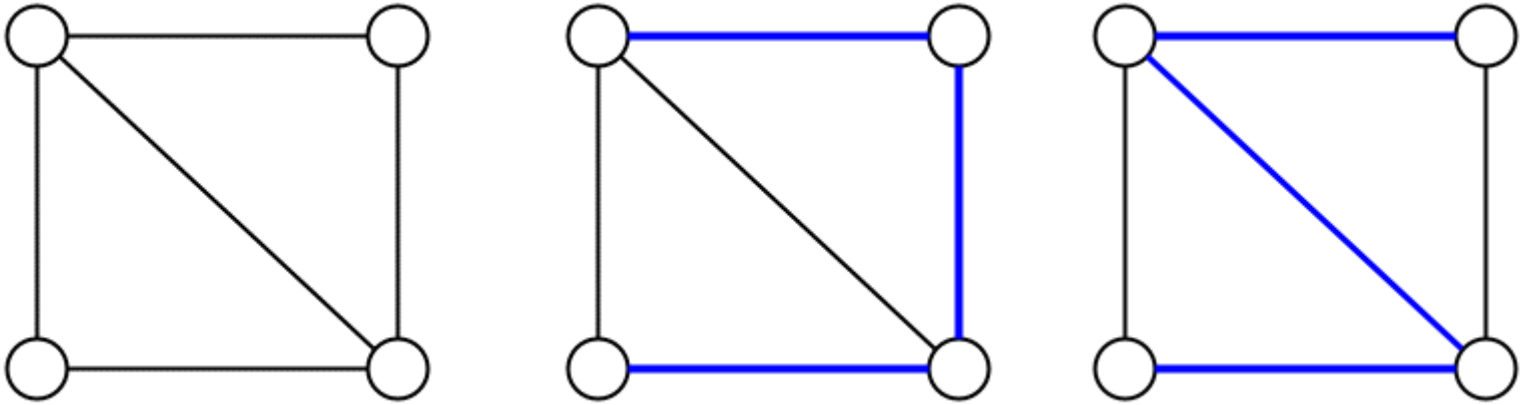
Введение в дискретную математику и математическую логику

Лекция №6
Остовные деревья. Продолжение

Апанович Зинаида Владимировна

1. Количество остовных деревьев

Количество остовных деревьев графа. Пример



Этот граф имеет 8 различных остовных деревьев = 4+4.

4 остовных дерева используют диагональное ребро и 4 остовных дерева не используют диагональное ребро.

Рекурсивная формула для вычисления количества остовных деревьев в графе

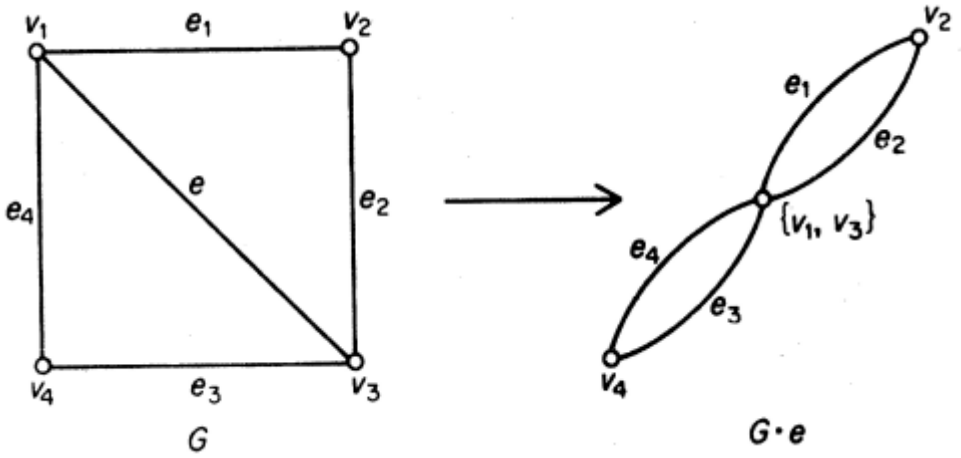
Существует простая и элегантная рекурсивная формула для определения количества остовных деревьев в графе.

Она включает в себя операцию стягивания ребра.

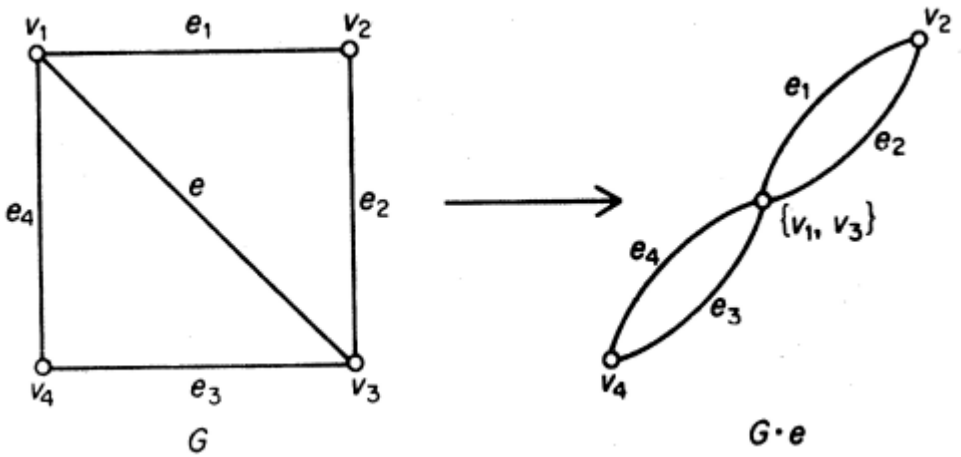
Ребро *e* графа *G* называется стянутым, если оно удалено и его концевые вершины объединены в одну вершину;

Полученный граф обозначается как $G \cdot e$.

На рисунке ниже показан результат стягивания ребра.



Рекурсивная формула для вычисления количества остовных деревьев в графе



Ясно, что если ребро e имеет различные концы в графе G , то $v(G \cdot e) = v(G) - 1$ и $e(G \cdot e) = e(G) - 1$ и число компонентов G $w(G \cdot e) = w(G)$.

Следовательно, если T — дерево, то и $T \cdot e$ — *дерево.*
Обозначим число остовных деревьев G через $T(G)$

Рекурсивная формула для вычисления количества остовных деревьев в графе

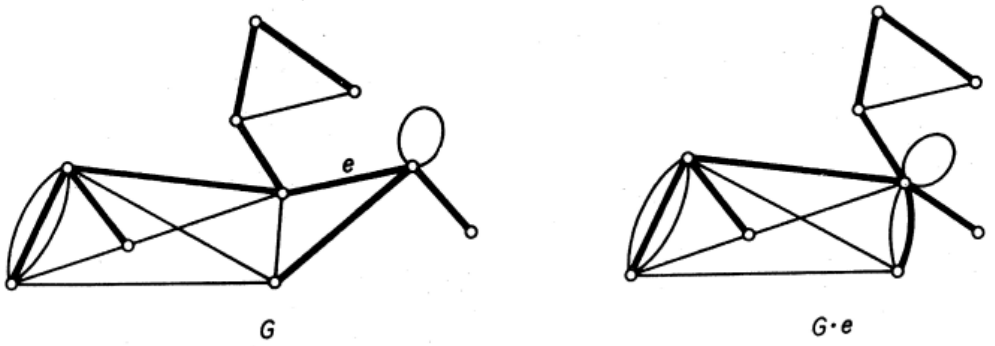
***Теорема* 1** Если ребро e имеет различные концы в G (e не является петлей),

то $T(G) = T(G - e)+T(G \cdot e)$.

Доказательство. Каждое остовное дерево графа G, которое не содержит e, также является остовным деревом графа $G - e$, и наоборот, $T(G - e)$ - это количество остовных деревьев графа G, которые не содержат ребро e. Также каждому остовному дереву T графа G, которое содержит ребро e , соответствует остовное дерево T $(G \cdot e)$.

