АЛГОРИТМЫ□ ЕВРЕСТИЧЕСКИЕ□ РАСКРАСКИ□ ГРАФА

ВВЕДЕНИЕ

Теория графов – теоретическая дисциплина, изучаемая в курсе «Дискретная математика». В практическом применении данная теория позволяет описывать и исследовать многие физические, экономические, социальные и другие системы, поэтому теория графов является интенсивно развивающимся разделом дискретной математики.

На практике бывает целесообразно изобразить некоторую систему в виде схем, составленных из точек (объектов) и линий, соединяющих определенные пары этих объектов и представляющих связи между ними. Такое изображение структур различных систем, которое однозначно ее описывает, и носит название – графы. На основе уже составленного графа изучаются свойства системы, причины и следствия взаимосвязи между ее объектами.

Цель изучения теории графов состоит в формировании у студентов умений и навыков, необходимых при исследовании различных систем.

В данных методических указаниях вводится стандартная терминология, используемая в теории графов, и разбираются основные задачи и алгоритмы, реализованные с помощью графов. В частности, организация поиска в структуре данных, раскраска графов, нахождение кратчайших путей на графе, задача коммивояжера. Кроме того, они содержат большое количество задач для проведения практических занятий, лабораторных и контрольных работ. Все задачи прошли апробацию в процессе обучения студентов по курсу «Дискретная математика».

ОСНОВНЫЕ ПОНЯТИЯ И СПОСОБЫ ПРЕДСТАВЛЕНИЯ

Граф – это совокупность двух множеств: множества вершин и множества ребер. Вершины и ребра называются элементами графа. Каждое ребро соединяет пару вершин. Вершины, прилегающие к одному и тому же ребру, называют *смежными* или *концевыми* данного ребра.

Граф Г = (V, A), V – множество вершин, A – множество ребер. Каждому элементу $a \in A$ соответствует пара (v_i, v_j) элементов множества V. Если пара считается упорядоченной, то ребро называется ориентированным, если же эта пара считается неупорядоченной, то ребро – неориентированное. Вершина и ребро называются *инцидентными* друг другу, если вершина является концевой для данного ребра. Вершина, не инцидентная никакому ребру, называется *изолированной*. Число ребер, инцидентных одной вершине, называется *степенью* данной вершины.

Маршрут (путь) на графе называется последовательность ребер $a_1 \ldots a_n$, где каждое ребро a_i и a_{i+1} имеют общую вершину.

Цепь – маршрут, у которого все ребра различны. Маршрут называется *замкнутым,* если $a_1 = a_n$.

Простая цепь – маршрут, у которого все вершины различны.

Цикл – замкнутая цепь*.*

Простой цикл – замкнутая простая цепь.

Граф называется *связным*, если найдется *путь* из любой вершины v_i в любую вершину v_j .

Графы можно представить разными способами:

– графическим изображением;

– прямым перечислением вершин и ребер;

– структурой смежности;

– матрицей смежности;

– матрицей инцидентности.

Графическое представление является наиболее наглядным способом задания графа. Графы изображаются в виде геометрических фигур. Вершины отмечаются точками, а ребра линиями, соединяющими эти точки. Пример графического представления графа (рис. 1 и рис. 2).

Прямое перечисление вершин и ребер. Для задания графа достаточно задать множество вершин и перечислить все его ребра, где каждое ребро представлено в виде пары вершин, которое оно соединяет.

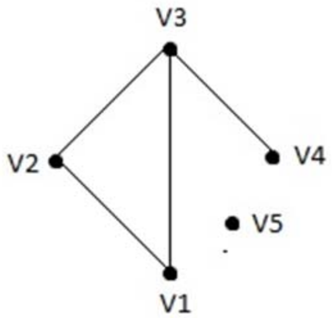


Рис. 1

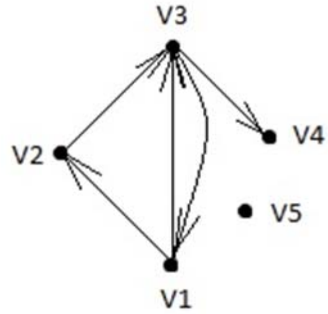


Рис. 2

$(\{v_1, v_2, v_3, v_4 \dots\}, (v_1, v_2), (v_1, v_3), (v_3, v_4), \dots)$

Пример 1.

Граф (рис. 1)

$(\{v_1, v_2, v_3, v_4, v_5\}, (v_1, v_2), (v_1, v_3), (v_2, v_3), (v_3, v_4))$

Пример 2.

Граф (*рис.*2)

$(\{v_1, v_2, v_3, v_4, v_5\}, (v_1, v_2), (v_1, v_3), (v_2, v_3), (v_3, v_1), (v_3, v_4))$

Структура смежности. Данная структура состоит из списка вершин графа смежных с вершиной v_i. Данные списки составляются для всех вершин графа.

Пример 1.

Граф (рис. 1)

$v_1 : v_2, v_3$

$v_2 : v_1, v_3$

$v_3 : v_1, v_2, v_4$

$v_4 : v_3$

$v_5 :$

Пример 2.

Граф (рис. 2)

$v_1 : v_2, v_3$

$v_2 : v_3$

$v_3 : v_1, v_4$

$v_4 :$

$v_5 :$

Матрица смежности. Один из самых удобных способов. Пусть некоторый граф Г имеет *n* вершин. Составим матрицу, имеющую *n* строк и *n* столбцов. Каждая строка и столбец матрицы соответствует некоторой вершине. Элемент матрицы a_{ij} равен 1, если ребро (v_i, v_j) существует и 0, если вершины v_i и v_j не связаны ребром.

Пример 1.

Граф (рис. 1)

$$
\begin{array}{c} \\ V1 \\ V2 \\ V3 \\ V4 \\ V5 \end{array}
\begin{array}{c} \begin{array}{ccccc} V1 & V2 & V3 & V4 & V5 \end{array} \\
\begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array}
$$

Пример 2.

