
Взвешенные деревья

Лектор: Завьялов Олег Геннадьевич
кандидат физико-математических наук, доцент

Взвешенные деревья

В компьютере все буквы и другие символы хранятся в виде строк из 1 и 0.

Если данных достаточно много, всегда желательно провести компактификацию.

Проблема: если строки, представляющие разные символы, имеют разную длину, то как узнать, где заканчивается строка одного символа и начинается строка другого.

Определение.

Однозначно декорируемый код для языка как множество, что каждая строка в языке может быть задана однозначно как конкатенация элементов.

В этом случае строки из единиц и нулей, представляющие элементы из A , будут кодом.

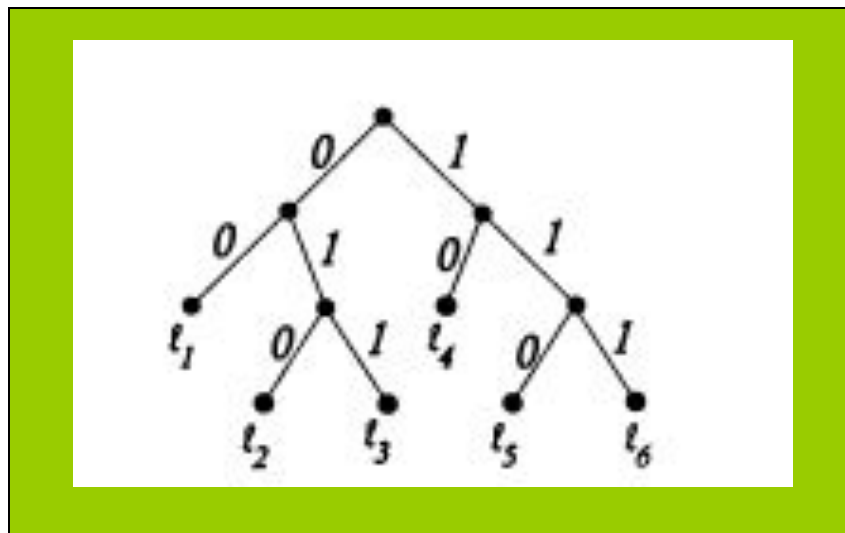
Эти строки образуют однозначно декодируемый код. Разделяя строки на элементы, представляющие A , знаем, что представление однозначно. Декодированные слова будут правильные.

Код C **префиксный**, если он обладает свойством, что никакой элемент кода не может быть начальной строкой другого элемента кода.

Конкатенация (сцепление) — операция склеивания объектов линейной структуры, обычно строк. Например, конкатенация слов «микро» и «мир» даст слово «микромир».

Пример.

Дерево с листьями



Пути к листьям $v_1, v_2, v_3, v_4, v_5, v_6$ обозначают 00, 010, 011, 10, 110, 111. Строку, соответствующую данному листу, называют **путевым кодом**.

Теорема.

В любом бинарном дереве путевые коды для листьев дерева являются префиксным кодом.

Чем меньше вес дерева, тем в большей степени достигнута цель.

Чтобы найти наилучший код для минимизации данных, необходимо найти код с минимальным весом.

Процесс построения такого дерева называется ***алгоритмом Хаффмана***.

Код, приписываемый символам согласно их путевому коду, называется ***кодом Хаффмана***.

Пример.

Имеются буквы и их частоты

СИМВОЛ	Частота
<i>A</i>	15
<i>B</i>	25
<i>C</i>	10
<i>D</i>	30
<i>E</i>	15
<i>F</i>	5

Бинарное дерево, что бы наиболее часто используемые элементы были возможно ближе расположены к корню.

Требуется найти такое дерево, что вес дерева

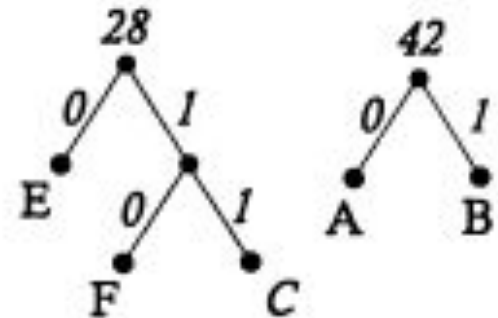
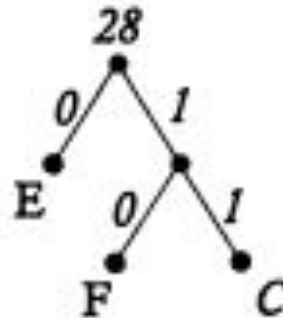
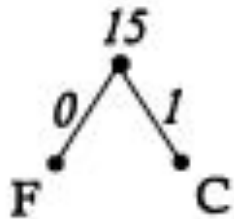
$$w = l_1 f_1 + l_2 f_2 + l_3 f_3 + \dots + l_n f_n = \sum_{i=1}^n l_i f_i$$

Пример (продолжение).

li и fi - уровень и частота заданного элемента.

Упорядочим частоты в списке частот (5, 10, 13, 17, 25, 30).

Деревья:



Списки частот: (13, 15, 17, 25, 30)

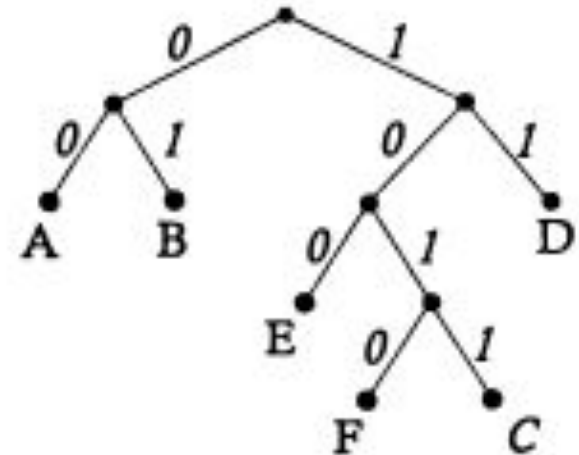
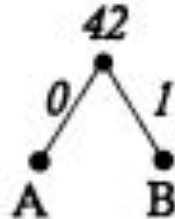
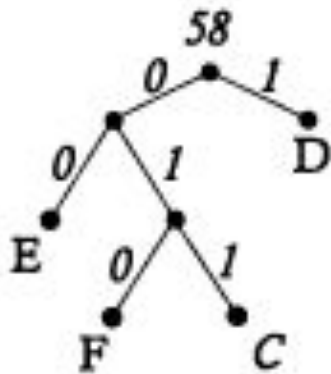
(17, 25, 28, 20)

(28, 30, 42)

(42, 58)

Пример (продолжение).

Объединяем два дерева и формируем исходное дерево



Лемма.

Для заданного множества из n символов и их частот существует бинарное дерево минимального веса с символами в качестве листьев.

Доказательство:

Существует только четное число бинарных деревьев с n листьями. Для каждого

$$w = l_1 f_1 + l_2 f_2 + l_3 f_3 + \dots + l_n f_n = \sum_{i=1}^n l_i f_i$$

и выберем дерево, которое обеспечивает наименьшее значение. Ч.т.д.

Лемма.

В дереве с минимальным весом на максимальном уровне листья присутствуют в парах, т.е. всюду, где есть сын, имеется и правый и левый сын и наоборот.

Лемма.

В дереве с минимальным весом два символа с минимальными частотами расположены на максимальном уровне.

Лемма.

Существует дерево с минимальным весом, в котором два символа с наименьшими частотами имеют одного родителя.

Теорема.

Для заданного множества символов с соответствующими частотами дерево Хаффмана является деревом с минимальным весом.

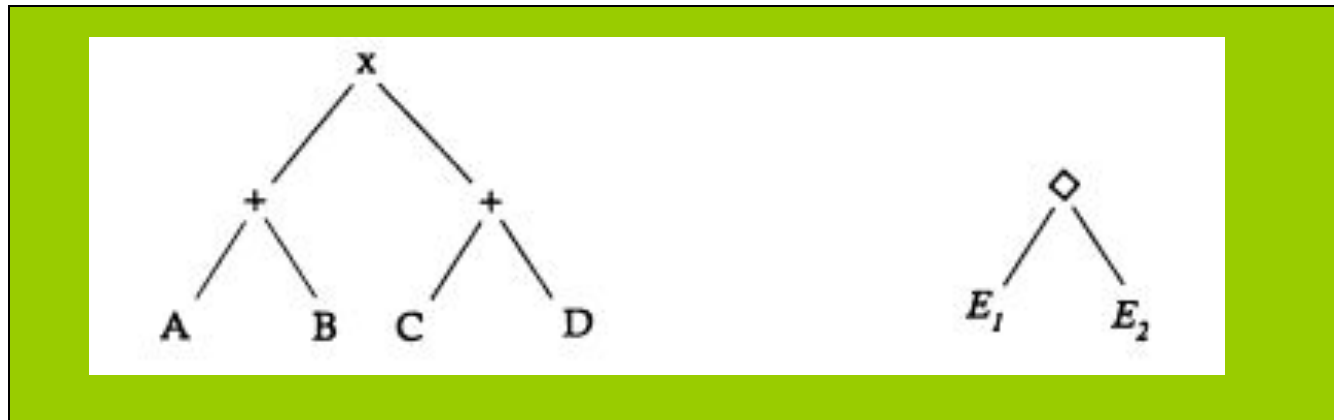
Обход бинарных деревьев.

Рассмотрим способ обхода бинарного дерева

Используем команду **обработать** (n), где n – узел.

Обход дерева в центрированном порядке.

Обращаем процесс, использованный при создании дерева. Если дерево:



◇ - бинарная операция над выражениями E_1 и E_2 , обрабатываем (печатаем) это как $E_1 \diamond E_2$

Получается выражение в **инфиксной записи**.

Алгоритм обхода дерева в центрированном порядке (ОПД).

лс (v) – левый сын вершины v .