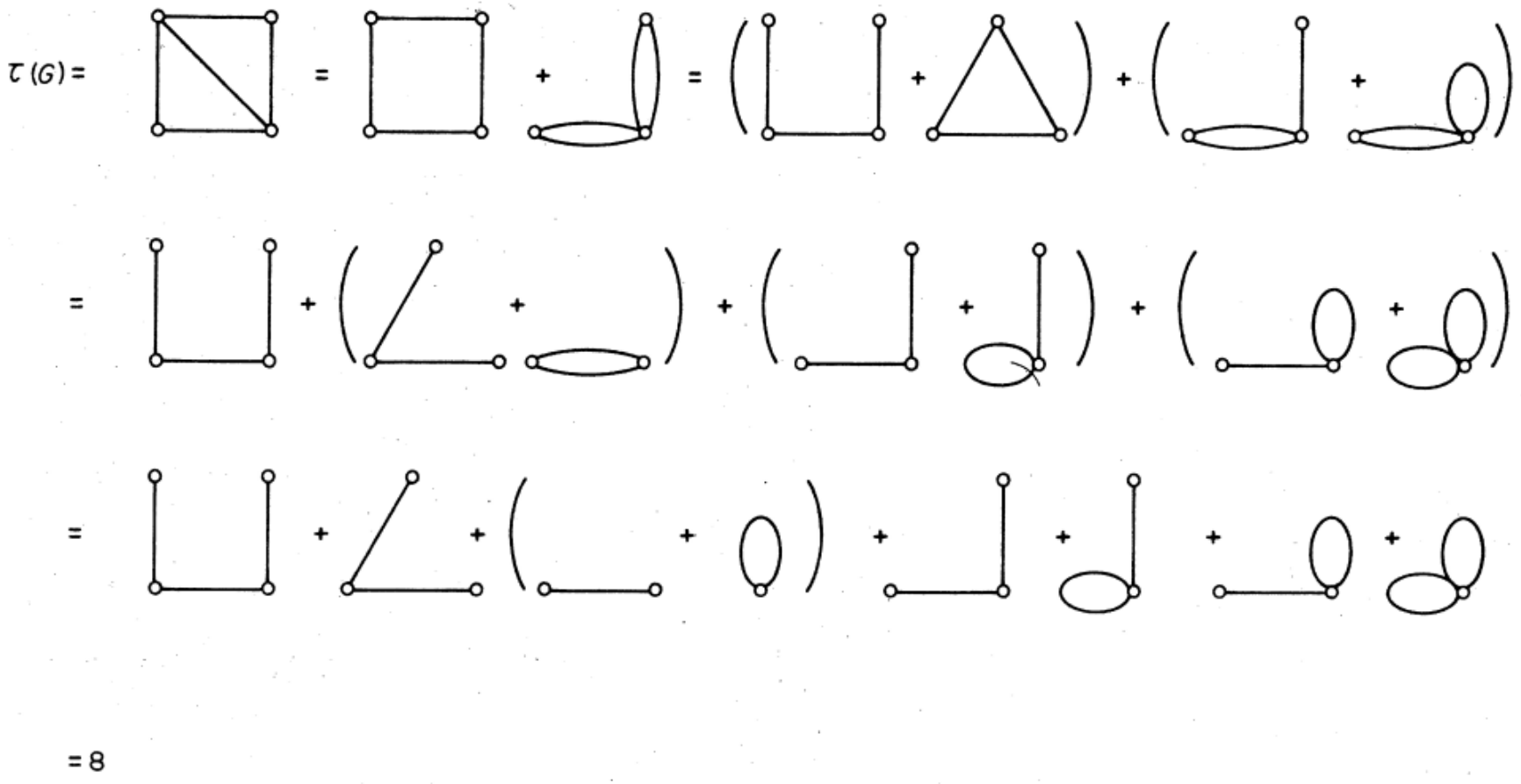
Это соответствие явно является биекцией (см. рисунок ниже).Поэтому T $(G \cdot e)$ — это в точности число остовных деревьев графа G, содержащих e.

Отсюда следует, что $T(G)=T(G-e)+T(G \cdot e)$.



Рекурсивная формула для вычисления количества остовных деревьев в графе

Рисунок ниже иллюстрирует рекурсивное вычисление $T(G)$ с помощью теоремы 1 ; количество остовных деревьев в графе символически представлено самим графом.



Количество остовных деревьев графа.

Хотя теорема 1 дает метод вычисления количества остовных деревьев в графе, этот метод не подходит для больших графов.
К счастью, существует формула для подсчета $T(G)$, которая выражает $T(G)$ при помощи определителя матрицы;

В особом случае, когда граф G является полным, простая формула для вычисления $T(G)$ была открыта Кэли (1889).

Приведенное нами доказательство принадлежит Прюферу (1918).

Количество остовных деревьев полного графа

Теорема 2 $T(K_n) = n^{n-2}$.

Доказательство Пусть множество вершин K_n равно $N = \{1, 2, \ldots, n\}$.

Отметим, что n^{n-2} — это количество последовательностей длины $n - 2$, которые можно образовать из элементов множества N.

Таким образом, для доказательства теоремы достаточно установить биекцию между множеством остовных деревьев графа K_n и множеством таких последовательностей. Каждому остовному дереву T графа K_n мы сопоставляем уникальную последовательность чисел $(t_1, t_2, .., t_{n-2})$ следующим образом.

Рассматривая N как упорядоченное множество, пусть s_1 будет первой вершиной степени 1 в T;

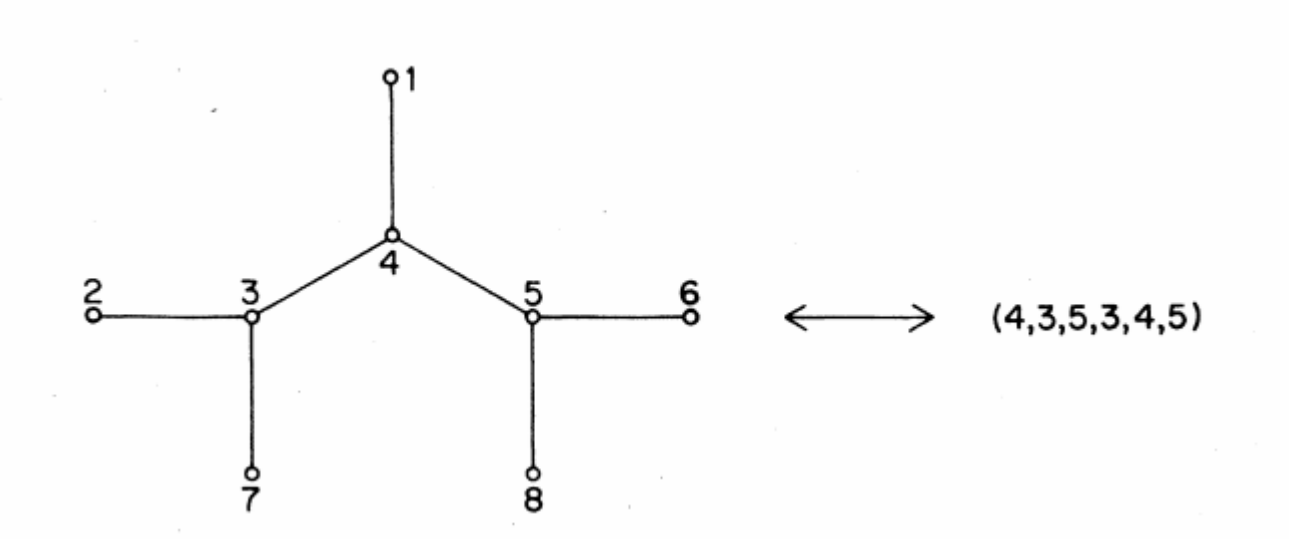
вершина, смежная с s_1, принимается за t_1.

