Министерство образования РФ
Астраханский государственный педагогический университет

Ю. Ю. Тарасевич

Элементы дискретной математики для программистов

Астрахань, 2002

ББК 22.174

Юрий Юрьевич Тарасевич
Элементы дискретной математики для программистов. — Электронное учебное пособие. — Астрахань: Астраханский государственный педагогический университет, 2002.

Оглавление

Введение

Дискретная математика включает следующие разделы: теорию графов, комбинаторный анализ, теорию кодирования, математическую логику, вычислительную математику.

Объектами исследования дискретной математики являются конечные группы, графы, конечные автоматы, машина Тьюринга, клеточные автоматы и т. д.

Глава 1

Теория графов

1.1. Введение

Обычно возникновении теории графов связывают со следующей задачей, которую в 1736 г. рассмотрел Леонард Эйлер и доказал её неразрешимость.

Задача 1.1 (Задача о Кёнигсбергских мостах). *Может ли пешеход обойти все мосты (рис. 1.1), пройдя по каждому из них только один раз, и вернуться в исходную точку?*

Еще одной классической задачей теории графов является задача четырех красок, сформулированная А. де Морганом в 1850 году.

Задача 1.2 (Задача четырех красок). *Можно ли любую карту так раскрасить четырьмя красками, чтобы никакие две области, имеющие общий участок границы, не были окрашены в один и тот же цвет?*

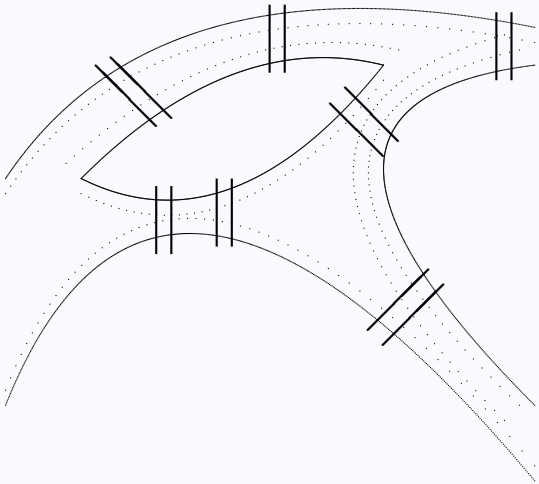


Рис. 1.1. Схема расположения мостов в Кёнигсберге.

К широко известным задачам относится так же следующая задача, предложенная Кэрроллом.

Задача 1.3 (Задача о домиках и колодцах). *Имеются три домика и три колодца. Можно ли проложить тропинки от каждого домика к каждому колодцу так, чтобы тропинки не пересекались?*

1.2. Основные определения и обозначения

Будем обозначать V — множество *точек* (*вершин*). Обозначение происходит от слова vertex — вершина. Обозначение $v \in V$ означает, что v — вершина из множества V.

Линии, соединяющие вершины (v_i, v_j), называются *рёбрами* E. От английского edge — ребро.

Альтернативное определение. *Ребро* — упорядоченная или неупорядоченная пара вершин. $E = (a, b), \quad a, b \in V$.

Граф (graph) G — множество вершин V и набор рёбер E.

$$G = G(V, E)$$

Иногда граф называют *сетью* (network), вершины — *узлами* (node), рёбра — *связями* (bond).

Неупорядоченная пара вершин называется (неориентированное) *ребро* (undirected edge).

$$E = (a, b) = (b, a)$$

Граф, содержащий только неориентированные рёбра, называется *неориентированный граф*.

Упорядоченная пара вершин называется *ориентированным ребром* (directed edge) или *дугой* (arc).

$$(a, b) \neq (b, a)$$

Граф, содержащий только ориентированные ребра (дуги), называется *ориентированным графом* (directed graph) или иногда *орграфом*.

Граф, содержащий как ориентированные, так и неориентированные ребра, называется *смешанным* или *частично ориентированным* графом.

Говорят, что ребро $E = (a, b)$ *инцидентно* вершинам a и b. Вершины a и b инцидентны ребру E.

Вершина, не инцидентная никакому ребру, называется *изолированной*.

Граф, состоящий только из изолированных вершин, называется *нуль-графом* или *вполне несвязным* или *пустым*.

Две вершины называются *смежными*, если существует соединяющее их ребро.

Два графа G_1 и G_2 называются *изоморфными*, если существует взаимно однозначное соответствие между множествами вершин, такое что, число ребер, соединяющих две любые вершины в G_1, равно числу ребер в G_2.

Степень или *валентность* вершины a — число ребер, инцидентных вершине $\rho(a)$.

Для изолированной вершины $\rho(a) = 0$.

Вершина степени 1 называется *висячей* или *концевой*.

В случае ориентированного графа говорят о *полустепени исхода* и *полустепени захода*, то есть числе ребер входящих в данную вершину и исхо-

дящих из нее.

Если полустепень исхода вершины равна нулю, то такая вершина называется *стоком*.

Если полустепень захода вершины равна нулю, то такая вершина называется *источником*.

Граф, имеющий только один источник и только один сток, называется *сетью*.

Полный граф — такой граф, ребрами которого являются *все* возможные пары вершин из V. В полном графе любые две вершины смежные. Полный граф с n вершинами обозначают K_n.

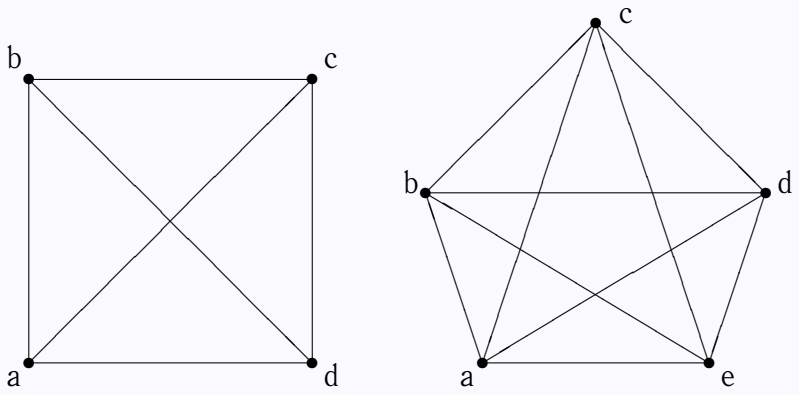


Рис. 1.2. Полные графы K_4 и K_5.

Упражнение 1.1. *В полном графе n узлов. Сколько в нем рёбер?*

Граф называется *двудольным*, если его вершины можно раскрасить двумя цветами так, что вершины одного цвета имеют смежные вершины только другого цвета. Полный двудольный граф обозначают $K_{n,m}$.

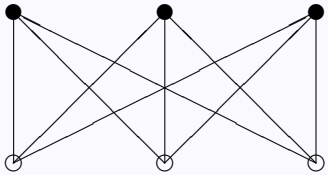


Рис. 1.3. Двудольный граф $K_{3,3}$.

Звездный граф — граф, который определяется вершиной a и состоит из ребер, имеющих a концевой точкой. Звездный граф является полным двудольным графом $K_{1,n}$.

Ребро, у которого обе концевые точки совпадают, называется *петлей* (loop).

$$L = (a, a)$$

Вершины могут соединяться несколькими ребрами. Такие ребра называются *кратными*.

$$E_i = (a, b)_i$$

Граф, который не содержит петель и кратных рёбер, называется *простым*.

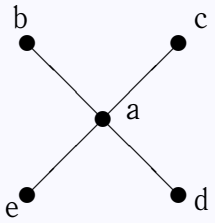


Рис. 1.4. Звёздый граф $K_{1,4}$.

Если число ребер [и число вершин] бесконечно, то граф называется *бесконечным*.

Если все локальные степени $\rho(a)$ конечны, то граф называется *локально конечным* (localy finite).

Граф называется *однородным* или *регулярным* (regular) степени n, если

$$\rho(a) = n, \text{ для всех } a \in V$$

Цепь (chain) — последовательность идущих друг за другом ребер.

Путь (path) — это цепь, в которой все ребра различны.

Если все вершины пути, за исключением может быть первой и последней, различны, то путь называется *простым*.

Замкнутая цепь называется *циклом*. Цикл — это связный регулярный граф степени 2. Цикл из n вершин обозначают C_n. Цикл называется простым, если он не проходит ни через одну вершину более одного раза.

Ориентированный граф называется *(слабо) связным* (connected), если связен соответствующий ему неориентированный граф.

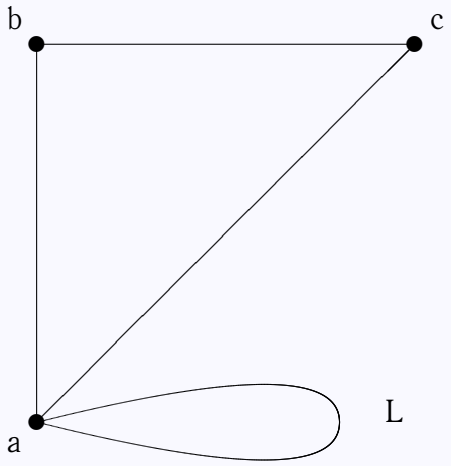


Рис. 1.5. Петля.

Ориентированный граф называется *(сильно) связным* (strongly connected), если любые два различных узла в нем соединены ориентированным путем.

Лес — это граф без циклов.

Дерево (tree) — это граф, в котором существует одна и только одна цепь, соединяющая каждую пару вершин. Таким образом, дерево — связанный граф без циклов.

Вершины степени 1 в дереве называются *листьями*.