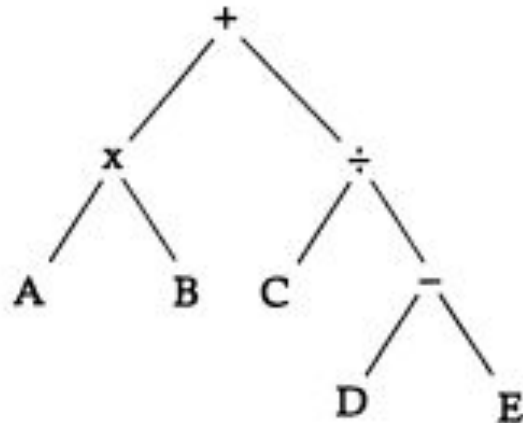
пс (v) – правый сын вершины v .

Алгоритм обхода дерева в центрированном порядке — ОЦП (корень)

(1) Если лс(корень) существует, то ОЦП(лс(корень)).

(2) Обработать(корень).

(3) Если пс(корень) существует, то ОЦП(пс(корень)).



Остовные деревья

Определение.

Остовное дерево – дерево T , которое является подграфом графа G таким, что каждая вершина в G является вершиной в T . Каждый связный граф имеет остовное дерево.

Два метода построения остовного дерева.

Первый – метод поиска в ширину,

Второй – метод поиска в глубину.

Первый метод: произвольную вершину v_0 графа G выбирают в качестве корня дерева T . Для каждой вершины v , смежной с вершиной v_0 , в дерево T добавляется вершина v и ребро $\{v, v_0\}$.

Это вершины уровня 1. Затем берем каждую вершину v_i уровня 1 и для каждой вершины v_j и ребро $\{v_i, v_j\}$.

На втором этапе – это вершины уровня 2.

Процесс продолжается, пока в графе G не останется вершин, которые можно добавить в дерево. $\Rightarrow T$ является деревом.

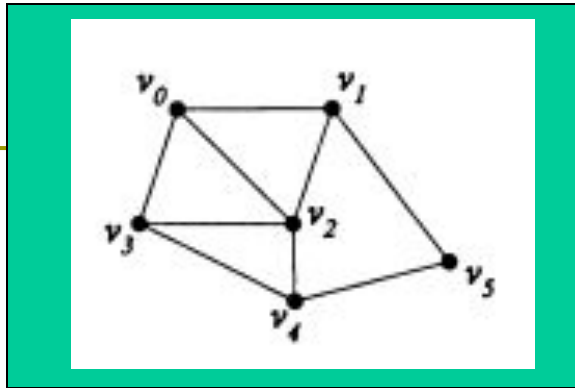
Если расстояние от v_0 до вершины v графа G равно n , то эта вершина будет добавлена в дерево на уровне n . $\Rightarrow T$ является остовным деревом.

Алгоритм поиска остовного дерева в ширину — ПОДШ(G)

- (1) Выбрать произвольный элемент v_0 графа G . Пусть $v_0 \in V^T$ и $L(v_0) = 0$.
- (2) Для всех $v \in V - V^T$ таких, что v смежна с v_0 , положить $v \in V^T$, $\{v_0, v\} \in E^T$ и $L(v) = 1$.
- (3) Пусть $i = 1$.
- (4) Выбрать $v_j \in V^T$ такое, что $L(v_j) = i$.
- (5) Выбрать $v \in V - V^T$. Если v смежна с v_j , положить $v \in V^T$, $\{v_0, v\} \in E^T$ и $L(v) = i + 1$.
- (6) Продолжать шаг 5, пока все элементы множества $V - V^T$ не будут рассмотрены. (Обратите внимание, что $V - V^T$ постоянно меняется.)
- (7) Повторять шаги 4, 5, и 6, пока все v_j такие, что $L(v_j) = i$, не будут выбраны.
- (8) Положить $i = i + 1$.
- (9) Повторять шаги 4–8 до $V = V^T$.

Пример.

Граф



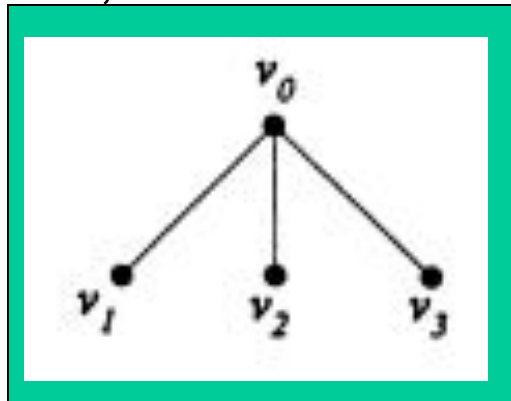
Пусть вершина v_0 выбрана в качестве первой вершины.

Тогда $L(v_0) = 0$ и $v_0 \in V^T$. v_1 является смежной вершиной с v_0 , положим $v_1 \in V^T$, $\{v_0, v_1\} \in E^T$ и $L(v_1) = 1$.

Вершина v_2 смежна с v_0 , положим $v_2 \in V^T$, $\{v_0, v_2\} \in E^T$ и $L(v_2) = 1$.

Вершина v_3 смежна с v_0 , положим $v_3 \in V^T$, $\{v_0, v_3\} \in E^T$ и $L(v_3) = 1$.

Получаем дерево



Пример (продолжение).

Рассмотрим вершины v , что $L(v) = 1$.

Начнем с v_1 , находим неиспользованные вершины, смежные с v_1 .

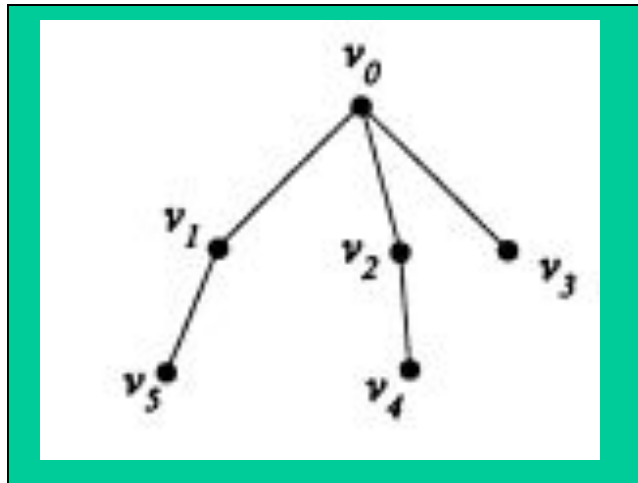
Вершина v_5 смежна с v_1 , положим $v_5 \in V^T$, $\{v_1, v_5\} \in E^T$ и $L(v_5) = 2$.

Больше нет неиспользованных и смежных с v_1 вершин, переходим к v_2 .

Вершина v_4 смежна с v_2 , положим $v_4 \in V^T$, $\{v_2, v_4\} \in E^T$ и $L(v_4) = 2$.

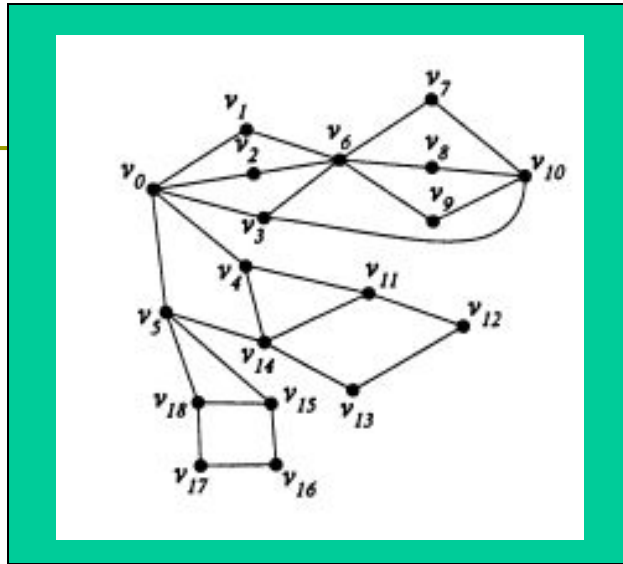
Все вершины использованы, процесс завершен.

Имеем дерево



Пример.

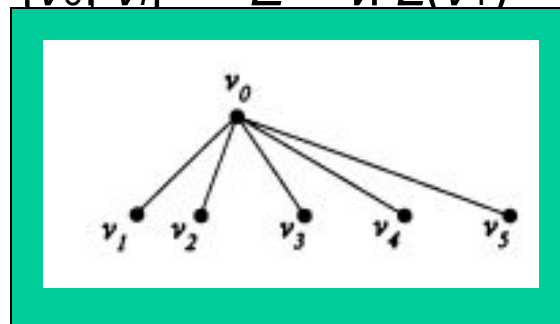
Граф



Выберем вершину v_0 как начальную.

Тогда $L(v_0) = 0$ и $v_0 \in V^T$. v_1, v_2, v_3, v_4, v_5 являются смежными с v_0 , положим $v_i \in V^T$, $\{v_0, v_i\} \in E^T$ и $L(v_i) = 1$, где $1 \leq i \leq 5$.

Получим дерево



Пример (продолжение).

Начинаем на уровне 1 с вершины v_1 .

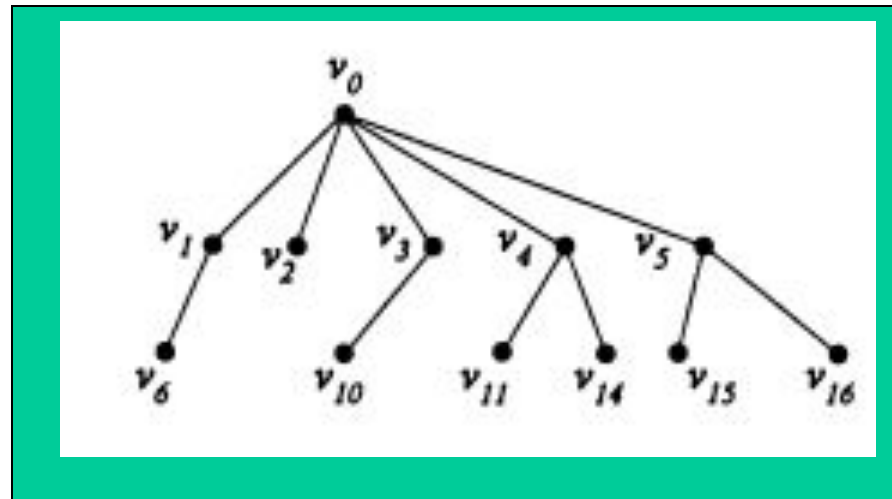
На этом уровне: вершина v_6 смежная с v_0 , вершина v_{10} смежная с v_3 ,
т.д.

