для нее построить табличную информационную модель.

Иерархические модели. Нас окружает множество различных объектов, каждый из которых обладает определенными свойствами. Однако некоторые группы объектов имеют одинаковые общие свойства, которые отличают их от объектов других групп. Группа объектов, обладающих одинаковыми общими свойствами, называется классом объектов. Внутри класса могут быть выделены подклассы, объекты которых обладают некоторыми особенными свойствами, в свою очередь, подклассы можно делить на еще более мелкие группы и т. д. Такой процесс называется процессом классификации. При классификации объектов часто применяются информационные модели, которые имеют иерархическую (древовидную) структуру. В иерархической информационной модели объекты распределены по уровням, причем элементы нижнего уровня – входят в состав одного из элементов более высокого уровня.

ТЗ. «Иерархическая информационная модель». На этом рисунке изображена информационная модель, которая позволяет классифицировать современные компьютеры.

Задание. Выбрать предметную область и построить для нее иерархическую информационную модель.

Форма представления информационной модели

Форма представления информационной модели зависит от способа кодирования (алфавита) и материального носителя.

Воображаемое (мысленное или интуитивное) моделирование - это мысленное представление об объекте.

Компьютерная модель – это созданный за счет ресурсов компьютера виртуальный образ, качественно и количественно отражающий внутренние свойства и связи моделируемого объекта, иногда передающий и его внешние характеристики.

Компьютерное моделирование включает в себя процесс реализации информационной модели на компьютере и исследование с помощью этой модели объекта моделирования – проведение вычислительного эксперимента.

Компьютерная модель представляет собой материальную модель, воспроизводящую внешний вид, строение или действие моделируемого объекта посредством электромагнитных сигналов. Разработке компьютерной модели предшествуют мысленные, вербальные, структурные, математические и алгоритмические модели.

Информационное моделирование в информатике – это компьютерное моделирование, применимое к различным предметным областям.

5. ОСНОВНЫЕ ЭТАПЫ КОМПЬЮТЕРНОГО МОДЕЛИРОВАНИЯ

1. Определение предмета компьютерного моделирования, постановка задачи. Предметом компьютерного моделирования могут быть: экономическая деятельность фирмы или банка, промышленное предприятие, информационно-вычислительная сеть, технологический процесс, любой реальный объект или процесс. Цели и задачи компьютерного моделирования могут быть различными, однако наиболее часто моделирование является центральной процедурой системного анализа (совокупность методологических средств, используемых для подготовки и принятия решений экономического, организационного, социального или технического характера).

2. Определение объекта моделирования – широкое понятие, включающее объекты живой или неживой природы, процессы и явления действительности. Сама модель может представлять собой либо физический, либо идеальный объект. Первые называются натурными моделями, вторые – информационными моделями. Например, макет здания – это натурная модель здания, а чертеж того же здания – это его информационная модель, представленная в графической форме (графическая модель).

3. Разработка концептуальной модели, выявление основных элементов системы и элементарных актов взаимодействия. Это абстрактная модель, определяющая состав и структуру объекта, свойства элементов и причинно-следственные связи, присущие анализируемой объекту и существенные для достижения целей моделирования.

4. Формализация (переход к математической модели). Это сведение содержания к форме. Формулы, описывающие физические процессы, – это формализация этих процессов. Радиосхема электронного устройства – это формализация функционирования этого устройства. Ноты, записанные на нотном листе, – это формализация музыки и т.п. Формализованная информационная модель – это определенные совокупности знаков (символов),

которые существуют отдельно от объекта моделирования, могут подвергаться передаче и обработке.

5. Создание алгоритма. Формы представления алгоритмов:

• *словесная* (записи на естественном языке);

• *графическая* (изображения из графических символов);

• *псевдокоды* (полуформализованные описания алгоритмов на условном алгоритмическом языке, включающие в себя как элементы языка программирования, так и фразы естественного языка, общепринятые математические обозначения и др.);

• *программная* (тексты на языках программирования).

Объект, который будет выполнять алгоритм, обычно называют исполнителем. Исполнитель – объект, который выполняет алгоритм.

6. *Написание программы.* Программисты часто начинают с создания *прототипа* программы. Прототип программы обычно содержит графический интерфейс пользователя (раскрывающиеся меню, кнопки, диалоговые окна).