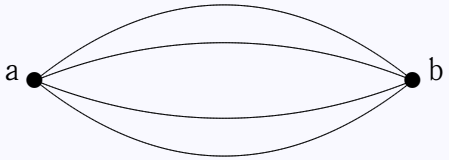


Рис. 1.6. Кратные рёбра.

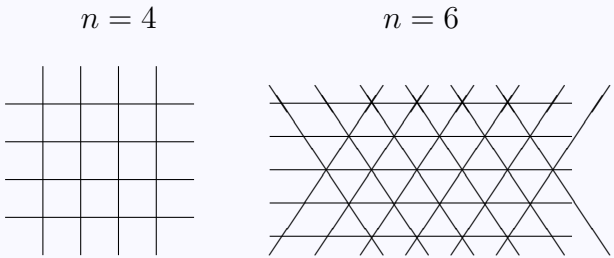


Рис. 1.7. Фрагменты регулярных графов.

Дерево, содержащее все вершины некоторого графа G, называется *покрывающим деревом* (spanning tree) графа G.

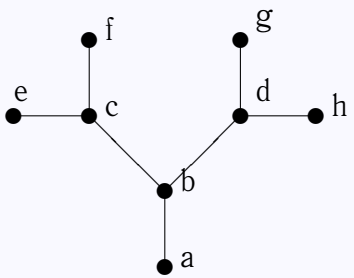


Рис. 1.8. Двоичное дерево.

Упражнение 1.2. *Каково максимальное возможное число ориентированных ребер в простом ориентированном графе без циклов, состоящем из n вершин?*

Упражнение 1.3. *Каково максимальное возможное число ребер в простом неориентированном графе без циклов, состоящем из n вершин?*

Гранью называют часть плоскости, ограниченной простым циклом и не содержащей внутри других циклов.

Граф называется *плоским* или *планарным* (planar), если он может быть изображен на плоскости так, что все пересечения ребер являются вершинами.

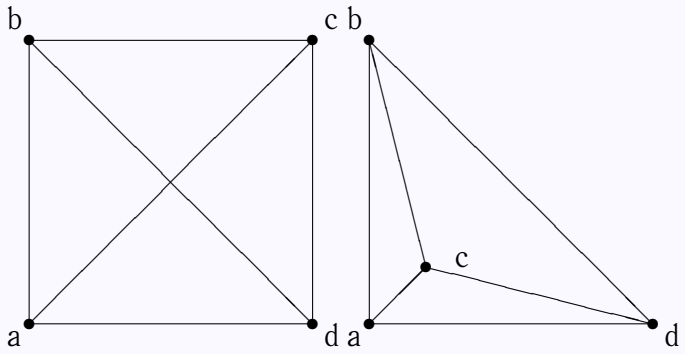


Рис. 1.9. Полный граф K_4 является плоским.

Теорема 1.1 (Фари). *Простой планарный граф всегда можно изобразить на плоскости так, что все его рёбра будут отрезками прямых линий.*

Упражнение 1.4. *Нарисуйте графы выпуклых многогранников (тетраэдра, куба, октаэдра, додекаэдра, икосаэдра). Являются ли эти графы плоскими?*

Простой граф называется *максимально плоским*, если нельзя добавить к нему ни одного ребра так, чтобы полученный граф был плоским. Все грани максимально плоского графа — треугольники, поэтому максимально плоский граф еще называют *триангулированным*.

Теорема 1.2. *Графы K_5 и $K_{3,3}$ не планарны.*

Два графа *гомеоморфны* (тождественны до вершин степени 2), если они могут быть получены из одного и того же графа включением в его ребра новых вершин степени 2.

Два любых циклических графа гомеоморфны.

1.2.1. Части графов

G_1 называется *подграфом* графа G, если

$$V(G_1) \in V(G), \qquad E(G_1) \in E(G)$$

Разделяющее множество связного графа G называется такое множество его ребер, удаление которых приводит к несвязному графу.

Разрез — разделяющее множество, никакое собственное подмножество которого не является разделяющим.

Разрез, состоящий из единственного ребра, называется *мост* или *перешеек*.

Дополнительная часть или *дополнение* $\bar{H}$ любой части H графа G — это часть, состоящая из всех ребер G, не принадлежащих H.

$$H = G - \bar{H}$$

Две вершины в $\bar{H}$ смежны тогда и только тогда, когда они смежны в H.

Объединение $G_1 \cup G_2$ графов G_1 и G_2 —

$$V_1 \cup V_2, \qquad E_1 \cup E_2$$

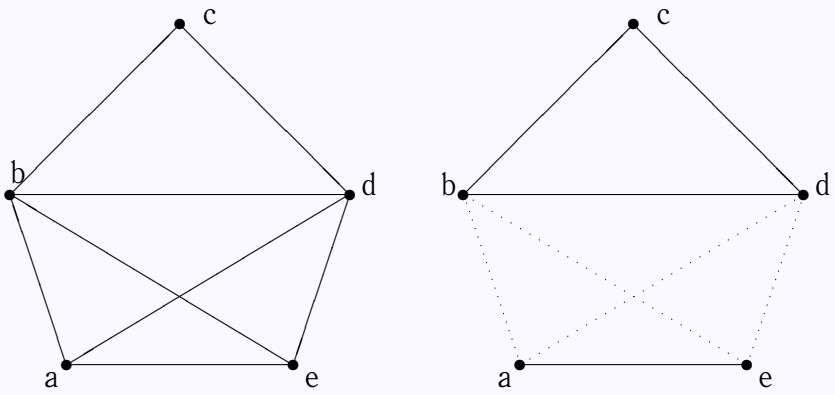


Рис. 1.10. Разделяющее множество (пунктир).

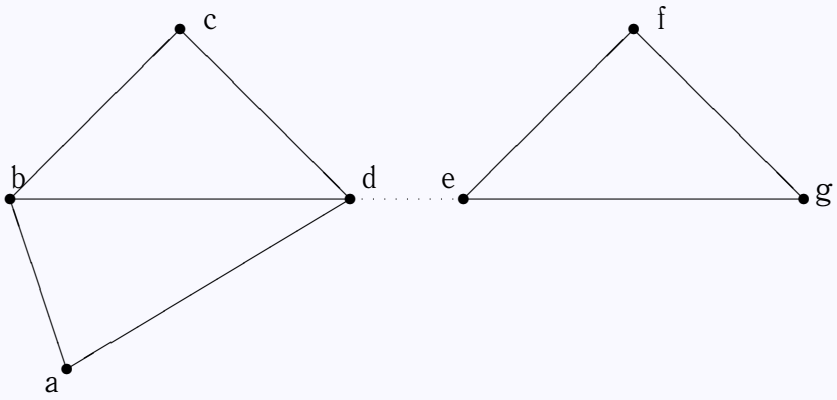


Рис. 1.11. Мост (пунктир).

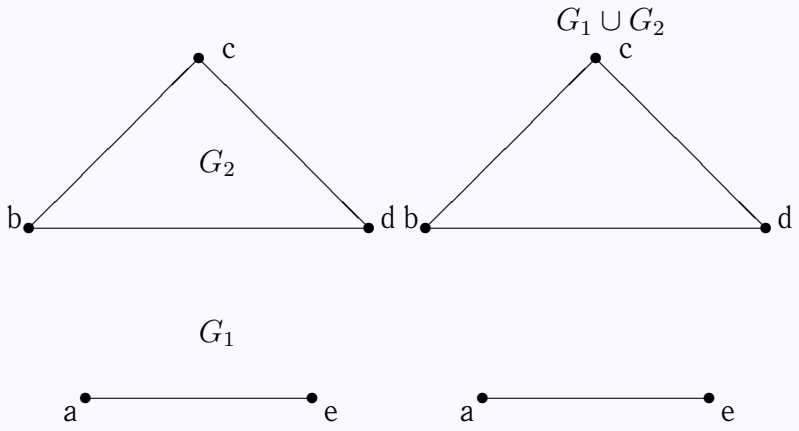


Рис. 1.12. Объединение графов G_1 и G_2.

Объединение звездого графа $K_{1,n}$ и цикла C_n называется *колесом* W_n.

$$K_{1,n} \cup C_n = W_n$$

Чтобы получить *соединение* $G_1 + G_2$ графов G_1 и G_2 необходимо объединить G_1 и G_2 и соединить каждую вершину G_1 с каждой вершиной G_2.

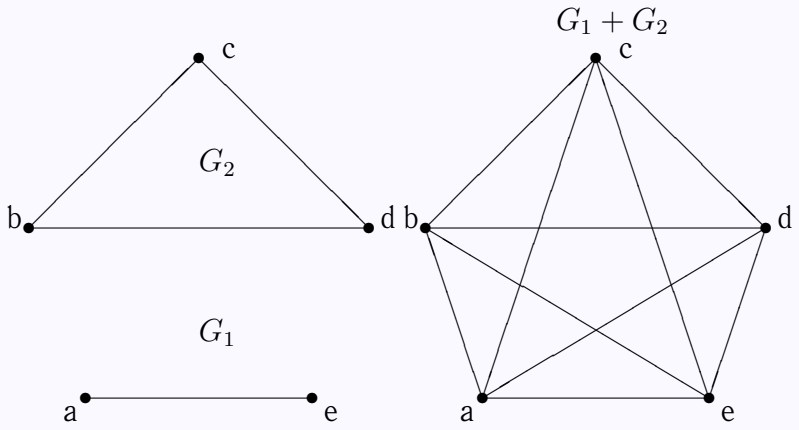


Рис. 1.13. Соединение графов G_1 и G_2.

Независимое множество I вершин графа $G = G(V, E)$ — это набор таких вершин графа G, что никакие две вершины из I не смежны в G.

Максимально независимое множество M графа G — это токое независимое множество, что если v — какая-либо вершина графа G, не принадлежащая M ($v \in V$ и $v \notin M$), то $M \cup v$ не является независимым множеством.

