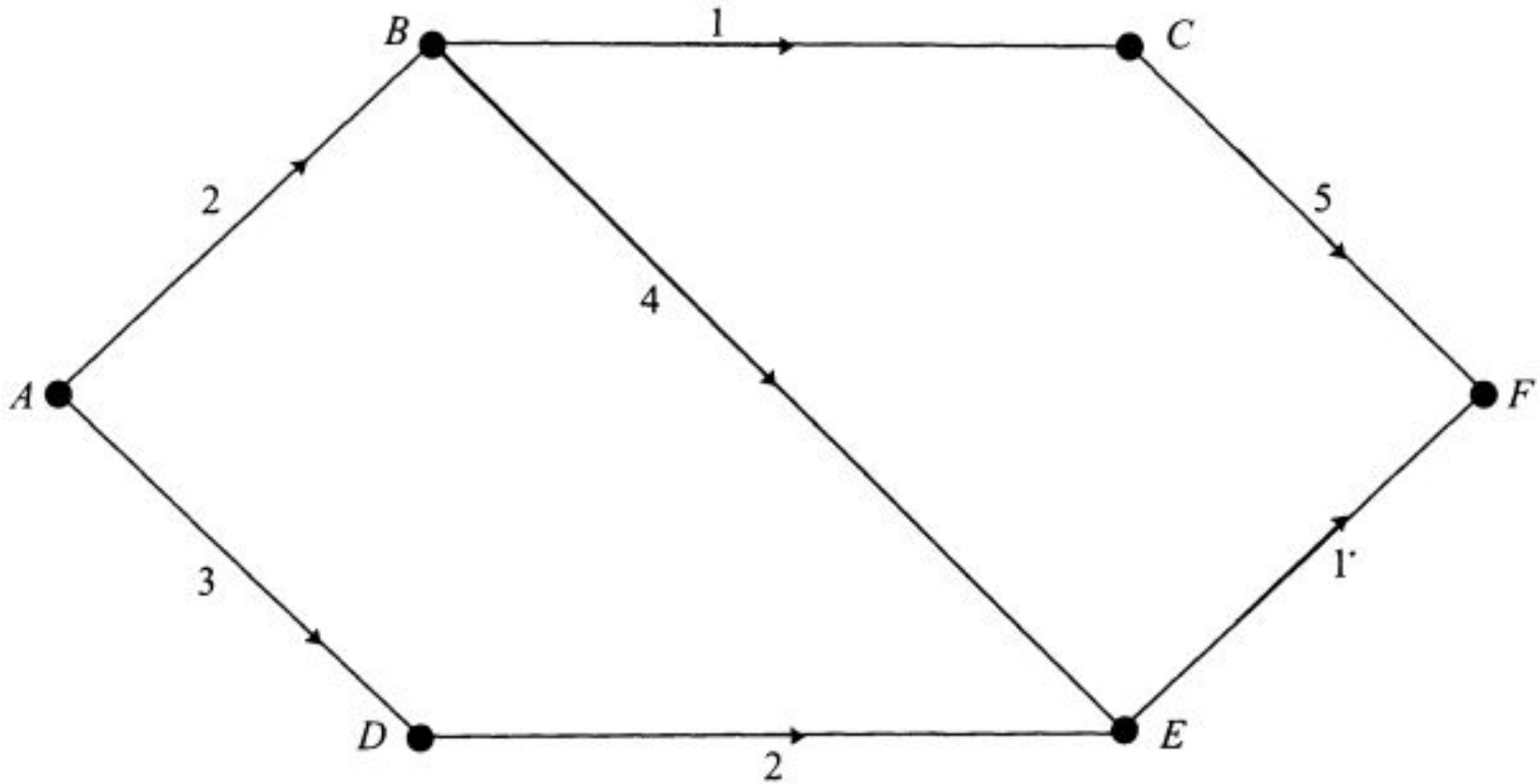
# Тема 5. Сетевые задачи дискретной математики

1. Основные понятия
2. Алгоритм Дейкстры
3. Алгоритм Беллмана-Мура
4. Алгоритм нахождения максимального пути
5. Теорема Форда-Фалкерсона
6. Система ПЕРТ