Теперь удалим s_1 из T, обозначим через s_2 первую вершину степени 1 в $T - s_1$, и возьмем вершину, смежную с s_2, в качестве t_2.

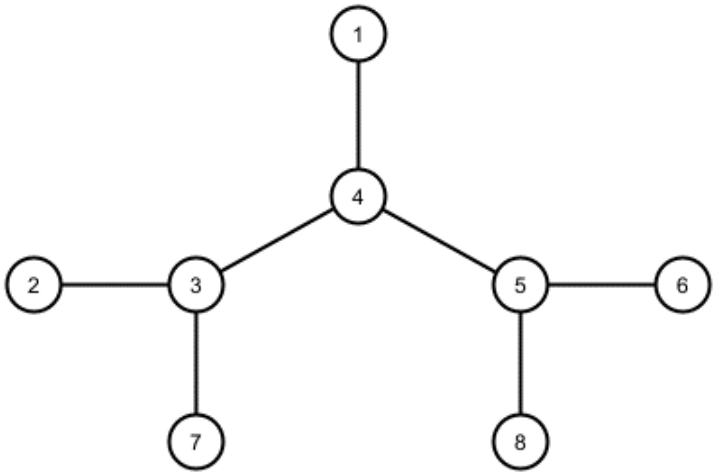
Эта операция повторяется до тех пор, пока не будет определено t_{n-2} и не останется дерево всего с 2 вершинами;

Количество остовных деревьев полного графа. Пример

Дерево на рисунке ниже соответствует последовательности (4, 3, 5, 3, 4, 5). Видно, что различные остовные деревья графа K_n определяют разные последовательности.

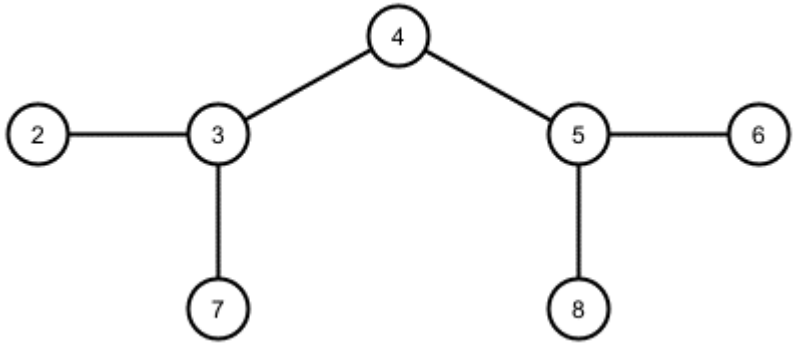


Количество остовных деревьев полного графа. Пример



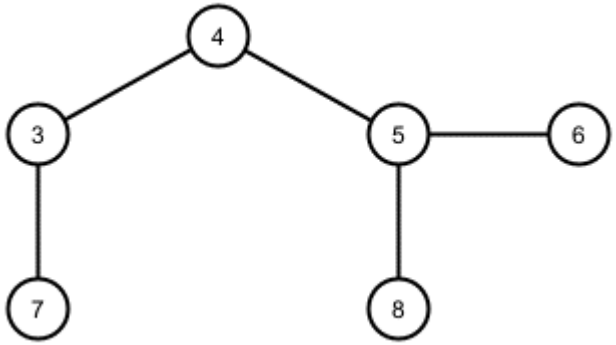
	1	2	3	4	5	6	7	8	s	t	Код Прюфера
deg(v)	1	1	3	3	3	1	1	1			
	0	1	3	2	3	1	1	1	1	4	4 ,

Количество остовных деревьев полного графа. Пример



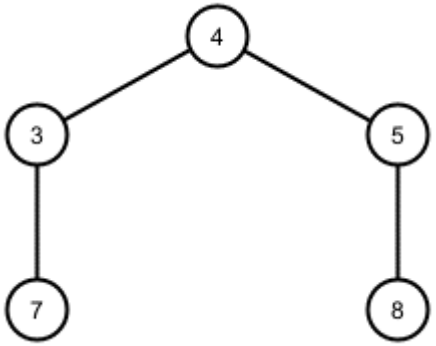
	1	2	3	4	5	6	7	8	s	t	Код Прюфера
deg(v)	1	1	3	3	3	1	1	1			
	0	1	3	2	3	1	1	1	1	4	4
	0	0	2	2	3	1	1	1	2	3	4, 3

Количество остовных деревьев полного графа. Пример



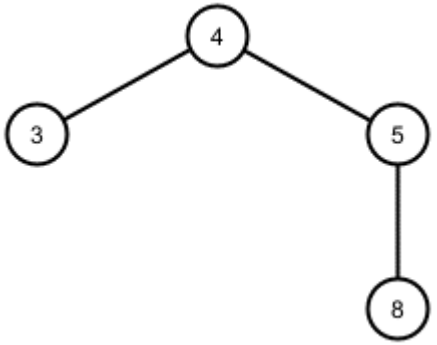
	1	2	3	4	5	6	7	8	s	t	Код Прюфера
Д(в)	1	1	3	3	3	1	1	1			
	0	1	3	2	3	1	1	1	1	4	4
	0	0	2	2	3	1	1	1	2	3	4,3
	0	0	2	2	2	0	1	1	6	5	4,3, 5

Количество остовных деревьев полного графа. Пример



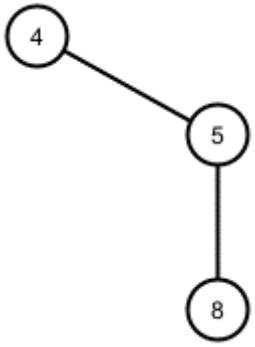
	1	2	3	4	5	6	7	8	s	t	Код Прюфера
deg(v)	1	1	3	3	3	1	1	1			
	0	1	3	2	3	1	1	1	1	4	4
	0	0	2	2	3	1	1	1	2	3	4,3
	0	0	2	2	2	0	1	1	6	5	4,3,5
	0	0	1	2	2	0	0	1	7	3	4,3,5, 3

Количество остовных деревьев полного графа. Пример



	1	2	3	4	5	6	7	8	s	t	Код Прюфера
deg(v)	1	1	3	3	3	1	1	1			
	0	1	3	2	3	1	1	1	1	4	4
	0	0	2	2	3	1	1	1	2	3	4,3
	0	0	2	2	2	0	1	1	6	5	4,3,5
	0	0	1	2	2	0	0	1	7	3	4,3,5,3
	0	0	0	1	2	0	0	1	3	4	4,3,5,3, 4

Количество остовных деревьев полного графа. Пример



	1	2	3	4	5	6	7	8	s	t	Код Прюфера
deg(v)	1	1	3	3	3	1	1	1			
	0	1	3	2	3	1	1	1	1	4	4
	0	0	2	2	3	1	1	1	2	3	4,3
	0	0	2	2	2	0	1	1	6	5	4,3,5
	0	0	1	2	2	0	0	1	7	3	4,3,5,3
	0	0	0	1	2	0	0	1	3	4	4,3,5,3,4
	0	0	0	0	1	0	0	1	4	5	4,3,5,3,4, 5

