ТЕМА 3.

Введение в теорию графов

1. НЕКОТОРЫЕ ОПРЕДЕЛЕНИЯ И ПОНЯТИЯ

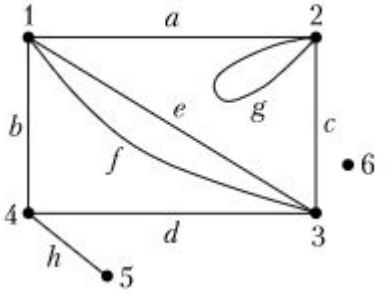
Графом (G) будем называть тройку объектов ($V, X, \theta(x)$)

V – множество n вершин.
X – конечное множество m ребер.
θ - функция инцидентности, которая каждому элементу множества X ставит в соответствие пару элементов из множества V.

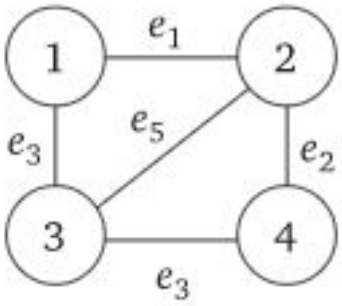
θ задана на множестве X.

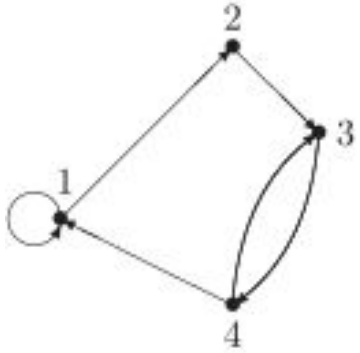
v_i – начало ребра,
v_k – конец ребра.

$\theta(x_i) = (v_i, v_k)$ – ребро инцидентно вершинам v_i и v_k.



Вершины, инцидентные одному и тому же ребру, называются **смежными.** Так же **смежными** называются ребра, инцидентные одной и той же вершине.





Если в значении функции инцидентности допускается перестановка вершин, то граф называется неориентированным. В противном случае граф называется ориентированным (Орграф).

Если одной и той же паре вершин инцидентно несколько ребер, то ребра называются кратными.

Если на некотором ребре x_0 $\theta(x_0) = (v_{j0}, v_{i0})$, то ребро называется петлей.

2. Способы задания графов

1. Аналитический.

Если вершине не инцидентно никакое ребро, то эта вершина называется **изолированной**.

Выписываются все ребра и пишутся напротив две пары вершин, которым они инцидентны.

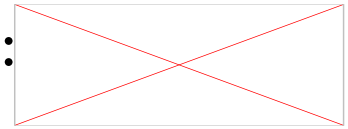
В конце выписываются все изолированные вершины.

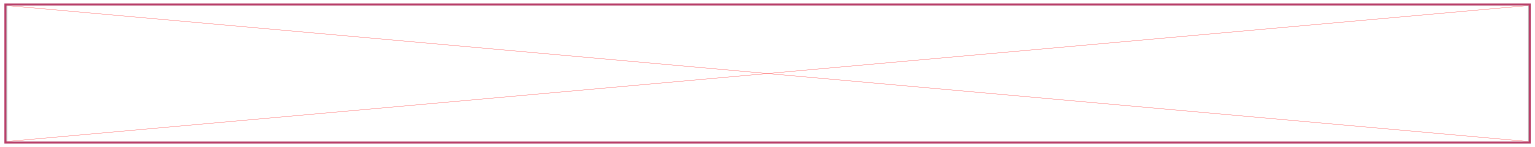
2. Геометрический.

Каждая вершина графа задается точкой. А ребра, инцидентные паре вершин – кривой.

Желательно рисовать кривые без пересечения. Если пересечения существуют, то их надо отличать от вершин.

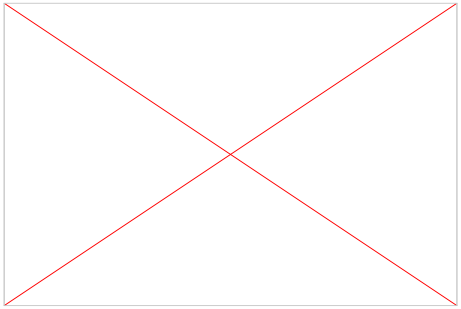
3. С помощью матрицы смежности.

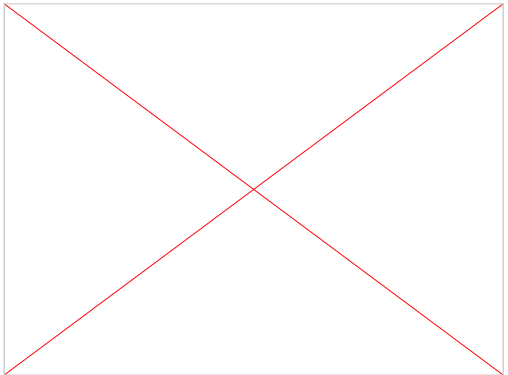
Задается одинаково для всех графов: 



Если граф не ориентирован, то матрица симметрична.

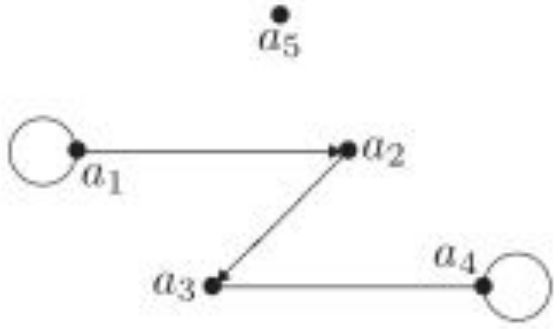
Пример.

На рисунке изображен граф: 

Его матрица смежности: 

ПРАКТИЧЕСКАЯ ЧАСТЬ

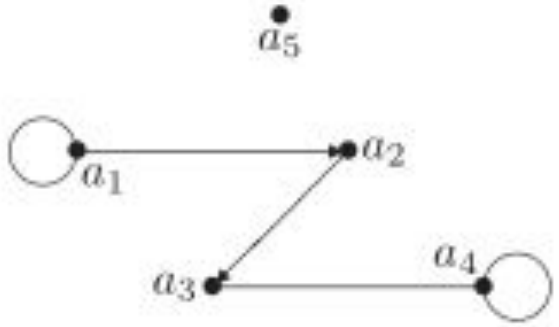
Записать матрицу смежности графа:



Решение.

ПРАКТИЧЕСКАЯ ЧАСТЬ

Записать матрицу смежности графа:



Решение.

Матрица смежности

$$B_{5\times5} = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

4. С помощью матрицы инцидентности.

$$A_{|V|\times|X|}$$

Для неориентированных графов:

$$a_{ij} = \begin{cases} \text{вес ребра } x_j, \text{если ребро } x_i \text{ инцидентно вершине } v_i, \\ 0 \text{ в противном случае} \end{cases}$$

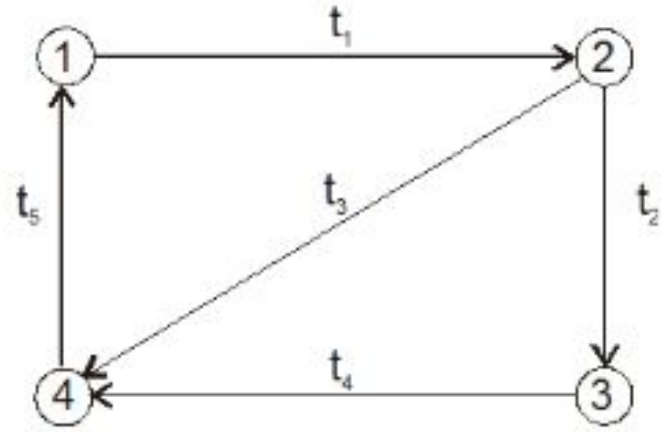
Для орграфов:

$$a_{ij} = \begin{cases} 1, \text{если } v_i - \text{конец ребра } x, \\ -1, \text{если } v_i - \text{начало ребра } x_j, \\ 0 \text{ в противном случае.} \end{cases}$$

Для петель нужны дополнительные предположения.