Граф (рис. 2)

$$
\begin{array}{c} \\ V1 \\ V2 \\ V3 \\ V4 \\ V5 \end{array}
\begin{array}{c} \begin{array}{ccccc} V1 & V2 & V3 & V4 & V5 \end{array} \\
\begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array}
$$

Матрица инцидентности. Пусть некоторый граф Г имеет n вершин и m ребер (рис. 3, рис. 4). Составим матрицу, имеющую n строк и m столбцов. Каждая строка матрицы соответствует некоторой вершине, а каждый столбец – ребру.

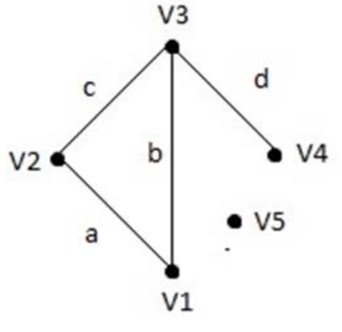


Рис. 3

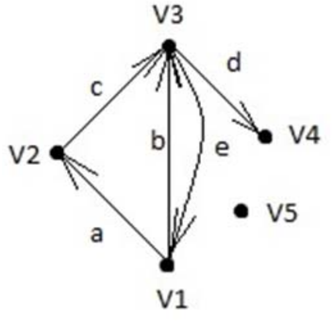


Рис. 4

Для неориентированного графа элемент матрицы b_{ij} равен 1, если вершина v_i инцидентна ребру a_j и 0, если вершина не инцидентна данному ребру. Для ориентированного графа элемент b_{ij} равен:

+1, если ребро a_j выходит из вершины v_i,

–1, если ребро a_j входит в вершину v_i,

0, если вершина v_i не инцидентна ребру a_j.

Пример 1.

Граф (рис. 3)

$$\begin{array}{c c} & \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{array}{c} V1 \\ V2 \\ V3 \\ V4 \\ V5 \end{array} & \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

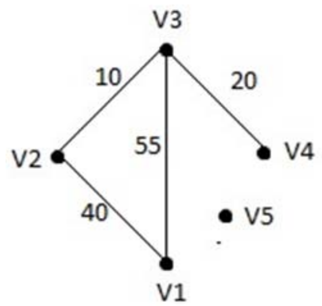
Пример 2.

Граф (рис. 4)

$$\begin{array}{c c} & \begin{array}{ccccc} a & b & c & d & e \end{array} \\ \begin{array}{c} V1 \\ V2 \\ V3 \\ V4 \\ V5 \end{array} & \begin{pmatrix} 1 & 1 & 0 & 0 & -1 \\ -1 & 0 & 1 & 0 & 0 \\ 0 & -1 & -1 & 1 & 1 \\ 0 & 0 & 0 & -1 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

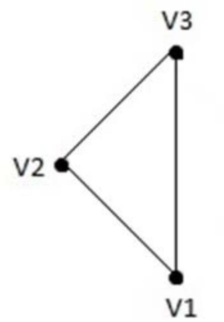
Матрица весов. Граф называется взвешенным, если каждому его ребру сопоставляется число (*вес*). Взвешенный граф может быть представлен матрицей весов W, где элемент данной матрицы w_{ij} – вес ребра, соединяющего вершины v_i и v_j.

Пример.



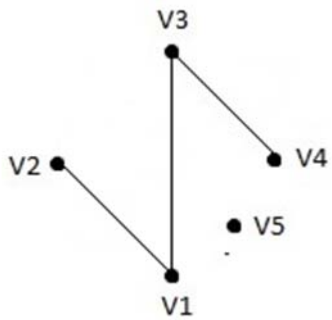
Подграф исходного графа Г = (V, A) – это граф Г' = (V', A'), где $V' \subseteq V$, а $A' \subseteq A$ (в A' входят только те ребра, оба конца которых принадлежат V').

Пример.



Остовной подграф – это подграф графа Г, для которого $V' = V$.

Пример.



Дерево – связный граф, не имеющий циклов. Вершина графа инцидентная только одному ребру называется *концевой* или *висячей*. Часто в дереве выделяют одну вершину, являющейся *«начальной»*. Такое дерево называют *корневым*, а эту вершину – *корнем*.

Остовное дерево связного графа – любой его *остовной подграф*, являющийся деревом.

МЕТОДЫ ПОИСКА В ГЛУБИНУ И В ШИРИНУ НА ГРАФЕ

Процедура поиска на графе, это способ систематического исследования вершин графа на основании данного алгоритма.

Выполнение поиска на графе:

Шаг 1. Пометить все ребра и вершины графа, как «новые» (неисследованные).

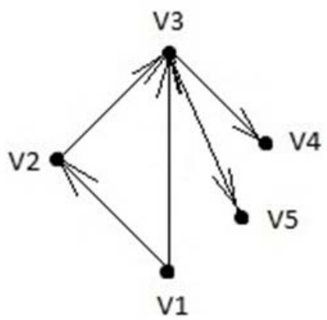
Шаг 2. (Выбор новой начальной вершины) Выбрать «новую» вершину и пометить ее как «старую» (исследованную). Если «новых» вершин нет, то остановиться.

Шаг 3. Если нет «новых» ребер, выходящих из «старых» вершин, то перейти к **шагу 2** (вся часть графа достижимая из текущей начальной вершины исследована). В противном случае выбрать «новое» ребро, выходящее из некоторой «старой» вершины, и пометить его как «старое». Если вторая вершина ребра «новая», то ее сделать «старой». Повторить **шаг 3**.

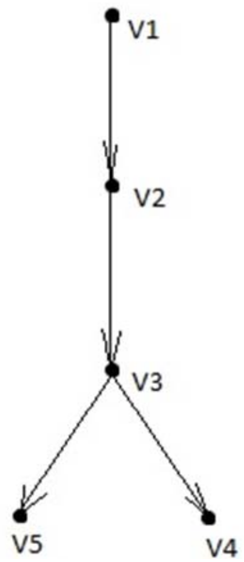
Предположим, что все вершины в графе достижимы из первой начальной вершины, выбранной на **шаге 2**, тогда поиск порождает *остовное дерево*. *Корнем остовного дерева* является начальная вершина, *ребром* – ребра, которые в момент их исследования на **шаге 3** ведут в новые вершины.