Восстановление дерева по коду Прюфера

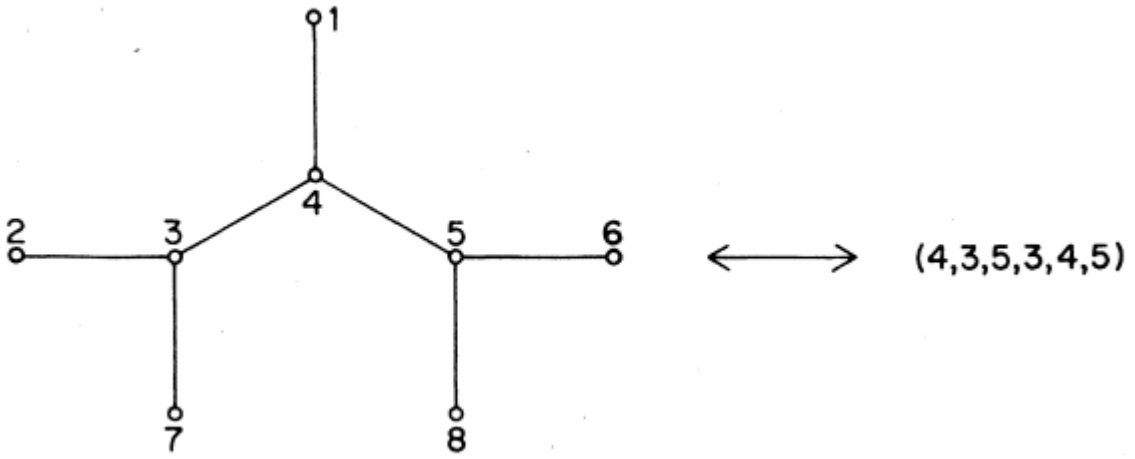
Код Прюфера : 4,3,5,3,4,5

Восстановление дерева по коду Прюфера

Обратная процедура столь же проста.

Заметим, во-первых, что любая вершина v из T встречается $d_T(v) - 1$ раз в последовательности $(t_1, t_2, \ldots, t_{n-2})$.

Таким образом, вершины степени 1 в T — это как раз те, которые не появляются в этой последовательности.



Восстановление дерева по коду Прюфера. Пример

Код Прюфера : 4,3,5,3,4,5

Следовательно, дерево имеет 8 узлов.

Отсутствующие узлы: 1, 2, 6, 7, 8

Это листья дерева.

Остальные вершины имеют следующие степени: deg(4) = 2+1, deg(3) = 2+1, deg(5) = 2+1

Последовательность степеней дерева следующая:

1	2	3	4	5	6	7	8
1	1	3	3	3	1	1	1

Восстановление дерева по коду Прюфера. Пример

Код Прюфера : 4,3,5,3,4,5

$V(T)$, которые не входят в $(t_1, t_2, \dots, t_{n-2}) = 1, 2, 6, 7, 8$

Чтобы восстановить T из $(t_1, t_2, \dots, t_{n-2})$, мы действуем следующим образом.

	1	2	3	4	5	6	7	8	s	t
Deg(v)	1	1	3	3	3	1	1	1		
	0	1	3	2	3	1	1	1	1	4

Пусть s_1 будет первой вершиной $V(G)$ не входящей в код Прюфера $(t_1, t_2, \dots, t_{n-2})$;
Соединим вершину s_1 ребром с вершиной t_1.

Восстановление дерева по коду Прюфера. Пример

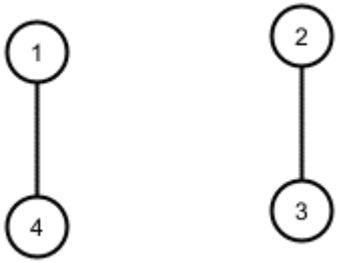
Далее, пусть s_2 будет первой вершиной $V(T) \setminus \{ s_1 \}$ не входящей в последовательность $(t_2, \dots, t_{n-2})$,

Соединим вершину s_2 с вершиной t_2.

Будем продолжать таким образом до тех пор, пока не будут определены n - 2 ребра $\{s_1, t_1\}, \{s_2, t_2\}, \dots, \{s_{n-2}, t_{n-2}\}$.

Код Прюфера : 4, 3, 5, 3, 4, 5

$V(T)$, которые не входят в $(t_1, t_2, \dots, t_{n-2})$ = 1, 2, 6, 7, 8



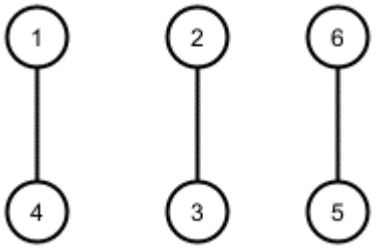
	1	2	3	4	5	6	7	8	s	t
deg(v)	1	1	3	3	3	1	1	1		
	0	1	3	2	3	1	1	1	1	4
	0	0	2	2	3	1	1	1	2	3

Восстановление дерева по коду Прюфера. Пример

Код Прюфера : 4,3, 5 , 3,4,5

$V(T)$, которые не входят в $(t_1, t_2, \dots, t_{n-2}) = 1, 2, 6, 7, 8$

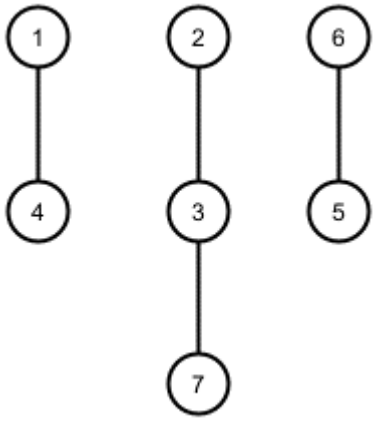
	1	2	3	4	5	6	7	8	s	t
deg(v)	1	1	3	3	3	1	1	1		
	0	1	3	2	3	1	1	1	1	4
	0	0	2	2	3	1	1	1	2	3
	0	0	2	2	2	0	1	1	6	5



Восстановление дерева по коду Прюфера. Пример

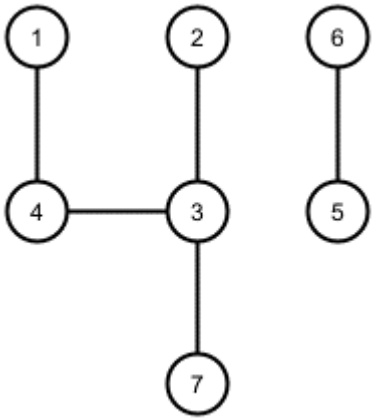
Код Прюфера : 4,3,5, 3 ,4,5

$V(T)$, которые не входят в $(t_1, t_2, \dots, t_{n-2})$ = 1, 2, 6, 7, 8



	1	2	3	4	5	6	7	8	s	t
deg(v)	1	1	3	3	3	1	1	1		
	0	1	3	2	3	1	1	1	1	4
	0	0	2	2	3	1	1	1	2	3
	0	0	2	2	2	0	1	1	6	5
	0	0	1	2	2	0	0	1	7	3

Восстановление дерева по коду Прюфера. Пример

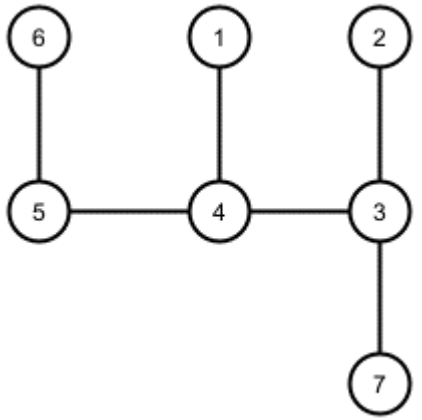


Код Прюфера : 4, 3, 5, 3, 4, 5

$V(T)$, которые не входят в $(t_1, t_2, \ldots, t_{n-2})$ = 1, 2, 6, 7, 8

	1	2	3	4	5	6	7	8	s	t	Прюфер
deg(v)	1	1	3	3	3	1	1	1			
	0	1	3	2	3	1	1	1	1	4	4
	0	0	2	2	3	1	1	1	2	3	4,3
	0	0	2	2	2	0	1	1	6	5	4,3,5
	0	0	1	2	2	0	0	1	7	3	4,3,5,3
	0	0	0	1	2	0	0	1	3	4	4,3,5,3,4