ПРАКТИЧЕСКАЯ ЧАСТЬ

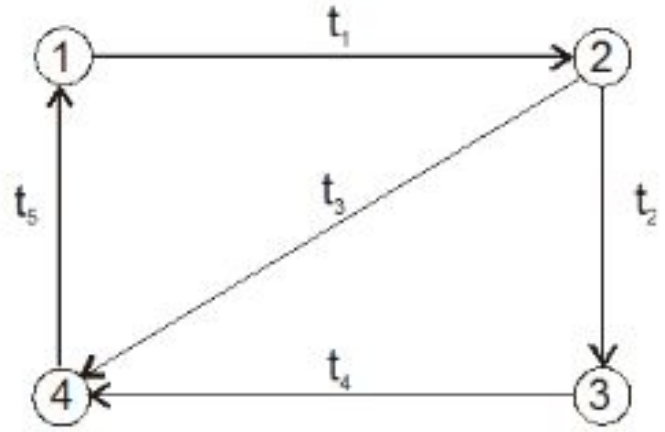
Записать матрицу инцидентности для орграфа на рисунке:



Решение.

ПРАКТИЧЕСКАЯ ЧАСТЬ

Записать матрицу инцидентности для орграфа на рисунке:



Решение.

Матрица инцидентности:

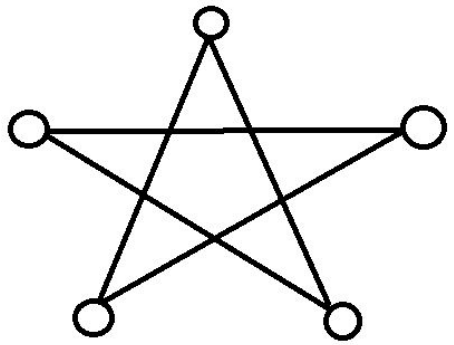
	t1	t2	t3	t4	t5
1	-1	0	0	0	1
2	1	-1	-1	0	0
3	0	1	0	-1	0
4	0	0	1	1	-1

Граф, в котором нет кратных ребер и петель, называется **простым**.

Простой граф называется **полным**, если любой паре его вершин инцидентно одно ребро.

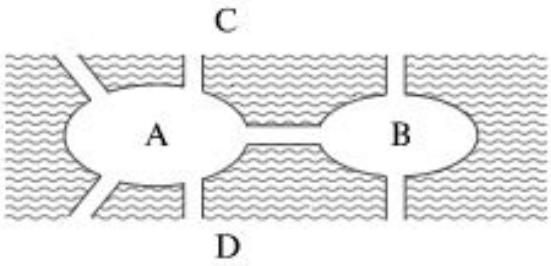
Пример.

Полный граф с пятью вершинами:



3. ЭЙЛЕРОВЫ ГРАФЫ

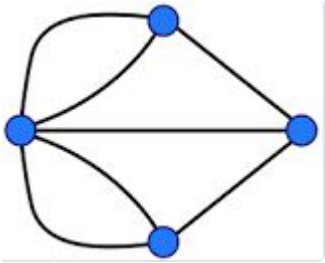
Задача о Кёнигсбергских мостах:



Можно ли так прогуляться по городу, чтобы пройти по каждому мосту ровно 1 раз?

В терминах теории графов:

Дан граф. Требуется найти в нем маршрут, проходящий по каждому ребру ровно один раз. Начало и конец – в одной вершине.

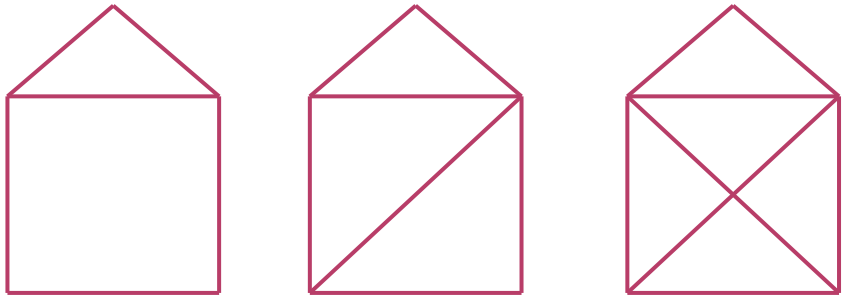


Такой маршрут называется **эйлеровым циклом**, а граф, в котором он существует, называется **эйлеровым графом.**

Эйлерова цепь — это цепь, содержащая все ребра графа.
Эйлеров цикл (ЭЦ) — это есть цикл, содержащий все ребра графа.

Проверь себя.

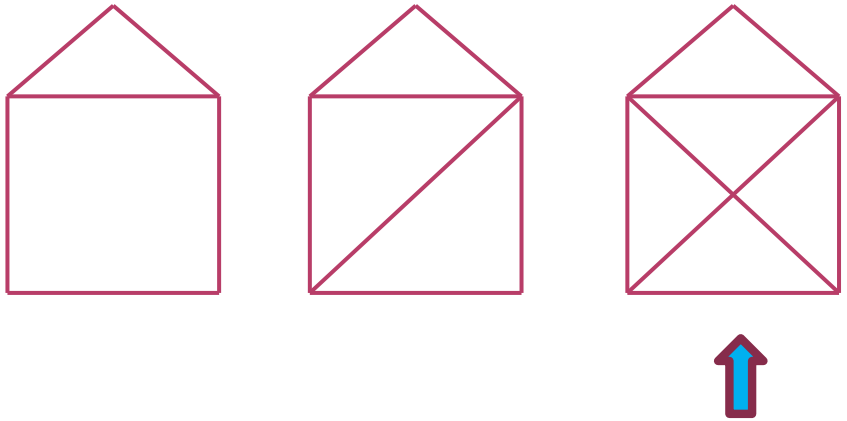
В каких графах на рисунке ниже есть эйлеров цикл?



Эйлерова цепь — это цепь, содержащая все ребра графа.
Эйлеров цикл (ЭЦ) — это есть цикл, содержащий все ребра графа.

Проверь себя.

В каких графах на рисунке ниже есть эйлеров цикл?



Степень вершины в графе – это число ребер, инцидентных этой вершине.

Критерий эйлеровости графа.

Для того, чтобы связный граф без петель был Эйлеровым, необходимо и достаточно, чтобы степень его вершины была четным числом.

3. МАРШРУТЫ, ЦИКЛЫ, СВЯЗНОСТИ.

Маршрут в графе $G=(V,E)$ — это последовательность $v_0 e_1 v_1 e_2 v_2,\ldots,e_n v_n$, $e_i \in E$, $v_i \in V$ такая, что $\forall i \in \{1,\ldots,n\}$ $e_i = \{v_{i-1}, v_i\}$, v_0 — начало маршрута, v_n — конец маршрута, n — длина маршрута.

Маршрут с началом u и концом v называется (u, v)-маршрутом.

Цепь — маршрут без повторяющихся ребер.

Простой маршрут — это маршрут, в котором вершины $v_0, v_1, \ldots, v_{n-1}$ попарно различны и $v_n \notin \{v_1, v_2, \ldots, v_{n-1}\}$.

Маршрут называется *циклическим,* если $v_0 = v_n$.

Циклическая цепь — это цикл.

$G=(V,E)$ — граф, $u, v \in V$.

$u \sim v$ (и связана с V) $\Leftrightarrow$ в G есть (u, v)-маршрут.

$\sim \subseteq V \times V$.

Дерево — связный граф без циклов.

Лес — граф без циклов.

4. ЗАДАЧА НАХОЖДЕНИЯ МИНИМАЛЬНОГО ОСТОВА.

Остовной подграф графа $G=(V,E)$ — это есть граф $H=(V,F)$, $F \subseteq E$, $k(H)=k(G)$.

Остов (каркас) графа G — есть остовной подграф с минимальным числом ребер.

Замечание

Каждая компонента остова — дерево. Иначе, удалив ребро в цикле, уменьшим число ребер, не изменив числа компонент.

Обратно: Если в остовном подграфе каждая компонента — дерево, то удаление любого ребра увеличивает число компонент связности, то есть то, что остается, — уже не остовной подграф.