Написание программы:

• (1) Формулировка общей идеи программы.

• (2) Принятие решения о потенциальных пользователях программы.

• (3) Принятие решения о типе компьютера, на котором программа будет выполняться.

• (4) Выбор языка программирования.

• (5) Проектирование структуры программы с помощью псевдокода или другого инструмента.

• (6) Написание программы.

• (7) Тестирование программы без участия пользователей. Этот этап называют *альфа-тестированием*.

• (8) Исправление ошибок, обнаруженных во время альфа-тестирования.

Этапы 7 и 8 повторяются многократно.

• (9) Передача копий программы пользователям для ее тестирования «в полевых условиях». Этот этап называют *бета-тестированием.*

• (10) Исправление ошибок, обнаруженных во время бета-тестирования.

Этапы 9 и 10 повторяются многократно.

• (11) Выпуск окончательной версии программы. Лишь с этого момента разработчики гарантируют безупречную работу программы (естественно, гарантия не 100%-ная).

Если выбор технических средств в настоящее время не вызывает особых затруднений, то выбор программных средств зачастую довольно сложен.

В настоящее время известно более 500 языков моделирования. Такое множество языков частично обусловлено разнообразием классов моделируемых систем, целей и методов моделирования. Однако желание упростить и ускорить процесс создания моделей привело к реализации идеи автоматизации программирования имитационных моделей. Программа создается автоматически по одной из формализованных схем на основании задаваемых исследователем параметров системы, внешних воздействий и особенностей функционирования. Это наиболее перспективное направление развития средств имитационного моделирования. Опыт развития теории и практики имитационного моделирования в нашей стране и за рубежом показывает, что наиболее эффективным средством являются специальные имитационные языки, которых к настоящему времени создано уже немало и многие из них эффективно используются, особенно за рубежом, где ни один крупный проект не реализуется без проверки на имитационной модели. Наиболее известны языки: GPSS, GASP, SIMSCRIPT и DYNAMO, реализующие различные подходы к моделированию.

7. Планирование и проведение компьютерных экспериментов: рандомизация эксперимента, т.е. проведение опытов в случайном порядке; одновременное варьирование всеми переменными; четкая стратегия проведения эксперимента, принятие обоснованных решений на каждом его этапе.

8. Анализ и интерпретация результатов – насколько близка созданная модель реально существующему явлению, (верификация, чаще всего решается ретроспективным методом или методом контрольных точек; обычно системе задаются такие значения параметров и начальных значений, в которые она должна прийти через определенное количество шагов модельного времени к состоянию, известному тем или иным образом исследователю); насколько пригодна данная модель для исследования новых, еще не опробованных значений аргументов и параметров системы.

6. *ЛР. ОСНОВНЫЕ ЭТАПЫ КОМПЬЮТЕРНОГО МОДЕЛИРОВАНИЯ. ОПРЕДЕЛЕНИЕ ОБЪЕКТА МОДЕЛИРОВАНИЯ*

Цель. Научиться называть не менее семи этапов и объектов компьютерного моделирования для компьютерного сопровождения профессиональной деятельности.

Ход работы

1. Завести папку с именем «Аббревиатура специальности_Номер группы_Фамилия»

2. Завести в своей папке папку данной лабораторной работы с именем «Аббревиатура специальности_Номер группы_Фамилия_Номер лабораторной работы».

3. Создать в этой папке документ «Аббревиатура специальности_Номер группы_Фамилия_Номер лабораторной работы_А». Указать в нем в правом верхнем углу информацию:«Аббревиатура специальности_Номер группы_Фамилия_Номер лабораторной работы_А». На середине следующей строки напечатать номер и название практикума.

4. Проанализировать Интернет-источники и выделить семи этапов и объектов компьютерного моделирования для компьютерного сопровождения профессиональной деятельности. Результат анализа представить в виде следующих таблиц.

Таблица 1. Основные этапы компьютерного моделирования

	Виды основных этапов компьютерного моделирования для своей профессиональной деятельности	Описание с указанием гиперссылки адреса источника

Таблица 2. Виды объектов компьютерного моделирования в своей профессиональной деятельности

	Виды объектов компьютерного моделирования в своей профессиональной деятельности	Описание с указанием гиперссылки адреса источника

5. Записать в созданный файл с именем «Аббревиатура специальности_Номер группы_Фамилия_Номер лабораторной работы_А» всю разработанную информацию.