1.2.2. Теоремы Понтрягина-Куратовского и Эйлера

Теорема 1.3 (Понтрягина-Куратовского). *Граф планарен тогда и только тогда, когда он не содержит подграфов гомеоморфных K_5 и $K_{3,3}$.*

Теорема 1.4 (Эйлер, 1752 г.). *Пусть G — связный граф, а v, e, f обозначают соответственно число вершин, рёбер и граней графа G. Тогда*

$$v + f = e + 2$$

Доказательство. Проведем доказательство по индукции.

Если $e = 0$, то $v = 1$ и $f = 1$ (бесконечная грань).

Пусть теорема верна для любого графа, имеющего e ребер. Добавим к графу новое ребро, тогда возможны следующие ситуации

1. Ребро является петлей:

$$\begin{aligned} v &\to v \\ f &\to f + 1 \\ e &\to e + 1 \end{aligned}$$

2. Ребро соединяет две вершины и расщепляет одну из граней на две

$$\begin{aligned} v &\to v \\ f &\to f + 1 \\ e &\to e + 1 \end{aligned}$$

3. Ребро инцидентно только одной вершине, тогда необходимо добавить новую вершину

$$v \to v + 1$$
$$f \to f$$
$$e \to e + 1$$

□

Следствие 1. *Если G — планарный простой связный граф с $v \geqslant 3$ вершинами, то*

$$e \leqslant 3v - 6$$

Каждая грань ограничена тремя ребрами, каждое ребро ограничивает не более двух граней, следовательно,

$$3f \leqslant 2e \Rightarrow v + \frac{2}{3}e \geqslant e + 2 \Rightarrow e \leqslant 3v - 6$$

Упражнение 1.5. *Используя теорему Эйлера доказать, что графы K_5 и $K_{3,3}$ не планарны.*

1.2.3. Эйлеровы и гамильтоновы графы

Связный граф называется *эйлеровым*, если существует замкнутая цепь, проходящая через каждое его ребро.

Связный граф называется *полуэйлеровым*, если существует цепь, проходящая через каждое его ребро.

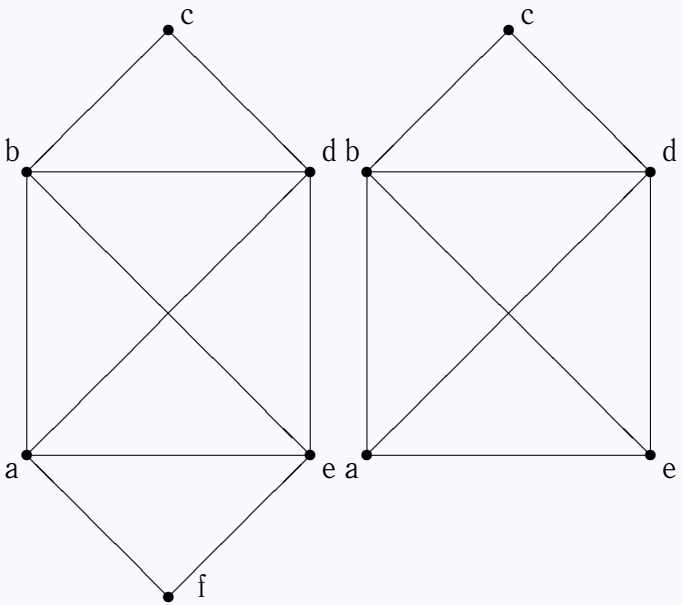


Рис. 1.14. Эйлеров и полуэйлеров графы.

Граф, содержащий замкнутую цепь, проходящую через все его вершины ровно один раз, называется *гамильтоновым*. Такая цепь называется *гамильтоновой*.

Граф, содержащий цепь, проходящую через все его вершины ровно один раз, называется *полугамильтоновым*.

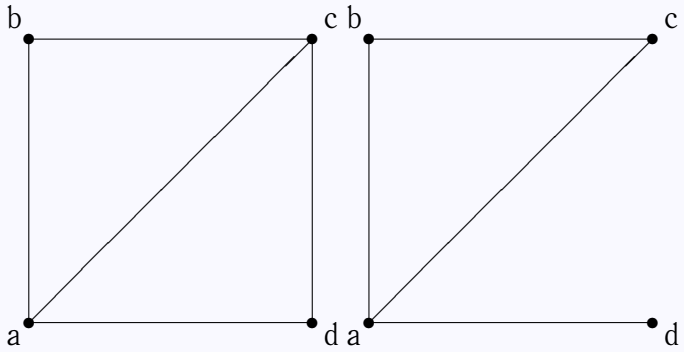


Рис. 1.15. Гамильтонов и полугамильтонов графы.

Теорема 1.5. *Связный граф является эйлеровым тогда и только тогда, когда каждая его вершина имеет чётную степень.*

Упражнение 1.6. *Нарисуйте граф к задаче 1.1. Является ли полученный граф эйлеровым? Почему?*

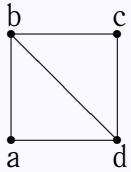
Для нахождения эйлерова пути на графе используют следующий алгоритм.

Алгоритм 1.1 (Флёри).

1. Начав движение из произвольной вершины, стираем ребра по мере их прохождения, а так же стираем изолированные вершины, которые при этом образуются.

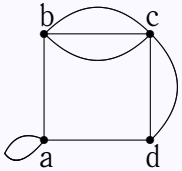
2. На каждом этапе идем по мосту только в том случае, когда нет других возможностей.

Матрица смежности вершин — это квадратная матрица, каждая строка и каждый столбец которой сопоставлены определенной вершине графа. Матричный элемент равен числу ребер (дуг), связывающих две вершины.



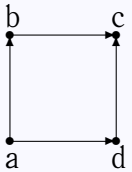
	a	b	c	d
a	0	1	0	1
b	1	0	1	1
c	0	1	0	1
d	1	1	1	0

простой граф



	a	b	c	d
a	1	1	0	1
b	1	0	3	0
c	0	3	0	2
d	1	0	2	0

граф с кратными ребрами и петлей



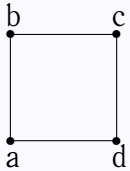
	a	b	c	d
a	0	1	0	1
b	0	0	1	0
c	0	0	0	0
d	0	0	1	0

ориентированный граф

Сумма элементов строки (столбца) матрицы смежности равна степени соответствующей вершины.

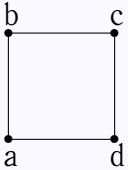
Если обозначить матрицу смежности $\mathbf{M}$, то матричный элемент $\|\mathbf{M}^m\|_{i,j}$ будет показывать, сколько различных путей длины m ведет из вершины i в вершину j.

Матрица смежности рёбер показывает, какие ребра являются смежными. Это квадратная матрица, каждая строка и каждый столбец которой сопоставлены определенному ребру. Матричный элемент равен нулю, если соответствующие ребра не смежны, и равен 1, если смежны.



	ab	bc	cd	da
ab	0	1	0	1
bc	1	0	1	0
cd	0	1	0	1
da	1	0	1	0

Матрица инцидентности считается компактным способом представления графа. Это прямоугольная матрица $n \times m$, где n — число вершин, а m — число ребер. Столбцы этой матрицы сопоставляют ребрам, а строки — вершинам графа. Матричный элемент равен нулю, если вершина не инцидентна ребру, и равен единице, если инцидентна.



	ab	bc	cd	da
a	1	0	0	1
b	1	1	0	0
c	0	1	1	0
d	0	0	1	1

Обобщение этих матриц — конечные и бесконечные матрицы, элементы которых вещественные числа $\rho(a, b)$, обозначающих меру, мощность, пропускную способность ребра.

1.3. Рёберные и дуальные графы

Если рассматривать матрицу смежности ребер как матрицу смежности вершин, то можно получить *смежностный* или *рёберный* или *покрывающий* граф. Такой переход соответствует преобразованию ребра в вершину (bond-to-site transformation).

Реберный граф $L(G)$ простого графа G — это такой граф, в котором вершины сопоставлены рёбрам G. Вершины в $L(G)$ смежные, если соответствующие им рёбра в G смежные.

Для построения смежностного графа поставим в середине каждого ребра исходного графа точки. Это — вершины смежностного графа. Соединим полученные вершины между собой, если они лежат на смежных ребрах.

Упражнение 1.7. *Нарисуйте смежностные графы выпуклых многогранников. Являются ли эти графы плоскими?*

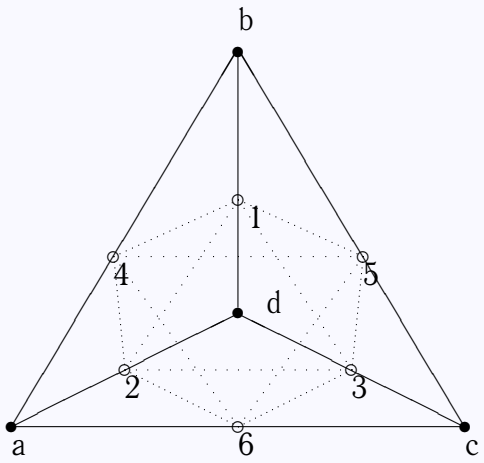


Рис. 1.16. Построение смежностного графа.

Для построения *дуального* или *двойственного* графа нужно поставить точку внутри каждой грани исходного графа, в том числе, и внутри бесконечной грани, если она имеется. Эти точки являются вершинами дуального графа. Соединяем две вершины между собой, если они принадлежат соседним граням. Будем проводить линии так, чтобы они пересекали только общее ребро соседних граней и не пересекали другие рёбра.