Следствие 2

Остовной подграф, у которого каждая компонента — дерево, является остовом.

5. І-Я ЗАДАЧА НАХОЖДЕНИЯ МИНИМАЛЬНОГО ПУТИ В ГРАФЕ.

6. II-Я ЗАДАЧА НАХОЖДЕНИЯ МИНИМАЛЬНОГО ПУТИ В ГРАФЕ.

Алгоритм Дейкстры.

Вход: $G = (V, E)$, $\omega : E \to R^+$, $u \in V$.

Выход: $\forall v \in V$, $\rho(u, v)$.

Требуется две функции: $d : V \to R^+$ (функция текущего расстояния)

и $e : V \to V$ (частичная функция текущего предшественника).

$S \subseteq V$ (список найденных, но не обработанных вершин).

$v \to S$ (вершину *v* поместить в список S).

$v \leftarrow S$ (из S вынимается вершина с наименьшим значением *d* и помещается в переменную *v*).

0) $\forall v \in V \; d(v) = +\infty$, $d(u) = 0$, $e(v) = NULL$, $u \to S$;

1) Если $S = \varnothing$, то *стоп*;

2) $v \leftarrow S$;

3) $\forall x \in V \; x$ — смежна с вершиной *v*.

если $d(x) > d(v) + \omega(v, x)$,

то $d(x) = d(v) + \omega(v, x)$.

$e(x) = v, \; x \to S$.

4) Переход на п. 1).

Пусть $M = \min \; d(v) \; v \in S$.

7. ДВУДОЛЬНЫЕ ГРАФЫ.

ПАРОСОЧЕТАНИЯ

Двудольным графом называется граф, у которого множество вершин можно разбить на два непересекающихся подмножества так, что ребра соединяют вершины из разных подмножеств.

Паросочетанием в двудольном графе называется любое множество попарно несмежных ребер (у них нет общих вершин).

Паросочетание называется максимальным для данного графа, если оно содержит наибольшее число ребер для всех возможных паросочетаний.

Паросочетание называется совершенным (из множества v в множество w), если число ребер в нем совпадает с числом вершин в подмножестве с.

Для любого подмножества S через ф(S) обозначим те вершины из множества w, которые соединяются ребрами с вершинами подмножества S.

Теорема Холла (без доказательства)

Для того, чтобы в двудольном графе существовало совершенное паросочетание, необходимо и достаточно, чтобы для любого подмножества S из множества V выполнялось условие [S] <= [ф(S)].

МАКСИМАЛЬНЫЕ ПАРОСОЧЕТАНИЯ

Венгерский алгоритм нахождения максимального паросочетания.

Дан двудольный граф. Все определения для графа справедливы.

Полным паросочетанием называется паросочетание (ПС), к которому нельзя добавить ни одного ребра графа, не нарушив условие несмежности ребер.

1. Перебираем все ребра в любом порядке. Все несмежные ребра включаем в паросочетание.

Ребра, входящие в полное паросочетание, будем называть толстыми. Остальные ребра считаем тонкими.

Вершины, которые соединены толстыми ребрами – насыщенные. Остальные – ненасыщенные.

Чередующейся цепью называется цепь, в которой тонкие и толстые ребра чередуются.

Тонкой чередующейся цепью называется чередующаяся цепь, соединяющая 2 ненасыщенные вершины (В ней тонких ребер на 1 больше, чем толстых).

1. Находим полное паросочетание.

2. Для этого паросочетания ищем тонкую цепь. Если ее нет, то данное паросочетание максимально и алгоритм закончен.

3. Если же она существует, то проводим перекраску ребер.

4. Толстые ребра тонкой цепи делаем тонкими, а тонкие – толстыми.

5. Получаем новое паросочетание, т. е. из исходного паросочетания удаляем те толстые ребра, которые входили в тонкую цепь и вместо них добавляем тонкие ребра из этой цепи.

6. Переходим на шаг 2.

Количество ребер в новом паросочетании увеличится на 1.

Максимальное паросочетание (МПС) найдено.

МАКСИМАЛЬНЫЕ ПАРОСОЧЕТАНИЯ

МАТРИЦА СМЕЖНОСТИ ДВУДОЛЬНОГО ГРАФА

[V] = M
[W] = N

$$A_{M\times N} = (a_{ij})$$

$$a_{ij} = \begin{cases} 1, \text{если есть ребро } (v_i, w_j) \\ 0, \text{если нет} \end{cases}$$

1	0	1	1
0	1	1	0
1	0	0	0

Чтобы найти полное паросочетание, нужно найти единицы, которые находятся в разных строках и разных столбцах.

При поисках мы можем двигаться по строкам и на углы в 90 градусов.

Задача об оптимальном назначении

ПОСТАНОВКА ЗАДАЧИ

Задача о наилучшем распределении некоторого числа работ между таким же числом исполнителей.

Есть m работников и m работ.

Каждый из работников выполняет каждую работу с определенной эффективностью, т. е. дана матрица эффективности $A_{m*m} = (a_{ij})$.

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

Задача о назначениях имеет много интерпретаций: распределение работ между механизмами, распределение целей между огневыми средствами для максимизации математического ожидания числа пораженных целей или среднего ущерба и т.д.

ПОСТАНОВКА ЗАДАЧИ (ПРОДОЛЖЕНИЕ)

Требуется распределить работы таким образом, чтобы:

- каждый работник выполнял только одну работу,
- выполнялись все работы,
- суммарная эффективность была максимальна среди всех возможных.

ПОСТАНОВКА ЗАДАЧИ (ПРОДОЛЖЕНИЕ)

В терминах матрицы эффективности задача состоит в нахождении m элементов, расположенных в разных строках и разных столбцах, чтобы их сумма была максимальной.

$$A = \begin{pmatrix} 1 & \textcircled{2} & 3 & 4 \\ 1 & 4 & \textcircled{4} & 2 \\ \textcircled{2} & 5 & 6 & 3 \\ 1 & 6 & 4 & \textcircled{1} \end{pmatrix}$$

В ТЕРМИНАХ ТЕОРИИ ГРАФОВ:

Дан двудольный полный граф с

$$|V| = m, \ |W| = m.$$

Даны длины ребер.

Задача состоит в нахождении ***совершенного паросочетания***, сумма длин всех ребер которого максимальна.

ОПРЕДЕЛЕНИЯ ЕЩЕ РАЗ:

Паросочетание называется ***совершенным*** (из множества v в множество w), если число ребер в нем совпадает с числом вершин в графе.

Определения еще раз:

Паросочетание называется ***совершенным*** (из множества v в множество w), если число ребер в нем совпадает с числом вершин в графе.

Паросочетание называется ***максимальным*** для данного графа, если оно содержит наибольшее число ребер для всех возможных паросочетаний.

Алгоритм. Шаг 1

1. Всем вершинам v_i даем метку x_i = max среди всех элементов i-й строки, i-1..m.

Всем w_j присваиваем метку y_j=0, j=1..m.

Алгоритм. Шаг 2

2. Ищем ребра, для которых выполняется условие

$$x_i + y_j = a_{ij}$$

Строим граф, в который входит все вершины исходного графа и найденные ребра.

Алгоритм. Шаг 3

3. В построенном графе ищем максимальное паросочетание. Если найденное паросочетание совершенно, то алгоритм закончен. Если нет, то переходим дальше.

Немного теории. Теорема Холла

Для того, чтобы в двудольном графе существовало совершенное паросочетание, необходимо и достаточно, чтобы для любого подмножества $S \subset V$ выполнялось условие

$$|S| \leq |\phi(S)|,$$

где $\phi(S)$ – множество вершин из W, которые соединяются ребрами с вершинами из S.

НЕМНОГО ТЕОРИИ. ТЕОРЕМА ХОЛЛА (ПРОДОЛЖЕНИЕ)

Теорема Холла. Для того, чтобы в двудольном графе существовало совершенное паросочетание, необходимо и достаточно, чтобы для любого подмножества $S \subset V$ выполнялось условие

$$|S| \leq |\phi(S)|.$$

Если на 3-м шаге алгоритма обнаружено, что найденное максимальное паросочетание **не совершенно**, то ищем подмножество вершин из V, такое что $|S| > |\phi(S)|$, т.е. ищем часть графа, где не выполняется условие теоремы Холла.