Восстановление дерева по коду Прюфера. Пример



Код Прюфера : 4,3,5,3,4, 5

$V(T)$, которые не входят в $(t_1, t_2, \dots, t_{n-2}) = 1, 2, 6, 7, 8$

	1	2	3	4	5	6	7	8	s	t	Код Прюфера
deg(v)	1	1	3	3	3	1	1	1			
	0	1	3	2	3	1	1	1	1	4	4
	0	0	2	2	3	1	1	1	2	3	4,3
	0	0	2	2	2	0	1	1	6	5	4,3,5
	0	0	1	2	2	0	0	1	7	3	4,3,5,3
	0	0	0	1	2	0	0	1	3	4	4,3,5,3,4
	0	0	0	0	1	0	0	1	4	5	4,3,5,3,4,5

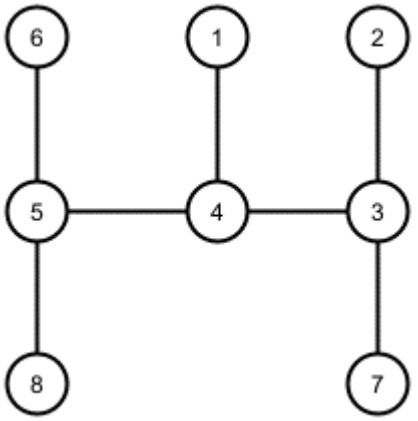
Восстановление дерева по коду Прюфера. Пример

T получается путем добавления ребра (8, 5), соединяющего 2 оставшиеся вершины

$V \setminus \{ s_1, s_2, \ldots, s_{n-2} \}$.

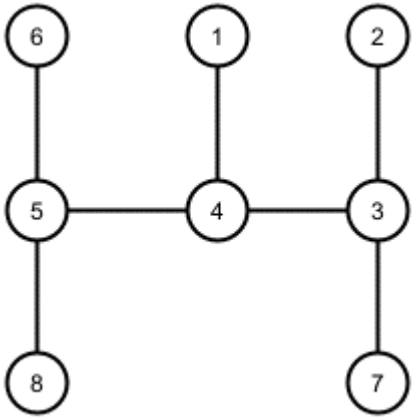
Легко проверить, что разные последовательности порождают разные остовные деревья K_n.

Таким образом, мы установили искомую биекцию.



Восстановление дерева по коду Прюфера. Пример

Последнее добавленное ребро — (8, 5)

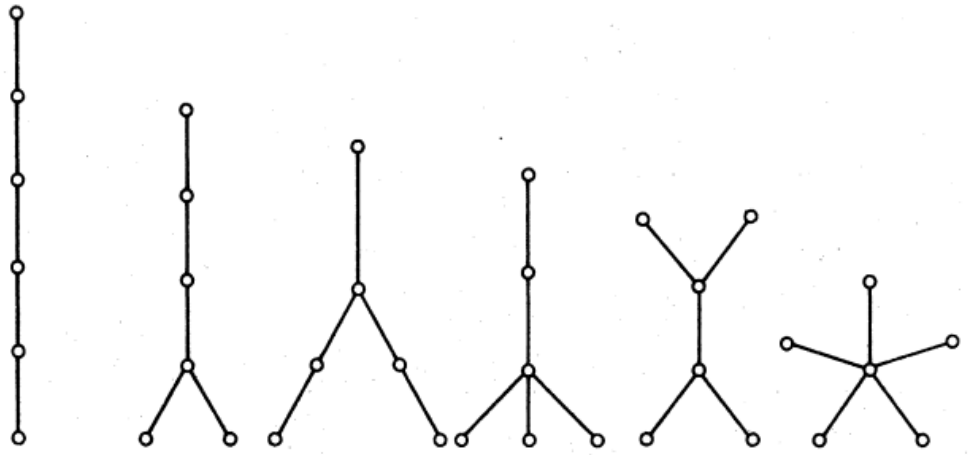


Количество остовных деревьев полного графа

Обратите внимание, что n^{n-2} — это не количество неизоморфных остовных деревьев K_n ,
а количество различных остовных деревьев K_n ;

Существует всего 6 неизоморфных остовных деревьев графа K_6 (см. рисунок ниже),

тогда как имеется $6^4 = 1296$ различных остовных деревьев графа K_6



Матрица Лапласа (Кирхгофа) для подсчета количества остовных деревьев

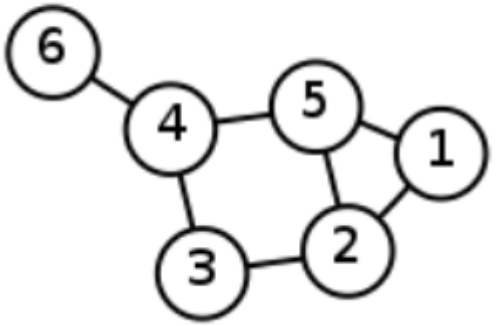
Для простого графа с вершинами его **матрица Лапласа (**матрица Кирхгофа) определяется поэлементно как

$$L_{i,j} := \begin{cases} \deg(v_i) & \text{if } i = j \\ -1 & \text{if } i \neq j \text{ and } v_i \text{ is adjacent to } v_j \\ 0 & \text{otherwise,} \end{cases}$$

или эквивалентно как матрица **L = D -A** ,
где **D** - это матрица степеней графа, **A** — матрица смежности графа.

Матрица Лапласа (Кирхгофа) для подсчета количества остовных деревьев. Пример

Labelled graph



Degree matrix

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

Adjacency matrix

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

Laplacian matrix

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

Матрица Лапласа (Кирхгофа) для подсчета количества остовных деревьев.

Каков спектр *LG?*

Мы наблюдаем, что вектор **e (**вектор*,* состоящий из всех единиц) является собственным вектором с собственным значением 0 для *LG*,

Факт 1 $\lambda_1 = 0$.

Факт 2 $\lambda_2 = 0 \Leftrightarrow$ *G не связен.*

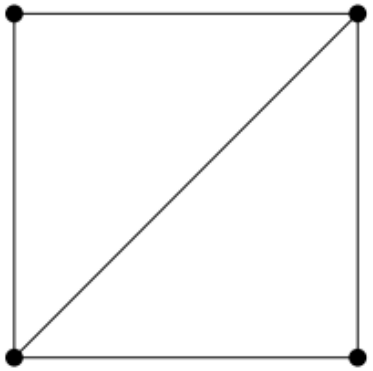
Факт 3 $\lambda_к = 0 \Leftrightarrow$ *G имеет по крайней мере k компонент связности.*

Матричная теорема о деревьях

Пусть $A[i]$ — матрица A , из которой удалены *i-я* строка и *i-ый* столбец.

Теорема 3 (Матричная теорема Кирхгофа деревьях) *Количество остовных деревьев в графе G определяется как* $\det(LG[i])$ *для любого* i .

Матричная теорема о деревьях. Пример



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

$$L^* = \begin{pmatrix} 3 & -1 & -1 \\ -1 & 2 & 0 \\ -1 & 0 & 2 \end{pmatrix},$$

Det(L^*) = 8.

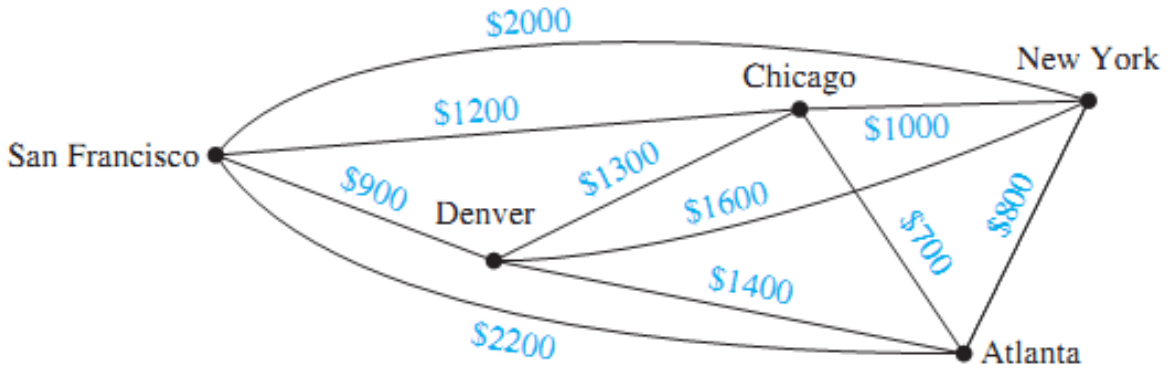
2.Минимальные остовные деревья

Введение

Компания планирует построить сеть связи, соединяющую пять ее компьютерных центров.

