Ткачев С.Б.
каф. Математического моделирования
МГТУ им. Н.Э. Баумана

ДИСКРЕТНАЯ МАТЕМАТИКА

ИУ5 — 4 семестр, 2015 г.

Лекция 12. ДЕРЕВЬЯ. МЕТОДЫ СИСТЕМАТИЧЕСКОГО ОБХОДА ВЕРШИН ГРАФА

12.1. Представление графа матрицей смежности вершин.

Матрица смежности вершин, или **булева матрица** графа — это квадратная матрица B порядка n, элементы которой определяют следующим образом:
для неориентированного графа

$$\begin{cases} b_{ij} = 1, i\text{-я и } j\text{-я вершины } \textit{смежные}; \\ 0, \text{иначе}; \end{cases}$$

для ориентированного графа

$$\begin{cases} b_{ij} = 1, \text{из } i\text{-й вершины в } j\text{-ю ведет дуга}; \\ 0, \text{иначе}; \end{cases}$$

В k -й строке матрицы ориентированного графа количество единиц равно **полустепени исхода** $\mathrm{dg}^+ v_k$ вершины v_k , а количество единиц в k -м столбце — **полустепени захода** $\mathrm{dg}^- v_k$.

Для неориентированного графа **матрица** смежности вершин **симметрическая**.

Эта матрица есть матрица бинарного отношения непосредственной достижимости на множестве вершин V .

Пример 12.1. Для ориентированного графа $G = (V, E)$, где $V = \{v_1, v_2, v_3\}$,
$E = \{\{v_1, v_2\}, \{v_1, v_3\}, \{v_2, v_3\}, \{v_3, v_2\}\}$.

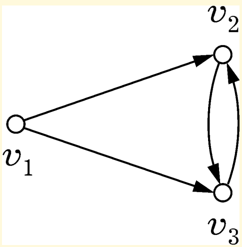

Рис. 1

Матрица смежности вершин:

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

12.2. Списки смежности

В ориентированном графе для задания множества **вершин**, **непосредственно достижимых** из вершины v, используют **линейный однонаправленный список**.

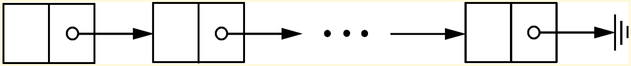

Рис. 2

Задать для вершины v ее список смежности означает в произвольном порядке в **данные** элементов списка поместить номера вершин u, для которых в ориентированном графе есть дуга из v в u ($v \to u$). **Список смежности вершины** v обозначают $L(v)$.
Если количество вершин ориентированного графа известно заранее, то ориентированный граф удобно задавать в виде структуры, называемой **массивом лидеров**.

Пример 12.2. Ориентированный граф $G = (V,\ E)$: $V = \{v_1,\ v_2,\ v_3,\ v_4,\ v_5\}$, $E = \{(v_1,\ v_2), (v_2,\ v_2), (v_2,\ v_3), (v_2,\ v_4), (v_2,\ v_5), (v_3,\ v_1), (v_5,\ v_2)\}$. Списки смежности, собранные в массив лидеров:

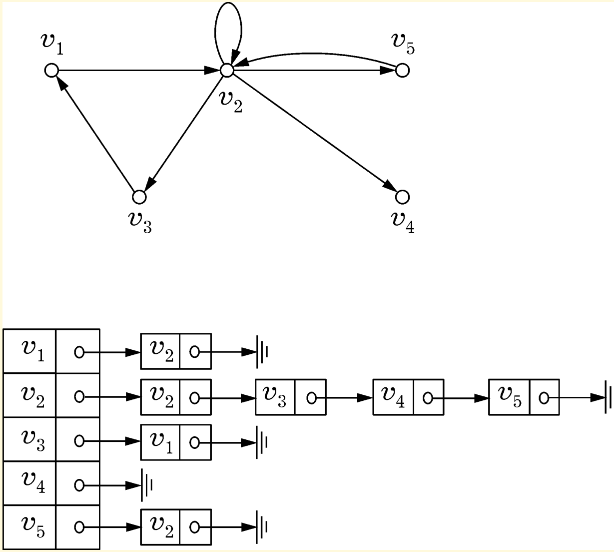

Рис. 3

Неориентированный граф задать с помощью списков смежности можно так же, как и ориентированный. Здесь в список смежности вершины v войдут все вершины, смежные с ней, а списки смежности могут быть собраны в массив лидеров.

12.3. Деревья

Определение 12.1. Неориентированным деревом называют **связный** и **ациклический неориентированный граф**.

Определение 12.2. Ориентированным деревом называют **бесконтурный ориентированный граф**, у которого **полустепень захода** любой вершины не больше 1 и существует ровно одна вершина, называемая **корнем ориентированного дерева**, **полустепень захода** которой равна 0.

В ориентированном дереве любая вершина **достижима** из корня.
Требование бесконтурности ориентированного графа в определении 12.2 является обязательным.

Определение 12.3. Вершину v ориентированного дерева называют **потомком** (**подлинным потомком**) вершины u , если существует путь из u в v (путь ненулевой длины из u в v). В этом же случае вершину u называют **предком** (**подлинным предком**) вершины v, а если длина пути из u в v равна 1, то вершину v называют **сыном** вершины u, которая при этом вполне естественно именуется **отцом** вершины v .
Вершину, не имеющую потомков, называют **листом**.

Пример 12.3.

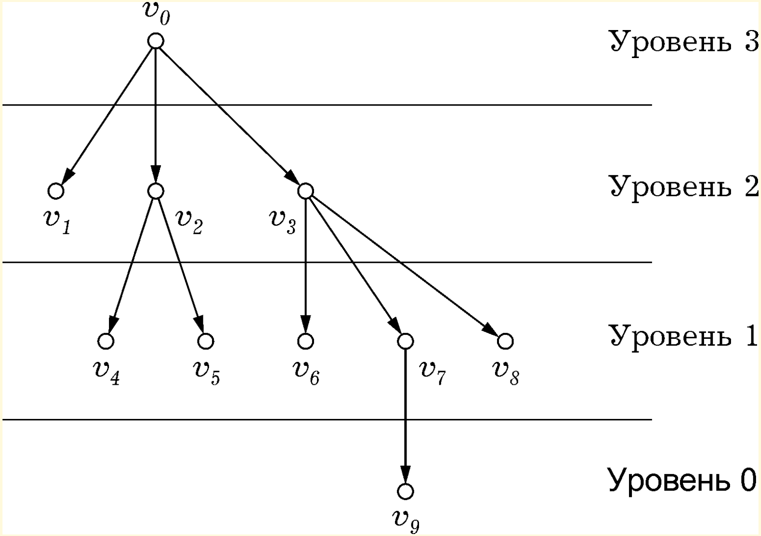

Рис. 4

Вершины v_4 и v_5 — сыновья вершины v_2 , которая, в свою очередь, является сыном вершины v_0 — корня дерева.
Вершины v_4 и v_5 являются подлинными потомками вершин v_0
и v_2 , которые соответственно будут их подлинными предками. Вершины v_1 , v_4 , v_5 , v_6 , v_9 , v_8 — листья дерева.
Взаимно недостижимые вершины ориентированного дерева (v_2 и v_9) не являются ни предком, ни потомком одна другой. Каждая вершина будет сама для себя предком и потомком, но не подлинным.

Определение 12.4. Ориентированное дерево, у которого каждая вершина, отличная от корня, есть лист, называют **кустом**.

Определение 12.5. Подграф неориентированного (ориентированного) дерева, являющийся неориентированным (ориентированным) деревом, называют **поддеревом** исходного дерева.

Компонентами ориентированного дерева являются его **подграфы**, порожденные множеством вершин, расположенных на некотором пути из корня в лист.

Подграф, порожденный множеством вершин $\{v_3, v_6, v_7, v_8, v_9\}$, является **поддеревом** ориентированного дерева.