Алгоритм. Шаг 4

4. Из теоремы Холла существует такое подмножество S из V, что $|S| > |\phi(S)|$.

Ищем это подмножество.

Для каждой вершины v_i из S метку x_i **уменьшают на 1**, а для каждой y_j из (S) метку y_j **увеличивают на 1.**

Алгоритм. Шаг 5

5. Переходим на начало шага 2 с новыми значениями меток.

ВЕСЬ АЛГОРИТМ

1. Всем вершинам v_i даем метку x_i = max среди всех элементов i-й строки, i-1..m. Всем w_j присваиваем метку y_j=0, j=1..m.

2. Ищем ребра, для которых выполняется условие $x_i + y_j = a_{ij}$. Строим граф, в который входит все вершины исходного графа и найденные ребра.

3. В построенном графе ищем максимальное паросочетание. Если найденное паросочетание совершенно, то алгоритм закончен. Если нет, то переходим дальше.

4. Из теоремы Холла существует подмножество S из V, $|S| > |\phi(S)|$. Ищем это подмножество. Для каждой вершины v_i из S метку x_i уменьшают на 1, а для y_j из $\phi(S)$ метку y_j увеличивают на 1.

5. Переходим на начало шага 2 с новыми значениями меток.

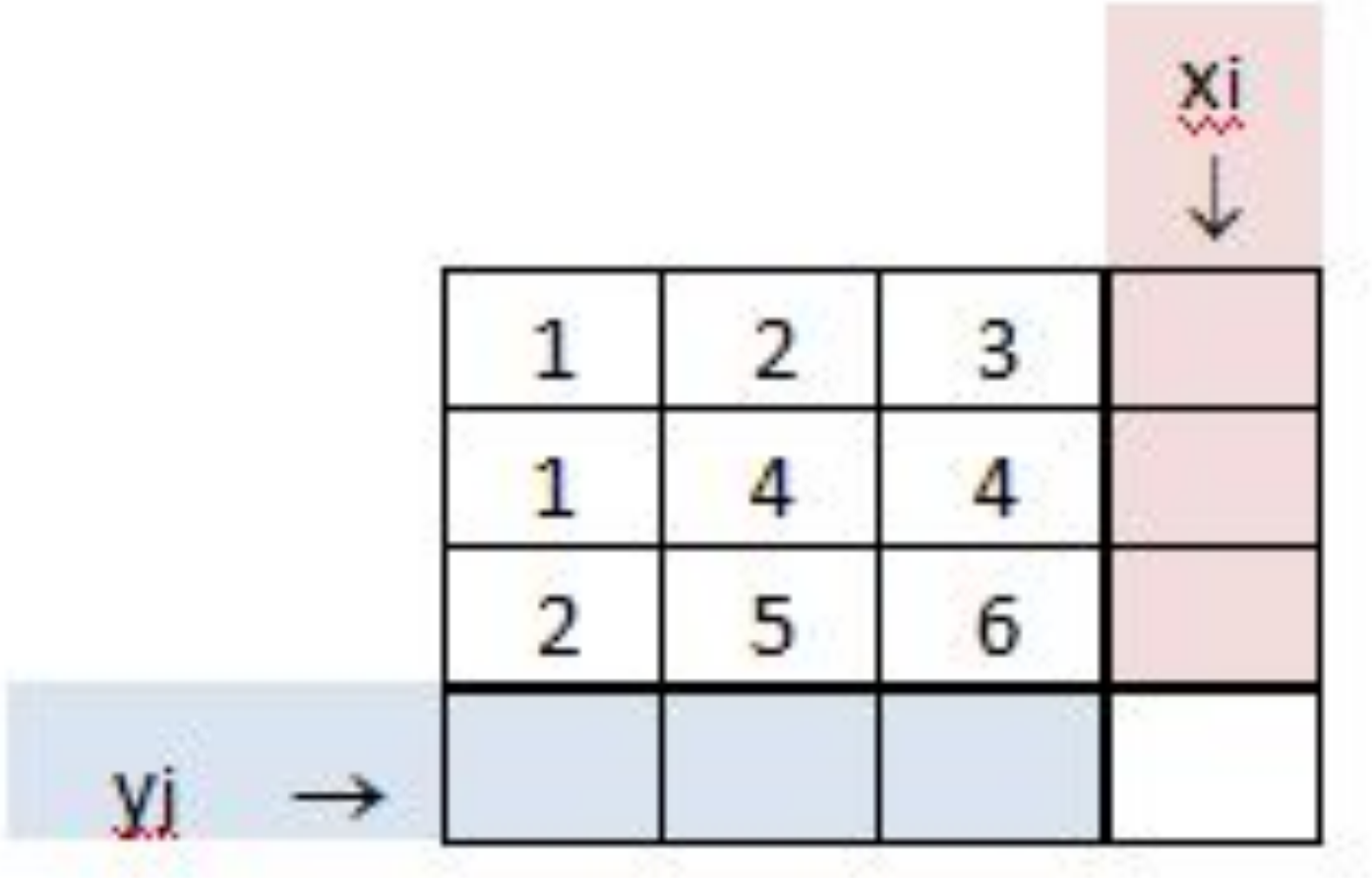
ПРИМЕР 1.

Дана матрица назначений:

1	2	3
1	4	4
2	5	6

ПРИМЕР 1.

Дана матрица назначений:



ПРИМЕР 1. ШАГ 1

- Расставляем метки:

1	2	3	3
1	4	4	4
2	5	6	6
0	0	0	

ПРИМЕР 1. ШАГ 2.

Ищем ребра, для которых выполняется условие

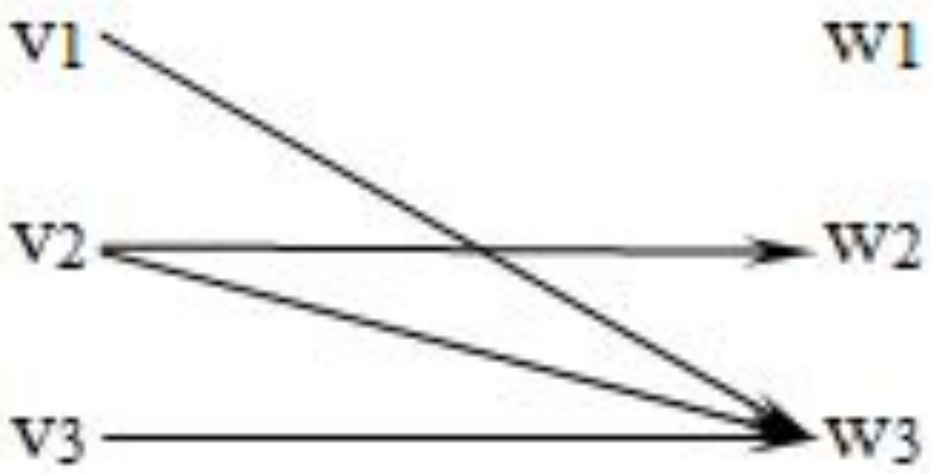
$$x_i + y_j = a_{ij}$$

1	2	3	3
1	4	4	4
2	5	6	6
0	0	0	

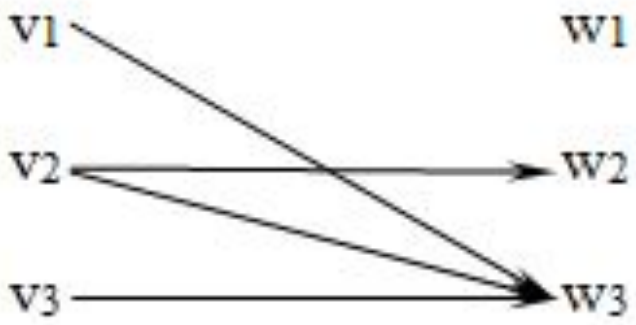
ПРИМЕР 1. ШАГ 2.

- Строим граф, в который входит все вершины исходного графа и найденные ребра.