Любую пару этих центров можно соединить арендованной телефонной линией.

Какие связи следует создать, чтобы обеспечить наличие пути между любыми двумя компьютерными центрами и минимизировать общую стоимость сети?



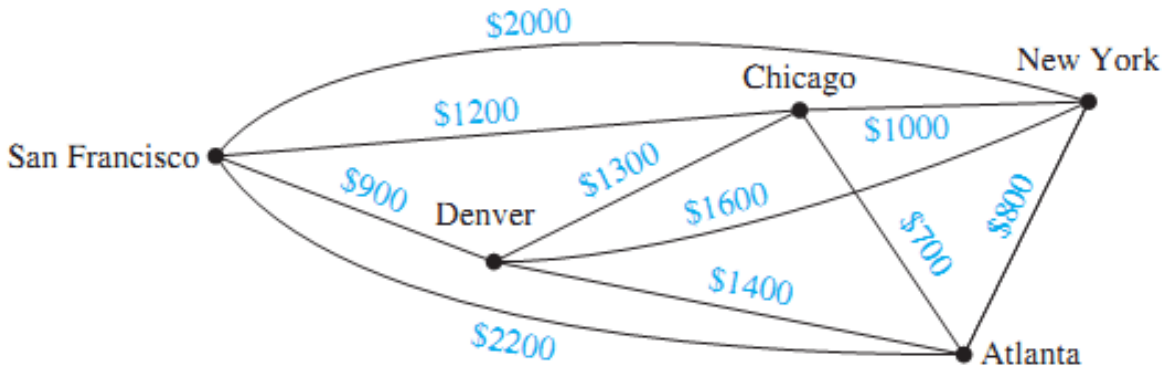
Мы можем смоделировать эту проблему, используя взвешенный граф, показанный ниже,

где

Вершины представляют собой компьютерные центры ,

ребра представляют собой возможные выделенные линии,

веса на ребрах — это ежемесячные ставки аренды линий, представленных ребрами.



Взвешенный граф, показывающий ежемесячную стоимость аренды линий в компьютерной сети.

Эту проблему можно решить, найдя остовное дерево так , чтобы сумма весов ребер дерева была минимальной.

Такое остовное дерево называется минимальным остовным деревом

(наименьшим остовным деревом) .

Алгоритмы для минимальных остовных деревьев

ОПРЕДЕЛЕНИЕ 1 Минимальное остовное дерево (наименьшее остовное дерево) в связном взвешенном графе — это остовное дерево, имеющее наименьшую возможную сумму весов ребер.

Рассмотрим два алгоритма построения минимальных остовных деревьев.

Оба алгоритма работают последовательно, добавляя к дереву ребра наименьшего веса с указанным свойством, которые еще не были использованы .

Оба алгоритма являются жадными .

(Жадный алгоритм — это процедура, которая делает оптимальный выбор на каждом из своих шагов.)

Оптимизация на каждом этапе не гарантирует, что будет получено оптимальное общее решение.

Однако два рассматриваемых алгоритма для построения минимальных остовных деревьев, являются жадными алгоритмами, которые действительно выдают оптимальные решения.

Первый рассматриваемый алгоритм, был первоначально открыт чешским математиком Войцехом Ярником в 1930 году.

Алгоритм стал широко известен после того, как в 1957 году его заново открыл Роберт Прим.

По этой причине он известен как алгоритм Прима (иногда как алгоритм Прима- Ярника).

Начните с выбора любого ребра с наименьшим весом и поместите его в остовное дерево.

Последовательно добавляйте к дереву ребра минимального веса, инцидентные вершине, уже имеющейся в дереве, никогда не образуя простой цикл с ребрами, уже имеющимися в дереве.

Остановитесь, когда будет добавлено $n - 1$ ребро.

АЛГОРИТМ 1 Алгоритм Прима.

АЛГОРИТМ 1 Алгоритм Прима.

процедура *Prim* (*G* : взвешенный связный неориентированный граф с *n* вершинами)

T := ребро минимального веса

для *i* := 1 до *n* − 2

e := ребро минимального веса, инцидентное вершине в *T* и не образующее простого цикла в *T* добавляется к *T*

$T := T \cup e$

вернуть *T* { *T* — минимальное остовное дерево *G* }

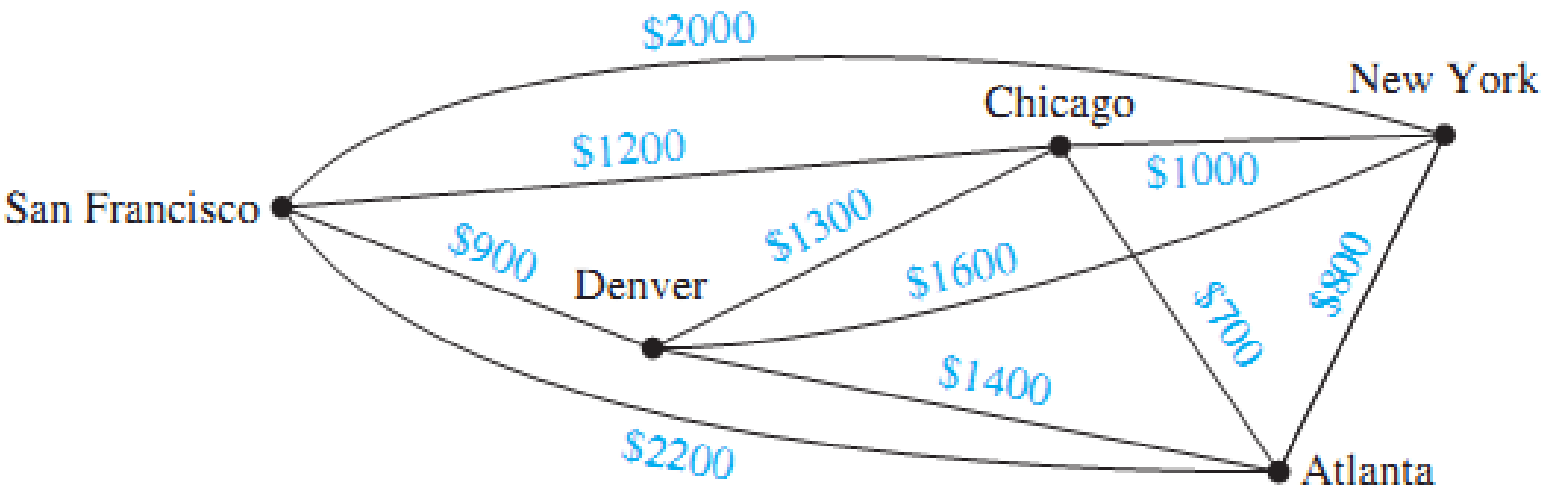
Обратите внимание, что выбор ребра для добавления на этапе алгоритма не являетсядетерминированным, когда есть более одного ребра с одним и тем же весом, которые удовлетворяют заданнымм критериям.

Чтобы сделать выбор детерминированным ребра должны быть упорядочены.

Также обратите внимание, что для заданногосвязного взвешенного простого графа может существовать более одного минимального остовного дерева.