$v_6, v_{10}, v_{11}, v_{14}, v_{15}, v_{18} \in V^T$,

$L(v_6) = L(v_{10}) = L(v_{11}) = L(v_{14}) = L(v_{15}) = L(v_{18}) = 2$

$\{v_1, v_6\}, \{v_3, v_{10}\}, \{v_4, v_{11}\}, \{v_4, v_{14}\}, \{v_5, v_{15}\}, \{v_5, v_{18}\} \in E^T$.

Получим дерево



Пример (продолжение).

Теперь на уровне 2 .

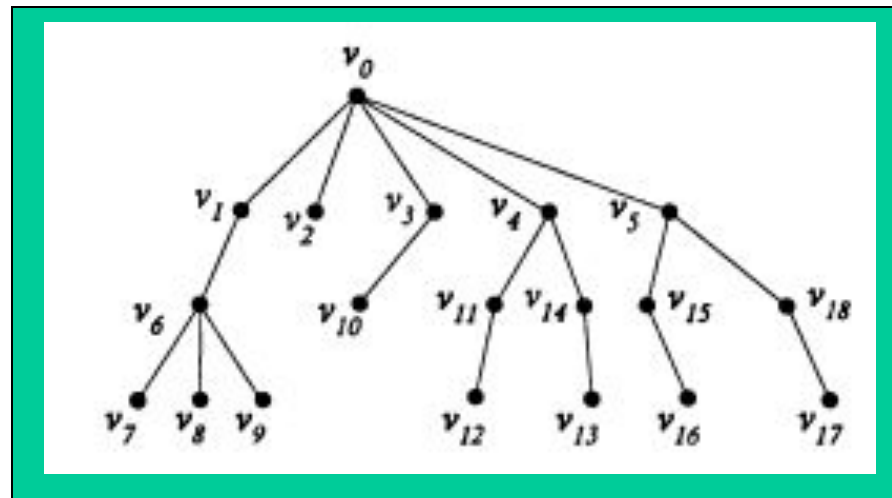
На этом уровне: вершины v_7, v_8, v_9 смежные с v_6 , v_{12} , смежная с v_{11} , v_{13} , смежная с v_{14} , v_{16} , смежная с v_{15} , v_{17} , смежная с v_{18} .

$v_7, v_8, v_9, v_{12}, v_{16}, v_{17} \in V^T$,

$L(v_7) = L(v_8) = L(v_9) = L(v_{12}) = L(v_{13}) = L(v_{16}) = L(v_{17}) = 3$

$\{v_6, v_7\}, \{v_6, v_8\}, \{v_6, v_9\}, \{v_{11}, v_{12}\}, \{v_{14}, v_{13}\}, \{v_{16}, v_{13}\}, \{v_{17}, v_{18}\} \in E^T$.

Получим дерево



Обратное дерево.

При поиске в ширину вначале отыскиваются все вершины, смежные с заданной вершиной. Потом осуществляется переход на следующий уровень.

Главная цель при поиске в ширину - построение дерева как можно более длинного пути.

Когда путь заходит в тупик и формирует лист, необходимо возвращаться к родителю листа и пытаться сформировать другой путь. Возврат к родителю происходит после попытки построить все возможные пути от сына этого родителя. Т.е. пытаемся достичь самый большой уровень, какой возможен.

Алгоритм начинается у заданной вершины графа, которую считают корнем. Выбирают вершину, смежную с корнем, формируют ребро дерева.

Затем выбирают вершину, смежную с ранее выбранной вершиной и формируют новое ребро. Необходимо помечать использованные вершины.

Обратное дерево.

Если, находясь в вершине v , выбираем другую вершину w и обнаруживается, что вершина w уже была добавлена в дерево, то ребро $\{v, w\}$ между этими вершинами не может быть добавлено в дерево. Такое дерево называют **обратным ребром**.

Чтобы объявить ребро обратным, необходимо проверить, не является ли вершина w родителем вершины v , т.к. ребро $\{v, w\}$ уже присутствует в дереве.

При выборе w следует избегать случая, когда вершина w является родителем v . Если $\{v, w\}$ - обратное ребро, то остаемся в v и выбираем новую смежную вершину, если это возможно.

Любое ребро графа – либо **ребро дерева**, либо **обратное ребро**.

В алгоритме множество ребер дерева называют ребра дерева, множество обратных ребер – обратные ребра.

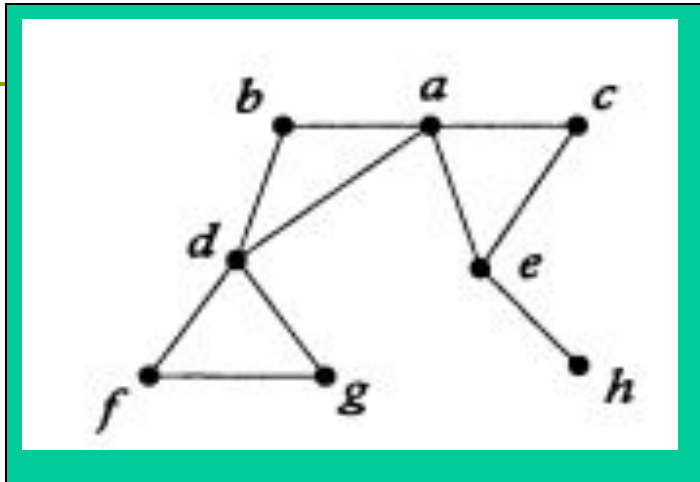
“исп.” – использованные вершины, “нов.” – новые вершины.

Алгоритм поиска остовного дерева в глубину — ПОДГ(G)

- (1) Пометить каждую вершину графа G символом n .
- (2) Выбрать произвольный элемент v_0 графа G и назвать его корнем дерева.
- (3) Заменить метку вершины v_0 с “нов” на “исп” и положить $v = v_0$
- (4) Пока имеются невыбранные вершины, смежные с v , выполнять следующие действия:
 - а) Выбрать вершину w , смежную с v .
 - б) Если w имеет метку “нов”, добавить (v, w) в РЕБРА ДЕРЕВА, поменять метку w на “исп”, положить $w = v$ и повторить шаг 4.
 - в) Если w имеет метку “исп” и не является родителем v , добавить (v, w) в ОБРАТНЫЕ РЕБРА и повторить шаг 4.
- (5) Если $v \neq a$, положить $v = (v)$ и повторить шаг 4.

Пример.

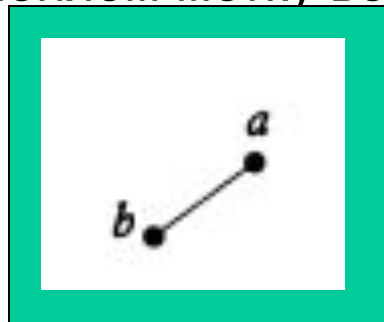
Граф



Произвольно выбирают вершину a в качестве корня.

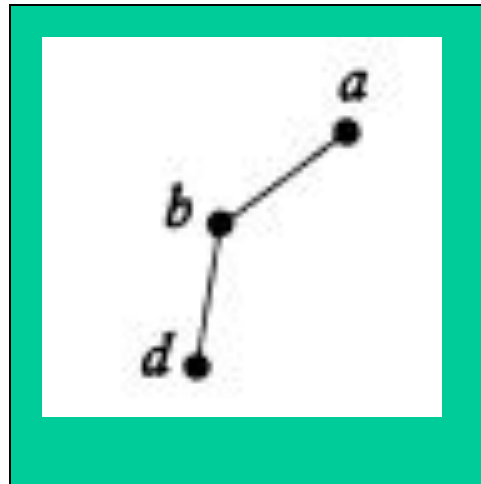
Меняем метку a с “нов.” на “исп.”

Вершина b смежна с a и имеет метку “нов.”, добавляем ребро $\{a, b\}$ в ребра дерева и меняем метку вершины b на “исп.”



Пример (продолжение).

От вершины b переходим к вершине d , т.к. она является смежной с b . Вершина d имеет метку “нов.”, поэтому добавляем ребро $\{b, d\}$ в **ребра дерева** и меняем метку вершины d на “исп.”



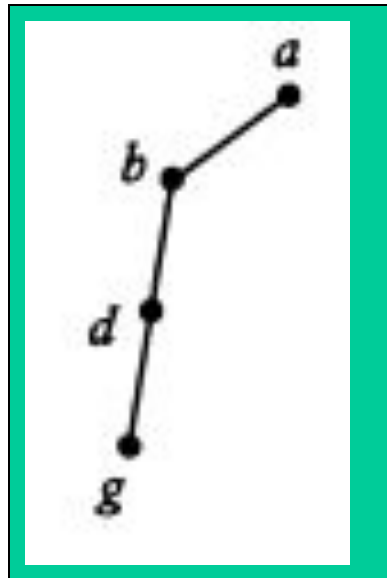
Пример (продолжение).

Выбираем вершину, смежную с вершиной d . (a , f или g)

Вершина d (поиск в глубину не единственный)

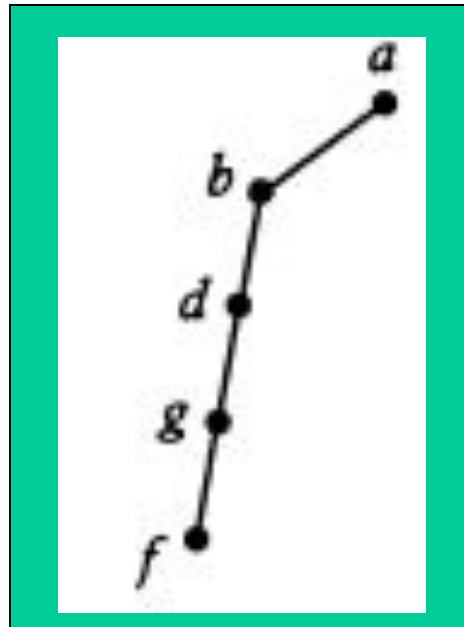
Следующей g (имеет метку “нов.”)