На основании различия выбора новых ребер на **шаге 3**, выделяют два метода реализации поиска: *поиск* в *глубину* и *поиск в ширину*. При поиске в *глубину* каждое новое выбранное ребро на **шаге 3**, это некоторое ребро, выходящее из последней *исследованной* вершины, имеющей *неисследованные* ребра. При поиске в ширину, ребро, выбираемое на **шаге 3**, это некоторое ребро, выходящее из первой *исследованной* вершины, имеющей *неисследованные* ребра. Такой поиск разбивает вершины на уровни, в зависимости от их расстояния от начальной вершины.

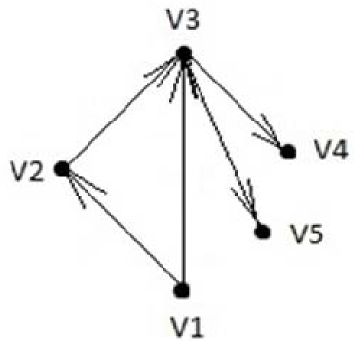
Пример 1. Поиск в глубину на данном графе.



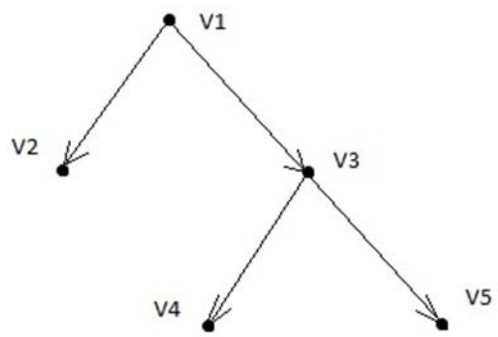
Выберем за начальную вершину V_1. Включаем в результат поиска ребра $(V_1; V_2) \to (V_2; V_3) \to (V_3; V_4)$. Выбор нового ребра из вершины V_4 не возможен, возвращаемся к предыдущей вершине V_3 и выбираем новое ребро, ведущее к новой вершине, $(V_3; V_5)$. Все вершины графа достигнуты, получено остовное дерево графа.



Пример 2. Поиск в ширину на данном графе.



Выберем за начальную вершину V_1. Берем все возможные ребра, ведущие из вершины V_1: (V_1; V_2) и (V_1; V_3). Переходим в вершину V_2, новых ребер ведущих из этой вершины в новые вершины нет. Переходим к вершине V_3: добавляем в дерево нашего поиска ребра (V_3; V_4) и (V_3; V_5). Все вершины графа достигнуты, получено остовное дерево графа.



РАСКРАСКА ГРАФОВ

Раскраска графов – это принцип разбиения вершин (ребер) графа на подмножества, при котором смежные вершины (ребра) графа являются элементами разных подмножеств. Количество полученных множеств соответствуют количеству *цветов* при раскраске графа. Суть задачи о раскраске графа состоит в нахождение такого разбиения, которое дает наименьшее количество цветов (подмножеств). Граф, на котором рассматривается данная задача, должен быть полностью неориентированным, без петель и кратных дуг.

Раскраска вершин

Наименьшее количество цветов, которое требуется для раскраски вершин графа, называется его *хроматическим числом* и обозначается Χ(Г). Для нахождения *хроматического числа* графа можно использовать алгоритм *неявного перебора*. Для реализации данного алгоритма упорядочим вершины графа. Присвоим каждой вершине номер V_i. Получим начальную допустимую раскраску. Для этого:

1. Окрасить вершину V_i в первый цвет.
2. Последовательно окрашиваем все оставшиеся вершины, используя принцип: вершина окрашивается в цвет наименьший по номеру (во множестве упорядоченных цветов), который еще не использовался при окраске смежных с ней вершин.

В результате получим r цветов, использованных для раскраски данного графа. Предположим, что существует раскраска с использованием r–1 цвета, т.е. вершины, окрашенные в цвет r, надо перекрасить, но это сделать невозможно, пока не будут перекрашены некоторые вершины смежные с ними. Для этого осуществим *шаг возвращения* следующем образом: из множества вершин смежных с вершиной V_i, окрашеной в цвет номера r, выберем ту вершину, у которой наибольший номер цвета r_i (пусть номер данной вершины $V_к$) и попытаемся перекрасить ее в цвет $r_i' \geq r_i + 1$. Если $r_i' < r$, то последовательно перекрашиваем все вершины, начиная с номера $V_{к+1}$, применяя правило под пунктом 2, учитывая то, что цвет под номером r использовать нельзя. Если потребуется использование цвета r, то от данной вершины опять исполним *шаг возвращения*. Так же осуществим данный шаг, если $r_i' = r$. Алгоритм заканчивает свою работу, когда на шаге возвращения мы достигаем вершину V_1.

Приближенные алгоритмы раскрашивания

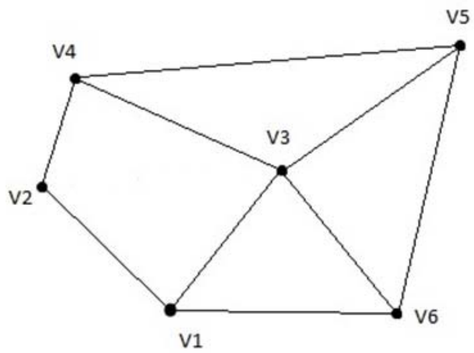
При раскраске графов с большим числом вершин вызывает затруднение использование точных алгоритмов раскрашивания, в связи с их большой трудоемкостью вычисления. Для таких графов были разработаны эвристические, т.е. приближенные, алгоритмы раскрашивания, которые дают очень хорошее приближение для определения хроматического числа. Один из таких методов можно построить, используя «жадный» алгоритм. Для его реализации надо упорядочить все вершины графа по неубыванию их степеней. До тех пор пока не раскрасим все вершины графа повторять:

Шаг 1. Раскрашиваем вершину V_i с наибольшей степенью в цвет i (на первом этапе $i = 1$) с учетом запрета наложенного на нее по цветам.

Шаг 2. На все вершины, смежные с вершиной V_i, накладываем запрет на цвет i.

Шаг 3. $i = i + 1$.