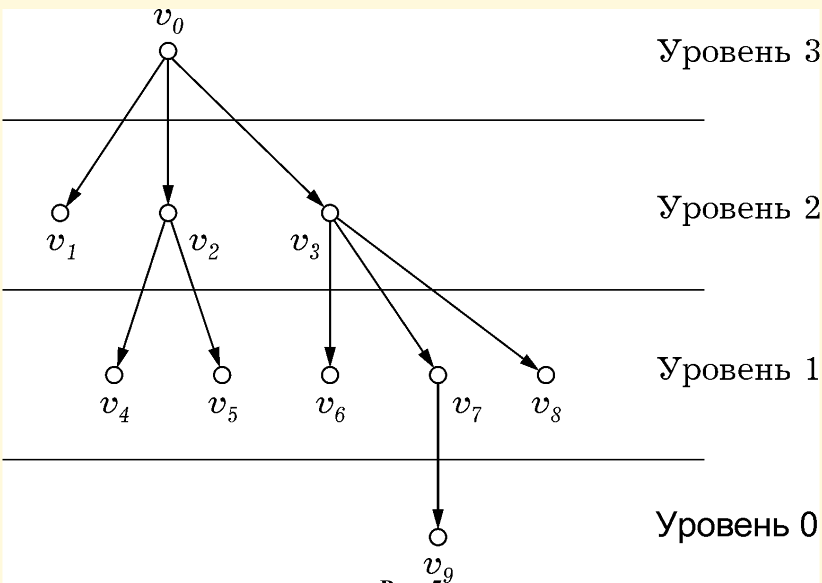

Рис. 5

Определение 12.6. Произвольный ациклический граф называют **неориентированным лесом**.
Если каждая **слабая компонента** ориентированного графа является ориентированным деревом, то такой граф называют **ориентированным лесом**.

Неориентированный лес — это неориентированный граф, каждая компонента которого является неориентированным деревом.

Примеры неориентированного и ориентированного леса:

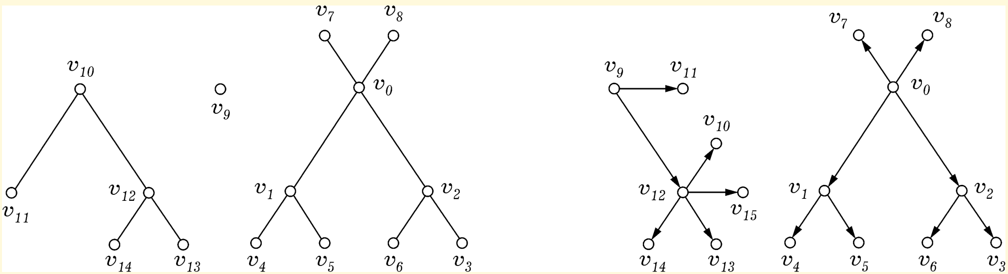

Рис. 6

Остовным лесом (**деревом**) неориентированного (ориентированного) графа называют любой его остовный подграф, являющийся лесом (деревом).

Определение 12.7. Высота ориентированного дерева — это наибольшая длина пути из корня в лист.
Глубина $\mathrm{d}(v)$ **вершины ориентированного дерева** v — это длина пути из корня в эту вершину.
Высота $\mathrm{h}(v)$ **вершины ориентированного дерева** v — это наибольшая длина пути из данной вершины в лист.
Уровень вершины ориентированного дерева — это разность между высотой ориентированного дерева и глубиной данной вершины.

Уровень корня равен высоте ориентированного дерева, но уровни различных листьев,так же как и их глубины, могут быть различными; **высота** любого **листа** равна **нулю**.

Определение 12.8. Ориентированное дерево называют **бинарным**, если полустепень исхода любой его вершины не больше 2.

Бинарное ориентированное дерево называют **полным**, если из любой его вершины, не являющейся листом, исходят ровно две дуги, а уровни всех листьев совпадают.

Теорема 1 (теорема о высоте бинарного ориентированного дерева с заданным числом листьев). Бинарное ориентированное дерево с n листьями имеет высоту, не меньшую $\log_2 n$.

◀ Покажем, что в **полном** бинарном ориентированном дереве высоты h ровно 2^h листьев. Используем метод математической индукции.

Ориентированное дерево высоты 0 имеет $2^0 = 1$ лист.

Полное бинарное ориентированное дерево высоты 1 имеет $2^1 = 2$ листа.

Пусть полное бинарное ориентированное дерево имеет высоту k и соответственно 2^k листьев.

Рассмотрим полное бинарное ориентированное дерево высоты $k+1$. Поскольку в полном бинарном ориентированном дереве уровни всех листьев совпадают, ориентированное дерево высоты $k+1$ можно получить из полного бинарного ориентированного дерева высоты k, если из каждого листа последнего провести по две дуги.

Тогда количество листьев в ориентированном дереве высоты $k+1$ будет в 2 раза больше, чем в ориентированном дереве высоты k, т.е. $2^k \cdot 2 = 2^{k+1}$.
В произвольном бинарном дереве листьев может быть только меньше, чем в полном.
Следовательно, в произвольном бинарном дереве высоты h не более 2^h листьев $(n \le 2^h)$.
Таким образом $h \ge \log_2 n$. ▶

12.4. Задача сортировки:

необходимо расположить строго по возрастанию элементы конечного линейно упорядоченного множества $\{a_1, \ldots, a_n\}$.

Эту задачу называют **задачей сортировки**, а любой алгоритм, ее решающий, — **алгоритмом сортировки**.

Все сравнения, которые могут быть проведены в процессе работы некоторого алгоритма, изображаются наглядно в виде ориентированного дерева, называемого **деревом решений**.

Пример 12.4. Дерево решений для трехэлементного множества $\{a, b, c\}$.

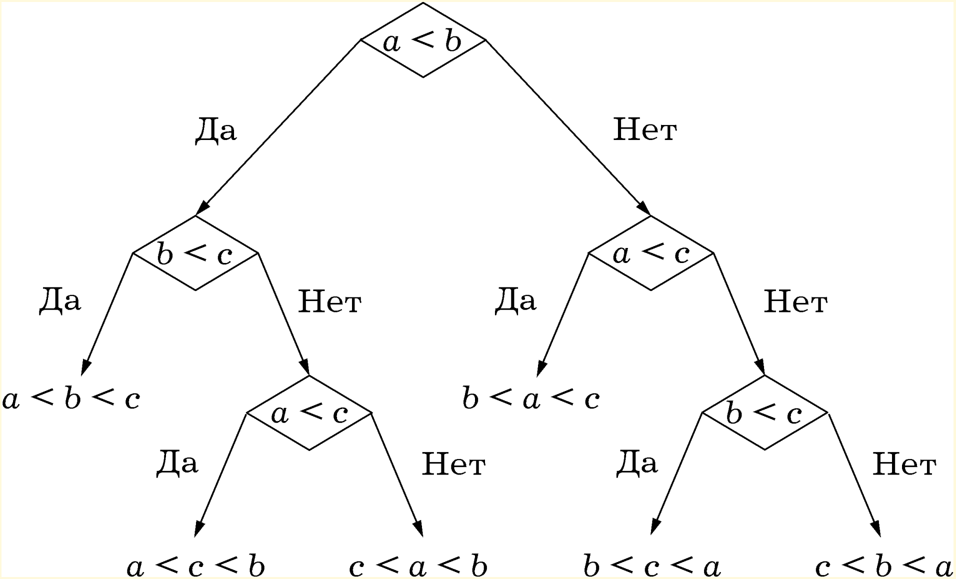

Рис. 7

С математической точки зрения алгоритм сортировки должен найти такую **перестановку** $\{a_{p_1}, \ldots, a_{p_n}\}$ элементов множества, которая была бы согласована с заданным на нем отношением $\leq$ линейного порядка, т.е. для любых k, l из справедливости неравенства $p_k < p_l$ должно следовать $a_{p_k} \leq a_{p_l}$.

Первоначально сортируемые элементы могут быть расположены в произвольном порядке, т.е. исходной может быть любая перестановка элементов сортируемого множества, и мы не имеем никакой априорной информации об этой перестановке.

Поскольку в результате сортировки может получиться любая перестановка исходного множества и каждой такой перестановке соответствует лист дерева решений, в общем случае количество листьев будет равно $n!$ — количеству перестановок n -элементного множества.

Следовательно, сортируя входную последовательность, алгоритм обязательно пройдет какой-то путь от корня дерева решений к одному из листьев, и, таким образом, число операций сравнения (число шагов алгоритма сортировки) будет величиной, пропорциональной высоте дерева решений, не меньшей чем $\log_2 n!$, в силу теоремы 1.

Используя для оценки факториала при больших n **формулу Стирлинга**

$$n! \approx \sqrt{2\pi n}\left(\frac{n}{e}\right)^n,$$

получаем, что дерево решений имеет высоту порядка $n\log_2 n$.

Таким образом, в общем случае задачу сортировки с помощью попарных сравнений нельзя решить быстрее, чем за указанное число шагов.