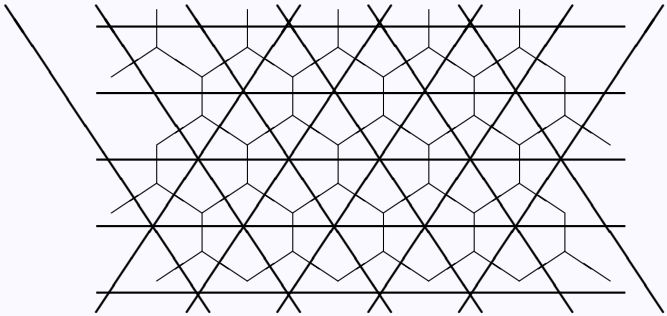


Рис. 1.17. Регулярный граф (жирные линии) и его дуальный (тонкие линии).

Упражнение 1.8. *Нарисуйте дуальные графы платоновых тел.*

1.4. Применение пакета Maple для решения некоторых задач теории графов

```
> with(networks):
```

Задаем полный граф из пяти вершин

```
> K5:=complete(5):
```

Определяем степень вершины 1

```
> vdegree(1,K5);
```

$$4$$

Определяем полустепень захода этой же вершины

```
> indegree(1,K5);
```

$$0$$

Определяем полустепень захода

```
> outdegree(1,K5);
```

$$0$$

(Поскольку граф неориентированный, то, естественно, обе полустепени равны 0.)

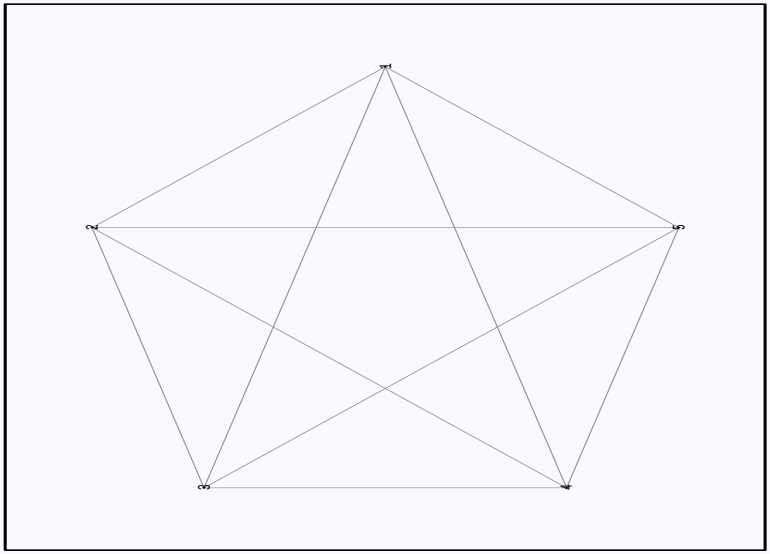
Определяем максимальную степень вершин.

```
> maxdegree(K5);
```

$$4$$

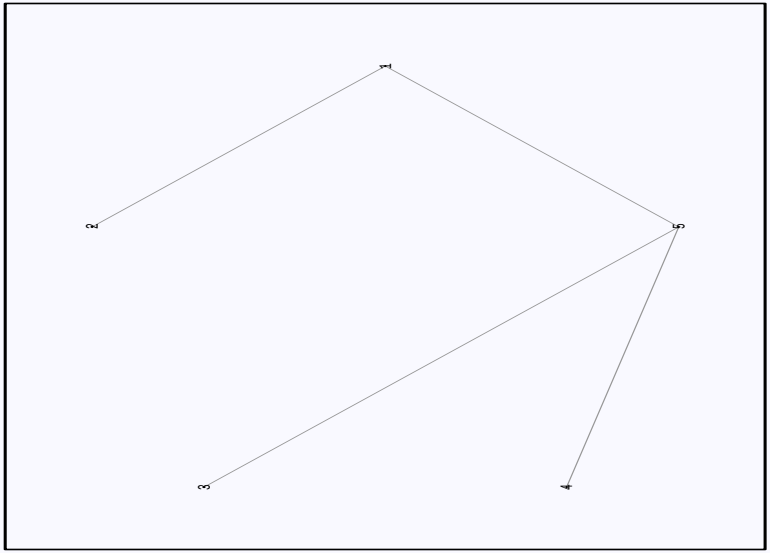
Рисуем граф

```
> draw(K5);
```



И его стягивающее дерево

```
> T:=spantree(K5):
> draw(T);
```



Находим матрицу инцидентности

```
> incidence(K5);
```

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

Определяем двудольный граф

```
> K33:=complete(3,3):
```

Находим его ребра

```
> edges(K33);
```

$$\{e2, e1, e8, e9, e7, e5, e6, e4, e3\}$$

и вершины

```
> vertices(K33);
```

$$\{1, 2, 3, 4, 5, 6\}$$

Находим матрицу смежности

```
> adjacency(K33);
```

$$\begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \end{bmatrix}$$

Проверяем, планарны ли графы

```
> isplanar(cube());
```

$$true$$

```
> isplanar(K5);
```

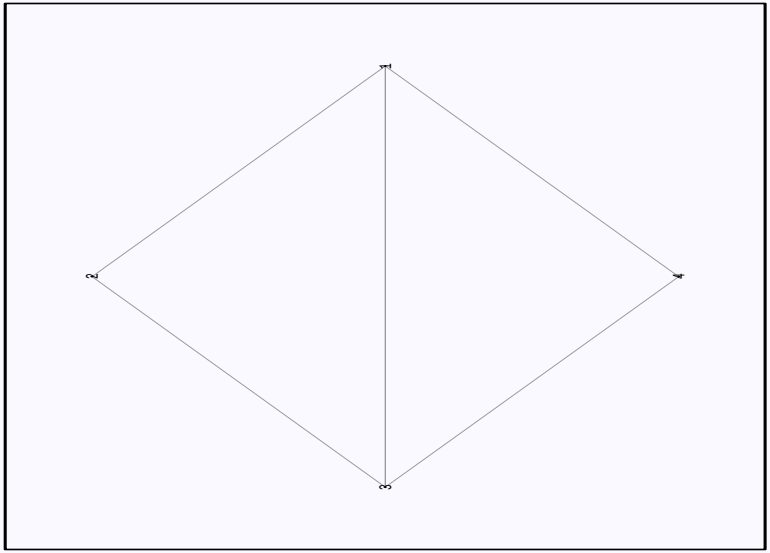
$$false$$

Создаем новый граф на основе цикла

```
> C4:=cycle(4):
> addedge({1,3},C4);
```

$$e5$$

```
> draw(C4);
```

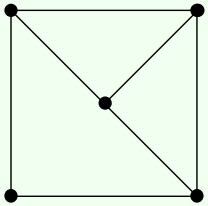

1
2
3
4

1.5. Контрольная работа

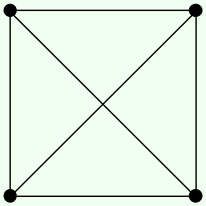
1. Дайте полную характеристику графа.
2. Составьте для данного графа матрицы смежности вершин, рёбер и матрицу инцидентности.
3. Для данного графа постройте смежностный и дуальный графы.

Ипользуйте графы, приведенные на рисунке, в соответствии с номером своего варианта.

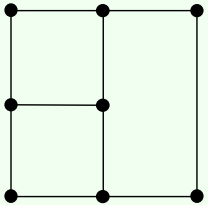
1



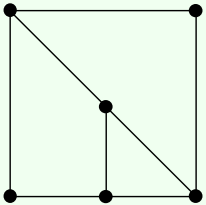
2



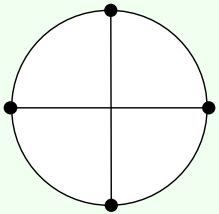
3



4



5



Глава 2

Комбинаторика

Комбинаторика возникла в XVI веке. Ее основоположниками были Тарталья, Паскаль, Ферма, Я. Бернулли, Лейбниц, Эйлер.

2.1. Основные определения

Каждая последовательность n различных предметов с учетом порядка называется *перестановкой* предметов.

Задача 2.1. *Сколькими способами можно расставить на полке 10 книг?*

На первое место можно поставить любую из десяти книг. На второе место — любую из оставшихся 9 и т. д. Таким образом,

$$P_{10} = 10 \cdot 9 \cdot \ldots \cdot 1 = 10! = 3628800$$

Задача 2.2. *Сколько имеется различных отображений, переводящих множество X из n элементов в себя ($X \mapsto X$)?*

$$(a,b) \mapsto (a,b)(b,a)$$

$$(a,b,c) \mapsto (a,b,c)(a,c,b)(b,a,c)(b,c,a)(c,a,b)(c,b,a)$$

Количество перестановок из n предметов обозначают P_n и вычисляют по формуле

$$P_n = n!$$

Взаимно однозначное отображение f конечного упорядоченного множества M из n элементов в себя называется *подстановкой* элементов множества M.

$$f(i) = j, \qquad i, j \in M$$

Всего имеется $n!$ различных подстановок.

Подстановку можно изобразить в виде матрицы из двух строк

$$f = \begin{pmatrix} a & b & c & d \\ d & a & c & b \end{pmatrix}$$

Если $f(i) = i$, то i называют *неподвижной точкой*.

Подстановку называют *тождественной* и обозначают f_e, если

$$f(i) = i, \text{ для всех } i.$$

Если последовательное применение r подстановок переводит некоторое i в себя, то имеется *цикл* длины r.

$$\underbrace{f \cdot f \cdot \ldots f}_{r}(i) = i$$

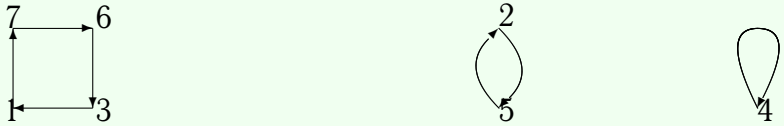
Неподвижная точка соответствует циклу единичной длины.