ПРИМЕР 1

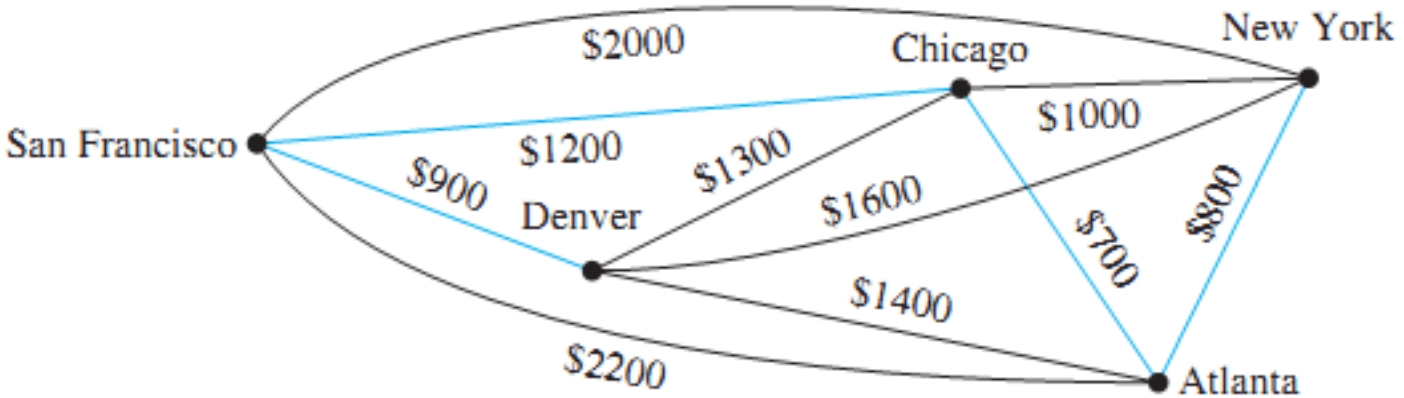
Используйте алгоритм Прима для проектирования сети связи с минимальной стоимостью, соединяющей все компьютеры, представленные на графе ниже.



Решение : Мы решаем эту проблему, находя минимальное остовное дерево в графе.

Алгоритм Прима реализуется путем выбора начального ребра минимального веса и последовательного добавления ребер минимального веса, инцидентных вершине дерева и не образующих простых циклов.

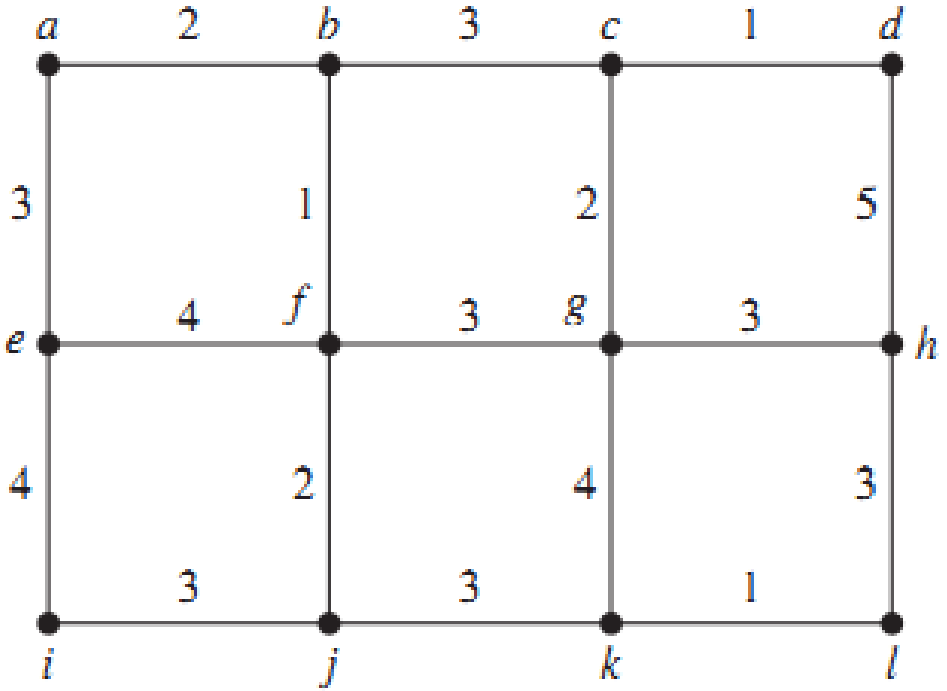
Синие ребра показывают минимальное остовное дерево, созданное алгоритмом Прима, с отображением выбора, сделанного на каждом шаге.



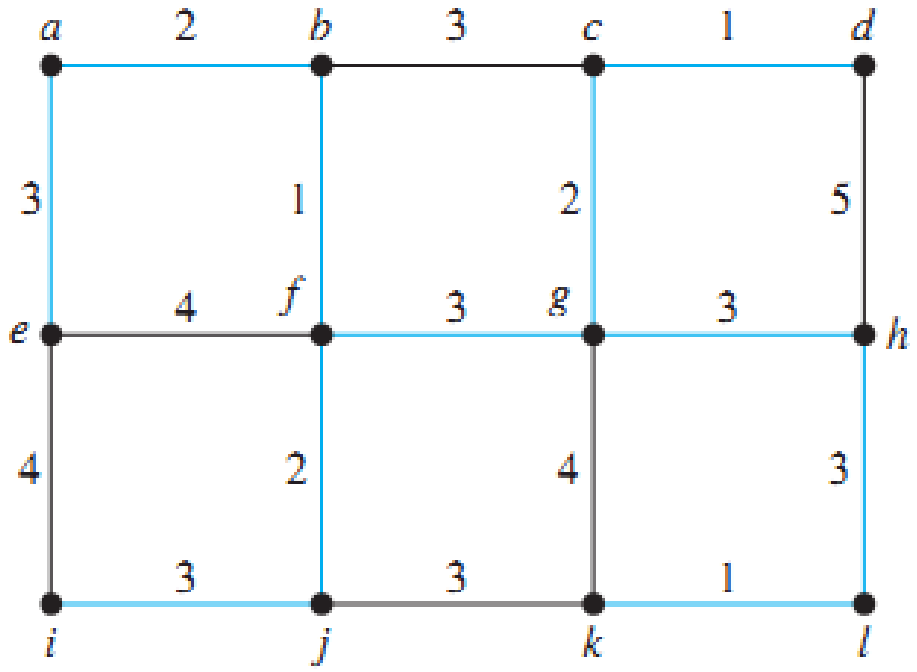
Choice	Edge	Cost
1	{Chicago, Atlanta}	$ 700
2	{Atlanta, New York}	$ 800
3	{Chicago, San Francisco}	$ 1200
4	{San Francisco, Denver}	$ 900
	Total:	$ 3600

ПРИМЕР 2

Используйте алгоритм Прима, чтобы найти минимальное остовное дерево в графе, показанном ниже.



Решение: Ниже показано минимальное остовное дерево, построенное с использованием алгоритма Прима. Отображаются последовательно выбранные ребра.



(a)

Choice	Edge	Weight
1	$\{b, f\}$	1
2	$\{a, b\}$	2
3	$\{f, j\}$	2
4	$\{a, e\}$	3
5	$\{i, j\}$	3
6	$\{f, g\}$	3
7	$\{c, g\}$	2
8	$\{c, d\}$	1
9	$\{g, h\}$	3
10	$\{h, l\}$	3
11	$\{k, l\}$	1
	Total:	24

(b)

Второй алгоритм, который будет рассмотрен, был открыт Джозефом Краскалом в 1956 году, хотя основные идеи, которые он использует, были описаны гораздо раньше.

Для выполнения алгоритма Краскала выберите ребро с минимальным весом.

Последовательно добавляйте ребра с минимальным весом, которые не образуют простого цикла с уже выбранными ребрами.

Остановитесь после того, как будет выбрано $n - 1$ ребро.