12.5. Методы систематического обхода вершин графа

Необходимо уметь обходить все вершины графа таким образом, чтобы каждая вершина была отмечена ровно один раз.
Обычно такое „путешествие“ по графу сопровождается нумерацией вершин графа в том порядке, в котором они отмечаются, а также определенной „маркировкой“ *ребер* (или *дуг*) графа.
Существуют две основные стратегии таких обходов: **поиск в глубину** и **поиск в ширину**.

Алгоритм поиска в глубину в неор. графе

Граф задан списками смежности, собранными в массив лидеров.

При поиске вершины графа нумеруются в порядке их посещения. Номер вершины v графа, присваиваемый ей при поиске в глубину, обозначим $D[v]$ и будем называть **D -номером**.

В процессе обхода будем находить **фундаментальные циклы** графа.

Пусть в неориентированном графе $G = (V, E)$ произвольно фиксирован **максимальный остовный лес**. Для связного графа это будет максимальное остовное дерево. Множество его ребер обозначим T. Все ребра из T назовем **древесными**, а ребра исходного графа G, не принадлежащие T, — **обратными**.

Любой цикл графа G, содержащий только одно обратное ребро, назовем фундаментальным.

Максимальный остовный лес, находимый с помощью алгоритма поиска в глубину, называют **глубинным остовным лесом**.

Классификация ребер зависит от хода работы алгоритма, который определяется стартовой вершиной и расположением вершин в списках смежности.

Для организации работы алгоритма поиска в глубину используется способ хранения данных, называемый **стеком**. Элементы в стеке упорядочиваются в порядке поступления. В стек можно добавлять новые элементы и из него можно извлекать элементы. При этом доступен только последний добавленный элемент — **вершина стека**.

В алгоритме поиска в глубину используется стек вершин.

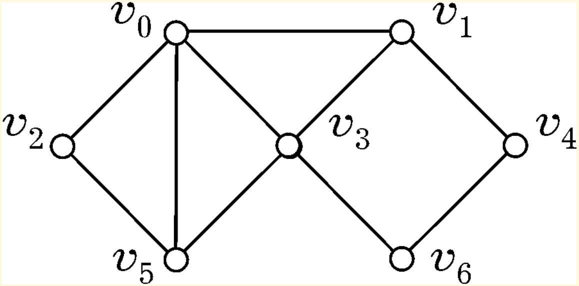

Рис. 8

Списки смежности

$V_0 \longrightarrow (V_1, V_2, V_3, V_5)$
$V_1 \longrightarrow (V_0, V_3, V_4)$
$V_2 \longrightarrow (V_0, V_5)$
$V_3 \longrightarrow (V_0, V_1, V_5, V_6)$
$V_4 \longrightarrow (V_1, V_6)$
$V_5 \longrightarrow (V_0, V_2, V_3)$
$V_6 \longrightarrow (V_3, V_4)$

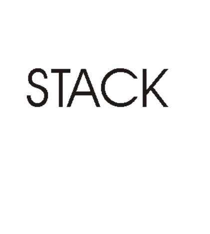

Рис. 9

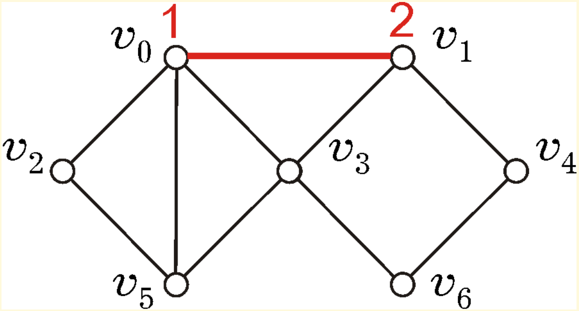

Рис. 10

Списки смежности

$$V_0 \longrightarrow (\cancel{V_1}, V_2, V_3, V_5)$$
$$V_1 \longrightarrow (V_0, V_3, V_4)$$
$$V_2 \longrightarrow (V_0, V_5)$$
$$V_3 \longrightarrow (V_0, V_1, V_5, V_6)$$
$$V_4 \longrightarrow (V_1, V_6)$$
$$V_5 \longrightarrow (V_0, V_2, V_3)$$
$$V_6 \longrightarrow (V_3, V_4)$$

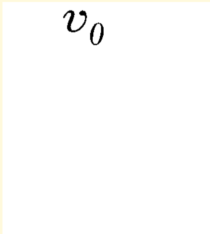

Рис. 11

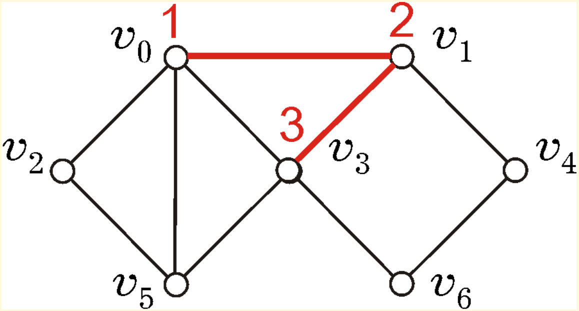

Рис. 12

Списки смежности

$$V_0 \longrightarrow (\cancel{V_1}, V_2, V_3, V_5)$$
$$V_1 \longrightarrow (\cancel{V_0}, \cancel{V_3}, V_4)$$
$$V_2 \longrightarrow (V_0, V_5)$$
$$V_3 \longrightarrow (V_0, V_1, V_5, V_6)$$
$$V_4 \longrightarrow (V_1, V_6)$$
$$V_5 \longrightarrow (V_0, V_2, V_3)$$
$$V_6 \longrightarrow (V_3, V_4)$$

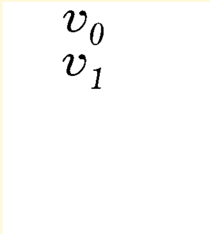

Рис. 13

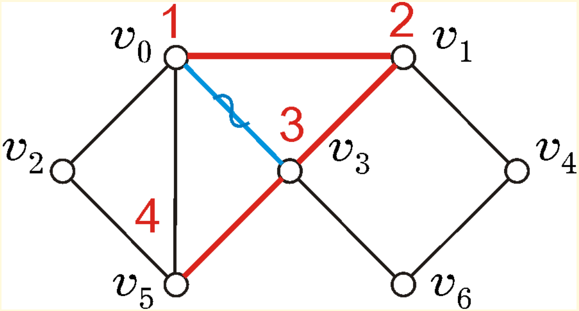

Рис. 14

Списки смежности

$$V_0 \longrightarrow (\cancel{V}_1, V_2, \cancel{V}_3, V_5)$$
$$V_1 \longrightarrow (\cancel{V}_0, \cancel{V}_3, V_4)$$
$$V_2 \longrightarrow (V_0, V_5)$$
$$V_3 \longrightarrow (\cancel{V}_0, \cancel{V}_1, \cancel{V}_5, V_6)$$
$$V_4 \longrightarrow (V_1, V_6)$$
$$V_5 \longrightarrow (V_0, V_2, V_3)$$
$$V_6 \longrightarrow (V_3, V_4)$$

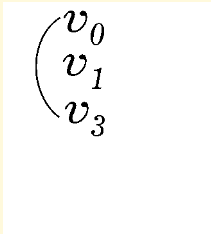

Рис. 15

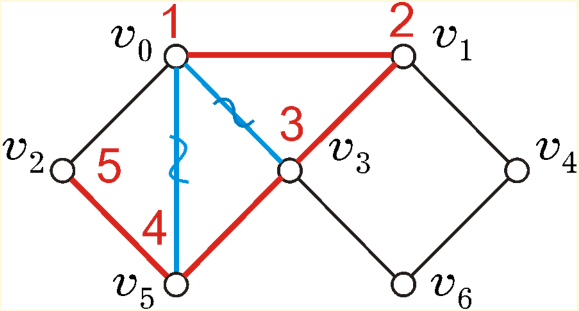

Рис. 16

Списки смежности

$$V_0 \longrightarrow (\cancel{V}_1, \cancel{V}_2, \cancel{V}_3, \cancel{V}_5)$$
$$V_1 \longrightarrow (\cancel{V}_0, \cancel{V}_3, V_4)$$
$$V_2 \longrightarrow (V_0, V_5)$$
$$V_3 \longrightarrow (\cancel{V}_0, \cancel{V}_1, \cancel{V}_5, V_6)$$
$$V_4 \longrightarrow (V_1, V_6)$$
$$V_5 \longrightarrow (\cancel{V}_0, \cancel{V}_2, \cancel{V}_3)$$
$$V_6 \longrightarrow (V_3, V_4)$$