6. Разработать требования к оформлению *электронной газеты* для чего заполнить таблицу

Таблица 3. Основные виды и требования к оформлению электронной газеты для выбранного предмета исследования

	Основные виды и требования к оформлению электронной газеты для выбранного предмета исследования	Требование 1	Требование 2	…

Выписать по убыванию частоты цитирования требования к оформлению электронной газеты: _____.

На основании проведенного анализа дать описание: *электронная газета – это…*

7. Создать второй файл с именем «Аббревиатура специальности_Номер группы_Фамилия_Номер лабораторной работы_Р». Указать в нем в правом верхнем углу информацию: «Аббревиатура специальности_Номер группы_Фамилия_Номер лабораторной работы_Р». Выберете по одной позиции в каждой таблице и опишите и результаты анализа, оформив в виде электронной газеты, используя любой процессор оформления газет и результаты таблицы 2.3. Записать эту газету во второй созданный файл с именем «Аббревиатура специальности_Номер группы_Фамилия_Номер лабораторной работы_Р»

8. Создать третий файл с именем «Аббревиатура специальности_Номер группы_Фамилия_Номер лабораторной работы_П», разместив в нем презентацию (не менее пяти слайдов) по защите результатов лабораторной работы.

7. ВЫБОР МЕТОДА РЕШЕНИЯ ЗАДАЧИ. ПОНЯТИЕ ЧИСЛЕННЫХ МЕТОДОВ

На практике в большинстве случаев найти точное решение возникшей математической задачи не удается. Это происходит главным образом не потому, что мы не умеем этого сделать, а потому, что искомое решение обычно не выражается в привычных для нас элементарных или других известных функциях. Поэтому разработаны *численные методы решения задач*, т.е. методы, позволяющие получить в результате применения последовательности осуществимых вычислительных операций численный ответ. Раздел математики, в котором разрабатываются численные методы решения задач, называется *теорией численных методов (или методы вычислений).*

Решение, полученное численным методом, обычно является приближенным, т.е. содержит некоторую погрешность, источниками погрешности приближенного решения являются:

1) несоответствие математической задачи (математической модели) изучаемому реальному явлению;

2) погрешность исходных данных (входных параметров);

3) погрешность метода решения;

4) погрешности округления в арифметических и других действиях над числами.

Численные методы в большинстве случаев сами по себе являются приближенными, т.е. даже при отсутствии погрешностей во входных данных и при идеальном выполнении арифметических действий они дают решение исходной задачи с некоторой погрешностью, *называемой погрешностью метода*.

К численному методу, кроме требования достижения заданной точности, предъявляется ряд других требований. Предпочтение отдается методу, который реализуется с помощью меньшего числа действий, требует меньшей памяти ЭВМ и, наконец, является логически более простым, что способствует более быстрой его реализации на ЭВМ.

8. ЧИСЛЕННЫЕ МЕТОДЫ РЕШЕНИЯ АЛГЕБРАИЧЕСКИХ УРАВНЕНИЙ

Рассмотрим уравнение с одной переменной. Это уравнение может быть записано в виде $F(x) = 0$ или $f(x) = \varphi(x)$. При решении уравнения необходимо установить:

a) существует ли решение;

b) вычислить решение точно или приближенно (с заданной точностью ε).

Приближенное решение (x^*) – корень уравнения: $a < x^* < b$ и $b-a \le \varepsilon$, т.е. вместо решения (x) находится последовательность $\{x_n\}$, которая сходится к x.

Процесс нахождения приближенного решения разбивается на два этапа:

(I) – отделение корней;

(II) – уточнение корня до заданной степени точности.

Отделение корней означает: необходимо найти интервалы, в которых содержатся по одному корню. Отделение корней можно проводить многими способами. Например, графически или аналитически.

Графический способ:

- представить уравнение в виде $f(x)=\varphi(x)$,

- построить графики $y=f(x)$, $y=\varphi(x)$,

- на чертеже найти интервал, на который приходится точка пересечения этих графиков.