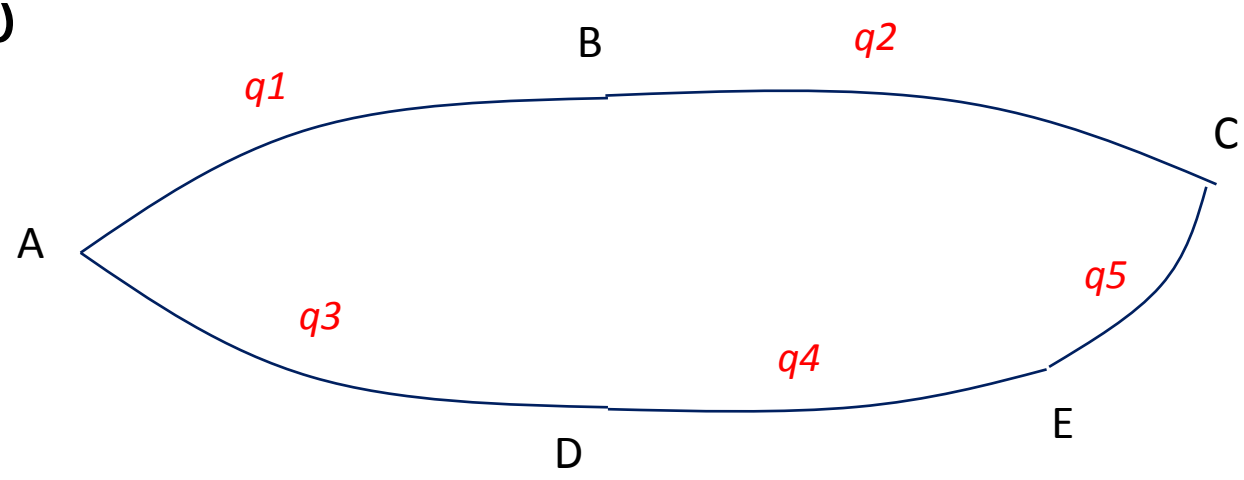
# 1. Основные понятия

- Пусть G=(S, U) – ориентированный граф, S – множество вершин (узлов), U – множество дуг
- Пусть каждой дуге $(x_i, x_j) \in U$ поставлено в соответствие некоторое число $\omega(x_i, x_j)$ – **вес**.
- Тогда граф называется **взвешенным** или **сетью**.
- *В зависимости от контекста, в качестве веса могут выступать стоимость, длина, пропускная способность и т.п.*

# Пример сети (нагруженного орграфа)



- Вес некоторого пути равен сумме весов дуг, его составляющих:
- ω(ABC)= ω(AB)+ ω(BC)=q1+q2
- ω(ADEC)= ω(AD)+ ω(DE)+ ω (EC)=q3+q4+q5



# 2. Алгоритм Дейкстры (алгоритм расстановки меток)

- Пусть G={S, U, Ω} – ориентированный граф со взвешенными дугами.
- S – множество вершин (узлов), U – множество дуг, Ω - весовая матрица
- Вершина s- начало пути, t - конец пути

- *Алгоритм был разработан в 1959 г нидерландским математиком **Эдсгером Дейкстрой** (1930–2002) в поисках путей оптимизации разводки печатных плат*

# Постановка задачи

- В сети найти кратчайший путь, соединяющий две заданные вершины.

# Замечания

- узлам сети $\mathbf{x_i \in S}$ приписываются числа (метки) $\mathbf{d(x_i)}$
- Если вершина $\mathbf{x_i}$ получила на некотором шаге метку $\mathbf{d(x_i)}$, то в графе **G** существует путь из **s** в $\mathbf{x_i}$, имеющий вес **d(xi)**
- Состояния меток – временные или постоянные
- если метка стала постоянной, то кратчайшее расстояние от вершины **s** до соответствующей найдено
- Ограничение алгоритма – веса дуг **положительны**

# Этапы алгоритма Дейкстры

- I этап – поиск длины кратчайшего пути от вершины s к вершине t,
- II этап – построение кратчайшего пути от вершины s к вершине t.

# Этап 1. Нахождение длины кратчайшего пути

- **Шаг 1. Присвоение вершинам начальных меток**. Пусть **d(s) = 0*** и считаем эту метку **постоянной** (*постоянные метки выделяются звездочкой*).
- Для остальных вершин $x_i \in S$, ($x_i \neq s$) полагаем $d(x_i) = \infty$ и считаем эти метки **временными**.
- Обозначим **x'** - текущая вершина.
- Пусть **x'= s**.

- *под знаком ∞ будем подразумевать неопределенность*

# Шаг 2. Изменение меток

Для каждой вершины $x_i$ с временной меткой, непосредственно
следующей за вершиной **x'**:

$$d_{нов}(x_i) = \min\{d_{стар}(x_i), d(x') + \omega(x', x_i)\} \quad (*)$$

# Шаг 3. Превращение метки из временной в постоянную

- Из всех вершин с временными метками выбираем вершину $x_j$ с наименьшим значением метки:

- **$d(x_j^*) = \min\{ d(x_j) \mid x_j \in S, d(x_j)$ – временная $\}$**

- Превращаем эту метку в постоянную и полагаем $\tilde{x} = x_j^*$.

# Шаг 4. Проверка завершения первого этапа

- Если **x’ = t**, то **d(x’)** – длина кратчайшего пути от **s** до **t**.
- В противном случае – возврат ко второму шагу.