Цикл длины 2 называется *транспозицией*.

Подстановка

$$f = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 7 & 5 & 1 & 4 & 2 & 3 & 6 \end{pmatrix}$$

имеет следующее разложение на циклы $f = [1763][25][4]$.



Для каждой подстановки f существует *обратная подстановка* f^{-1}, такая что

$$f \cdot f^{-1} = f^{-1} \cdot f = f_e.$$

Если в матрице подстановок встречается два столбца

$$f = \begin{pmatrix} \cdots & s_i & \cdots & s_j & \cdots \\ \cdots & t_i & \cdots & t_j & \cdots \end{pmatrix},$$

для которых $s_i < s_j$, но $t_i > t_j$ (или $s_i > s_j$, но $t_i < t_j$), то такая пара столбцов называется *инверсией* подстановки f.

Подстановка называется *четной* или *нечетной* в зависимости от того, четно или нечетно число встречающихся в ней инверсий.

Любой упорядоченный набор из m *различных* элементов множества, состоящего из n элементов, называется *размещением*. Число размещений

$$A_n^m = \frac{n!}{(n-m)!}$$

Любой упорядоченный набор из m элементов множества, состоящего из n элементов, называется *размещением с повторениями*. Число размещений с повторениями

$$\tilde{A}_n^m = m^n$$

Любое подмножество из m *различных* элементов множества, состоящего из n элементов, называется *сочетанием*. Число сочетаний

$$C_n^m = \binom{m}{n} = \frac{n!}{m!(n-m)!}$$

Основные соотношения для числа сочетаний

$$C_n^m = C_n^{n-m}$$

$$\sum_{i=0}^{n} C_n^i = 2^n$$

$$\sum_{i=0}^{n} (-1)^i C_n^i = 0$$

$$C_n^m = C_{n-1}^m + C_{n-1}^{m-1}$$

Упражнение 2.1. *Докажите формулу*

$$(C_n^0)^2 + (C_n^1)^2 + (C_n^2)^2 + \cdots + (C_n^n)^2 = C_{2n}^n$$

Задача 2.3. *Рассмотрим прямоугольную сетку из n горизонтальных линий и n вертикальных линий. Сколькими различными способами можно переместиться из левого нижнего угла сетки в ее праый верхний угол, если перемещаться можно только вправо и вверх?*

Будем кодировать каждый возможный путь из левого нижнего угла сетки в ее правый верхний угол в виде последовательности нулей и единиц. Ноль будет обозначать шаг в вертикальном направлении, а единица — в горизонтальном. Каждая такая последовательность состоит из $2n$ элементов. Для размещения n единиц на $2n$ имеющихся мест существует C_{2n}^n возможностей. Таким образом, имеется C_{2n}^n различных путей.

Упражнение 2.2. *Решить задачу* (2.3) *для случая прямоугольной сетки* $n \times m$.

Упражнение 2.3. *Решить задачу* (2.3) *при условии, что нельзя делать подряд два шага в вертикальном направлении.*

Число перестановок с поворениями определяется выражением

$$C_n(i_1, i_2, \ldots, i_p) = \frac{n!}{i_1!\, i_2! \ldots i_p!}$$

Упражнение 2.4. *Сколько различных шестизначных чисел можно составить из цифр 1, 1, 1, 5, 5, 9?*

2.1.1. Матрица перестановок

Квадратная матрица $\mathbf{P}$ называется *матрицей перестановки*, если в любом ее столбце и в любой ее строке один элемент равен 1, а все остальные равны 0. При умножении матрицы $\mathbf{M}$ на такую матрицу слева переставляются строки, а при умножении справа — столбцы матрицы $\mathbf{M}$.

Например, при умножении иатрицы

$$\mathbf{P} = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

на вектор

$$\begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$$

имеем

$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} = \begin{pmatrix} 2 \\ 1 \\ 3 \end{pmatrix}$$

Для того, чтобы поменять местами i—ю и j—ю строку матрицы, ее нужно

умножить слева на матрицу

$$
\begin{array}{cc}
 & \begin{array}{cccccccccc} & & & i\text{-й столбец} & & & j\text{-й столбец} & & & \end{array} \\
\mathbf{P} = & \begin{pmatrix}
1 & & & \vdots & & & & \vdots & & \\
 & \ddots & & \vdots & & & & \vdots & & \\
 & & 1 & \vdots & & & & \vdots & & \\
\cdots & \cdots & \cdots & 0 & \cdots & \cdots & \cdots & 1 & \cdots & \cdots \\
 & & & \vdots & 1 & & & \vdots & & \\
 & & & \vdots & & \ddots & & \vdots & & \\
 & & & \vdots & & & 1 & \vdots & & \\
\cdots & \cdots & \cdots & 1 & \cdots & \cdots & \cdots & 0 & \cdots & \cdots \\
 & & & \vdots & & & & \vdots & \ddots & \\
 & & & \vdots & & & & \vdots & & 1
\end{pmatrix}
\begin{array}{l} \\ \\ \\ i\text{-я строка} \\ \\ \\ \\ j\text{-я строка} \\ \\ \\ \end{array}
\end{array}
$$

которая также является частным случаем матрицы перестановок и называется *матрицей транспозиции*.

Определитель матрицы перестановок равен ± 1. Кроме того, $\mathbf{P}^T = \mathbf{P}^{-1}$.

Упражнение 2.5. *Вычислите* $\det(\mathbf{P}^{-1}\mathbf{A}\mathbf{P})$.

2.2. Рекуррентные соотношения

Соотношение вида

$$r_n = a_1 r_{n-1} + \ldots + a_k r_{n-k} \tag{2.1}$$

называется *линейным рекуррентным соотношением k-го порядка*.

Решение линейного рекуррентного соотношения с постоянными коэффициентами (2.1) называется *общим*, если оно содержит k произвольных постоянных.

Если задать k первых элементов последовательности, то числовая последовательность будет определена однозначно.

Рекуррентному соотношению (2.1) ставят в соответствие *характеристическое уравнение*

$$\lambda^k = a_1\lambda^{k-1} + a_2\lambda^{k-2} \ldots + a_k \tag{2.2}$$

Если среди корней λ_i характеристического уравнения (2.2) нет кратных, то числовая последовательность, задавемая рекуррентным соотношением (2.1), имеет вид

$$r_n = C_1\lambda_1^n + C_2\lambda_2^n + \ldots + C_k\lambda_k^n$$

Постоянные C_i определяются из условий

$$\begin{aligned} r_0 &= C_1 + C_2 + \ldots + C_k \\ r_1 &= C_1\lambda_1 + C_2\lambda_2 + \ldots + C_k\lambda_k \\ &\ldots \\ r_k &= C_1\lambda_1^k + C_2\lambda_2^k + \ldots + C_k\lambda_k^k \end{aligned}$$

Если r^n является решением некоторого рекуррентного соотношения, то и r^{n-1} так же является решением этого рекуррентного соотношения.

Если r_1^n и r_2^n являются решениями некоторого рекуррентного соотношения, то и $ar_1^n + br_2^n$ тоже является решением этого рекуррентного соотношения.

Если среди корней характеристического уравнения имеются кратные

$$r_m = r_{m+1} = \ldots = r_{m+l},$$

то общее решение имеет вид

$$f(n) = C_1 r_1^n + C_2 r_2^n + \ldots + C_m r_m^n + C_{m+1} n r_m^{n-1} + \ldots + C_{m+l} n^m r_m^{n-l} \ldots + C_n r_n^n$$

Задача 2.4. *Найти общее решение рекуррентного соотношения*

$$f(n+1) = af(n)$$

Это линейное рекуррентное соотношение первого порядка. Его характеристическое уравнение имеет вид

$$\lambda = a.$$

Следовательно, $f(n) = C \cdot a^n$

2.3. Производящие функции

Пусть имеется функция $f(x)$ такая, что в некоторой окрестности числа x_0 (действительного или комплексного) она может быть представлена в виде степенного ряда

$$f(x) = \sum_{i=0}^{\infty} a_i x^i. \tag{2.3}$$

Функция $f(x)$ нзывается *производящей* (generating) для числовой последовательности $a_0, a_1, \ldots$.

При нахождении числовых последовательностей бывает полезно использовать следующие действия с производящими функциями.

Умножение производящей функции на число

$$Af(x) = A\sum_{i=0}^{\infty} a_i x^i = \sum_{i=0}^{\infty}(Aa_i)x^i.$$

Сумма двух производящих функций

$$f(x) + g(x) = \sum_{i=0}^{\infty} a_i x^i + \sum_{i=0}^{\infty} b_i x^i = \sum_{i=0}^{\infty}(a_i + b_i)x^i.$$

Произведение Коши

$$f(x) \cdot g(x) = \sum_{i=0}^{\infty} a_i x^i \cdot \sum_{j=0}^{\infty} b_j x^j = \sum_{k=0}^{\infty} c_k x^k,$$

где

$$c_k = \sum_{l=0}^{k} a_l b_{k-l}$$

2.3.1. Числа Фибоначчи

Задача 2.5 (Фибоначчи). *Пара кроликов приносит раз в месяц приплод из двух крольчат (самки и самца), причем новорожденные крольчата через два месяца после рождения приносят приплод. Сколько пар кроликов появится через год, если в начале была одна пара кроликов?*

Обозначим количество кроликов в начальный момент времени f_0. Количество кроликов в последующие месяцы будем обозначать f_i. Очевидно, что $f_0 = 1$ и $f_1 = 1$. За исключением этих месяцев, во все последующие месяцы количество кроликов определяется соотношением

$$f_i = f_{i-1} + f_{i-2}. \tag{2.4}$$

То есть кроликов столько, сколько их было в предыдущий месяц, плюс потомство от всех тех кроликов, которые родились два месяца назад и ранее.