Аналитический способ основан на теореме 1:

Если функция F(x) непрерывна на отрезке [a,b], принимает на концах отрезка значения разных знаков, а производная F'(x) сохраняет постоянный знак внутри отрезка, то внутри отрезка существует корень уравнения F(x)=0, и притом единственный.

Пример 1. Решить графически уравнение $\ln|x|+\cos(\pi x)=0$.

1) Представим уравнение в виде $\ln|x|=-\cos(\pi x)$.

2) Построим в одной системе координат графики функций $y=\ln|x|, \quad y=-\cos(\pi x)$ (Рис.1).

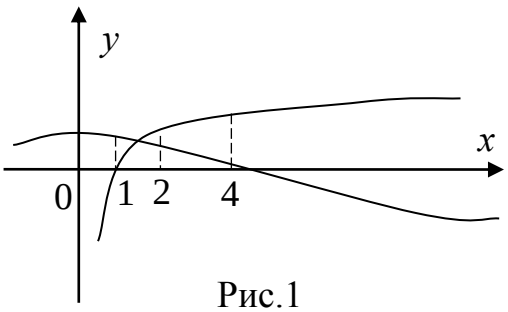


Рис.1

3) очевидно, что корень уравнения $x^*\in[1,2]$ и $x\approx 1.6$.

Уточнение корней можно проводить различными методами:

1) *метод половинного деления* заключается в том, что отрезок (в котором содержится корень) делится пополам и выбирается та половина, на концах

которого функция приобретает значения разных знаков, и этот процесс продолжается до получения необходимой точности ε.

Рассмотрим алгоритм нахождения корня уравнения *f*(x)=0, находящегося на интервале [a,b] с заданной точностью ε. Входными данными являются границы интервала и заданная точность. В блоке вычисления находим середину интервала [a,b]. Затем находим из двух полученных отрезок, на концах которого значения функции разных знаков и продолжаем вычисления до получения заданной точности ε, т.е. проверяем условие |b-a|< ε.

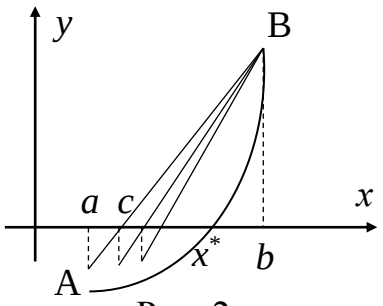


Рис. 2

2) *Метод хорд* заключается в том, что мы находим близкую к точке x^* точку пересечения хорды AB с осью Ox. Графическая интерпретация метода хорд изображена на рис.2.

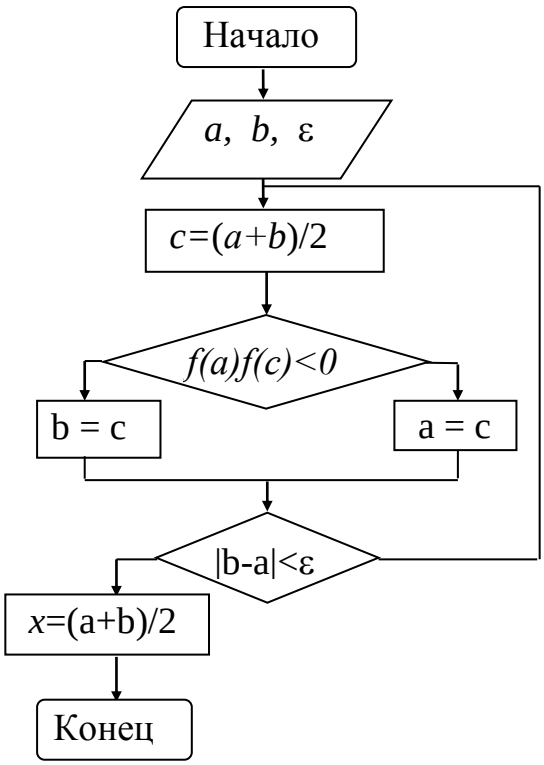
Уравнение хорды: $\frac{x-a}{b-a}=\frac{f(x)-f(a)}{f(b)-f(a)}$. При $f(x)=0$ $x = c$ – это точка пересечения хорды с осью Ox. Следовательно, $\frac{c-a}{b-a}=\frac{0-f(a)}{f(b)-f(a)}$.