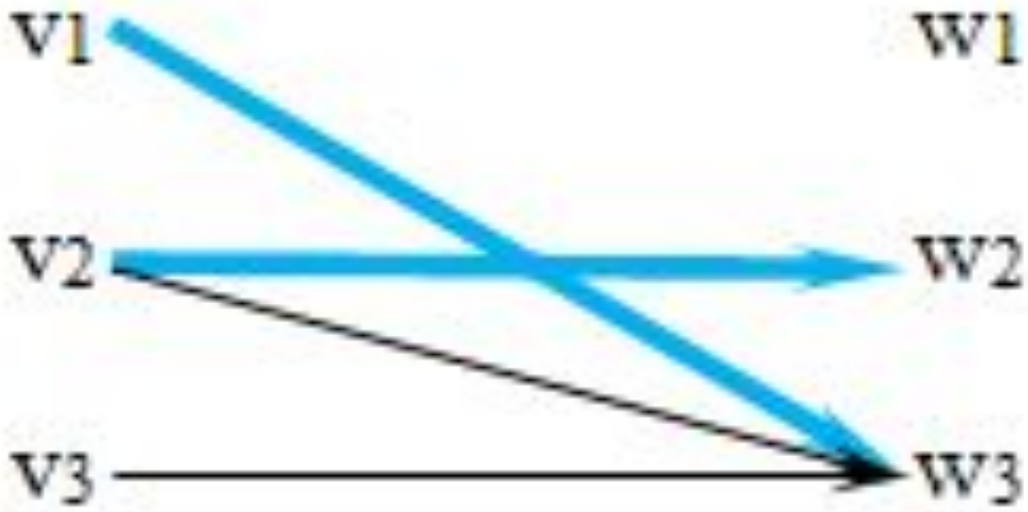
1	2	3	3
1	4	4	4
2	5	6	6
0	0	0	



ПРИМЕР 1. ШАГ 3.



В построенном графе ищем максимальное паросочетание.

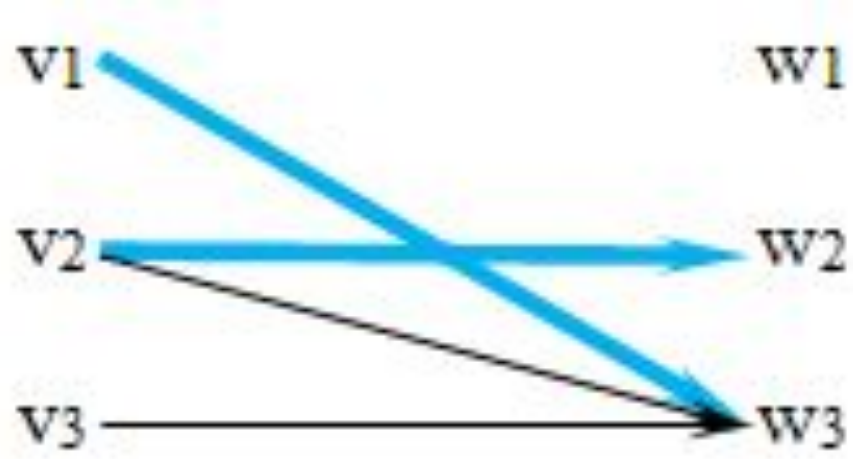


Алгоритм нахождения максимального паросочетания

1. Перебираем все ребра в любом порядке. Все несмежные ребра включаем в паросочетание.
2. Находим полное паросочетание.
3. Для этого паросочетания ищем тонкую цепь. Если ее нет, то данное паросочетание максимально и алгоритм закончен.
4. Если же она существует, то проводим перекраску ребер.
5. Толстые ребра тонкой цепи делаем тонкими, а тонкие – толстыми.
6. Получаем новое паросочетание, т. е. из исходного паросочетания удаляем те толстые ребра, которые входили в тонкую цепь и вместо них добавляем тонкие ребра из этой цепи.
7. Переходим на шаг 3.

ПРИМЕР 1. ШАГ 4.

Найдем подмножество S из V, такое что|S|> |ϕ(S)|.



$S=\{ v_1, v_2, v_3 \}, \varphi(S)=\{ w_2, w_3 \}.$

ПРИМЕР 1. ШАГ 4.

Поставим новые метки: для каждой вершины v_i из S метку x_i уменьшим на 1, а для y_j увеличим на 1.

$S=\{ v_1, v_2, v_3 \}$, $\varphi(S)=\{ w_2, w_3 \}$.

1	2	3	3	2
1	4	4	4	3
2	5	6	6	5
0	0	0		
	1	1		

ПРИМЕР 1. ШАГ 5.

Перейдем на шаг 2 с новыми значениями меток.

1	2	3	2
1	4	4	3
2	5	6	5
0	1	1	

ПРИМЕР 1. ШАГ 2.

Снова ищем ребра, для которых выполняется условие

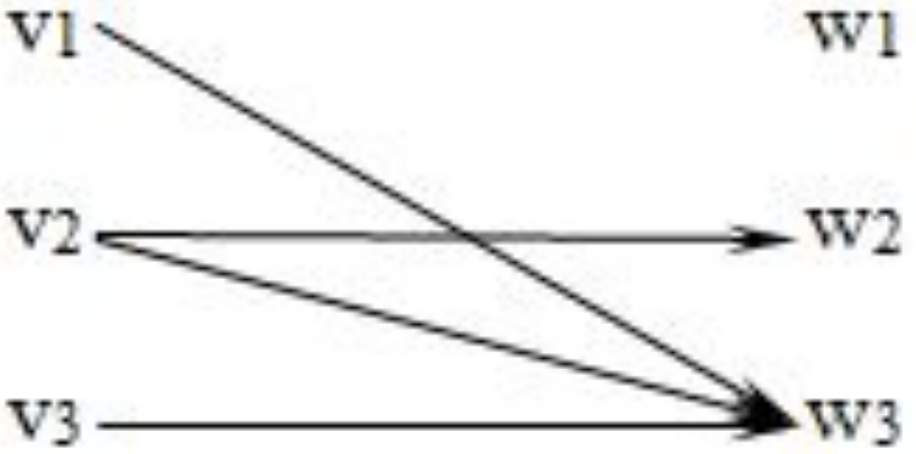
$x_i + y_j = a_{ij}$

1	2	3	2
1	4	4	3
2	5	6	5
0	1	1	

ПРИМЕР 1. ШАГ 2.