Пример:



Шаг 0. Упорядочим вершины графа и определим их степень.

Цвет раскраски						
вершина	V_1	V_2	V_3	V_4	V_5	V_6
степень	3	2	4	3	3	3
запретные цвета						

Шаг 1. Выберем вершину с наибольшей степенью – V_3 и раскрасим ее в цвет №1. Вычеркиваем данную вершину из рассмотрения. Всем вершинам, смежным с V_3, ставим запрет на первый цвет и уменьшаем их степень на единицу.

Цвет раскраски			№1			
вершина	V_1	V_2	V_3	V_4	V_5	V_6
степень	3 2	2 2	4 –	3 2	3 2	3 2
запретные цвета	№1			№1	№1	№1

Шаг 2. Выбираем V_1. Для данной вершины уже стоит запрет на цвет №1, следовательно, раскрашиваем ее в цвет №2. Накладываем запрет на смежные вершины и уменьшаем их степень.

Цвет раскраски	№2		№1			
вершина	V_1	V_2	V_3	V_4	V_5	V_6
степень	3 2 –	2 2 1	4 – –	3 2 2	3 2 2	3 2 1
запретные цвета	№1	 №2		№1	№1	№1 №2

Шаг 3.

Цвет раскраски	№2		№1	№2		
вершина	V_1	V_2	V_3	V_4	V_5	V_6
степень	3 2 – –	2 2 1 0	4 – – –	3 2 2 –	3 2 2 1	3 2 1 1
запретные цвета	№1	 №2		№1	№1 №2	№1 №2

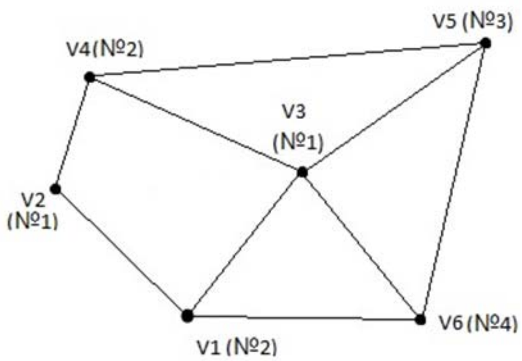
Шаг 4.

Цвет раскраски	№2		№1	№2	№3	
вершина	V_1	V_2	V_3	V_4	V_5	V_6
степень	3 2 – – –	2 2 1 0 0	4 – – – –	3 2 2 – –	3 2 2 1 –	3 2 1 1 0
запретные цвета	№1	 №2		№1	№1 №2	№1 №2 №3

Шаг 5. Раскрашиваем последние вершины, с учетом наложенных запретов.

Цвет раскраски	№2	№1	№1	№2	№3	№4
вершина	V_1	V_2	V_3	V_4	V_5	V_6
степень	3 2 – – –	2 2 1 0 0	4 – – – –	3 2 2 – –	3 2 2 1 –	3 2 1 1 0
запретные цвета	№1	№2		№1	№1 №2	№1 №2 №3

Получаем оптимальную раскраску графа:

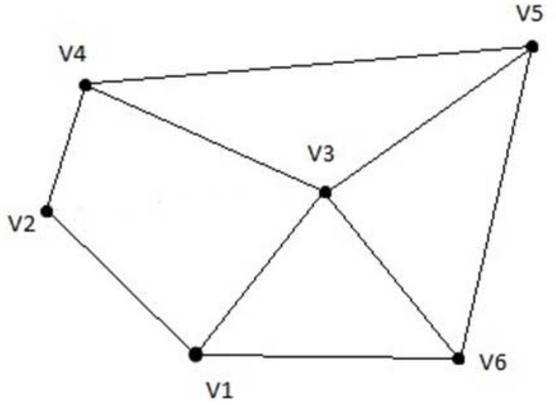


Алгоритм раскрашивания графа, основанный на действие со строками матрицы смежности. Зададим граф матрицей смежности, учитывая, что вершина смежная сама себе, т.е. диагональные элементы матрицы равны 1.

Шаг 1. $l = 1$, $k = 1$.

Шаг 2. В строке i находим первый элемент равный 0, стоящий в столбце $j = n$, и изменяем строку i, путем «сложения» данной строки со строкой $j = n$. Операция «сложения» производим согласно правилу: $0 + 0 = 0$, $0 + 1 = 1$, $1 + 0 = 1$, $1 + 1 = 1$. Строка i и строка j образуют цветовую группу под номером k. Вычеркиваем строку j из дальнейшего рассмотрения. Повторяем действия со строкой i пока в ней остаются 0, добавляя в k-ю цветовую группу новые строчки j. Как только строка i будет состоять из одних 1, формирование k-ой цветовой группы закончено. Кладем $k = k + 1$, находим новую не зачеркнутую строчку с номером N, приравниваем $i = N$ и повторяем **шаг 2**, если новых, еще не зачеркнутых строк больше не осталось, то следует остановиться.

Пример.



Составим матрицу смежности для данного графа.

$$\begin{array}{c} \\ V1\\ V2\\ V3\\ V4\\ V5\\ V6 \end{array}\begin{array}{c} \begin{matrix} V1 & V2 & V3 & V4 & V5 & V6 \end{matrix}\\ \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 1\\ 1 & 1 & 1 & 1 & 0 & 0\\ 1 & 1 & 0 & 1 & 1 & 1\\ 0 & 1 & 1 & 1 & 1 & 0\\ 0 & 0 & 1 & 1 & 1 & 1\\ 1 & 0 & 1 & 0 & 1 & 1 \end{pmatrix} \end{array}$$

Цвет №1. Ведущая строка – первая. Находим первый 0, он стоит в четвертом столбце. Добавляем четвертую строчку в текущий цвет и «складываем» ее с первой. Получаем:

$$\begin{array}{cc} & \\ №1 & V1\\ & V2\\ & V3\\ №1 & V4\\ & V5\\ & V6 \end{array}\begin{array}{c} \begin{matrix} V1 & V2 & V3 & V4 & V5 & V6 \end{matrix}\\ \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1\\ 1 & 1 & 1 & 1 & 0 & 0\\ 1 & 0 & 1 & 1 & 1 & 1\\ \cancel{0} & \cancel{1} & \cancel{1} & \cancel{1} & \cancel{1} & \cancel{0}\\ 0 & 0 & 1 & 1 & 1 & 1\\ 1 & 0 & 1 & 0 & 1 & 1 \end{pmatrix} \end{array}$$

Больше 0 в ведущей строке нет, формирование цвета №1 закончено.