Из рекуррентного соотношения (2.4) находим, что последовательность чисел Фибоначчи $1, 1, 2, 3, 5, 8, 13, 21, \ldots$.

Перепишем выражение (2.3) с учетом рекуррентного соотношения (2.4). Поскольку для первых двух чисел Фибоначчи рекуррентное соотношение не выполняется, мы вынесем их из под знака суммы.

$$F(x) = f_0 + f_1 x + \sum_{i=2}^{\infty} (f_{i-1} + f_{i-2}) x^i =$$

$$= f_0 + f_1 x + \sum_{i=2}^{\infty} f_{i-1} x^i + \sum_{i=2}^{\infty} f_{i-2} x^i \tag{2.5}$$

$$\sum_{i=2}^{\infty} f_{i-1}x^i =$$

$$= \sum_{i=2}^{\infty} f_{i-1}x^{i-1}x = x\sum_{i=2}^{\infty} f_{i-1}x^{i-1} = x\left(\sum_{i=2}^{\infty} f_{i-1}x^{i-1} + f_0 - f_0\right) =$$

$$= x\left(\sum_{i=1}^{\infty} f_{i-1}x^{i-1} - f_0\right) = x\left(\sum_{i=0}^{\infty} f_i x^i - f_0\right) = x(F(x) - f_0)$$

$$\sum_{i=2}^{\infty} f_{i-2}x^i = \sum_{i=2}^{\infty} f_{i-2}x^{i-2}x^2 = x^2\sum_{i=2}^{\infty} f_{i-2}x^{i-2} = x^2\sum_{i=0}^{\infty} f_i x^i = x^2F(x)$$

$$F(x) = f_0 + f_1x + x(F(x) - f_0) + x^2F(x) =$$
$$1 + x + x(F(x) - 1) + x^2F(x) = 1 + (x + x^2)F(x)$$

$$F(x) = \frac{1}{1 - x - x^2} \tag{2.6}$$

Упражнение 2.6. *Докажите следующие тождества для чисел Фибоначчи.*

1. $f_0 + f_1 + \cdots + f_n = f_{n+2} - 1$
2. $f_0 + f_2 + \cdots + f_{2n} = f_{2n+1}$
3. $f_1 + f_3 + \cdots + f_{2n-1} = f_{2n-1}$
4. $f_0^2 + f_1^2 + \cdots + f_n^2 = f_n f_{n+1}$

2.3.2. Числа Каталана

Задача 2.6. *Сколько существует неизоморфных двоичных деревьев с n вершинами?*

Задача 2.7. *Сколько существует вариантов правильного расположения скобок в произведении n сомножителей? (Например, в случае двух сомножителей имеем: $(ab)c$ и $a(bc)$.)*

Задача 2.8. *Сколькими способами можно разбить выпуклый $(n+2)$-угольник на треугольники, проведя $n-1$ непересекающихся диагоналей?*

В приведенных выше задачах, а так же в множестве других разнообразных задач ответом будут служить *числа Каталана*.

Числа Каталана (1, 1, 2, 5, 14, 42, 132, ...) задаются следующим рекуррентным соотношением при $i > 0$

$$c_{i+1} = c_0 c_i + c_1 c_{i-1} + \ldots + c_i c_0 \tag{2.7}$$

с начальным условием $c_0 = 1$.

Найдем производящую функцию для чисел Каталана

$$F(x) = \sum_{i=0}^{\infty} c_i x^i = 1 + \sum_{i=1}^{\infty} \left(\sum_{j=0}^{i-1} c_j c_{i-j} \right) x^i =$$

$$= 1 + x \sum_{i=1}^{\infty} \left(\sum_{j=0}^{i-1} c_j c_{i-j} \right) x^{i-1} = 1 + xF(x)^2 \tag{2.8}$$

Таким образом, производящая функция равна

$$F(x) = \frac{1 - \sqrt{1 - 4x}}{2x}$$

Рекуррентное соотношение для чисел Каталана может быть записана более компактно

$$c_i = \frac{1}{i+1}\binom{2i}{k}$$

или

$$c_{i+1} = c_i \frac{4i+2}{i+2}.$$

Упражнение 2.7. *Какие числовые последовательности порождают следующие производящие функции?*

1. $\dfrac{1}{1-x}$ 2. $\dfrac{1}{1-x^2}$ 3. $\dfrac{1}{(1-x)(1-x^2)}$

4. $\dfrac{1}{1-2x}$ 5. $\dfrac{1}{1-3x}$ 6. $\dfrac{1}{1-x} + \dfrac{1}{1-x^2}$

2.4. Неоднородные рекуррентные соотношения

Запись рекуррентного соотношения может оказаться обманчивой. Не всегда линейное рекуррентное соотношение можно узнать с первого взгляда. Рассмотрим простейший пример.

В случае арифметической последовательности можно записать

$$u_{n+1} = u_n + d. \tag{2.9}$$

На первый взгляд кажется, что полученное соотношение не может быть решено изложенным ранее способом. Однако, это не так. Запишем еще одно соотношение

$$u_{n+2} = u_{n+1} + d. \tag{2.10}$$

Теперь вычтем из уравнения (2.9) уравнение (2.10)

$$u_{n+2} - u_{n+1} = u_{n+1} - u_n$$

или

$$u_{n+2} = 2u_{n+1} - u_n. \tag{2.11}$$

Уравнение (2.11) представляет собой линейное рекуррентное соотношение 2-го порядка.

Упражнение 2.8. *Найдите общее и частное решение, а также производящую функцию для следующих рекуррентных соотношений*

1. $f_0 = f_1 = 1, \qquad f_n = 2f_{n-2} + 1$
2. $f_0 = f_1 = 1, \qquad f_n = f_{n-1} + n^2$

2.5. Применение пакета Maple для решения некоторых комбинаторных задач

Точное вычисление факториала

```
> 50!;
```

$$30414093201713378043612608166064768844377641568960512000000000000$$

Вычисление биномиальных коэффициентов

```
> binomial(10,5);
```

$$252$$

Вычисление чисел Фибоначчи

```
> fibonacci(10);
```

$$55$$

Нахождение перестановок

```
> with(combinat,permute):
> permute({A,B,C});
```

$$[[B, A, C], [B, C, A], [A, B, C], [A, C, B], [C, B, A], [C, A, B]]$$

Определение производящей функции для числовой последовательности, заданной рекуррентным соотношением

```
> RF:=f(n)=f(n-1)+f(n-2):
> rsolve({RF,f(0)=1,f(1)=1},f(n),'genfunc'(x));
```

$$-\frac{1}{-1+x+x^2}$$

Нахождение частного решения рекуррентного соотношения

```
> rsolve({RF,f(0)=1,f(1)=1},f);
```

$$\frac{2}{5}\,\frac{\sqrt{5}\,(\frac{2}{\sqrt{5}-1})^n}{\sqrt{5}-1}+\frac{2}{5}\,\frac{\sqrt{5}\,(-\frac{2}{\sqrt{5}+1})^n}{\sqrt{5}+1}$$

Нахождение общего решения рекуррентного соотношения

```
> rsolve(RF,f);
```

$$\frac{(-\frac{1}{5}\,\mathrm{f}(1)\,\sqrt{5}+\mathrm{f}(1)+\frac{3}{5}\,\mathrm{f}(0)\,\sqrt{5}-\mathrm{f}(0))\,(\frac{2}{\sqrt{5}-1})^n}{\sqrt{5}-1}$$
$$-\frac{1}{5}\,\frac{\sqrt{5}\,(\mathrm{f}(1)+\mathrm{f}(1)\,\sqrt{5}-3\,\mathrm{f}(0)-\mathrm{f}(0)\,\sqrt{5})\,(-\frac{2}{\sqrt{5}+1})^n}{\sqrt{5}+1}$$

2.6. Контрольная работа

Вариант 1

1. Сколько различных слов из 4-х букв можно составить из букв слова КОМБИНАТОРИКА?

2. Найти решение рекуррентного соотношения
$a_{n+2} - 7a_{n+1} + 12a_n = 0$, если $a_0 = 5,\ a_1 = 6$

3. Найти производящую функцию для полученной в задаче 2 числовой последовательности.

4. Доказать для чисел Фибоначчи следующее соотношение
$u_2 + u_4 + \ldots + u_{2n} = u_{2n+1} - 1$

5. Вычислить $C_n^0 - 2C_n^1 + 3C_n^2 - \ldots + (-1)^n (n+1) C_n^n$

6. Сколько различных кратчайших путей ведут из левого нижнего угла в правый верхний угол прямоугольника 5 на 4 клетки? Сколько будет путей, если два вертикальных участка не могут идти подряд?

Вариант 2

1. Сколько различных слов из 5-и букв можно составить из букв слова КОМБИНАТОРИКА?

2. Найти решение рекуррентного соотношения
$a_{n+2} + 3a_{n+1} - 10a_n = 0$, если $a_0 = 1,\ a_1 = 1$

3. Найти производящую функцию для полученной в задаче 2 числовой последовательности.

4. Доказать для чисел Фибоначчи следующее соотношение
$u_1 + u_3 + \ldots + u_{2n-1} = u_{2n}$

5. Вычислить $C_n^0 + 3C_n^1 + 5C_n^2 + \ldots + (2n+1)C_n^n$

6. Сколько различных кратчайших путей ведут из левого нижнего угла в правый верхний угол прямоугольника 5 на 5 клетки? Сколько будет путей, если два вертикальных участка не могут идти подряд?