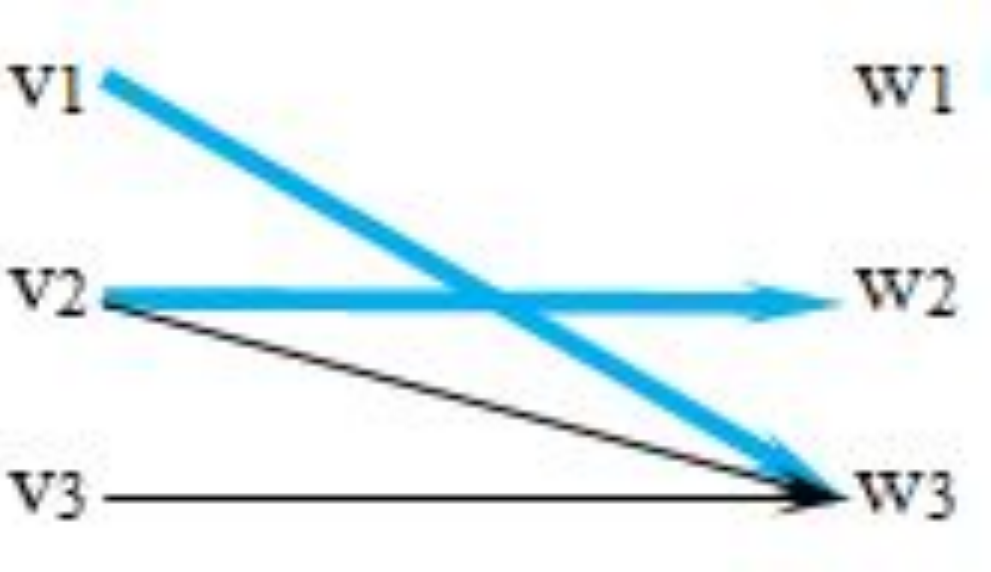
Строим граф из выбранных ребер

1	2	3	2
1	4	4	3
2	5	6	5
0	1	1	

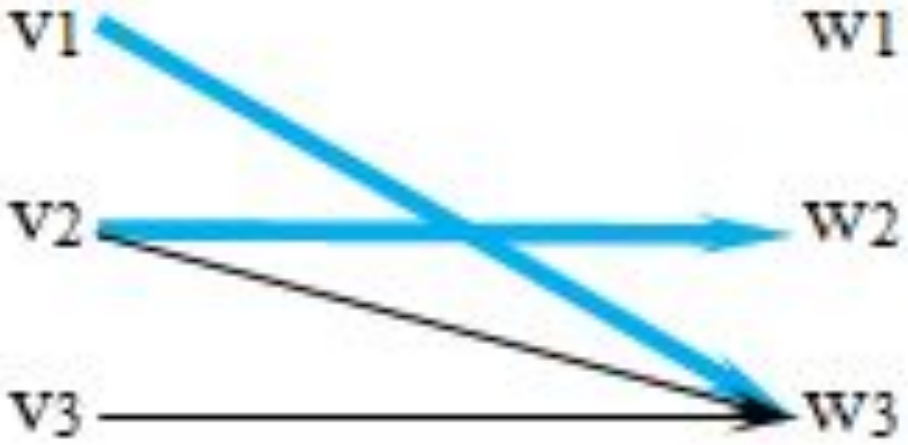


ПРИМЕР 1. ШАГ 3.

Ищем в нем максимальное паросочетание.



ПРИМЕР 1. ШАГ 4.



$S=\{ v_1, v_2, v_3 \}$, $\varphi(S)=\{ w_2, w_3 \}$.

ПРИМЕР 1. ШАГ 4.

Изменяем метки.

$S=\{ v_1, v_2, v_3 \}$, $\varphi(S)=\{ w_2, w_3 \}$.

1	2	3	2	1
1	4	4	3	2
2	5	6	5	4
0	1	1		
	2	2		

ПРИМЕР 1. ШАГ 2.

1	2	3	1
1	4	4	2
2	5	6	4
0	2	2	

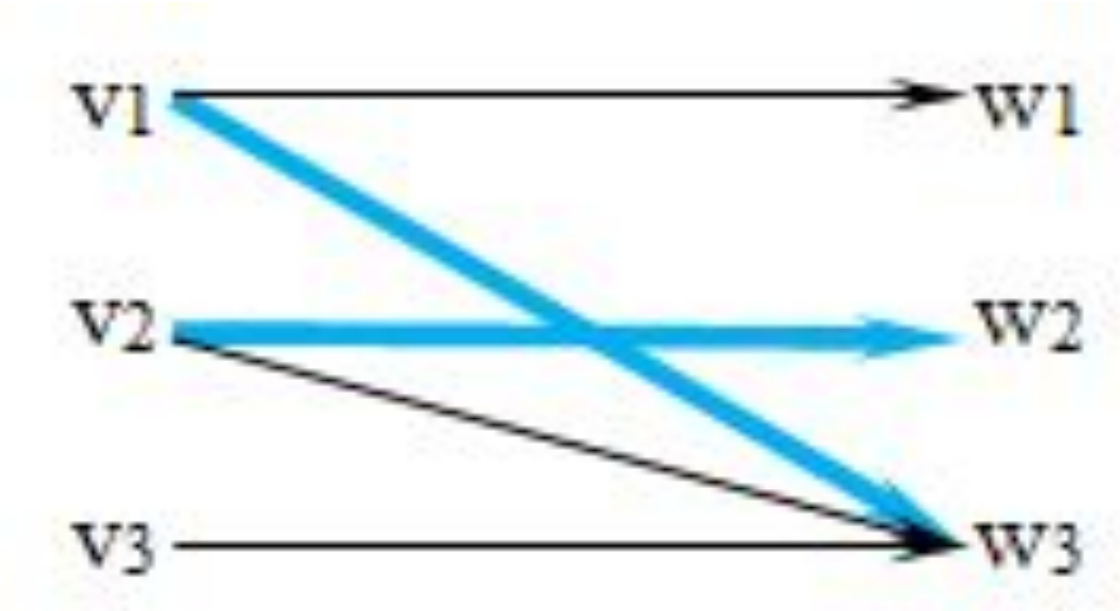
ПРИМЕР 1. ШАГ 2.

Ищем ребра, для которых $x_i + y_j = a_{ij}$

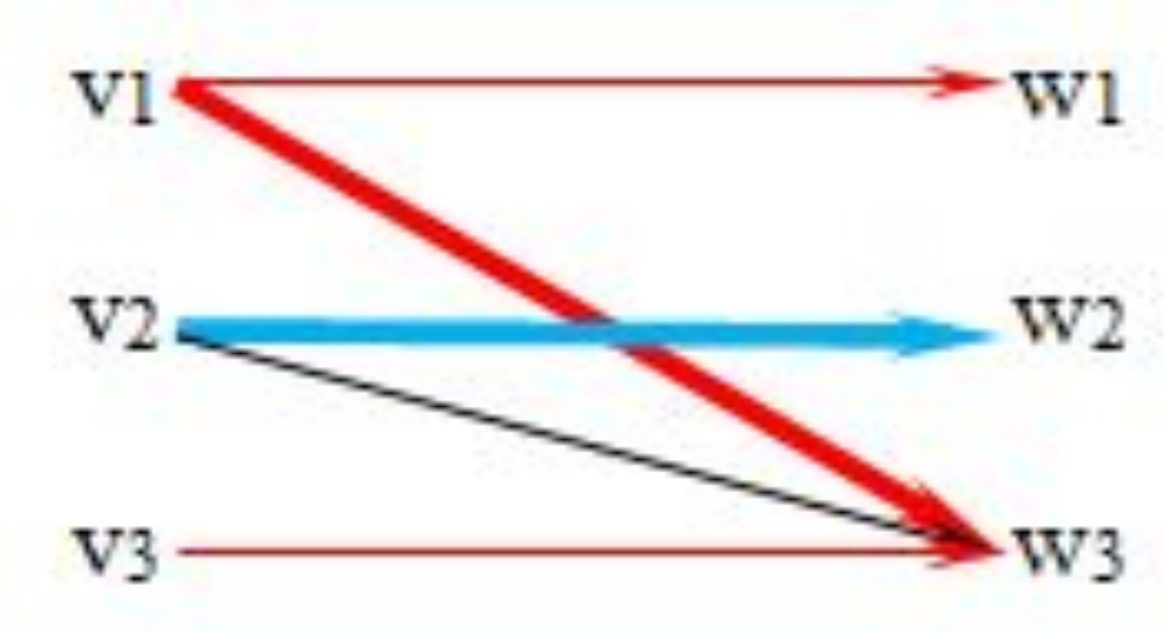
1	2	3	1
1	4	4	2
2	5	6	4
0	2	2	

ПРИМЕР 1. ШАГ 3.

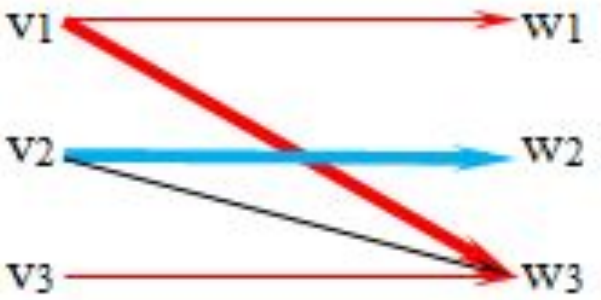
Строим граф и ищем макимальное паросочетание.