Цвет №2. Ведущая строка – вторая.

$$\begin{array}{cc} & \\ №1 & V1\\ №2 & V2\\ & V3\\ №1 & V4\\ & V5\\ & V6 \end{array}\begin{array}{c} \begin{matrix} V1 & V2 & V3 & V4 & V5 & V6 \end{matrix}\\ \begin{pmatrix} \cancel{1} & \cancel{1} & \cancel{1} & \cancel{1} & \cancel{1} & \cancel{1}\\ 1 & 1 & 1 & 1 & 0 & 0\\ 1 & 0 & 1 & 1 & 1 & 1\\ \cancel{0} & \cancel{1} & \cancel{1} & \cancel{1} & \cancel{1} & \cancel{0}\\ 0 & 0 & 1 & 1 & 1 & 1\\ 1 & 0 & 1 & 0 & 1 & 1 \end{pmatrix} \end{array}$$

Первый 0 в пятом столбце. «Складываем» вторую и пятую строчку.

$$\begin{array}{cc} & \begin{matrix} V1 & V2 & V3 & V4 & V5 & V6 \end{matrix} \\ \begin{matrix} \text{№1} & V1 \\ \text{№2} & V2 \\ & V3 \\ \text{№1} & V4 \\ \text{№2} & V5 \\ & V6 \end{matrix} & \begin{pmatrix} \cancel{1} & \cancel{1} & \cancel{1} & \cancel{1} & \cancel{1} & \cancel{1} \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 \\ \cancel{0} & \cancel{1} & \cancel{1} & \cancel{1} & \cancel{1} & \cancel{0} \\ \cancel{0} & \cancel{0} & \cancel{1} & \cancel{1} & \cancel{1} & \cancel{1} \\ 1 & 0 & 1 & 0 & 1 & 1 \end{pmatrix} \end{array}$$

Нулей больше нет, цвет №2 сформирован.

Цвет №3. Ведущая строка третья.

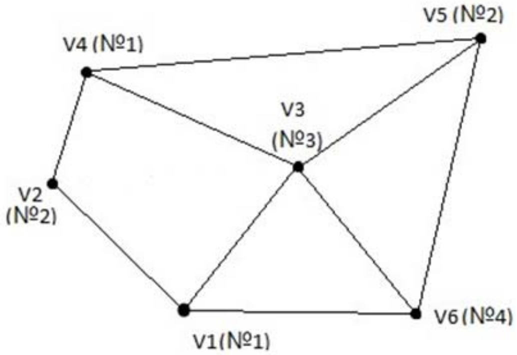
$$\begin{array}{cc} & \begin{matrix} V1 & V2 & V3 & V4 & V5 & V6 \end{matrix} \\ \begin{matrix} \text{№1} & V1 \\ \text{№2} & V2 \\ & V3 \\ \text{№1} & V4 \\ \text{№2} & V5 \\ & V6 \end{matrix} & \begin{pmatrix} \cancel{1} & \cancel{1} & \cancel{1} & \cancel{1} & \cancel{1} & \cancel{1} \\ \cancel{1} & \cancel{1} & \cancel{1} & \cancel{1} & \cancel{1} & \cancel{1} \\ 1 & 0 & 1 & 1 & 1 & 1 \\ \cancel{0} & \cancel{1} & \cancel{1} & \cancel{1} & \cancel{1} & \cancel{0} \\ \cancel{0} & \cancel{0} & \cancel{1} & \cancel{1} & \cancel{1} & \cancel{1} \\ 1 & 0 & 1 & 0 & 1 & 1 \end{pmatrix} \end{array}$$

В данной строке 0 находится во втором столбце, но вторая строчка уже распределена. Следовательно, цвет сформирован.

Цвет №4. Строка шесть.

$$\begin{array}{cc} & \begin{matrix} V1 & V2 & V3 & V4 & V5 & V6 \end{matrix} \\ \begin{matrix} \text{№1} & V1 \\ \text{№2} & V2 \\ \text{№3} & V3 \\ \text{№1} & V4 \\ \text{№2} & V5 \\ \text{№4} & V6 \end{matrix} & \begin{pmatrix} \cancel{1} & \cancel{1} & \cancel{1} & \cancel{1} & \cancel{1} & \cancel{1} \\ \cancel{1} & \cancel{1} & \cancel{1} & \cancel{1} & \cancel{1} & \cancel{1} \\ \cancel{1} & \cancel{0} & \cancel{1} & \cancel{1} & \cancel{1} & \cancel{1} \\ \cancel{0} & \cancel{1} & \cancel{1} & \cancel{1} & \cancel{1} & \cancel{0} \\ \cancel{0} & \cancel{0} & \cancel{1} & \cancel{1} & \cancel{1} & \cancel{1} \\ 1 & 0 & 1 & 0 & 1 & 1 \end{pmatrix} \end{array}$$

Полученная раскраска графа:

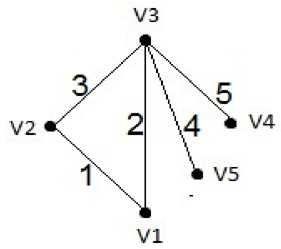


Раскраска ребер

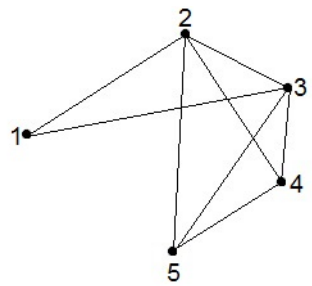
Зачастую в теории графов требуется осуществить раскраску не вершин графа, а его ребер, с условием, что любые два смежных ребра графа, имеющих общую концевую вершину, должны иметь различный цвет. Наименьшее количество цветов, используемых в данном случае, называют *хроматическим индексом графа*.