Откуда

$$c = a - \frac{b-a}{f(b)-f(a)} f(a),$$

$$c_1 = c - \frac{b-c}{f(b)-f(c)} f(c), \quad \ldots \quad \ldots \quad \ldots$$

$$c_{n+1} = c_n - \frac{b-c_n}{f(b)-f(c_n)} f(c_n).$$



Данные формулы выведены для фиксированной точки B.

Признак фиксирования начальной точки: значения функции и *f''* в этой точке должны иметь один знак.

9. ЗАДАНИЯ

Задание: Формулы для фиксированной точки А вывести самостоятельно.

Пример. Найдем приближенное значение корня уравнения $x^3+x^2-3=0$.

1) Графически находим интервал корня – [0.5, 1.5].

Докажем, что в этом интервале существует единственный корень: найдем значения функции $f(x)=x^3+x^2-3$ в концах отрезка – $f(0.5)<0$ и $f(1.5)>0$; найдем значение производной функции в точках отрезка – $f'(x)=3x^2+2x > 0$ на всем отрезке [0.5, 1.5].

Действуя по алгоритму метода половинного деления, вычисляем: $c_1=(0.5+1.5)/2=1$ и $f(1)<0 \Rightarrow a=1$,

$c_2=(1+1.5)/2=1.25$ и $f(1.25)>0 \Rightarrow b=1$,

$c_3= (1.25+1)/2=1.125$ (вычислим точность корня на данном шаге по формуле (1): $|c_3-c_2|=0.125=\varepsilon$).

Тогда $x^*=1.1875$(при $\varepsilon=0.125$).

Реализация методов на ЭВМ

Составление словесного и графического алгоритма решения уравнения.

I. Отделение корней:

а) Графическим или аналитическим способом находим наименьший интервал [a,b], в котором находится корень.

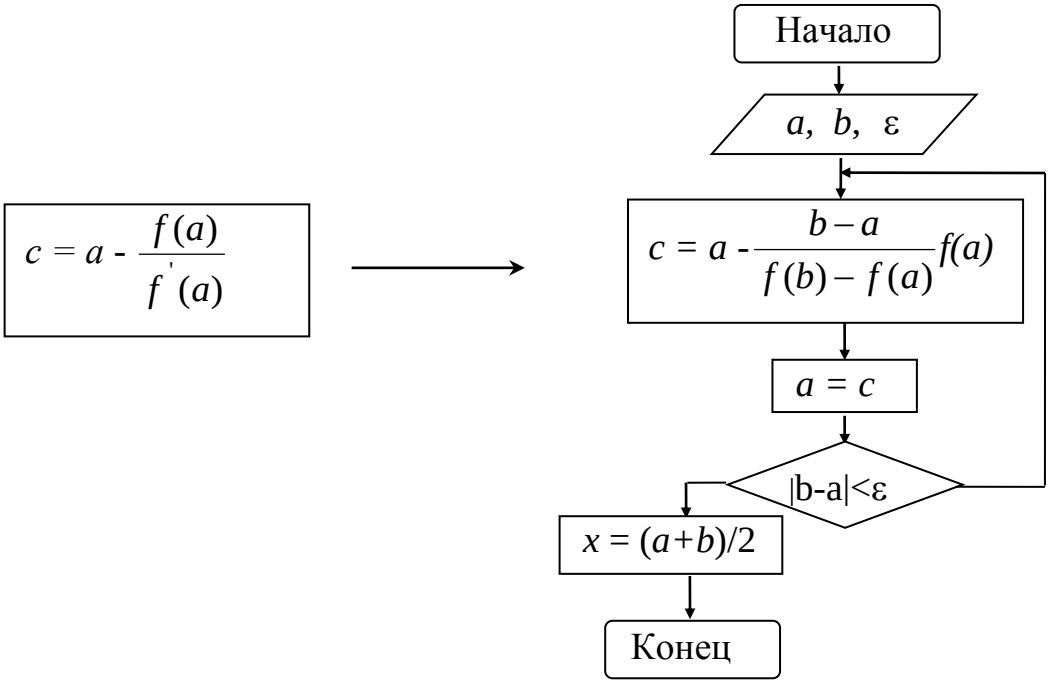
б) Докажем, что этот корень единственный, т.е. проверим условие: $f(a)f(b)<0$ и f' сохраняет знак на всем интервале.