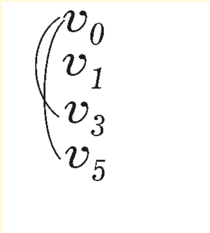

Рис. 17

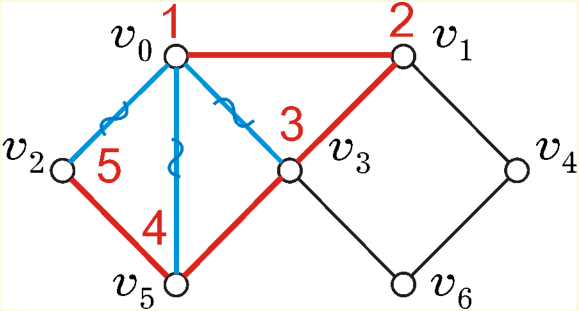

Рис. 18

Списки смежности

$$V_0 \longrightarrow (\cancel{V}_1, \cancel{V}_2, \cancel{V}_3, \cancel{V}_5)$$
$$V_1 \longrightarrow (\cancel{V}_0, \cancel{V}_3, V_4)$$
$$V_2 \longrightarrow (\cancel{V}_0, \cancel{V}_5)$$
$$V_3 \longrightarrow (\cancel{V}_0, \cancel{V}_1, \cancel{V}_5, V_6)$$
$$V_4 \longrightarrow (V_1, V_6)$$
$$V_5 \longrightarrow (\cancel{V}_0, \cancel{V}_2, \cancel{V}_3)$$
$$V_6 \longrightarrow (V_3, V_4)$$

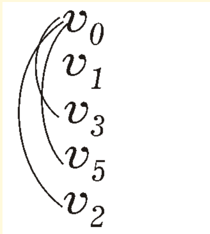

Рис. 19

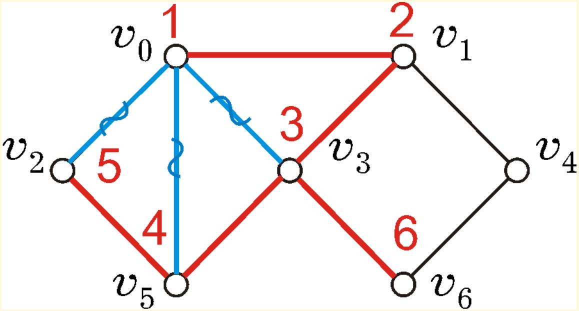

Рис. 20

Списки смежности

$$V_0 \longrightarrow (\cancel{V}_1,\ \cancel{V}_2,\ \cancel{V}_3,\ \cancel{V}_5)$$
$$V_1 \longrightarrow (\cancel{V}_0,\ \cancel{V}_3,\ V_4)$$
$$V_2 \longrightarrow (\cancel{V}_0,\ \cancel{V}_5)$$
$$V_3 \longrightarrow (\cancel{V}_0,\ \cancel{V}_1,\ \cancel{V}_5,\ \cancel{V}_6)$$
$$V_4 \longrightarrow (V_1, V_6)$$
$$V_5 \longrightarrow (\cancel{V}_0,\ \cancel{V}_2,\ \cancel{V}_3)$$
$$V_6 \longrightarrow (V_3, V_4)$$

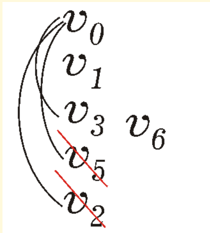

Рис. 21

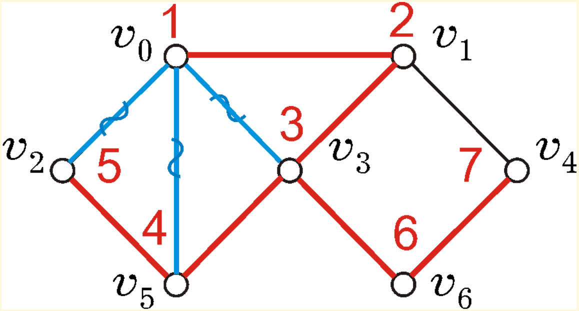

Рис. 22

Списки смежности

$$V_0 \longrightarrow (\cancel{V}_1, \cancel{V}_2, \cancel{V}_3, \cancel{V}_5)$$
$$V_1 \longrightarrow (\cancel{V}_0, \cancel{V}_3, V_4)$$
$$V_2 \longrightarrow (\cancel{V}_0, \cancel{V}_5)$$
$$V_3 \longrightarrow (\cancel{V}_0, \cancel{V}_1, \cancel{V}_5, \cancel{V}_6)$$
$$V_4 \longrightarrow (V_1, V_6)$$
$$V_5 \longrightarrow (\cancel{V}_0, \cancel{V}_2, \cancel{V}_3)$$
$$V_6 \longrightarrow (\cancel{V}_3, \cancel{V}_4)$$

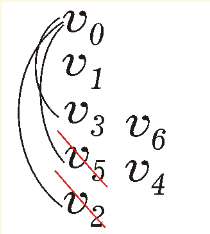

Рис. 23

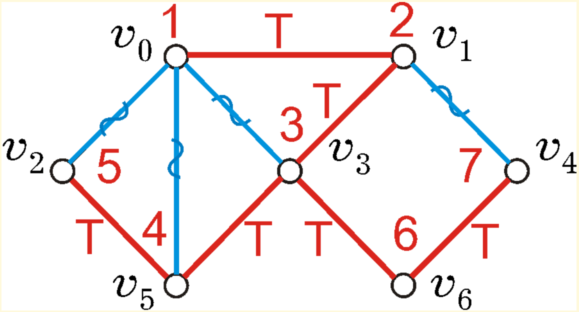

Рис. 24

Списки смежности

$V_0 \longrightarrow (\cancel{V}_1, \cancel{V}_2, \cancel{V}_3, \cancel{V}_5)$

$V_1 \longrightarrow (\cancel{V}_0, \cancel{V}_3, \cancel{V}_4)$

$V_2 \longrightarrow (\cancel{V}_0, \cancel{V}_5)$

$V_3 \longrightarrow (\cancel{V}_0, \cancel{V}_1, \cancel{V}_5, \cancel{V}_6)$

$V_4 \longrightarrow (\cancel{V}_1, \cancel{V}_6)$

$V_5 \longrightarrow (\cancel{V}_0, \cancel{V}_2, \cancel{V}_3)$

$V_6 \longrightarrow (\cancel{V}_3, \cancel{V}_4)$

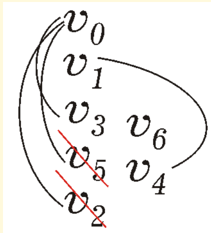

Рис. 25

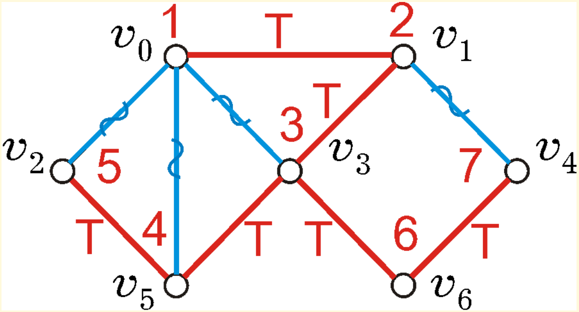

Рис. 26

Списки смежности

$$V_0 \longrightarrow (\cancel{V}_1,\ \cancel{V}_2,\ \cancel{V}_3,\ \cancel{V}_5)$$
$$V_1 \longrightarrow (\cancel{V}_0,\ \cancel{V}_3,\ \cancel{V}_4)$$
$$V_2 \longrightarrow (\cancel{V}_0,\ \cancel{V}_5)$$
$$V_3 \longrightarrow (\cancel{V}_0,\ \cancel{V}_1,\ \cancel{V}_5,\ \cancel{V}_6)$$
$$V_4 \longrightarrow (\cancel{V}_1,\ \cancel{V}_6)$$
$$V_5 \longrightarrow (\cancel{V}_0,\ \cancel{V}_2,\ \cancel{V}_3)$$
$$V_6 \longrightarrow (\cancel{V}_3,\ \cancel{V}_4)$$

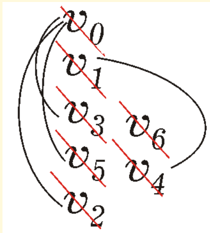

Рис. 27

Алгоритм поиска в глубину в ор. графе

В ориентированном графе вершинам также присваиваются D -номера. Классификация дуг:

1) **древесные дуги** — каждая такая *дуга ведет* от *отца* к *сыну* в глубинном остовном лесу;

2) **прямые дуги** — каждая такая дуга ведет от *подлинного предка* к *подлинному потомку* (но не от отца к сыну) в глубинном остовном лесу;

3) **обратные дуги** — от *потомков* к *предкам* (включая все петли);

4) **поперечные дуги** — все дуги, не являющиеся ни древесными, ни прямыми, ни обратными.