Доказательство того, что алгоритм Крускала создает минимальное остовное дерево для каждого связного взвешенного графа, оставлено в качестве упражнения.

АЛГОРИТМ 2 Алгоритм

АЛГОРИТМ 2 Алгоритм *Крускала* .

процедура Краскал (*G* : взвешенный связный неориентированный граф с *n* вершинами)

T := пустой граф

для *я* := 1 **к** *н* − 1

e := любое ребро в *G* с наименьшим весом, которое не образует простого цикла

при добавлении к *T*

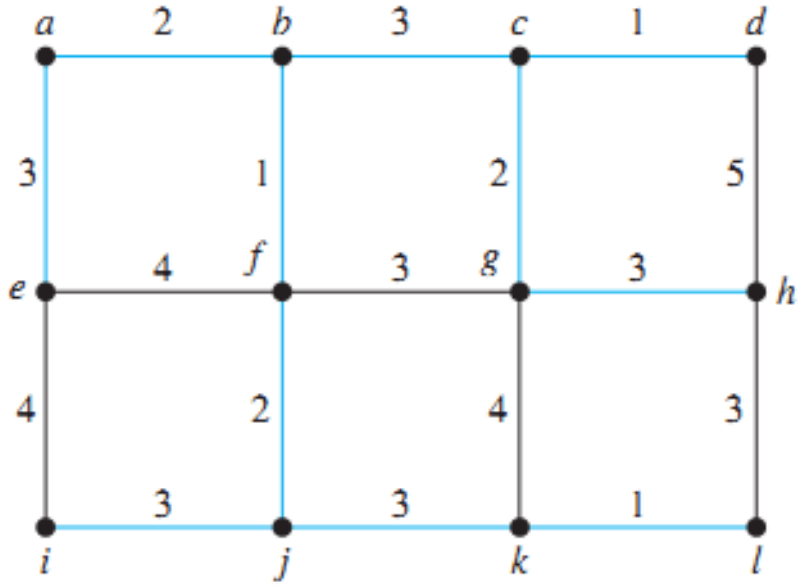
$T := T \cup e$

вернуть *T* { *T* — минимальное остовное дерево *G* }

ПРИМЕР

Используйте алгоритм Крускала, чтобы найти минимальное остовное дерево во взвешенном графе G , показанном ниже.

Решение: показано



(a)

Choice	Edge	Weight
1	{c, d}	1
2	{k, l}	1
3	{b, f}	1
4	{c, g}	2
5	{a, b}	2
6	{f, j}	2
7	{b, c}	3
8	{j, k}	3
9	{g, h}	3
10	{i, j}	3
11	{a, e}	3
	Total:	24

(b)

Обратите внимание на разницу между алгоритмами Прима и Краскала .

В алгоритме Прима выбираются ребра минимального веса, инцидентные вершине, уже имеющейся в дереве, и не образующие цикла;

тогда как в алгоритме Крускала выбираются ребра минимального веса, которые

не обязательно инцидентны вершине, уже находящейся в дереве , и не образуют контур.

Обратите внимание, что, как и в алгоритме Прима, если ребра не упорядочены, выбор ребра для добавления не является детерминированным.

Следовательно, чтобы процедура была детерминированной, ребра необходимо упорядочить.

Теорема Алгоритм Прима создает минимальное остовное дерево связного взвешенного графа.

Доказательство: Пусть G — связный взвешенный граф.

Предположим, что последовательными ребрами, выбранными алгоритмом Прима, являются e_1, e_2, ..., e_{n-1}.

Пусть S — дерево с ребрами e_1, e_2, ..., e_{n-1}, а S_k — дерево с ребрами e_1, e_2, ..., e_k.

Пусть T — минимальное остовное дерево графа G, содержащее ребра $e_1, e_2, \ldots, e_k$, где k — максимальное целое число со свойством, что существует минимальное остовное дерево, содержащее первые k ребер, выбранных алгоритмом Прима.

Теорема верна, если мы можем показать, что $S = T$.

Предположим, что $S \neq T$, так что $k < n - 1$.

Следовательно, T содержит ребра e_1, e_2 , ..., e_k, но не e_{k+1} .

Рассмотрим граф, состоящий из T вместе с ребром e_{k+1} .

Поскольку этот граф связен и имеет n ребер, в нем слишком много ребер, чтобы он был деревом.

Значит, он должен содержать простой цикл.

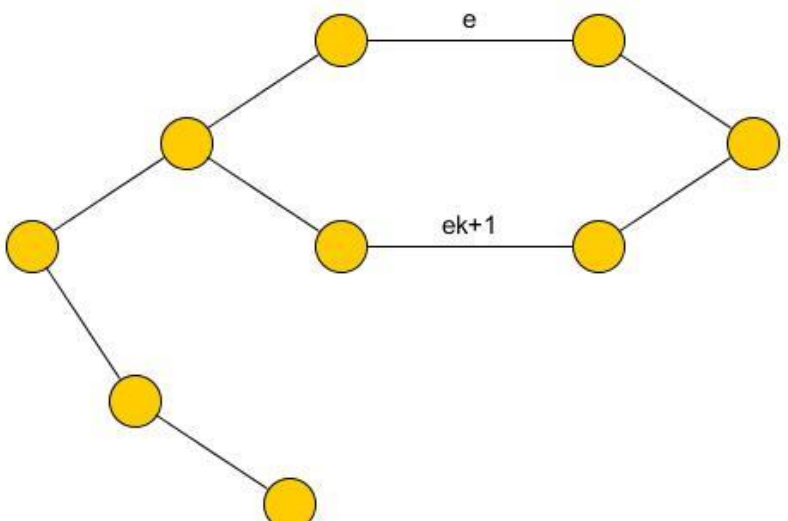
Этот простой цикл должен содержать ребро $e_{го+1}$ потому что в T не было простого цикла.

Кроме того, в простом цикле должно быть ребро, которое не принадлежит

$S_{k+1,}$ поскольку S_{k+1} является деревом .

Начиная с концевой вершины ребра e_{k+1} , которая также является концевой вершиной одного из ребер e_1, ..., e_k, и следуя по циклу до тех пор, пока он не достигнет ребра, не принадлежащего S_{k+1} , мы можем найти ребро e,

не принадлежащее S_{k+1} которое в качестве концевой вершины имеет вершину, инцидентную также одному из ребер e_1, e_2 , ..., $e_{k.}$



Удаляя из T ребро e и добавляя вместо него e_{k+1} , мы получаем дерево T' с $n - 1$ ребрами (оно является деревом, поскольку не имеет простых циклов).

Обратите внимание, что дерево T' содержит e_1 , e_2 , ..., e_k , e_{k+1} .

Более того, поскольку ребро e_{k+1} было выбрано алгоритмом Прима на

(k + 1)-*ом* шаге, и ребро e также было доступно на этом шаге, вес ребра

e_{k+1} не больше чем вес e .

Из этого наблюдения следует, что *T' также является минимальным остовным деревом*, поскольку сумма весов его ребер не превышает сумму весов ребер дерева T.

Это противоречит выбору k как максимального целого числа, такого, что существует минимальное остовное дерево, содержащее e_1 , ..., e_k .

Следовательно, $k = n - 1$ и $S = T$.

Отсюда следует, что алгоритм Прима создает минимальное остовное дерево.

Можно показать, что для нахождения минимального остовного дерева графа с m ребрами и n вершинами алгоритм Краскала может быть выполнен с использованием $O (m \log m)$ операций, а алгоритм Прима может быть выполнен с использованием $O (m \log n)$ операции.

Следовательно, предпочтительнее использовать алгоритм Краскала для разреженных графов , то есть для графов, где m очень мало по сравнению с $n(n - 1)/2$, общим числом возможных ребер в неориентированном графе с n вершинами.

В остальном сложность этих двух алгоритмов практически не отличается.

Ваши вопросы?

Контакты лектора:
apanovich_09@mail.ru