Вариант 3

1. Сколько различных слов из 3-х букв можно составить из букв слова КОМБИНАТОРИКА?

2. Найти решение рекуррентного соотношения
$a_{n+2} - 4a_{n+1} + 13a_n = 0$, если $a_0 = 1,\ a_1 = 2$

3. Найти производящую функцию для полученной в задаче 2 числовой последовательности.

4. Доказать для чисел Фибоначчи следующее соотношение
$u_1^2 + u_2^2 + \ldots + u_n^2 = u_n u_{n+1}$

5. Вычислить $C_n^2 + 2C_n^3 + 3C_n^4 + \ldots + (n-1)\,C_n^n$

6. Сколько различных кратчайших путей ведут из левого нижнего угла в правый верхний угол прямоугольника 6 на 3 клетки? Сколько будет путей, если два вертикальных участка не могут идти подряд?

Вариант 4

1. Сколько различных слов из 6-и букв можно составить из букв слова КОМБИНАТОРИКА?

2. Найти решение рекуррентного соотношения
$a_{n+2} - 5a_{n+1} - 6a_n = 0$, если $a_0 = 1,\ a_1 = -7$

3. Найти производящую функцию для полученной в задаче 2 числовой последовательности.

4. Доказать для чисел Фибоначчи следующее соотношение
$u_1u_2 + u_2u_3 + \ldots + u_{2n-1}u_{2n} = u_{2n}^2$

5. Вычислить $C_n^0 + 2C_n^1 + 3C_n^2 + \ldots + (n+1)\,C_n^n$

6. Сколько различных кратчайших путей ведут из левого нижнего угла в правый верхний угол прямоугольника 7 на 2 клетки? Сколько будет путей, если два вертикальных участка не могут идти подряд?

Вариант 5

1. Сколько различных слов из 7-и букв можно составить из букв слова КОМБИНАТОРИКА?

2. Найти решение рекуррентного соотношения
$a_{n+2} - 4a_{n+1} + 4a_n = 0$, если $a_0 = 2,\ a_1 = 4$

3. Найти производящую функцию для полученной в задаче 2 числовой последовательности.

4. Доказать для чисел Фибоначчи следующее соотношение
$u_{n+1}^2 = u_n u_{n+2} + (-1)^n$

5. Вычислить $C_n^1 + 2C_n^2 + 3C_n^3 + \ldots + nC_n^n$

6. Сколько различных кратчайших путей ведут из левого нижнего угла в правый верхний угол прямоугольника 6 на 6 клетки? Сколько будет путей, если два вертикальных участка не могут идти подряд?

Глава 3

Алгоритмы и программы

3.1. Алгоритмы обхода двоичного дерева

Двоичные деревья наиболее широко применяются в программировании. В частности, они возникают при анализе арифметических выражений. В двоичном дереве степень любой вершины не превышает 2. Для двоичных деревьев принята своеобразная терминология. Узел, принимаемый за начальный, называется *корнем*. У этого узла нет предков, но есть потомки. Вершины со степенью 1, т. е. вершины, у которых нет потомков, но есть предки, называются *терминальными* или *листьями*. На рис. 3.1 вершина 1 является корнем. Вершины 2 и 3 — ее потомки. Вершины 4 и 5 — листья. У них нет потомков.

На практике используются следующие алгоритмы обхода: левый – корень – правый (ЛКП или inorder), корень – левый – правый (КЛП или

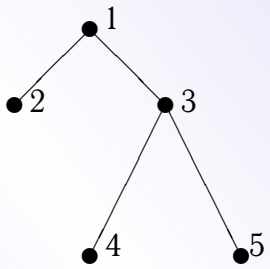


Рис. 3.1. Двоичное дерево

preorder), левый – правый – корень (ЛПК или postorder).

Прменяя алгоритм ЛКП к дереву на рис. 3.2, получаем обычную запись арифметического выражения.

$$2 * ((a + b) - (c + d))$$

Прменяя алгоритм КЛП, получаем *префиксную* запись арифметического выражения.

$$*2 - +ab + cd$$

Прменяя алгоритм ЛПК, получаем *постфиксную или обратную польскую* запись арифметического выражения.

$$2ab + cd + -*$$

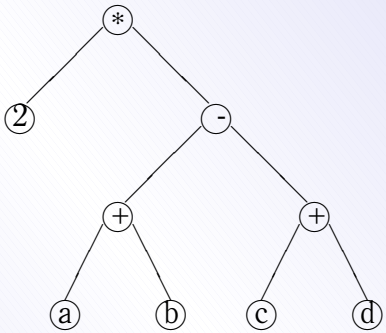


Рис. 3.2. Дерево разбора арифметического выражения.

3.2. Задача о коммивояжере

Задача 3.1. *Пусть имеется n городов. Некоторые из них связаны между собой дорогами различной длины. Бродячий торговец хотел бы посетить все эти города по одному разу, пройдя при этом наименьшее расстояние, и вернуться в город, из которого он начал свой путь.*

Эта задача называется *задачей о бродячем торговце* (задачей о коробейнике, задачей о коммивояжере). С точки зрения теории графов эта задача сводится к отысканию минимального гамильтонового пути на заданном графе.

Точно задача может быть решена методом полного перебора, что приводит к колоссальному объему вычислений в реальных задачах. А задачи такие

возникают очень часто, особенно на транспорте. В то же самое время можно найти решение, близкое к оптимальному, используя метод Монте-Карло.

Обозначим через a некоторую последовательность посещения городов, а через $L(a)$ — соответствующую этой последовательности длину пути. Мы хотим найти такую последовательность a, при которой длина пути была бы минимальной. Будем применять для этой цели алгоритм, подобный алгоритму Метрополиса, который применяется при моделировании магнитных систем.

1. Создаем какой-либо гамильтонов путь. Города, то есть вершины графа, можно выбирать случайным образом из имеющегося списка. Когда список будет исчерпан, то последний узел надо соединить с первым.

2. Случайным образом выбираем пару узлов и меняем их местами. При этом длина пути меняется на величину ΔL.

3. Если перестановка приводит к уменьшению пути, то мы ее принимаем, если перестановка приводит к увеличению пути, то она принимается с вероятностью $\exp(-\Delta L/(kT))$, где k — постоянная Больцмана, T — температура.

4. Повторяем пункты 2,3, пока не достигнем миниммальной длины пути.

Температура — это некоторый управляющий параметр. Первоначально он выбирается достаточно большим, а затем медленно понижается. Если при дальнейшем понижении температуры длина пути перестает меняться, то значит минимум достигнут.

Поскольку алгоритм основан на использовании случайного выбора узлов, то он относится к группе методов Монте-Карло. Понятно, что при малом

числе городов трудоемкость метода будет чрезвычайно большой. В то же время, поскольку трудоемкость обычных методов растет экспоненциально с ростом системы, этот метод становится весьма эффективным именно для больших систем.

3.3. Алгоритм Хошена-Копельмана

Алгоритм Хошена-Копельмана или алгоритм многократной маркировки кластеров был предложен в 1976 г. для решения задач теории перколяции. Этот алгоритм позволяет выделить связные подграфы (кластеры — по терминологии теории перколяции) некоторого случайного графа. Этот алгоритм может быть, в частности, применен для определения областей заливки экрана, на котором уже имеется некоторое изображение. Ещё одна область применения алгоритма — поиск пути в лабиринте, заданном в виде графа.

Важной особенность алгоритма Хошена-Копельмана является его однопроходность. За один проход алгоритм позволяет выяснить, к какому кластеру относится тот или иной узел решетки. Идея алгоритма заключается в том, что всем занятым узлам решетки приписываются различные кластерные метки.

Рассмотри работу алгоритма на примере квадратной решетки. Следует иметь в виду, что существует множество вариантов алгоритма, мы рассмотрим лишь простейший. Моделировать решетку будет массив a размером $L \times L$ случайным образом заполненный нулями и единицами.

Для того, чтобы работа со всеми элементами массива была единообразной, удобно добавить к нему одну строку с номером 0 и один столбец с

номером 0, заполненные нулями. Кроме того, создадим массив кластерных меток cl. Первоначально значения элементов этого массива совпадают с их номерами. В процессе проверки эти номера могут изменяться. Если значение элемента массива стало меньше его номера, то такую кластерную метку называют неправильной. Правильное значение кластерной метки — значение элемента массива.

Начнем просматривать последовательно все элементы массива, начиная с элемента $a[1,1]$. Нулевой столбец будем пропускать. Возможны следующие ситуации.

- Если значение текущего элемента массива равно 0, то переходим к следующему элементу.
- Если текущее значение равно 1, то осуществляем следующую проверку:
  - Если сосед слева $a[i,j-1]$ и сосед сверху $a[i-1,j]$ имеют значения 0, то в качестве рабочей гипотезы принимаем, что данный элемент входит в новый кластер, присваиваем текущему элементу номер очередной кластерной метки. Конечно, вполне может оказаться, что новый, как нам кажется, кластер является просто веткой какого-то другого кластера. Выяснить это мы сможем только просмотрев весь массив целиком.
  - Если сосед сверху имеет значение 0, а сосед слева не равен нулю, то текущая ячейка и ее сосед слева принадлежат одному и тому же кластеру. Текущему элементу присваиваем номер кластерной метки соседа слева.

– Если сосед сверху имеет значение, отличное от нуля, а сосед слева нулевое значение, то текущая ячейка и ее сосед сверху принадлежат одному и тому же кластеру. Однако, у соседа сверху кластерная метка может быть неправильной, так как в результате последующих проверок могло выясниться, что кластер, к которому принадлежит эта ячейка, слился с другим кластером. Поэтому текущему элементу следует присвоить номер *правильной кластерной метки* соседа сверху.