Результат работы алгоритма: множества $Tree$ — древесных дуг, $Back$ — обратных дуг, $Forward$ — прямых дуг, C — поперечных дуг и массив D, содержащий D -номера вершин.

Особенности алгоритма.

Так, если очередная вершина w, извлеченная из списка смежности текущей вершины v, новая, то дуга $(v,\, w)$ является древесной.

Если вершина w не новая $(w \notin V_0)$, то дуга $(v,\, w)$ будет либо прямой, либо обратной, либо поперечной.

Если D-номер вершины v строго меньше D-номера вершины w ($D[v] < D[w]$), то дуга $(v,\, w)$ является прямой.

Если D-номер вершины v не меньше D-номера вершины w ($D[v] \geq D[w]$), необходимо проверить, есть ли в стеке $STACK$ вершина w. Если вершина w находится в стеке, то дуга $(v,\, w)$ является обратной.Если вершины w в стеке нет, то дуга является поперечной.

Если стек пуст, но не все вершины ориентированного графа обработаны, поиск продолжают из любой необработанной вершины.

В случае ориентированного графа поиск контуров на базе поиска в глубину существенно сложнее.

Ориентированный граф является бесконтурным тогда и только тогда, когда при поиске в глубину от некоторой начальной вершины множество обратных дуг оказывается пустым.

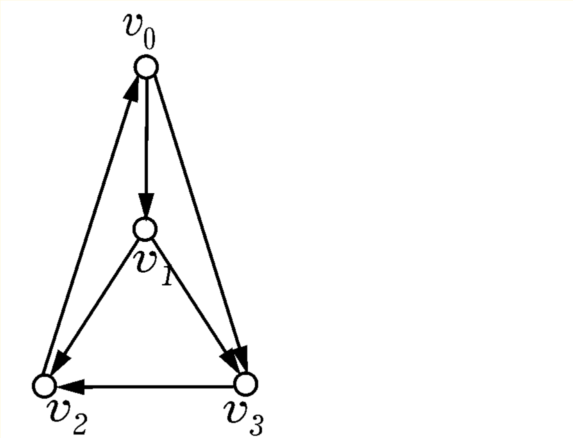

Рис. 28

Списки смежности

$$V_0 \longrightarrow (V_1, V_3)$$
$$V_1 \longrightarrow (V_2, V_3)$$
$$V_2 \longrightarrow (V_0)$$
$$V_3 \longrightarrow (V_2)$$

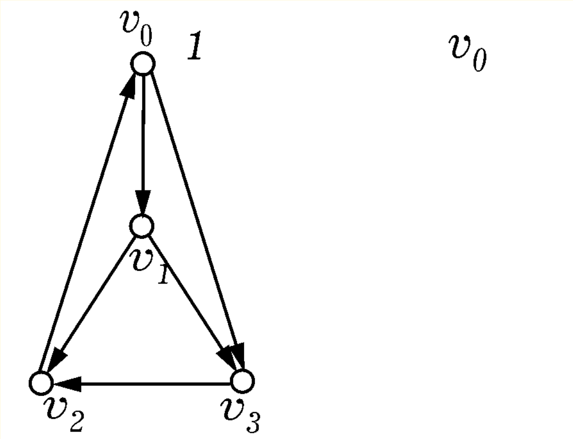

Рис. 29

Списки смежности

$V_0 \longrightarrow (V_1, V_3)$
$V_1 \longrightarrow (V_2, V_3)$
$V_2 \longrightarrow (V_0)$
$V_3 \longrightarrow (V_2)$

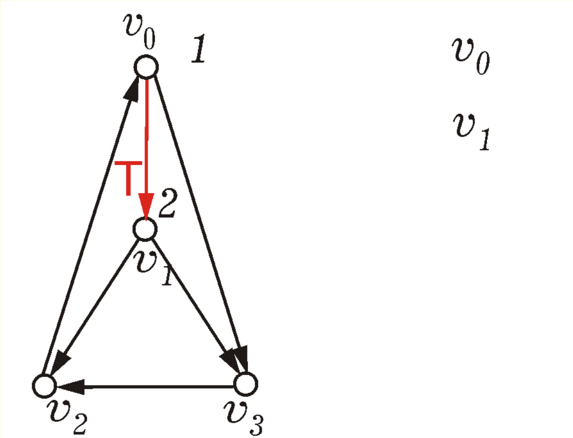

Рис. 30

Списки смежности

$V_0 \longrightarrow (\cancel{V}_1, V_3)$
$V_1 \longrightarrow (V_2, V_3)$
$V_2 \longrightarrow (V_0)$
$V_3 \longrightarrow (V_2)$

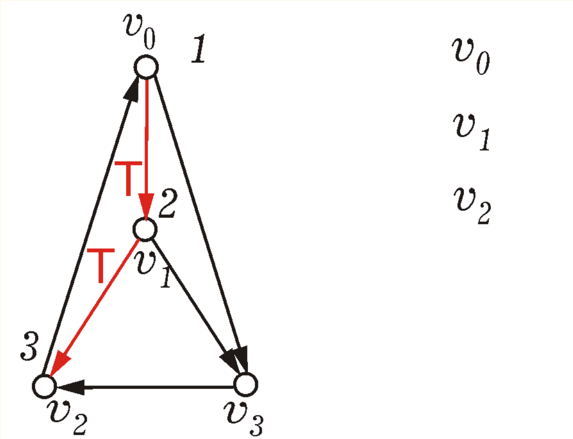

Рис. 31

Списки смежности

$V_0 \longrightarrow (\cancel{V_1}, V_3)$
$V_1 \longrightarrow (\cancel{V_2}, V_3)$
$V_2 \longrightarrow (V_0)$
$V_3 \longrightarrow (V_2)$

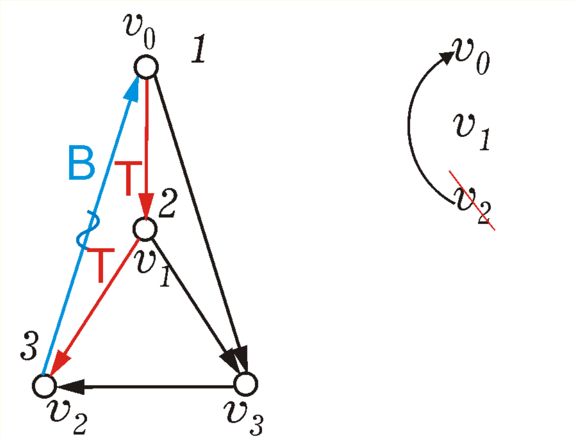

Рис. 32

Списки смежности

$V_0 \longrightarrow (\cancel{V}_1, V_3)$
$V_1 \longrightarrow (\cancel{V}_2, V_3)$
$V_2 \longrightarrow (\cancel{V}_0)$
$V_3 \longrightarrow (V_2)$

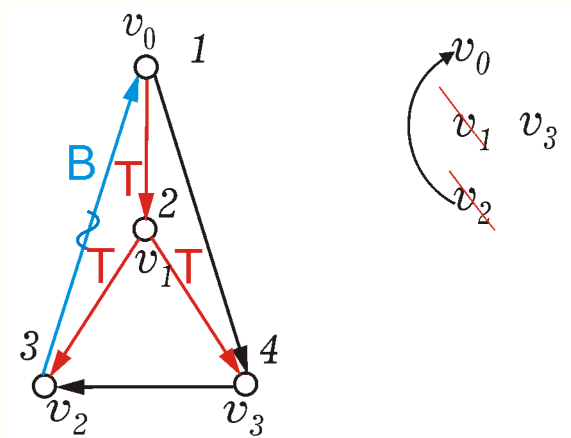

Рис. 33

Списки смежности

$V_0 \longrightarrow (\cancel{V_1}, V_3)$
$V_1 \longrightarrow (\cancel{V_2}, \cancel{V_3})$
$V_2 \longrightarrow (\cancel{V_0})$
$V_3 \longrightarrow (V_2)$