Простейший алгоритм для данной задачи состоит в составление нового двойственного графа, в котором ребра исходного графа соответствуют вершинам нового графа. Данные вершины соединяются ребром в том случае, если им соответствующие ребра были смежными. После чего переходим к раскраске наилучшим способом вершин двойственного графа, получая раскраску ребер исходного графа.

Пример.



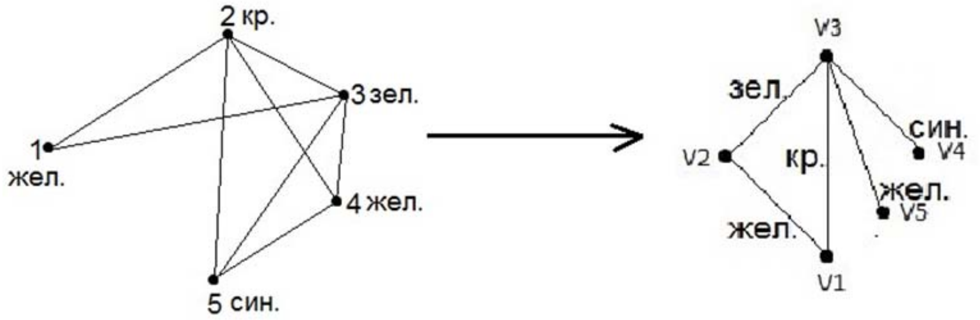
Составим двойственный граф, в котором смежные вершины соответствуют смежным ребрам исходного графа.



Найдем раскраску полученного графа.

Цвет раскраски	жел.	кр.	зел.	жел.	син.
вершина	1	2	3	4	5
степень	2 1 0 0	4 – – –	4 3 – –	3 2 1 –	3 2 1 0
запретные цвета	кр. зел.		кр.	кр. зел.	кр. зел. жел.

По раскраске данного графа раскрасим ребра исходного графа.



КОНТРОЛЬНЫЕ ЗАДАНИЯ №1

1. Дан неориентированный граф.

– Задать данный граф графически, в виде структуры смежности, матрицей смежности, матрицей инцидентности.

– Раскрасить вершины графа:

1) с помощью «жадного» алгоритма,

2) с помощью действий над строками матрицы смежности.

1. ({1,2,3,4,5,6,7}: (1,2), (1,3), (1,4), (2,3), (2,7), (3,4), (4,1), (4,5), (4,6), (5,6), (6,7), (7,1)).

2. ({1,2,3,4,5,6,7}: (1,2), (1,3), (1,4), (2,3), (3,4), (4,6), (5,6), (5,7), (7,1)).

3. ({1,2,3,4,5,6,7}: (1,6), (1,7), (2,3), (2,5), (2,6), (3,4), (4,5), (4,6), (5,6), (6,7)).

4. ({1,2,3,4,5,6,7}: (1,4), (2,7), (3,4), (3,6), (3,7), (4,5), (4,6), (5,6), (6,7)).

5. ({1,2,3,4,5,6,7}: (1,2), (1,3), (1,4), (3,4), (3,6), (4,5), (5,6), (5,7), (6,7), (7,1)).

6. ({1,2,3,4,5,6,7}: (1,2), (1,6), (2,3), (2,5), (2,7), (3,4), (3,5), (3,7), (5,6), (6,7)).

7. ({1,2,3,4,5,6,7}: (1,2), (1,3), (1,7), (2,3), (2,4), (2,5), (2,6), (3,4), (3,6), (4,5), (5,6), (5,7), (6,7)).

8. ({1,2,3,4,5,6,7}: (1,2), (1,3), (1,6), (1,7), (2,3), (2,6), (2,7), (3,4), (3,5), (4,5), (5,6), (6,7)).

9. ({1,2,3,4,5,6,7}: (1,4), (1,5), (2,5), (2,6), (2,7), (3,6), (3,7), (4,5), (5,6), (6,7)).

10. ({1,2,3,4,5,6,7}: (1,2), (1,4), (2,3), (2,5), (3,4), (3,7), (4,5), (4,7), (5,6), (6,7)).

11. ({1,2,3,4,5,6,7}: (1,2), (1,5), (2,5), (3,4), (3,5), (3,7), (4,5), (5,6), (6,7)).

12. ({1,2,3,4,5,6,7}: (1,3), (1,4), (1,5), (2,6), (2,7), (3,7), (4,5), (4,6), (5,7)).

13. ({1,2,3,4,5,6,7}: (1,2), (1,5), (2,3), (2,5), (2,6), (3,6), (3,7), (4,5), (4,7), (5,6), (6,7)).

14. ({1,2,3,4,5,6,7}: (1,2), (1,3), (1,4), (2,3), (2,5), (2,6), (3,4), (4,5), (4,6), (4,7), (5,6), (6,7)).

15. ({1,2,3,4,5,6,7}: (1,2), (1,3), (1,7), (2,3), (2,6), (3,4), (3,5), (3,6), (3,7), (5,6), (6,7)).

16. ({1,2,3,4,5,6,7}: (1,2), (1,3), (1,7), (2,3), (2,6), (3,4), (3,5), (3,6), (3,7), (4,5), (4,6), (5,6), (5,7), (6,7)).

17. ({1,2,3,4,5,6,7}: (1,2), (1,5), (1,6), (2,3), (2,6), (2,7), (3,5), (3,7), (4,5), (4,7), (5,6), (6,7)).

18. ({1,2,3,4,5,6,7}: (1,2), (1,5), (1,7), (2,3), (2,5), (3,4), (3,5), (3,6), (3,7), (4,5), (5,6), (6,7)).

19. ({1,2,3,4,5,6,7}: (1,2), (1,3), (1,4), (1,5), (1,6), (1,7), (2,3), (3,1), (3,4), (3,5), (3,6), (3,7), (4,5), (5,6), (5,7), (6,7)).

20. ({1,2,3,4,5,6,7}: (1,2), (1,3), (2,3), (3,1), (3,4), (3,5), (3,6), (3,7), (5,6), (6,7), (7,1)).