Добавляем ребро $\{d, g\}$ в **ребра дерева** и меняем метку вершины g на “исп.”



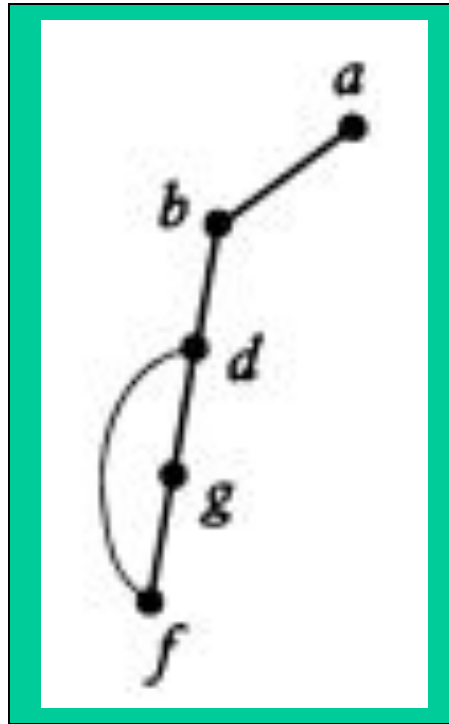
Пример (продолжение).

Из вершины g выбираем вершину f , т.к. она является смежной с g .
Вершина f имеет метку “нов.”, поэтому добавляем ребро $\{g, f\}$ в **ребра дерева** и меняем метку вершины f на “исп.”



Пример (продолжение).

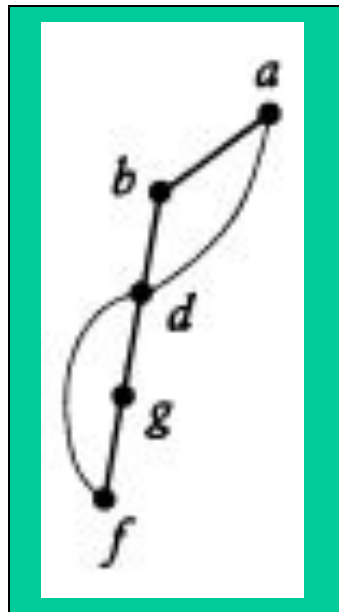
Из вершины f выбираем вершину d , т.к. она является смежной с f . Однако вершина d уже имеет метку “исп.”, поэтому добавляем ребро $\{d, f\}$ в **обратные ребра** дерева



Пример (продолжение).

Больше нет ребер для проверки смежности с вершиной f , кроме родителя, возвращаемся к вершине g .

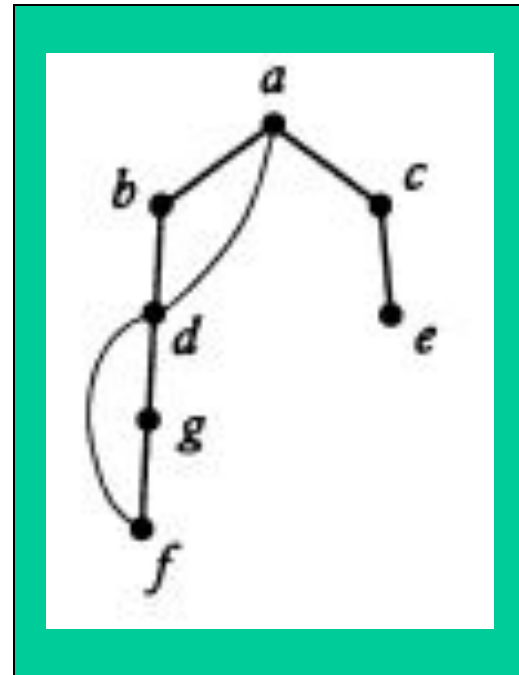
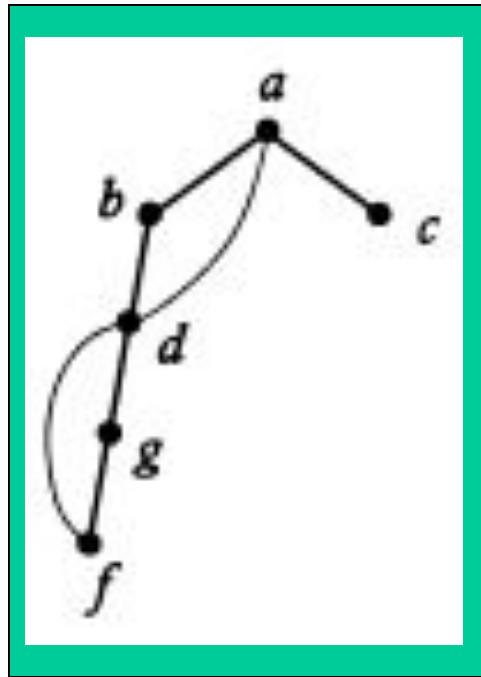
Иных ребер для проверки на смежность с вершиной g тоже нет, потому возвращаемся к вершине d . Единственной вершиной для проверки является вершина a , но a уже имеет метку “исп.”, поэтому ребро $\{d, f\}$ добавляем в **обратные ребра** и возвращаемся в вершину b



Пример (продолжение).

Ребер для проверки на смежность с вершиной b больше нет, возвращаемся к вершине a . Выбираем c или e .

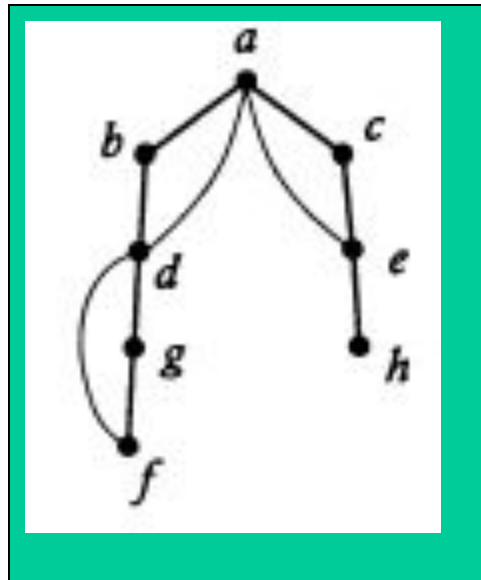
Предположим, выбираем c (имеет метку “нов.”). Добавляем ребро $\{a,c\}$ в **ребра дерева** и меняем метку вершины c на “исп.”



Пример (продолжение).

Вершина h смежна с вершиной e .

Добавляем ребро $\{e, h\}$ в **ребра дерева** и меняем метку e на “исп.”



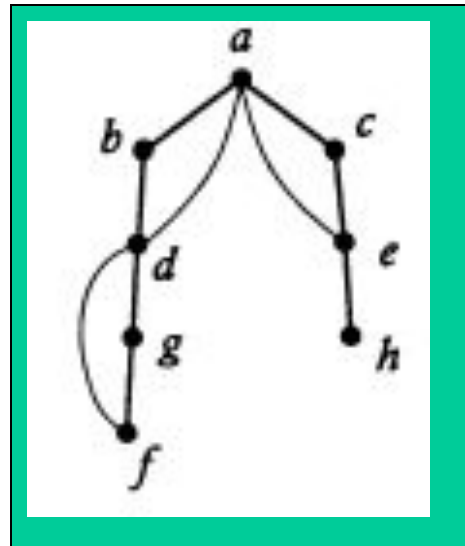
Пример (продолжение).

Больше нет вершин для проверки из вершины h , возвращаемся в вершину e .

Больше нет вершин для проверки из вершины e , возвращаемся в вершину c .

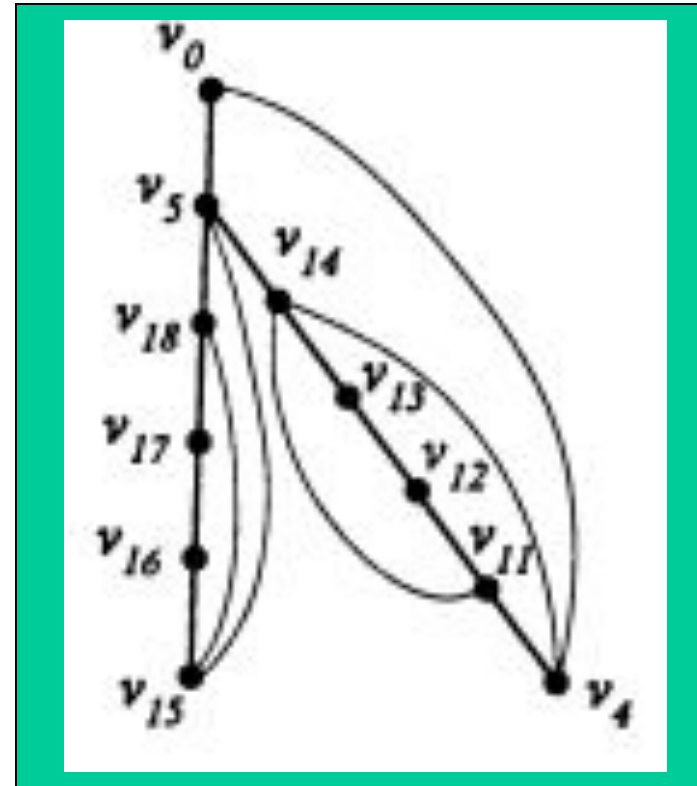
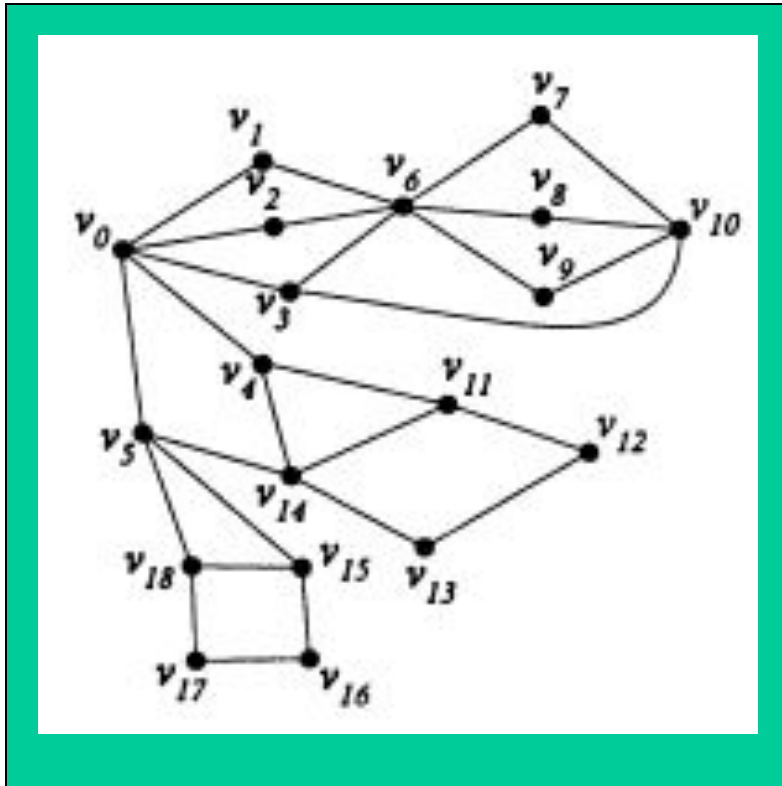
Больше нет вершин для проверки из вершины c , возвращаемся в вершину a .