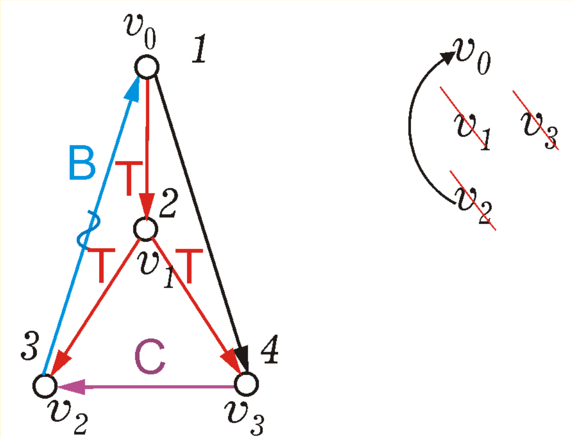

Рис. 34

Списки смежности

$$V_0 \longrightarrow (\cancel{V_1}, V_3)$$
$$V_1 \longrightarrow (\cancel{V_2}, \cancel{V_3})$$
$$V_2 \longrightarrow (\cancel{V_0})$$
$$V_3 \longrightarrow (\cancel{V_2})$$

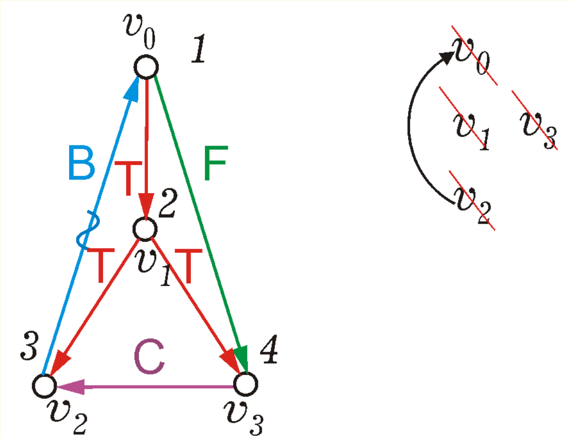

Рис. 35

Списки смежности

$$V_0 \longrightarrow (\cancel{V}_1,\ \cancel{V}_3)$$
$$V_1 \longrightarrow (\cancel{V}_2,\ \cancel{V}_3)$$
$$V_2 \longrightarrow (\cancel{V}_0)$$
$$V_3 \longrightarrow (\cancel{V}_2)$$

12.6. Алгоритм поиска в ширину в ор. графе

Вход. Граф $G = (V, E)$, заданный списками смежности; v_0 — начальная вершина (не обязательно первый элемент массива лидеров).

Выход. Массив M меток вершин, где каждая метка равна длине пути от v_0 до v.

0. Очередь Q положить пустой $(Q := \varnothing)$. Все вершины пометить как недостижимые из вершины v_0, присваивая элементам массива M значение $+\infty$ $(M[v_i] := +\infty$, $i = \overline{1,\ N}$).

Стартовую вершину v_0 пометить номером 0, т.е. длину пути от стартовой вершины v_0 до самой себя положить равной 0 ($M[v_0] := 0$). Поместить вершину v_0 в очередь Q. Перейти на шаг **1**.

1. Если очередь Q не пуста $(Q \neq \varnothing)$, то из „головы“ очереди извлечь (с удалением из очереди) вершину u и перейти на шаг **2**. Если очередь пуста, перейти на шаг **3**.

2. Если список смежности $L(u)$ вершины u пуст, вернуться на шаг **1**.

Если список смежности $L(u)$ вершины u не пуст, для каждой вершины w из списка смежности, где $M[w] = +\infty$, т.е. вершины, которую еще не посещали, положить длину пути из стартовой вершины v_0 до вершины w равной длине пути от v_0 до вершины u плюс одна дуга $(M[w] := M[u] + 1)$, т.е. отметить вершину w и поместить ее в очередь Q . После просмотра всех вершин списка смежности $L(u)$ вернуться на шаг **1**.

3. Распечатать массив M . Закончить работу.

Алгоритм поиска в ширину может быть дополнен процедурой „обратного хода“, определяющей номера вершин, лежащих на кратчайшем пути из вершины v_0 в данную вершину u.

Для этого необходимо завести массив PR размера $|V|$, каждый элемент $PR[w]$ которого содержит номер той вершины, из которой был осуществлен переход в вершину w при ее пометке.

Если вершина w находится в списке смежности $L(u)$ вершины u, заполнение элемента массива $PR[w]$ происходит при изменении метки вершины w $M[w]$ с $+\infty$ на единицу. При этом в элементе $PR[w]$ сохраняется номер вершины u $(PR[w] := u)$. Для начальной вершины $PR[v_0]$ можно положить равным 0, в предположении, что начальная вершина v_0 имеет номер 0 и остальные вершины пронумерованы от 1 до N.

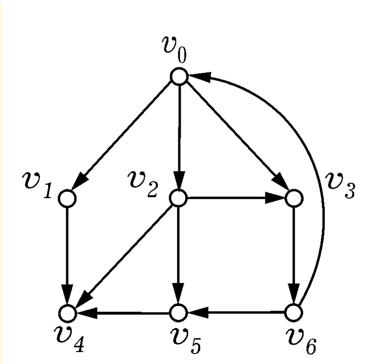

Рис. 36

Списки смежности

$v_0 \longrightarrow (v_1, v_2, v_3)$

$v_1 \longrightarrow (v_4)$

$v_2 \longrightarrow (v_4, v_5, v_3)$

$v_3 \longrightarrow (v_6)$

$v_4 \longrightarrow ()$

$v_5 \longrightarrow (v_4)$

$v_6 \longrightarrow (v_5, v_0)$

	v_0	v_1	v_2	v_3	v_4	v_5	v_6
1.	0	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$

$Q = (v_0)$

$PR(v_0) = \varnothing$

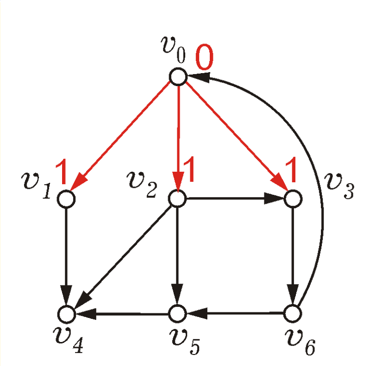

Рис. 37

Списки смежности

$v_0 \longrightarrow (v_1, v_2, v_3)$

$v_1 \longrightarrow (v_4)$

$v_2 \longrightarrow (v_4, v_5, v_3)$

$v_3 \longrightarrow (v_6)$

$v_4 \longrightarrow ()$

$v_5 \longrightarrow (v_4)$

$v_6 \longrightarrow (v_5, v_0)$

	v_0	v_1	v_2	v_3	v_4	v_5	v_6
1.	0	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$
2.	0	1	1	1	$+\infty$	$+\infty$	$+\infty$

$Q = (\cancel{v_0}, v_1, v_2, v_3)$
$PR(v_0) = \varnothing, PR(v_1) = v_0, PR(v_2) = v_0,$
$PR(v_3) = v_0$

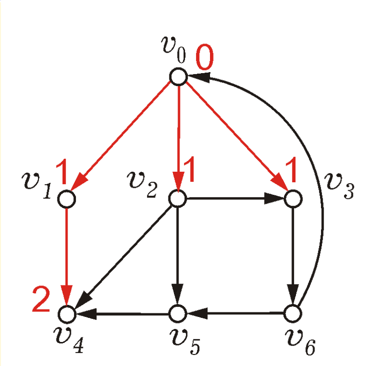

Рис. 38

Списки смежности

$v_0 \longrightarrow (v_1, v_2, v_3)$

$v_1 \longrightarrow (v_4)$

$v_2 \longrightarrow (v_4, v_5, v_3)$

$v_3 \longrightarrow (v_6)$

$v_4 \longrightarrow ()$

$v_5 \longrightarrow (v_4)$

$v_6 \longrightarrow (v_5, v_0)$

	v_0	v_1	v_2	v_3	v_4	v_5	v_6
1.	0	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$
2.	0	1	1	1	$+\infty$	$+\infty$	$+\infty$
3.	0	1	1	1	2	$+\infty$	$+\infty$

$Q = (\cancel{v_0}\ \cancel{v_1}, v_2, v_3, v_4)$
$PR(v_0) = \varnothing, PR(v_1) = v_0, PR(v_2) = v_0,$
$PR(v_3) = v_0, PR(v_4) = v_1$