21. ({1,2,3,4,5,6,7}: (1,4), (1,5), (1,6), (2,3), (2,5), (2,6), (3,4), (3,6), (3,7), (4,5), (5,7), (6,7)).

22. ({1,2,3,4,5,6,7}: (1,2), (1,3), (1,4), (1,7), (2,7), (3,4), (3,6), (3,7), (4,5), (4,6), (5,6), (5,7), (6,7)).

23. ({1,2,3,4,5,6,7}: (1,2), (1,5), (1,6), (1,7), (2,3), (2,7), (3,4), (3,6), (3,7), (4,5), (4,6), (5,6), (5,7), (6,7)).

24. ({1,2,3,4,5,6,7}: (1,3), (1,6), (1,7), (2,5), (3,4), (3,7), (4,5), (4,6), (5,6), (6,7)).

25. ({1,2,3,4,5,6,7}: (1,2), (1,3), (1,5), (1,6), (2,3), (2,7), (3,4), (3,5), (3,7), (4,5), (4,6), (5,6), (5,7)).

26. ({1,2,3,4,5,6,7}: (1,2), (1,4), (2,3), (2,6), (2,7), (3,4), (3,5), (3,7), (4,5), (5,6), (5,7), (6,7)).

27. ({1,2,3,4,5,6,7}: (1,2), (1,4), (1,6), (1,7), (2,3), (2,5), (2,6), (3,6), (3,7), (4,5), (4,6), (5,6), (5,7)).

28. ({1,2,3,4,5,6,7}: (1,2), (1,4), (1,7), (2,3), (2,5), (2,6), (3,4), (3,6), (4,5), (4,6), (4,7), (5,6), (6,7)).

29. ({1,2,3,4,5,6,7}: (1,3), (1,4), (2,3), (2,6), (2,7), (3,4), (4,5), (4,6), (5,6), (5,7)).

30. ({1,2,3,4,5,6,7}: (1,2), (1,3), (1,5), (2,6), (2,7), (3,4), (3,7), (4,5), (4,7), (5,6), (6,7)).

2. Дан ориентированный граф.

1. Задать данный граф графически, в виде структуры смежности, матрицей смежности, матрицей инцидентности.

2. Организовать на графе поиск в глубину и поиск в ширину, изобразить результирующее дерево поиска.

1. ({1,2,3,4,5,6,7}: (1,2), (1,3), (1,4), (2,3), (2,7), (3,4), (4,1), (4,5), (4,6), (5,6), (6,7), (7,1)).

2. ({1,2,3,4,5,6,7}: (1,3), (1,4), (2,1), (2,3), (3,1), (3,4), (4,6), (4,7), (5,7), (6,5), (7,1)).

3. ({1,2,3,4,5,6,7}: (1,6), (1,7), (2,3), (2,5), (2,6), (3,2), (3,4), (4,5), (4,6), (5,2), (5,4), (6,5), (6,7)).

4. ({1,2,3,4,5,6,7}: (1,4), (3,6), (3,7), (4,1), (4,3), (4,5), (6,4), (6,5), (6,7), (7,2)).

5. ({1,2,3,4,5,6,7}: (1,2), (1,3), (1,4), (3,4), (3,6), (4,5), (5,7), (6,5), (6,7), (7,1)).

6. ({1,2,3,4,5,6,7}: (1,6), (2,1), (2,3), (2,5), (2,6), (3,4), (3,5), (3,7), (6,5), (6,7), (7,2), (7,3)).

7. ({1,2,3,4,5,6,7}: (1,2), (1,3), (1,7), (2,3), (2,4), (2,6), (3,1), (3,4), (3,6), (5,2), (5,4), (5,7), (6,3), (6,5), (6,7)).

8. ({1,2,3,4,5,6,7}: (1,3), (1,6), (2,1), (2,3), (2,7), (3,4), (3,5), (5,3), (5,4), (5,6), (6,2), (7,1), (7,2), (7,6)).

9. ({1,2,3,4,5,6,7}: (1,4), (1,5), (2,6), (2,7), (4,5), (5,2), (6,3), (6,2), (6,5), (6,7), (7,3)).

10. ({1,2,3,4,5,6,7}: (1,2), (1,4), (2,5), (3,2), (3,7), (4,3), (4,5), (5,6), (6,7), (7,3), (7,4)).

11. ({1,2,3,4,5,6,7}: (1,2), (2,5), (3,5), (3,7), (4,3), (5,1), (5,2), (5,4), (5,6), (6,7), (7,3)).

12. ({1,2,3,4,5,6,7}: (1,3),(1,4),(2,6),(3,7),(4,6),(5,1),(5,4),(7,2),(7,5)).

13. ({1,2,3,4,5,6,7}: (1,2), (1,5), (2,5), (2,6), (3,2), (4,5), (5,1), (5,4), (5,6), (6,3), (6,7), (7,3), (7,4)).

14. ({1,2,3,4,5,6,7}: (1,3), (1,4), (2,1), (2,5), (2,6), (3,2), (4,5), (4,6), (4,7), (6,5), (6,7), (7,4)).

15. ({1,2,3,4,5,6,7}: (1,2), (1,3), (2,3), (3,1), (3,4), (3,5), (3,6), (3,7), (5,6), (6,7), (7,1)).

16. ({1,2,3,4,5,6,7}: (1,2), (1,3), (1,7), (2,6), (3,1), (3,4), (3,5), (3,6), (4,5), (5,6), (5,7), (6,2), (6,4), (6,7), (7,3)).

17. ({1,2,3,4,5,6,7}: (1,2), (1,5), (1,6), (2,3), (2,7), (3,5), (4,7), (5,4), (5,6), (6,2), (6,7), (7,3)).

18. ({1,2,3,4,5,6,7}: (1,2), (1,5), (1,7), (2,3), (3,4), (3,6), (3,7), (4,5), (5,2), (5,3), (6,5), (6,7), (7,3)).

19. ({1,2,3,4,5,6,7}: (1,2), (1,3), (2,3), (3,1), (3,4), (3,5), (3,6), (3,7), (5,6), (6,7), (7,1)).

20. ({1,2,3,4,5,6,7}: (1,2), (1,3), (1,4), (1,5), (3,2), (3,6), (3,7), (4,3), (5,4), (5,7), (6,1), (6,5), (7,1), (7,6)).