Других вершин для проверки из вершины a тоже нет. Процесс завершен.

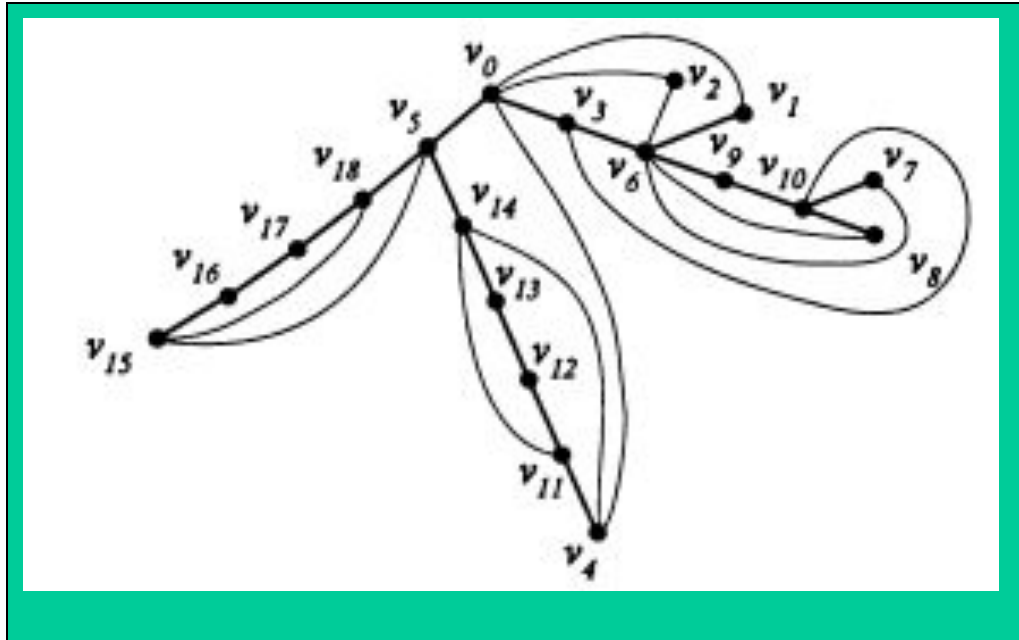


Пример.

Найти остовное дерево



Пример (продолжение).



Определение.

Лес остовных деревьев называется **остовным лесом**.

Для построения остовного дерева следует выбрать вершину для корня первого дерева и построить остовное дерево.

Теорема (для построения остовного дерева в глубину).

Если T – глубинное остовное дерево графа $G(V, E)$ и $\{a, b\}$ – ребро графа $G(V, E)$, то либо a является потомком b , либо b является потомком a .

Доказательство:

Если $\{a, b\}$ – ребро дерева T , то вывод очевиден.

Если не так, то одна из вершин (a) должна быть помещена в дерево первой. Но вершина b не была помещена в дерево на шаге 4 алгоритма ПОДГ, то поиск продолжается из вершины a до тех пор, пока не будет найдена вершина b .

Поэтому вершина b является потомком a . Ч.т.д.

Теорема.

Пусть T – глубинное остовное дерево графа $G(V, E)$.

Вершина $a \in V$ является точкой сочленения графа $G(V, E)$ тогда и только тогда, когда вершина a (1) либо является корнем дерева T и имеет более одного сына, либо (2) вершина a не является корнем дерева T , и существует такой сын s , что между s , или одним из его потомков, и собственным предком вершины a не существует обратного ребра.

OP – обратное ребро

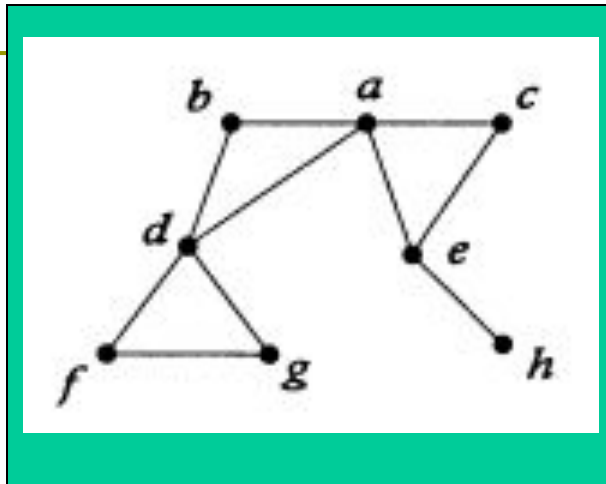
ЗС – значение счета (с является n -ой вершиной)

Алгоритм поиска точек сочленения — ПТС(v)

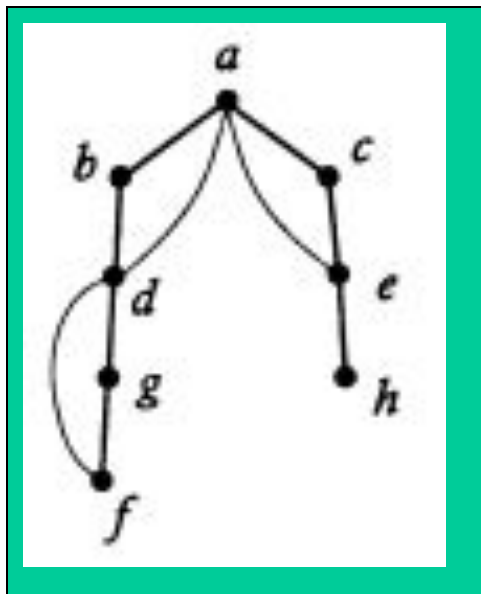
- (1) Пометить v символом “исп”.
- (2) Присвоить i значение $i + 1$.
- (3) Положить $OP(v) = ЗС(v) = i$.
- (4) Пока имеется невыбранная вершина, смежная с v , выполнять следующее:
 - (а) Выбрать вершину w , которая является смежной с v .
 - (б) Если w имеет метку “нов”, то:
 - (i) добавить (v, w) в РЕБРА ДЕРЕВА;
 - (ii) положить $v = \text{parent}(w)$;
 - (iii) вызвать ПТС(w);
 - (iv) если $OP(w) \geq ЗС(v)$, то v — точка сочленения. Ребра, ниже v , которые охватывают компоненту, удаляются;
 - (v) положить $OP(v) = \min(OP(v), OP(w))$.
 - (г) Если w имеет метку “исп” и w не является $\text{parent}(w)$, то $OP(v) = \min(OP(v), ЗС(w))$.

Пример.

Граф



дерево



Теорема (Формула Кэли для дерева).

Число остовных деревьев для n размеченных вершин равно n^{n-2}

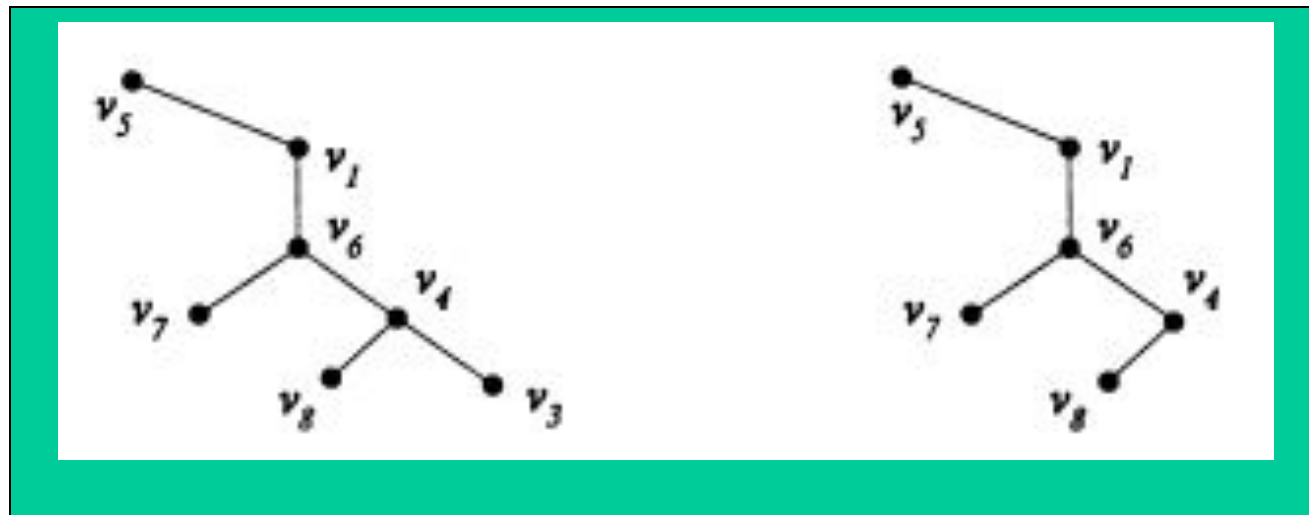
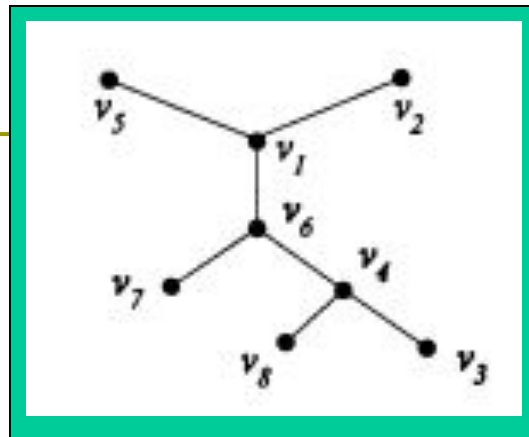
Алгоритм преобразования остовного дерева в последовательность.