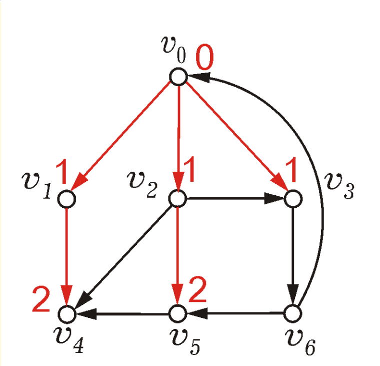

Рис. 39

Списки смежности

$v_0 \longrightarrow (v_1, v_2, v_3)$

$v_1 \longrightarrow (v_4)$

$v_2 \longrightarrow (v_4, v_5, v_3)$

$v_3 \longrightarrow (v_6)$

$v_4 \longrightarrow ()$

$v_5 \longrightarrow (v_4)$

$v_6 \longrightarrow (v_5, v_0)$

	v_0	v_1	v_2	v_3	v_4	v_5	v_6
1.	0	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$
2.	0	1	1	1	$+\infty$	$+\infty$	$+\infty$
3.	0	1	1	1	2	$+\infty$	$+\infty$
4.	0	1	1	1	2	2	$+\infty$

$Q = (\cancel{v_0}\ \cancel{v_1}\ \cancel{v_2}, v_3, v_4, v_5)$

$PR(v_0) = \varnothing, PR(v_1) = v_0, PR(v_2) = v_0,$

$PR(v_3) = v_0, PR(v_4) = v_1, PR(v_5) = v_2$

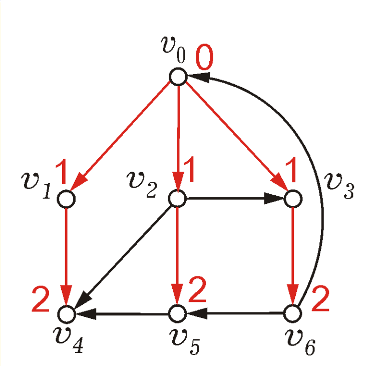

Рис. 40

Списки смежности

$v_0 \longrightarrow (v_1, v_2, v_3)$

$v_1 \longrightarrow (v_4)$

$v_2 \longrightarrow (v_4, v_5, v_3)$

$\color{red}{v_3} \longrightarrow (v_6)$

$v_4 \longrightarrow ()$

$v_5 \longrightarrow (v_4)$

$v_6 \longrightarrow (v_5, v_0)$

	v_0	v_1	v_2	v_3	v_4	v_5	v_6
1.	0	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$
2.	0	1	1	1	$+\infty$	$+\infty$	$+\infty$
3.	0	1	1	1	2	$+\infty$	$+\infty$
4.	0	1	1	1	2	2	$+\infty$
5.	0	1	1	1	2	2	2

$Q = (\cancel{v_0}\ \cancel{v_1}\ \cancel{v_2}\ \cancel{v_3}, v_4, v_5, v_6)$

$PR(v_0) = \varnothing, PR(v_1) = v_0, PR(v_2) = v_0,$
$PR(v_3) = v_0, PR(v_4) = v_1, PR(v_5) = v_2,$
$PR(v_6) = v_3$

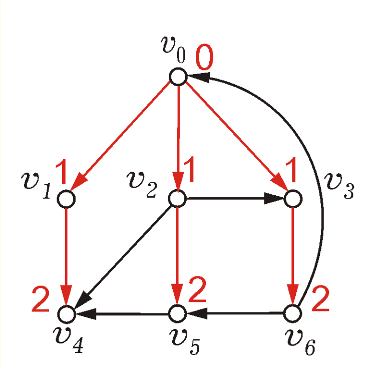

Рис. 41

Списки смежности

$v_0 \longrightarrow (v_1, v_2, v_3)$

$v_1 \longrightarrow (v_4)$

$v_2 \longrightarrow (v_4, v_5, v_3)$

$v_3 \longrightarrow (v_6)$

$v_4 \longrightarrow ()$

$v_5 \longrightarrow (v_4)$

$v_6 \longrightarrow (v_5, v_0)$

	v_0	v_1	v_2	v_3	v_4	v_5	v_6
1.	0	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$
2.	0	1	1	1	$+\infty$	$+\infty$	$+\infty$
3.	0	1	1	1	2	$+\infty$	$+\infty$
4.	0	1	1	1	2	2	$+\infty$
5.	0	1	1	1	2	2	2

$Q = (\cancel{v_0}\ \cancel{v_1}\ \cancel{v_2}\ \cancel{v_3}\ \cancel{v_4}\ \cancel{v_5}\ \cancel{v_6})$

$PR(v_0) = \varnothing, PR(v_1) = v_0, PR(v_2) = v_0,$
$PR(v_3) = v_0, PR(v_4) = v_1, PR(v_5) = v_2,$
$PR(v_6) = v_3$

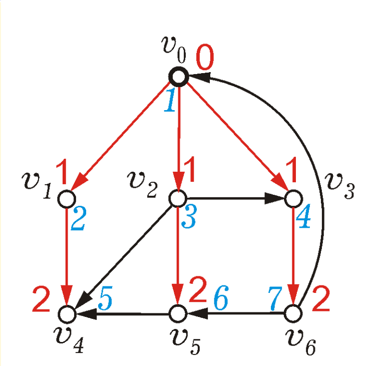

Рис. 42

Списки смежности

$v_0 \longrightarrow (v_1, v_2, v_3)$

$v_1 \longrightarrow (v_4)$

$v_2 \longrightarrow (v_4, v_5, v_3)$

$v_3 \longrightarrow (v_6)$

$v_4 \longrightarrow ()$

$v_5 \longrightarrow (v_4)$

$v_6 \longrightarrow (v_5, v_0)$

	v_0	v_1	v_2	v_3	v_4	v_5	v_6
1.	0	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$
2.	0	1	1	1	$+\infty$	$+\infty$	$+\infty$
3.	0	1	1	1	2	$+\infty$	$+\infty$
4.	0	1	1	1	2	2	$+\infty$
5.	0	1	1	1	2	2	2

$Q = \varnothing$

$PR(v_0) = \varnothing, PR(v_1) = v_0, PR(v_2) = v_0,$
$PR(v_3) = v_0, PR(v_4) = v_1, PR(v_5) = v_2,$
$PR(v_6) = v_3$

Материал для самостоятельного изучения

Матрица достижимости.

Это квадратная матрица C порядка $|V|$, каждый элемент c_{ij} которой равен 1, если j -я *вершина достижима* из i -й вершины, и равен 0 если иначе.

Согласно определению достижимости, элементы $c_{ii} = 1$. Матрица достижимости несет важную информацию об ориентированном графе. Ее анализ позволяет, например, найти его **бикомпоненты**.

Пример 12.5.

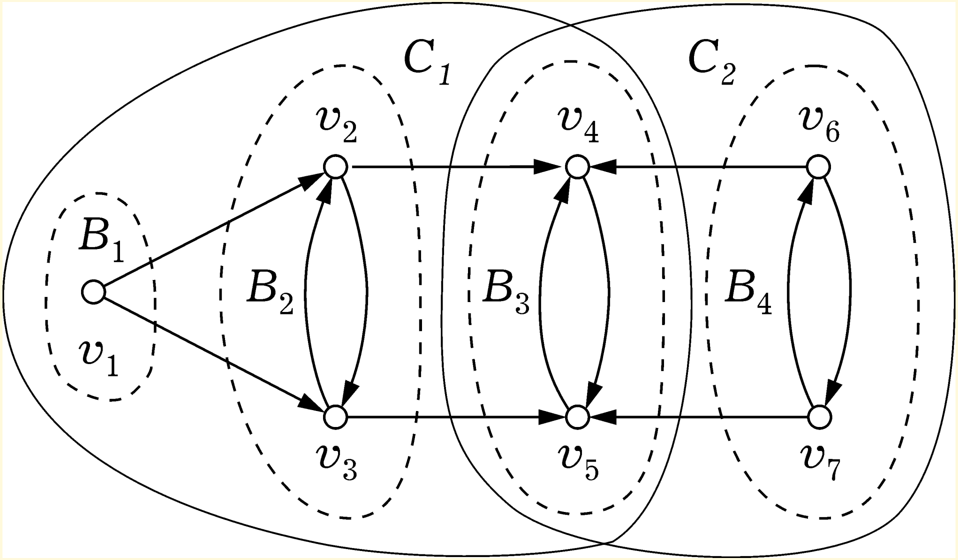

Рис. 43

Матрица достижимости:

$$C = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}.$$

Бикомпоненты.
Для первой вершины множество $P_1 = \{1\}$ включает только ее саму.
Для второй вершины — $P_2 = \{2,\ 3\}$,
для четвертой — $P_4 = \{4,\ 5\}$, для шестой — $P_6 = \{6,\ 7\}$.
Получили B_1, B_2, B_3, B_4.