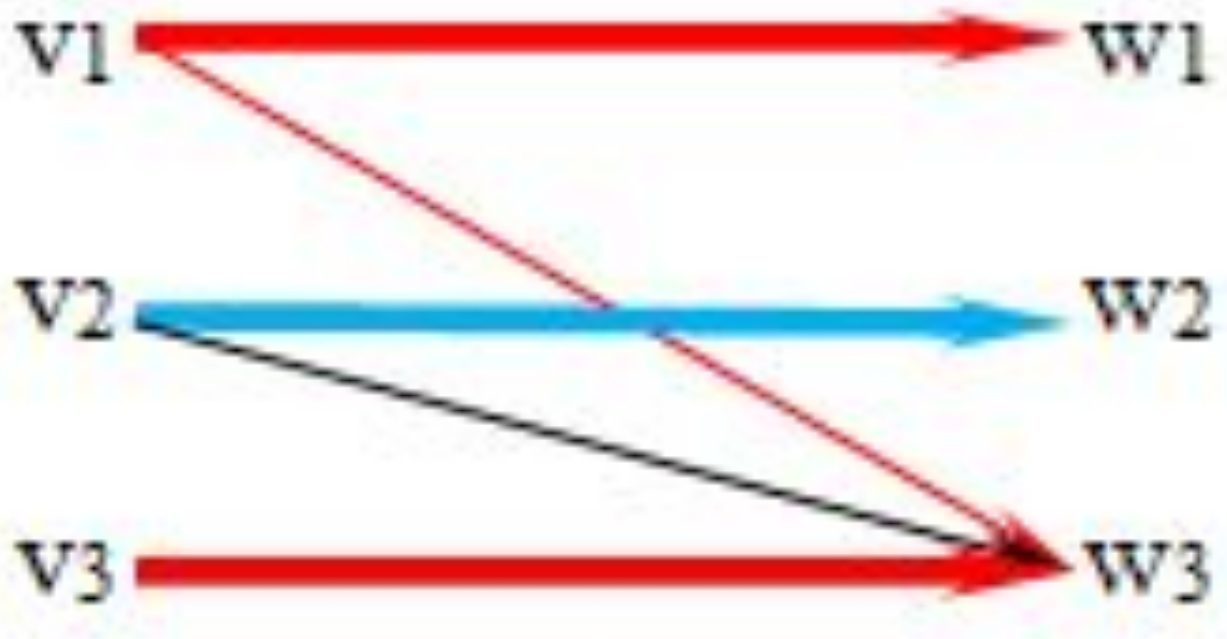
ПРИМЕР 1. ШАГ 3.



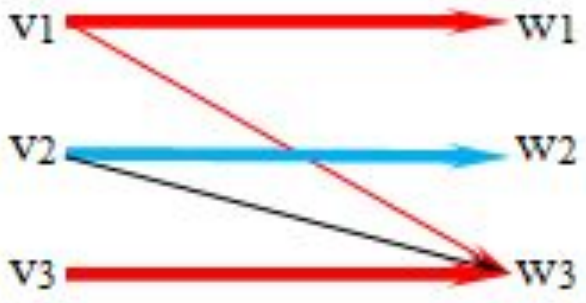
ПРИМЕР 1. ШАГ 3.



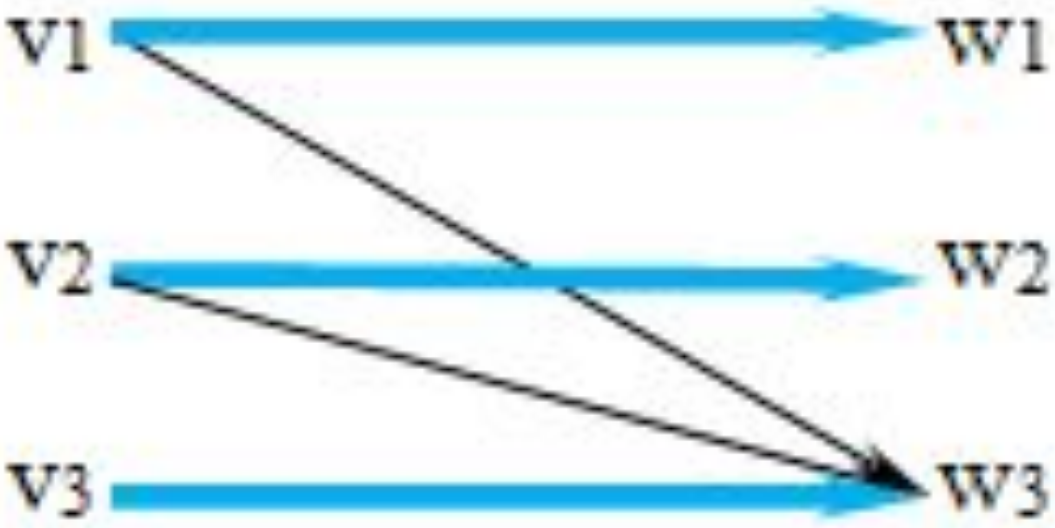
Перекрашиваем тонкую цепь.



ПРИМЕР 1. ШАГ 3.



Получившееся в результате паросочетание совершенно, алгоритм закончен.



ПРИМЕР 1. ОТВЕТ

1	2	3
1	4	4
2	5	6

ПРИМЕР 2.

Дана матрица назначений (эффективности):

1	2	3	4
1	4	4	2
2	5	6	3
1	6	4	1

8.
Сети Петри.

Для моделирования динамических дискретных систем используется математический аппарат, называемый **сетями Петри.** Сети Петри разрабатывались специально для моделирования тех систем, которые содержат взаимодействующие параллельные компоненты.

Определение 1. Сеть Петри *C* является четвёркой, $C=(P, T, I, O)$. $P=\{p_1, p_2, ..., p_n\}$– конечное *множество позиций*, $n^3 0$. $T=\{t_1, t_2, ..., t_m\}$ – конечное *множество переходов*, $m^3 0$. Множество позиций и множество переходов не пересекаются, P Ç T = Æ. *Входная функция I* осуществляет отображение из переходов в комплекты позиций*:* $T \to P^\infty$. *Выходная функция O* осуществляет отображение из позиций в комплекты переходов*:* $T \to P^\infty$.

Для моделирования вычислительных систем наибольший интерес представляют временные, стохастические и функциональные сети Петри (СП).

Временная СП характеризуется тем, что вводятся данные задержки при перемещении маркеров. Задержки можно относить к переходам или позициям.

Стохастическая СП характеризуется случайными задержками, в них возможно введение вероятностей срабатывания разрешенных переходов.

Функциональная СП характеризуется тем, что отражает не только последовательность событий, но и процессы обработки некоторого потока данных. Для этого в описание каждого перехода добавляется алгоритм обработки данных.

Цветная СП используется тогда, когда требуется отличать друг от друга некоторые группы маркеров. Например, детали разных типов.

Автоматная СП – это сеть, в которой каждый переход имеет только один вход и один выход.

Общие свойства сетей Петри

СП ***безопасна***, если безопасны все позиции сети. Позиция сети является ***безопасной***, если число фишек в ней никогда не превышает 1.

СП ***k-ограничена***, если все ее позиции *k*-ограничены. Позиция является ***k-ограниченной***, если количество фишек в ней не может превышать целое число *k*.

СП называется ***строго сохраняющей***, если общее число фишек в сети остается постоянной.

Достижимость СП заключается в возможности достижения заданных маркировок.

СП ***живая***, если все ее переходы являются живыми. Переход называется ***живым***, если из любого состояния, достижимого из начального, возможен переход в любое другое достижимое состояние.

Дерево достижимостей

Дерево достижимости представляет множество достижимости СП.

Каждая i-я вершина дерева связывается с расширенной разметкой $\mu(i)$. В расширенной разметке число меток в позиции может быть либо неотрицательным целым, либо бесконечно большим. Бесконечное число меток обозначим символом ω. Каждая вершина классифицируется или как граничная, терминальная, дублирующая вершина, или как внутренняя. ***Граничными*** являются вершины, которые еще не обработаны алгоритмом. После обработки граничные вершины становятся либо терминальными, либо дублирующими, либо внутренними. ***Терминальные*** (пассивные) маркировки – это маркировки в которых нет разрешенных переходов. ***Дублирующие*** маркировки – это маркировки, ранее встречающиеся в дереве.

Алгоритм начинает свою работу с определения начальной разметки. До тех пор, пока имеются граничные вершины, они обрабатываются алгоритмом.

Пусть x – граничная вершина, которую необходимо обработать, и с которой связана разметка $\mu(x)$.