Алгоритм перевода дерева в последовательность — ДвП(T) для $n \geq 3$

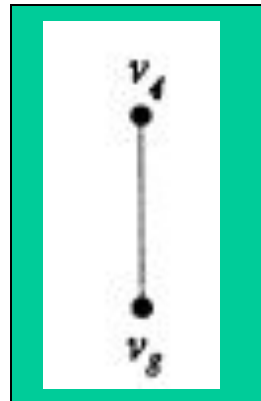
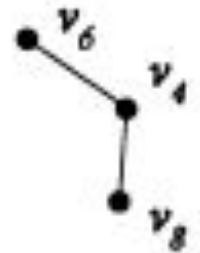
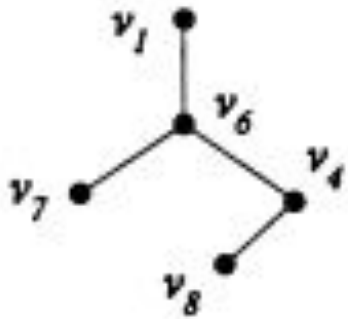
- (1) Для выбора a_1 взять лист v_j с наименьшим значением j . Всегда существует единственное k такое, что $\{v_j, v_k\}$ является ребром дерева. Удалить это ребро и положить $a_1 = k$.
- (2) Если выбрано a_{i-1} , то для выбора a_i из оставшегося дерева взять лист v_j с наименьшим значением j . Всегда существует единственное k такое, что $\{v_j, v_k\}$ является ребром дерева. Удалить это ребро и положить $a_i = k$.
- (3) Продолжать, пока не будет выбрано a_{n-2} .

Пример.

T - дерево

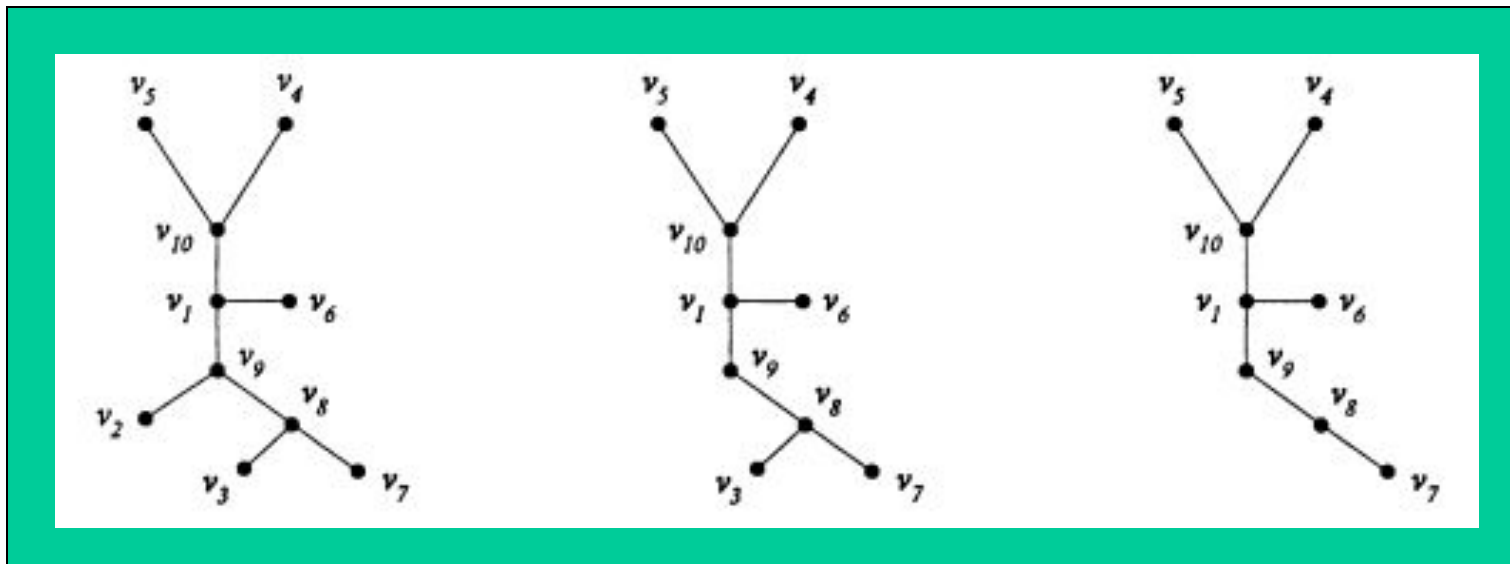


Пример (продолжение).



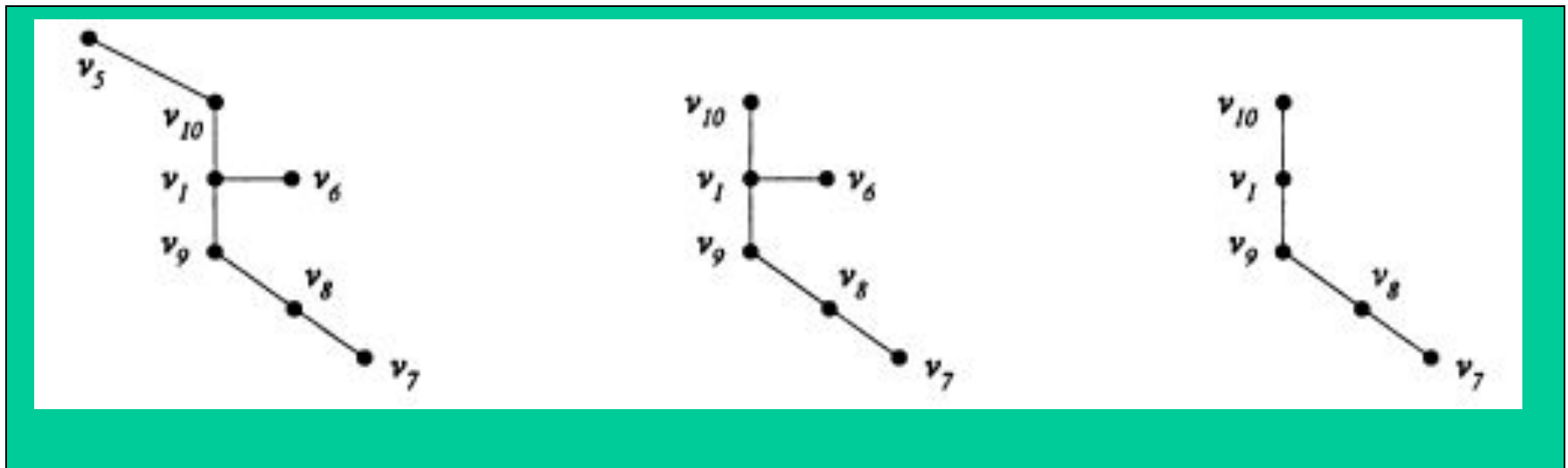
Пример (!).

Дерево T . v_2 – лист с наименьшим индексом. Удаляем ребро $\{v_2, v_9\}$ и полагаем $a_1 = 9$. В оставшемся дереве v_3 – лист с наименьшим индексом, удаляем ребро $\{v_3, v_8\}$ и полагаем $a_2 = 8$.



Пример (продолжение).

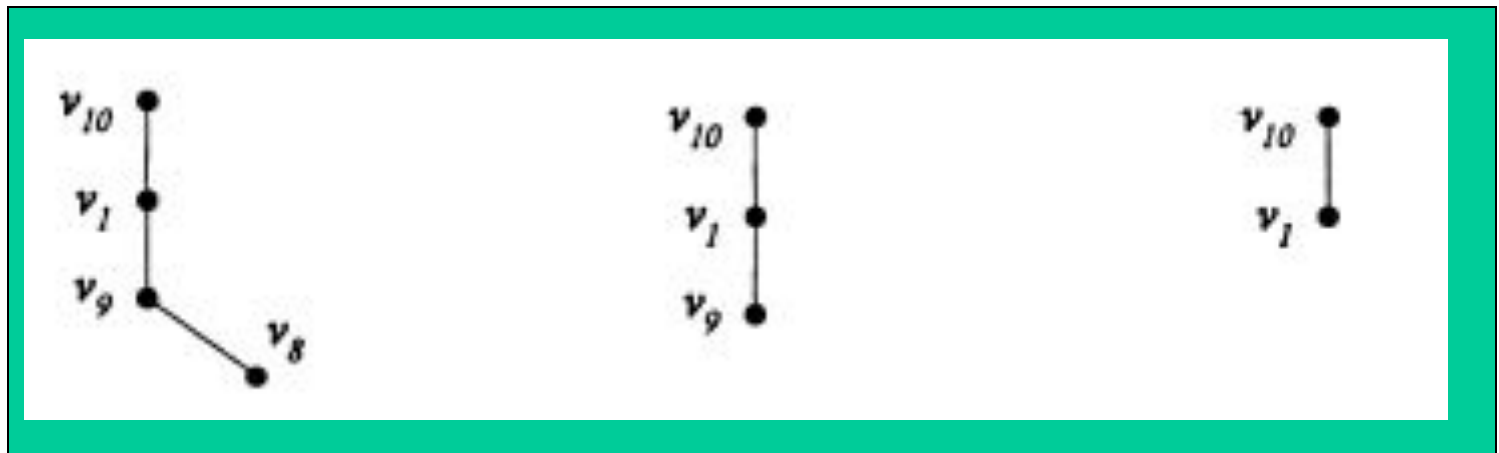
Вершина v_4 стала листом с наименьшим индексом, удаляем ребро $\{v_4, v_{10}\}$ и полагаем $a_3 = 10$. Вершина v_5 стала листом с наименьшим индексом, удаляем ребро $\{v_5, v_{10}\}$ и полагаем $a_4 = 10$. Вершина v_6 стала листом с наименьшим индексом, удаляем ребро $\{v_6, v_1\}$ и полагаем $a_5 = 1$.