II. Уточнение корня:

а) Выбираем метод приближения корня.

б) Для метода хорд и касательных выберем фиксированную точку из условия $f(z)f''(z)>0$, где z-крайние точки интервала.

с) Составим алгоритм в виде блок-схемы для приближения методом хорд и фиксированной точки – *b* (для преобразования данного алгоритма в метод



касательных необходимо заменить блок вычислений).

Задание для самостоятельной работы.

Выясните, какие изменения необходимо внести в алгоритм, в котором фиксированная точка – *a*.

Разработка способов реализации алгоритмов на ЭВМ.

Построение программы для решения данных уравнений:

1) $x-3\cos^2 x=0$,2) $2\ln x-1/x=0$ с точностью $\varepsilon=10^{-4}$.

Решение 1). Действуя по алгоритму, сначала найдем:

Интервал, в котором находится положительный корень. $x^* \in [0.5,1]$. Докажем, что этот корень единственный – f(0.5)<0, f(1)>0. Присвоим a=0.5, b=1. Составим для полученных данных программу приближения корня и найдем приближенные корни уравнения.

Ответы: 1) половинного деления – 0.9755;

2) хорд – 0.9798;

Задание для выполнения

1. Отделить корни уравнения графическим способом и доказать существование единственного корня на каждом отрезке.

2. Один из корней уточнить с точность до 10^{-4} методом:

а) итерации,

б) комбинированным

– касательных и хорд,

– касательных и половинного деления,

– хорд и половинного деления.

3. Оценить количество операций по разным методам, сравнить время приближения.

4. Сделать вывод об экономичности методов для данного уравнения и данного корня.

Варианты заданий:

*Уравнение***:**

1. $x^3-2x-5=0$
2. $x^3-2x^2+3x-5=0$
3. $x^3-3x^2-10=0$
4. $x^4-4x^3+3x^2-2x-2=0$
5. $x^4-x^3-9x^2+102x-10=0$
6. $x^4-3x^3+2x^2-5x-1=0$
7. $x^4+4x^3+6x^2+12x-10=0$
8. $x^4-5x^3-4x^2-3x+12=0$
9. $x^4-3x^3+3x^2-12=0$
10. $x^4-8x^3-2x^2+16x-3=0$

10. ПРОГРАММЫ РЕШЕНИЯ АЛГЕБРАИЧЕСКИХ УРАВНЕНИЙ

1) program метод касательных;

```
const e=0.00001;
var a,b,c,x,x1: real;
Function f(x:real): real;
```

```
begin
f:=        ; {заданная функция}
      end;

      Function f1(x:real): real;
begin
f1:=        ; {первая производная функции}
end;
Function f2(x:real): real;
      begin
      f2:=        ; {вторая производная функции}
end;
      begin
      write ('a = ');
      readln (a);
      write ('b = ');
      readln (b);
      if f(a)*f2(b)<0 then
      begin
      c:=a;
      a:=b;
      b:=c;
      end;
      x:=a;
      repeat
      x1:=x;
      x:=x1-f(x1)/f1(x1);
      until abc(x-x1)<e;
      writeln ('решение = ', x);
```

```
end.

2) program метод хорд;
const e=0.00001
var  a,b,c,x,x1: real;
Function f(x:real): real;
begin
f:=          ;
end;
Function f2(x:real): real;
begin
f2:=         ;
end;
begin
write ('a = ');
readln (a);
write ('b = ');
readln (b);
if f(a)*f2(b)<0 then
begin
c:=a;
a:=b;
b:=c;
end;
x:=a;
repeat
x1:=x;
x:=x1-(b-f(x1))*f(x1)/(f(b)-f(x1));
until abc(x-x1)<e;
```

```
writeln ('решение = ', x);
end.
```

3)
```
program Метод итерации;
const e=0.00001
var a,b,c,x,y: real;
Function f(x:real): real;
begin
f:=          ;
end;
begin
x:=0;
repeat
x1:=x;
x:=f(x1);
until abc(x-x1)<e;
writeln ('решение = ', x);
end.
```