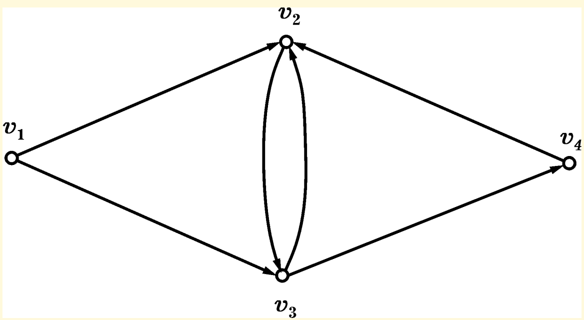

Рис. 44

$$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & \color{red}{1} & \color{red}{1} & \color{red}{1} \\ 0 & \color{red}{1} & \color{red}{1} & \color{red}{1} \\ 0 & \color{red}{1} & \color{red}{1} & \color{red}{1} \end{pmatrix}$$

Найдем бикомпоненты графа изображенного на рисунке. Для первой вершины множество $P_1 = \{1\}$ включает только ее саму. Для второй вершины имеем $P_2 = \{2, 3, 4\}$. Соответственно полученные множества вершин порождают бикомпоненты B_1, B_2.

12.7. Остовное дерево наименьшего веса

Неориентированный (ориентированный) граф, у которого каждому ребру (дуге) сопоставлено некоторое действительное число, называют **взвешенным** или **размеченным** графом.

Это число называют **весом** или **меткой** ребра (дуги).

Алгоритм Краскала вычисляет для заданного взвешенного неориентированного графа G *остовное дерево* с наименьшей суммой весов ребер — **остовное дерево наименьшего веса**.

При описании алгоритма будем использовать способ хранения данных, называемый **очередью**.

Элементы данных в очереди упорядочиваются по времени поступления. Элементы можно добавлять в очередь и извлекать из очереди.

В каждый момент времени доступен только один элемент, который был помещен в очередь раньше других, — „голова“ очереди.

При добавлении новый элемент помещается в „хвост“ очереди, т.е. работа ведется по обычному для очереди правилу — „первым пришел — первым вышел“.

Чтобы извлечь из очереди некоторый элемент, не доступный в текущий момент, надо извлечь все ранее поступившие элементы, начиная с „головы“ очереди.

Обычно очередь реализуется в виде списка.

Алгоритм Краскала

Рассмотрим алгоритм нахождения остовного дерева наименьшего веса. Пусть дан связный неориентированный граф $G = (V, E)$ с числовыми неотрицательными весами ребер. Вес ребра e обозначим $\varphi(e)$.

В результате работы алгоритма получим остовное дерево $T = (V, H)$ графа G, такое, что сумма $\sum\limits_{e \in H} \varphi(e)$ является наименьшей.

Отсортируем все ребра исходного графа по возрастанию весов и сформируем из них очередь так, чтобы в „голове“ очереди находилось ребро с наименьшим весом, а в „хвосте“ — с наибольшим и веса ребер не убывали от „головы“ очереди к „хвосту“.

Метод состоит в „сшивании“ искомого дерева из *компонент* остовного леса. Первоначально *остовный лес* представляет собой множество изолированных вершин исходного графа, т.е. его множество ребер пусто. На первом шаге из очереди извлекается ребро наименьшего веса и добавляется к множеству ребер исходного дерева.

На последующих шагах алгоритма из очереди извлекается по одному ребру. Если это ребро соединяет вершины, принадлежащие разным компонентам текущего остовного леса, то оно добавляется к текущему множеству ребер искомого дерева, а указанные компоненты сливаются в одну. Иначе ребро отбрасывается. Процесс повторяется до тех пор, пока число компонент остовного леса не окажется равным 1. Можно показать, что эта компонента и будет искомым остовным деревом наименьшего веса.

1. Множество ребер H искомого остовного дерева полагаем пустым ($H = \varnothing$).
2. Формируем множество $V_S = \{\{v_1\}, \dots, \{v_n\}\}$, элементами которого являются множества вершин, соответствующих компонентам исходного остовного леса. Каждая такая компонента состоит из единственной вершины.
3. Сортируем множество ребер E исходного графа по возрастанию весов и формируем очередь Q , элементами которой являются ребра графа G .
4. Если множество V_S содержит более одного элемента (т.е. остовный лес состоит из нескольких компонент) и очередь Q не пуста, переходим на шаг **5**, если иначе — на шаг **7**.

5. Извлекаем из очереди Q ребро e. Если *концы* ребра e принадлежат различным множествам вершин V_i и V_j из V_S, то переходим на шаг **6**, если иначе, то отбрасываем извлеченное ребро и возвращаемся на шаг **4**.
6. Объединяем множества вершин V_i и V_j (полагая $W = V_i \cup V_j$), удаляем множества V_i и V_j из множества V_S и добавляем в V_S множество W. Добавляем ребро e в множество H. Возвращаемся на шаг **4**.
7. Прекращаем работу. Множество H есть множество ребер полученного остовного дерева.

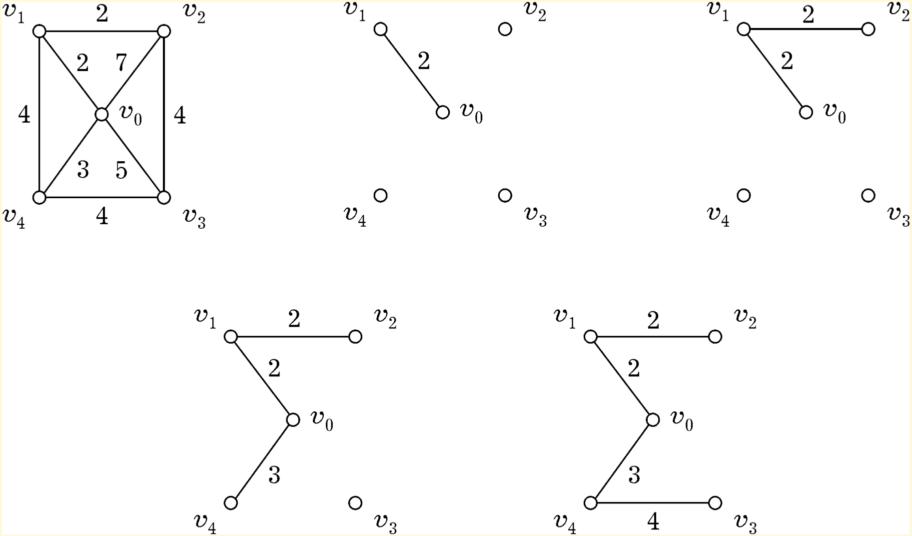
v_1
2
v_2
2
7
4
v_0
4
3
5
v_4
4
v_3
v_1
v_2
2
v_0
v_4
v_3
v_1
2
v_2
2
v_0
v_4
v_3
v_1
2
v_2
2
v_0
3
v_4
v_3
v_1
2
v_2
2
v_0
3
v_4
4
v_3

Рис. 45

Исходный граф изображен на рисунке *а*. На рисунке *б* проиллюстрирован результат выполнения первого шага алгоритма.
На *в* показан результат добавления следующего ребра $\{v_1, v_2\}$ с весом 2 из очереди. На *г* приведен результат добавления ребра $\{v_0, v_4\}$ с весом 3 .
Если следующим в очереди ребром будет $\{v_1, v_4\}$, оно будет отброшено.

Дальнейший ход работы алгоритма зависит от того, в каком порядке в очереди размещены ребра $\{v_2,\ v_3\}$ и $\{v_3,\ v_4\}$ с весами 4. Любое из них может быть добавлено в множество ребер остовного дерева, и на этом алгоритм закончит работу. На *д* приведено остовное дерево, полученное после добавления ребра $\{v_3,\ v_4\}$.

Для приведенного графа оба ребра с весом 2 войдут в остовное дерево независимо от порядка их расположения в очереди после сортировки, а ребро $\{v_1,\ v_4\}$ не войдет ни в какое остовное дерево наименьшего веса.