– Если и сосед сверху и сосед слева имеют ненулевые значения, то все три ячейки принадлежат одному кластеру. Текущему элементу присваиваем наименьший из номеров правильных кластерных меток соседа сверху и соседа сверху (рис. 3.3). Корректируем массив кластерных меток. Для этого в элемент массива cl, соответсвующий большей из кластерных меток заносим номер правильной кластерной метки.

В результате выполнения алгорима Хошена-Копельмана все узлы графа будут разделены по их принадлежности к тому или иному связному подграфу. Все узлы, имеющие одинаковые правильные кластерные метки принадлежат одному подграфу.

3.4. Алгоритм поиска в глубину

Алгоритм поиска в глубину (Depth First Search — DFS) может использоваться для поиска путей на графе или покрывающего дерева графа. Алгоритм

может быть разделен на два шага:

1. Создать связанный список. Каждый элемент должен (нести) переменную типа Boolean «проверено», установленную на (FALSE).

2. Выбрать непроверенный элемент из списка. Пометить его как проверенный. Выполнить процедуру поиска в глубину для соседнего непроверенного элемента. Повторять, пока список не исчерпается.

```
function DFS (vertex.V,i);
checked.V:=TRUE;
for vertex.W in (соседи V) do
     if not checked.W do DFS(vertex.W,i);
return; {конец рекурсивной функции DFS}
```

Программа 3.1. Поиск в глубину

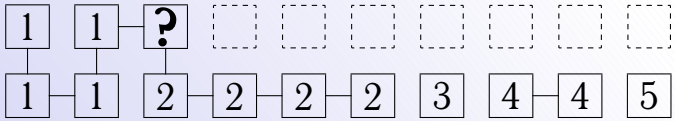


Рис. 3.3. Конфликт меток

Вся программа, включая создание связанного списка, не превысит одной страницы, и, что еще важнее, логика программы основана всего на пяти этих строках[1]. Вместо рекурсивной функции можно использовать стек.

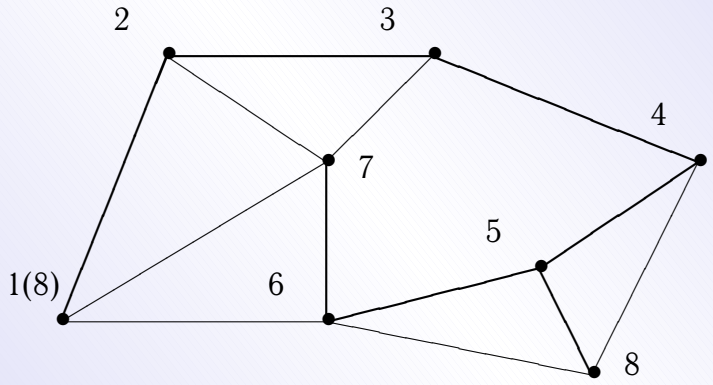


Рис. 3.4. Порядок просмотра узлов при поиске сначала в глубину. Жирными линиями показано покрывающее дерево, полученное в результате выполнения алгоритма.

[1]Нотация, используемая здесь, не следует строго конкретному языку программирования, но, можно надеяться, является достаточно ясной.

3.5. Алгоритм поиска в ширину

Алгоритм поиска в ширину (Breadth First Search — BFS) сходен с алгоритмо DFS, с той лишь разницей, что узлы помещаются не в стек, а в очередь. Алгоритм может применяться, чтобы найти кратчайший путь между двумя узлами.

1. Создайте связанный список. Каждый элемент должен нести две переменные типа (boolean): **проверенный** и **в очереди**, обе помеченные как (FALSE), и переменную типа (integer) ((LEVEL)), помеченную (0).

2. Пометьте два узла (скажем, (1) для узла с самой большой y-координатой и (2) для узла с наименьшим y).

3. Начните поиск в ширину элемента с меткой (2) с узла (1). Когда элемент с меткой (2) найден, BFS может вернуть длину наикратчайшего пути между (1) и (2), которая пропорциональна уровню поиска, достигнутого к тому моменту.

Функция BFS может быть использована для нахождения наикратчайшего пути следующим образом:

Ход программы для этой функции показан графически на рис. 3.5. Числа в круглых скобках соответствуют переменной (LEVEL), присвоенной узлу. Другое число соответствует порядку, в котором узлы входят в очередь.

Прямоугольники внизу, нарисованные сплошной линией, представляет собой очередь, заполненную до второго выполнения инструкций Read (queue) и Pop (queue). Стрелки показывают изменения в содержимом очереди, перед третьим исполнения упомянутых инструкций.

```
function BFS (vertex.V);
создать очередь с единственным элементом V;
InTheQueue.V:=TRUE;
while queue not empty
V:=Read (queue);
Pop (queue);
checked.V:=TRUE;
for vertex.W in (соседи V)
if ((not InTheQueue.W) AND (not checked.W)) do
Level.W:=Level.V + 1;
InTheQueue.W:=TRUE;
Inject (queue, W);
end; {do}
end; {for}
end; {while}
return; {конец функции BFS}
```

Программа 3.2. Поиск в ширину

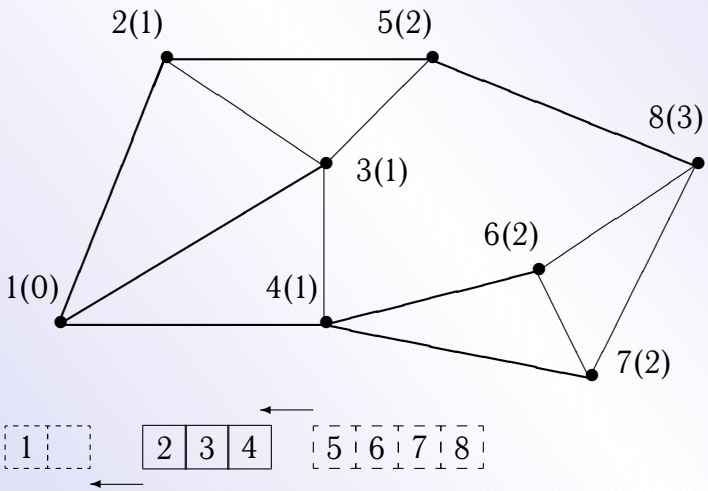


Рис. 3.5. Пример нумерации узлов при поиске в ширину. Внизу показана очередь, необходимая для второго шага.

После завершения процедуры целое число (Level), написанное на каждом элементе, означает наикратчайший путь из соответствующего узла к узлу, где поиск начался — начальный пункт поиска в ширину. Если поиск начался с узла, помеченного (1) (см. выше), значение (Level), присвоенное узлу с меткой (2), даст наикратчайший путь между этими двумя узлами (предполагается, что все соединения имеют равную длину).

3.6. Контрольная работа

Реализуйте на одном из языков программирования два любых из рассмотренных выше алгоритма.

Литература

1. Басакер Р., Саати Т. Конечные графы и сети. — М.: Наука, 1974. — 366 с.
2. Берж К. Теория графов и ее применение. — М.: ИЛ, 1962.
3. Биркгоф Г., Барти Т. Современная прикладная алгебра: М.: Мир, 1976.
4. Виленкин Н. Я. Комбинаторика. — М.: Наука, 1969.
5. Гаврилов Г. П., Сапоженко А. А Сборник задач по дискретной математике. — М.: Наука, 1977.
6. Горбатов В. А. Основы дискретной математики. — М.: ВШ, 1986.
7. Грэхем Р., Кнут Д., Паташник О. Конкретная математика. Основания информатики. — М.: Мир, 1998.
8. Емеличев В. А., Мельников О. И., Сарванов В. А., Тышкевич Р. И. Лекции по теории графов. — М.: Наука, 1990.

9. Ерусалимский Я. М. Дискретная математика: теория, задачи, приложения. — М.: Вузовская книга, 2000. — 280 с.

10. Зыков А. А. Основы теории графов. — М.: Наука, 1987.

11. Кристофидес Н. Теория графов. — М.: Мир, 1978.

12. Кузнецов О. П., Адельсон-Вельский Г. М. Дискретная математика для инженеров. — М.: Энергоатомиздат, 1988.

13. Кук Д., Бейз Г. Компьютерная математика. — М.: Наука, 1990.

14. Липский В. Комбинаторика для программистов. — М.: Мир, 1988. — 213 с., ил.

15. Нефедов В. Н., Осипова В. А. Курс дискретной математики. — М.: Изд-во МАИ, 1992. — 264 с.

16. Новиков Ф. А. Дискретная математика для программистов. — СПб.: Питер, 2000.

17. Оре О. Графы и их применение. — М.: Мир, 1965.

18. Оре О. Теория графов. — М.: Наука, 1968.

19. Райзер Г. Дж. Комбинаторная математика. — М.: Мир, 1970.

20. Рейнгольд Э., Нивергельт Ю., Део Н. Комбинаторные алгоритмы. Теория и практика. — М.: Мир, 1980. — 476 с.

21. Риордан Дж. Введение в комбинаторный анализ. — М.: ИЛ, 1963.

22. Сачков В. Н. Введение в комбинаторные методы дискретной математики. — М.: Наука, 1982.

23. Уилсон Р. Введение в теорию графов. — М.: Мир, 1977.

24. Харари Ф. Теория графов. — М.: Мир, 1973.

25. Холл М. Комбинаторный анализ. — М.: ИЛ, 1963.

26. Холл М. Комбинаторика. — М.: Мир, 1970.

27. Яблонский С. В. Введение в дискретную математику. — М.: Наука, 1979.