21. ({1,2,3,4,5,6,7}: (1,4), (1,5), (2,5), (2,6), (3,2), (3,6), (4,3), (5,6), (5,7), (6,1), (6,3), (7,3), (7,6)).

22. ({1,2,3,4,5,6,7}: (1,2), (1,4), (1,7), (2,7), (3,1), (3,4), (3,7), (4,7), (5,4), (5,6), (6,3), (6,4), (6,7), (7,3), (7,4)).

23. ({1,2,3,4,5,6,7}: (1,5), (1,6), (1,7), (2,1), (2,3), (3,4), (3,6), (3,7), (4,5), (4,6), (5,7), (6,5), (7,3), (7,6)).

24. ({1,2,3,4,5,6,7}: (1,3), (1,7), (2,5), (3,7), (4,3), (5,2), (5,4), (6,1), (6,4), (6,5), (7,3), (7,6)).

25. ({1,2,3,4,5,6,7}: (1,2), (1,5), (1,6), (2,7), (3,1), (3,2), (3,4), (4,6), (5,3), (5,4), (6,5), (7,3), (7,5)).

26. ({1,2,3,4,5,6,7}: (1,2), (1,4), (1,7), (2,3), (2,6), (2,7), (3,5), (4,3), (5,4), (5,6), (6,7), (7,3), (7,5)).

27. ({1,2,3,4,5,6,7}: (1,2), (1,4), (1,6), (1,7), (2,5), (2,6), (3,2), (4,3), (4,5), (5,7), (6,1), (6,2), (6,3), (6,4), (6,5), (7,3)).

28. ({1,2,3,4,5,6,7}: (1,2), (1,4), (2,3), (2,5), (3,6), (4,3), (4,5), (4,6), (5,6), (6,2), (6,7), (7,1), (7,4)).

29. ({1,2,3,4,5,6,7}: (1,2), (1,4), (2,6), (2,7), (3,2), (4,3), (4,5), (5,6), (5,7), (6,4), (6,5), (7,2), (7,3)).

30. ({1,2,3,4,5,6,7}: (1,2), (1,3), (1,5), (2,6), (3,1), (3,4), (4,5), (4,7), (6,5), (6,7), (7,2), (7,3)).

3. На заданном графе реализовать реберную раскраску.

1. ({1,2,3,4}: (1,2), (1,3), (1,4), (2,4), (3,4)).

2. ({1,2,3,4}: (1,3), (1,4), (2,3), (2,4), (3,4)).

3. ({1,2,3,4}: (1,2), (1,4), (2,3), (2,4), (3,4)).

4. ({1,2,3,4}: (1,2), (1,3), (1,4), (2,3), (2.4)).

5. ({1,2,3,4}: (1,2), (1,3), (2,3), (2,4), (3,4)).

6. ({1,2,3,4}: (1,2), (1,3), (1,4), (2,3), (3,4)).

7. ({a,b,c,d}: (a,b), (a,d), (a,c), (b,d), (c,d)).

8. ({a,b,c,d}: (a,c), (a,d), (b,c), (b,d), (c,d)).
9. ({a,b,c,d}: (a,b), (a,c), (b,c), (b,d), (c,d)).
10. ({a,b,c,d}: (a,b), (a,c), (a,d), (b,c), (b,d)).
11. ({a,b,c,d}: (a,b), (a,d), (b,c), (b,d), (c,d)).
12. ({a,b,c,d}: (a,b), (a,c), (a,d), (b,c), (c,d)).
13. ({x,y,z,v}: (x,y), (x,z), (y,z), (y,v), (z,v)).
14. ({x,y,z,v}: (x,z), (x,v), (y,z), (y,v), (z,v)).
15. ({x,y,z,v}: (x,y), (x,z), (x,v), (y,v), (z,v)).
16. ({x,y,z,v}: (x,y), (x,z), (x,v), (y,z), (y,v)).
17. ({x,y,z,v}: (x,y), (x,v), (y,z), (y,v), (z,v)).
18. ({x,y,z,v}: (x,y), (x,z), (x,v), (y,z), (z,v)).
19. ({00,01,10,11}: (00,01), (00,10), (00,11), (01,10), (10,11)).
20. ({00,01,10,11}: (00,01), (00,10), (01,10), (01,11), (10,11)).
21. ({00,01,10,11}: (00,01), (00,10), (00,11), (01,11), (10,11)).
22. ({00,01,10,11}: (00,01), (00,10), (00,11), (01,10), (01,11)).
23. ({00,01,10,11}: (00,01), (00,11), (01,10), (01,11), (10,11)).
24. ({00,01,10,11}: (00,10), (00,11), (01,10), (01,11), (10,11)).
25. ({k,l,m,n}: (k,l), (k,m), (l,m), (l,n), (m,n)).
26. ({k,l,m,n}: (k,m), (k,n), (l,m), (l,n), (m,n)).
27. ({k,l,m,n}: (k,l), (k,m), (k,n), (l,n), (m,n)).
28. ({k,l,m,n}: (k,l), (k,m), (k,n), (l,m), (l,n)).
29. ({k,l,m,n}: (k,l), (k,n), (l,n), (l,m), (m,n)).
30. ({k,l,m,n}: (k,m), (k,n), (l,m), (l,n), (m,n)).

СПИСОК ЛИТЕРАТУРЫ

1. Хаггарти, Р. Дискретная математика для программистов / Р. Хаггарти. – М.: ТЕХНОСФЕРА, 2016. – 400 с.

2. Новиков, Ф.А. Дискретная математика: учебник для вузов / Ф.А. Новиков. – СПб.: Питер, 2017. – 496 с.

3. Лекции по теории графов: учеб. пособие / В.А. Емеличев, О.И. Мельников, В.И. Сарванов, Р.И. Тышкевич. – М.: ЛЕНАНД, 2017. – 390 с.

4. Харари, Ф. Теория графов / Ф. Харари. – М.: ЕДИТОРИАЛ УРСС, ЛЕНАНД, 2015. – 304 с.

СОДЕРЖАНИЕ