Пример (продолжение).

В оставшемся дереве вершина v_7 стала листом с наименьшим индексом, удаляем ребро $\{v_7, v_8\}$ и полагаем $a_6 = 8$. Вершина v_8 стала листом с наименьшим индексом, удаляем ребро $\{v_8, v_9\}$ и полагаем $a_7 = 9$.

Оставшаяся вершина v_9 стала листом с наименьшим индексом, удаляем ребро $\{v_9, v_1\}$ и полагаем $a_8 = 1$. Осталось ребро (справа)



Последовательность имеет вид

9, 8, 10, 10, 1, 8, 9, 1

Алгоритм перевода последовательности в дерево.

Алгоритм перевода последовательности в дерево —

ПвД(a_1, a_2, \dots, a_{n-2}) для $n \geq 3$

- (1) Для каждого v_i , если i появляется в последовательности n_i раз, положить $\deg(v_i) = n_i + 1$.
- (2) Прочитать a_1 и сформировать ребро $\{v_{a_1}, v_j\}$, где j — наименьшая метка, такая что $\deg(v_j) = 1$.
- (3) Уменьшить $\deg(v_{a_1})$ и $\deg(v_j)$ на 1.
- (4) Предположим, что a_{i-1} прочитано, читать a_i и формировать ребро $\{v_{a_i}, v_j\}$, где j — наименьшая такая метка, что $\deg(v_j) = 1$.
- (5) Уменьшить $\deg(v_{a_i})$ и $\deg(v_j)$ на 1.
- (6) После того, как a_{n-2} прочитано, создать ребро между двумя оставшимися вершинами степени 1.

Пример.

Задана последовательность

1, 4, 1, 6, 6, 4

Восстановить дерево.

1, 4, 6 появляются в последовательности дважды,

$$\deg(v_1) = \deg(v_4) = \deg(v_6) = 3.$$

2, 3, 5 не встречаются вообще,

$$\deg(v_2) = \deg(v_3) = \deg(v_5) = \deg(v_7) = \deg(v_8) = 1.$$

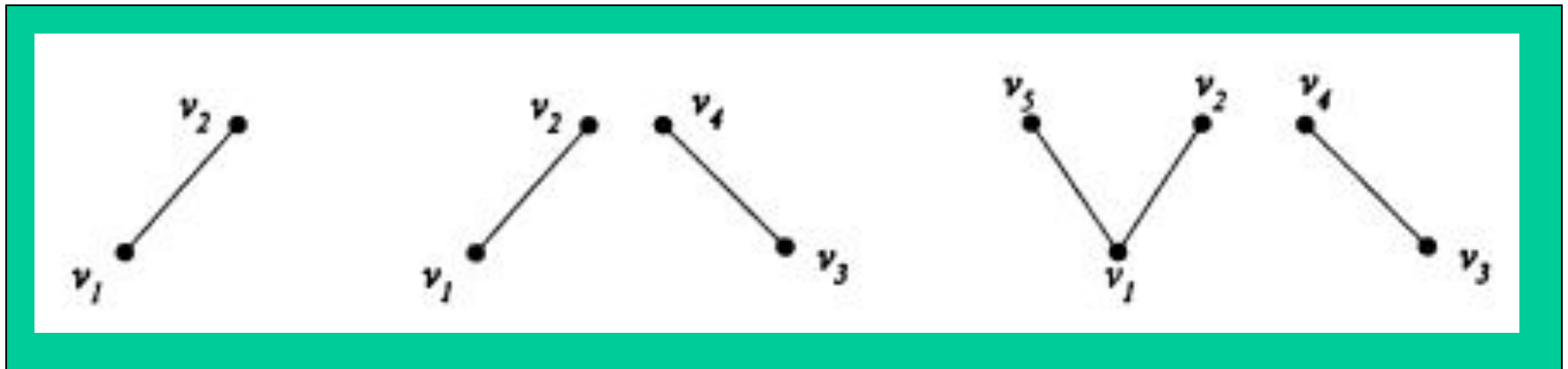
Запишем с их помощью восьмерки из чисел (3, 1, 1, 3, 1, 3, 1, 1), которую называют восьмеркой степеней.

Считаем $a_1 = 1$, v_2 среди всех восьми вершин степени 1 имеет наименьший индекс, создаем ребро $\{v_1, v_2\}$ (рис. слева).

Положим $\deg(v_1) = 2$ и $\deg(v_2) = 0$, поэтому восьмерка степеней имеет вид (2, 0, 1, 3, 1, 3, 1, 1).

Пример (продолжение).

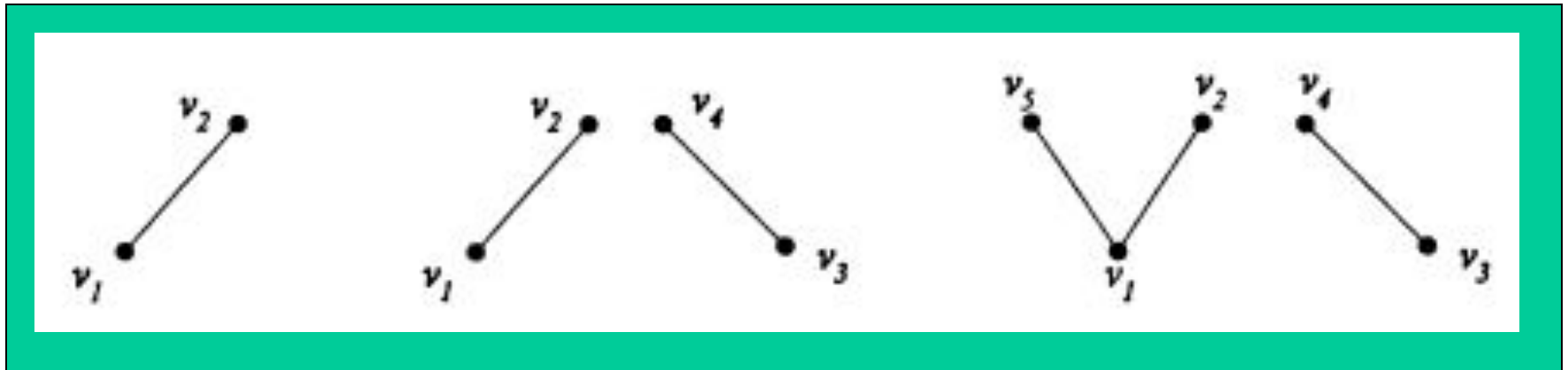
Далее считаем $a_2 = 4$, т.к. v_3 среди всех вершин степени 1 имеет наименьший индекс, создаем ребро $\{v_3, v_4\}$. Получаем граф (посередине)



Пример (продолжение).

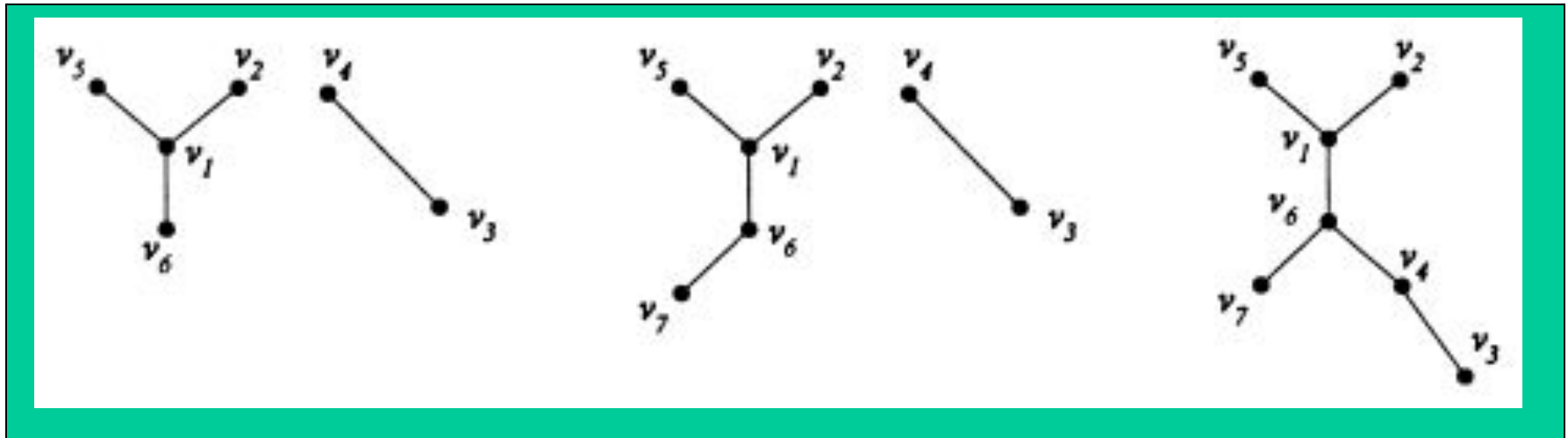
Полагаем $\deg(v_4) = 2$ и $\deg(v_3) = 0$, восьмерка степени имеет вид
(2, 0, 0, 2, 1, 3, 1, 1)

Теперь считаем $a_3 = 1$, т.к. v_5 среди всех вершин степени 1 имеет наименьший индекс, создаем ребро $\{v_1, v_5\}$. Получаем граф (слева)