1. Если в дереве имеется другая вершина y, не являющаяся граничной, и с ней связана разметка $\mu(y)=\mu(x)$, то вершина x дублируется.
2. Если для разметки $\mu(x)$ ни один из переходов неразрешим, т.е. $\mu(x)$ тупиковая разметка, то x терминальная вершина.
3. Для любого перехода t_j, из множества T разрешенного в разметке $\mu(x)$, создать новую вершину z дерева достижимости. Разметка $\mu(z)$, связанная с этой вершиной, определяется для каждой позиции p_i следующим образом:

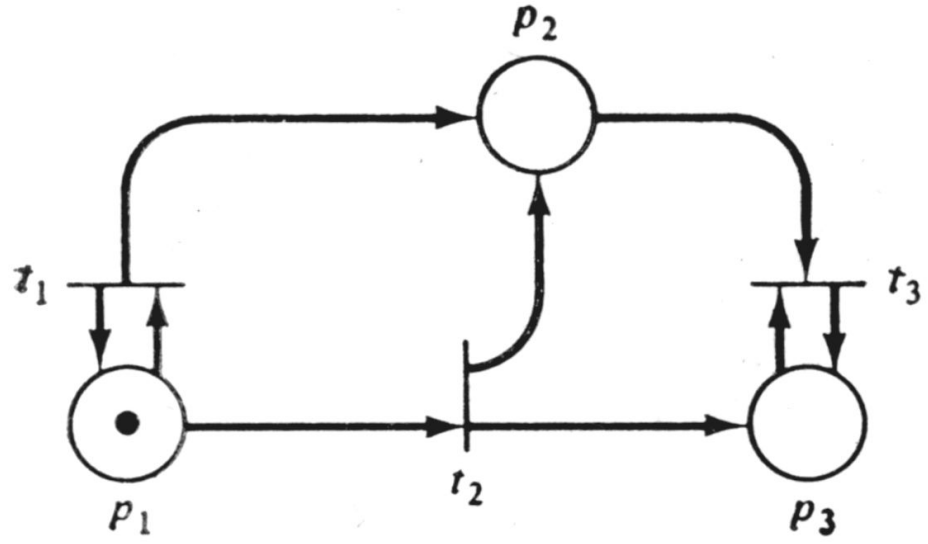
   а) если $\mu(x)_i=\omega$, то $\mu(z)_i=\omega$;

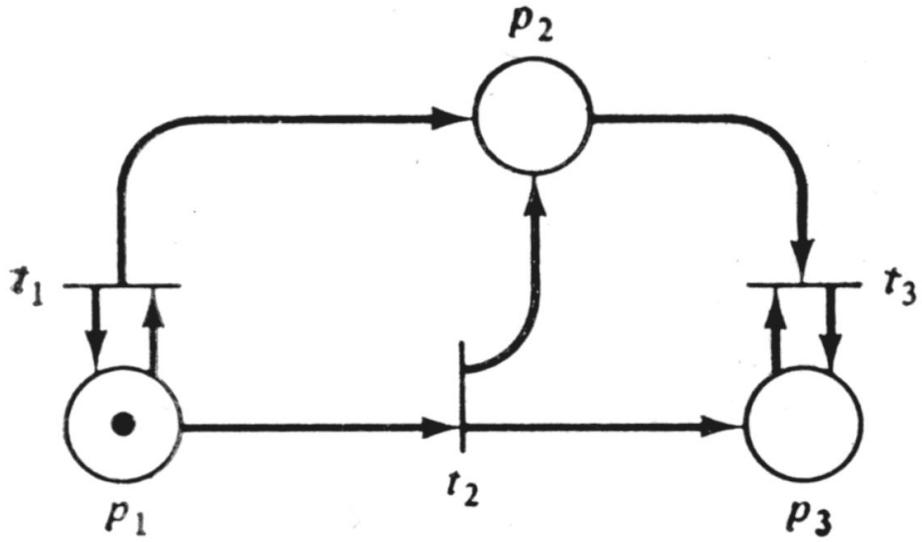
   б) если на пути от корневой вершины к x существует вершина y такая, что

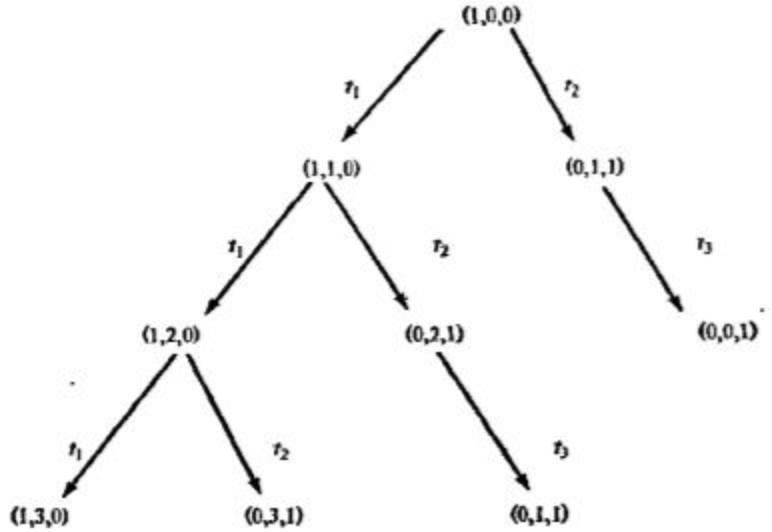
   $\mu(y)\rightarrow^{tj}\mu(x)$, $\mu(y)<\mu(x)$ и $\mu(y)_i<\mu(x)_i$, то $\mu(z)_i=\omega$;

   в) в противном случае $\mu(z)_i=\mu(x)_i$.

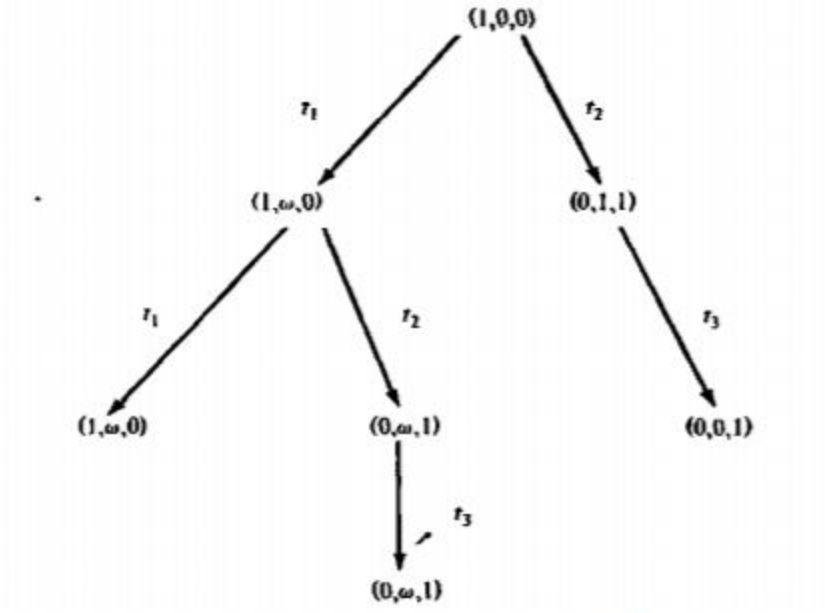
Дуга, помеченная t_j, направлена от вершины x к вершине z. Вершина x переопределяется как внутренняя, вершина z становится граничной. Когда все вершины дерева становятся терминальными, дублирующими или внутренними, алгоритм останавливается.


p_2
t_1
t_3
p_1
t_2
p_3


p_2
t_1
t_3
t_2
p_1
p_3


(1,0,0)
t_1
t_2
(1,1,0)
(0,1,1)
t_1
t_2
t_3
(1,2,0)
(0,2,1)
(0,0,1)
t_1
t_2
t_3
(1,3,0)
(0,3,1)
(0,1,1)

а)


(1,0,0)
t_1
t_2
(1,ω,0)
(0,1,1)
t_1
t_2
t_3
(1,ω,0)
(0,ω,1)
(0,0,1)
t_3
(0,ω,1)

б)