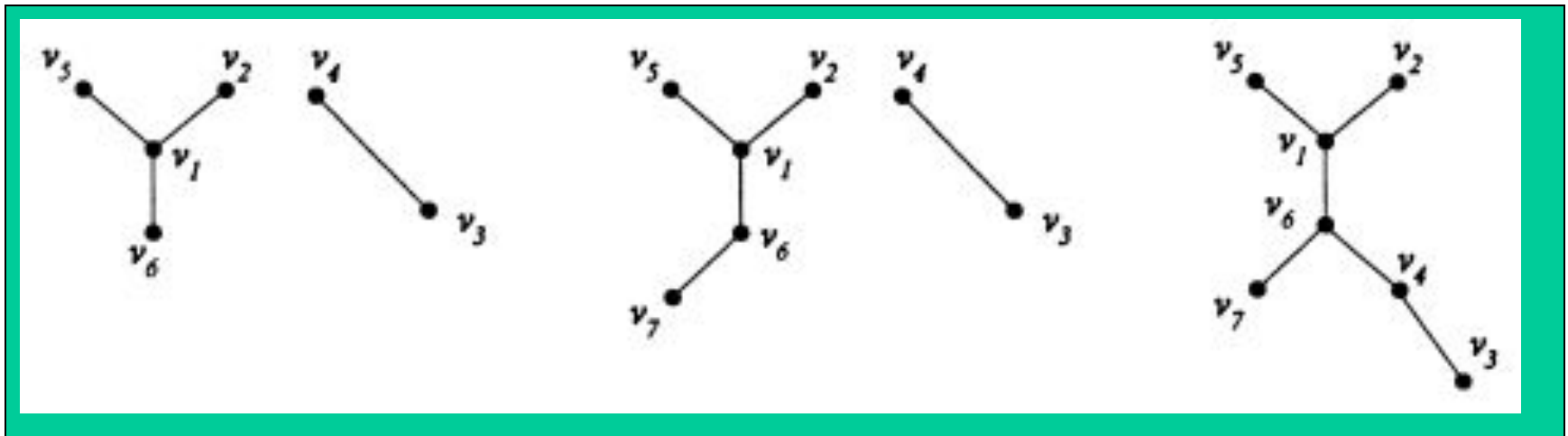
Пример (продолжение).

Полагаем $\deg(v_1) = 1$ и $\deg(v_5) = 0$, поэтому восьмерка степеней имеет вид $(1, 0, 0, 2, 0, 3, 1, 1)$. Считаем $a_4 = 6$, т.к. v_1 среди всех вершин степени 1 имеет наименьший индекс, создаем ребро $\{v_1, v_6\}$. Получаем граф (слева)



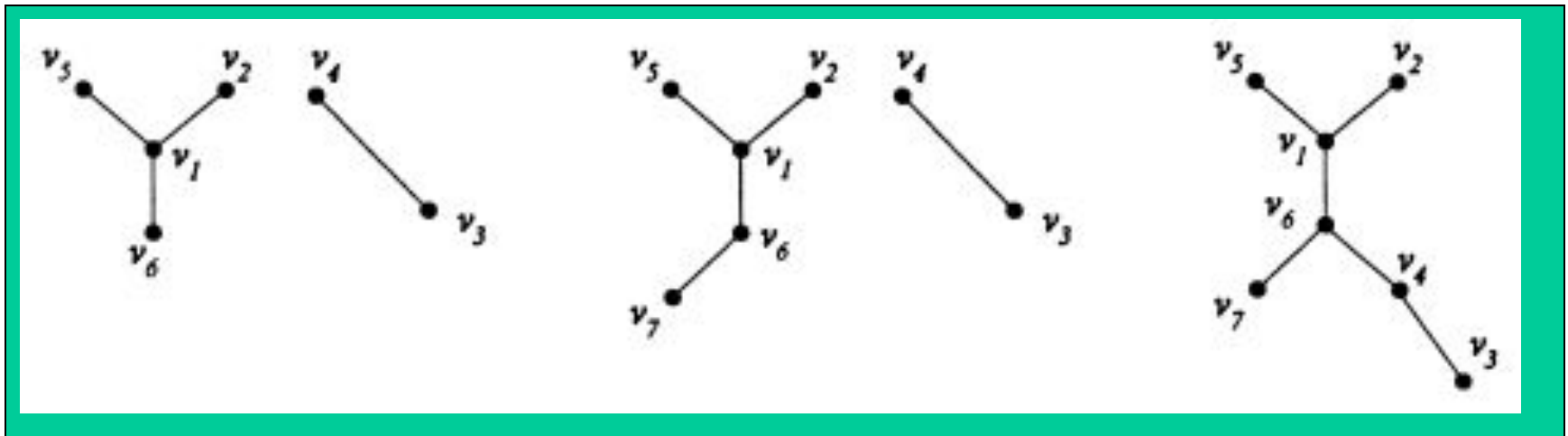
Пример (продолжение).

Полагаем $\deg(v_1) = 1$ и $\deg(v_6) = 2$, поэтому восьмерка степеней имеет вид $(0, 0, 0, 2, 0, 2, 1, 1)$. Считаем $a_5 = 6$, т.к. v_7 среди всех вершин степени 1 имеет наименьший индекс, создаем ребро $\{v_7, v_6\}$. Получаем граф (посередине)



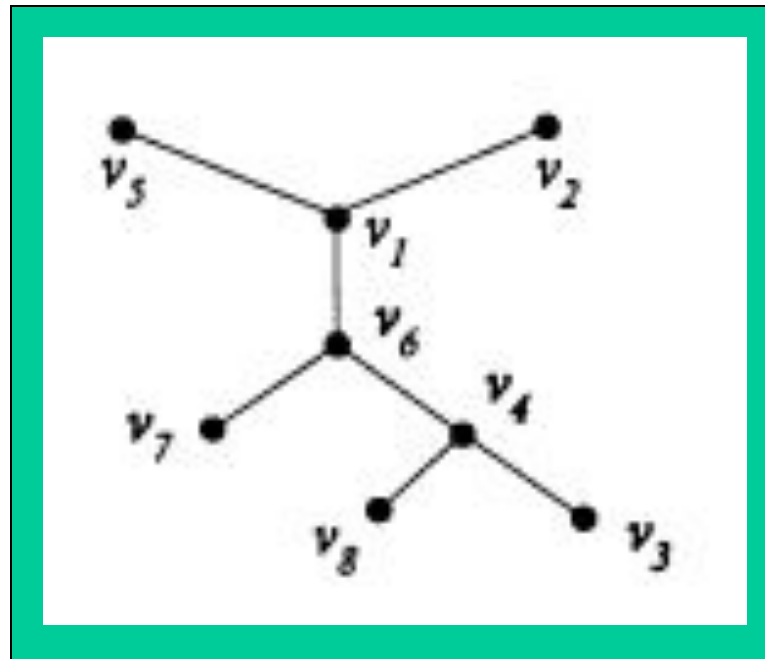
Пример (продолжение).

Полагаем $\deg(v_7) = 1$ и $\deg(v_6) = 1$, поэтому восьмерка степеней имеет вид $(0, 0, 0, 2, 0, 1, 0, 1)$. Считаем $a_6 = 4$, т.к. v_6 среди всех вершин степени 1 имеет наименьший индекс, создаем ребро $\{v_4, v_6\}$. Получаем граф (справа)



Пример (продолжение).

Полагаем $\deg(v_4) = 1$ и $\deg(v_6) = 0$, поэтому восьмерка степеней имеет вид $(0, 0, 0, 1, 0, 0, 0, 1)$. Все последовательности прочитаны и $\deg(v_4) = \deg(v_8) = 1$, формируем ребро $\{v_4, v_8\}$. Получаем искомое дерево



Определение.

Пусть G – граф с n размеченными вершинами $v_1, v_2, v_3, \dots, v_n$.
Матрицей степеней графа G называется $n \times n$ матрица D ,
определенная следующим образом:

$D_{ij} = 0$, если $i \neq j$, и D_{ii} равно степени вершины v_i .

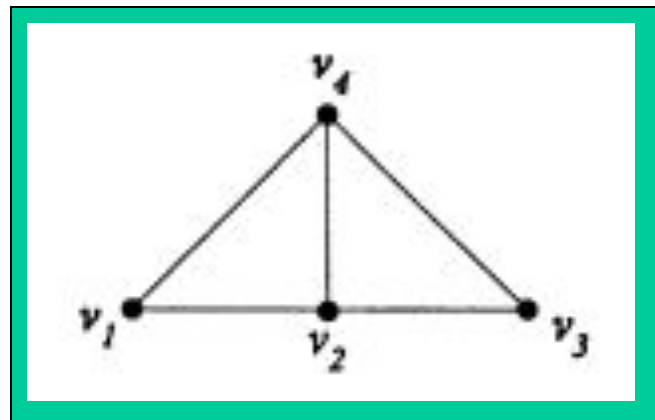
Теорема (матричная формула Кирхгофа).

Пусть G – граф с помеченными вершинами. Пусть $K = D - A$,
где A – матрица смежности графа G ,
а D – матрица степеней графа G .

Число остовных деревьев графа G равна любому из алгебраических дополнений матрицы K .

Пример.

Найти количество остовных деревьев графа



Поскольку $\deg(v_1) = \deg(v_3) = 2$ и $\deg(v_2) = \deg(v_4) = 3$,

$$C = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 3 \end{bmatrix}.$$

Пример (продолжение).

Матрица A имеет вид

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

и

$$K = C - A = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 3 \end{bmatrix} - \begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 2 & -1 & 0 & -1 \\ -1 & 3 & -1 & -1 \\ 0 & -1 & 2 & -1 \\ -1 & -1 & -1 & 3 \end{bmatrix}.$$

Пример (продолжение).

Алгебраическое дополнение K_{11}

$$\det \begin{bmatrix} 3 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 3 \end{bmatrix} = 3 \det \begin{bmatrix} 2 & -1 \\ -1 & 3 \end{bmatrix} - (-1) \det \begin{bmatrix} -1 & -1 \\ -1 & 3 \end{bmatrix} + \\ + (-1) \det \begin{bmatrix} -1 & 2 \\ -1 & -1 \end{bmatrix} = 3(5) - 4 - 3 = 8.$$

Следовательно, существует восемь остовных деревьев

Последний слайд лекции

!!