# Этап 2. Построение кратчайшего пути

- Шаг 5. Последовательный поиск дуг кратчайшего пути

- Среди вершин, непосредственно предшествующих вершине $\tilde{x}$ с постоянными метками, находим вершину $\mathbf{x_i}$, удовлетворяющую соотношению $\tilde{x}$
- $\mathbf{d(\ \ ) = d(x_i) + \omega(x_i,\ \ )}$
- Включаем дугу $\mathbf{(x_i,\ \ )}$ в искомый путь и полагаем $\tilde{x}$ $= \mathbf{x_i}$.

# Шаг 6. Проверка на завершение второго этапа

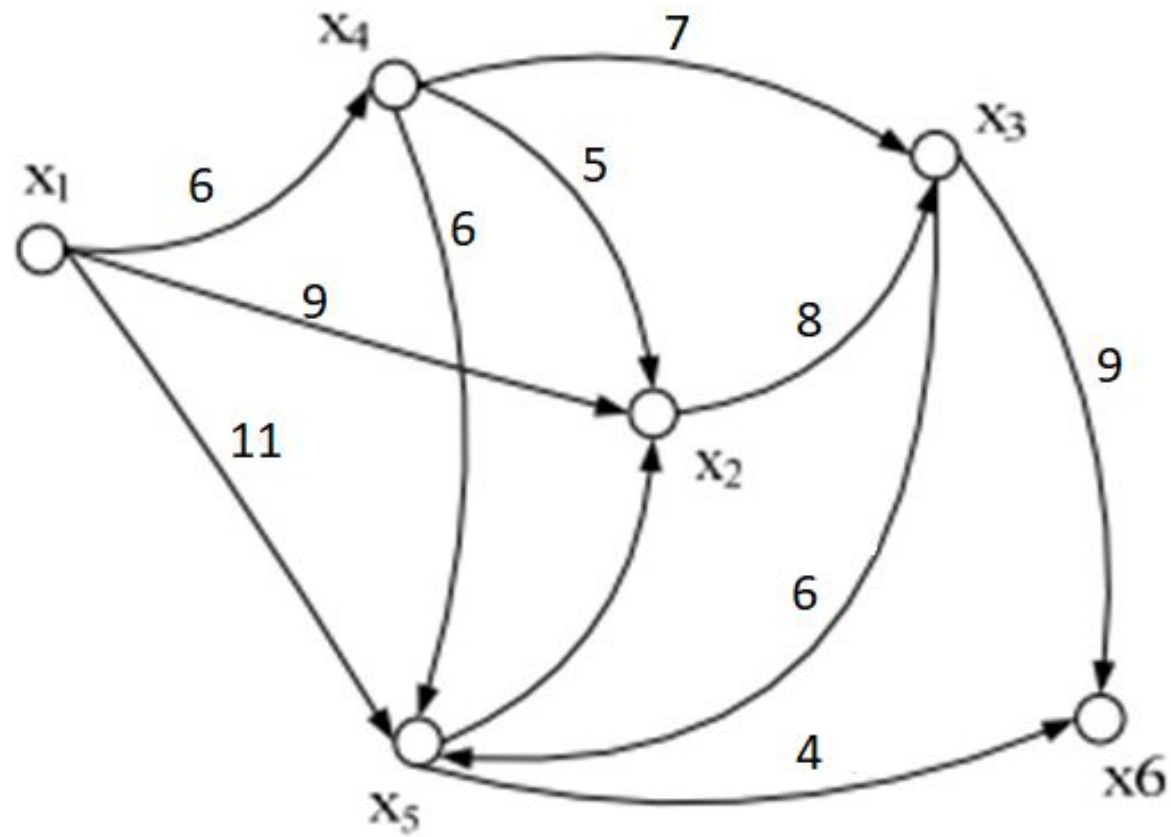
- Если $\tilde{x}$ = **s**, то кратчайший путь найден – последовательность дуг, полученных на пятом шаге и выстроенных в **обратном порядке**.
- В противном случае – возврат к пятому шагу.

# Пример

Задана весовая матрица Ω графа G. Необходимо найти минимальный путь из вершины $x_1$ в вершину $x_6$ по алгоритму Дейкстры.

$$\Omega = \begin{array}{c} \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{array} \begin{array}{c} \begin{array}{cccccc} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \end{array} \\ \begin{pmatrix} - & 9 & \infty & 6 & 11 & \infty \\ \infty & - & 8 & \infty & \infty & \infty \\ \infty & \infty & - & \infty & 6 & 9 \\ \infty & 5 & 7 & - & 6 & \infty \\ \infty & 6 & \infty & \infty & - & 4 \\ \infty & \infty & \infty & \infty & \infty & - \end{pmatrix} \end{array}$$

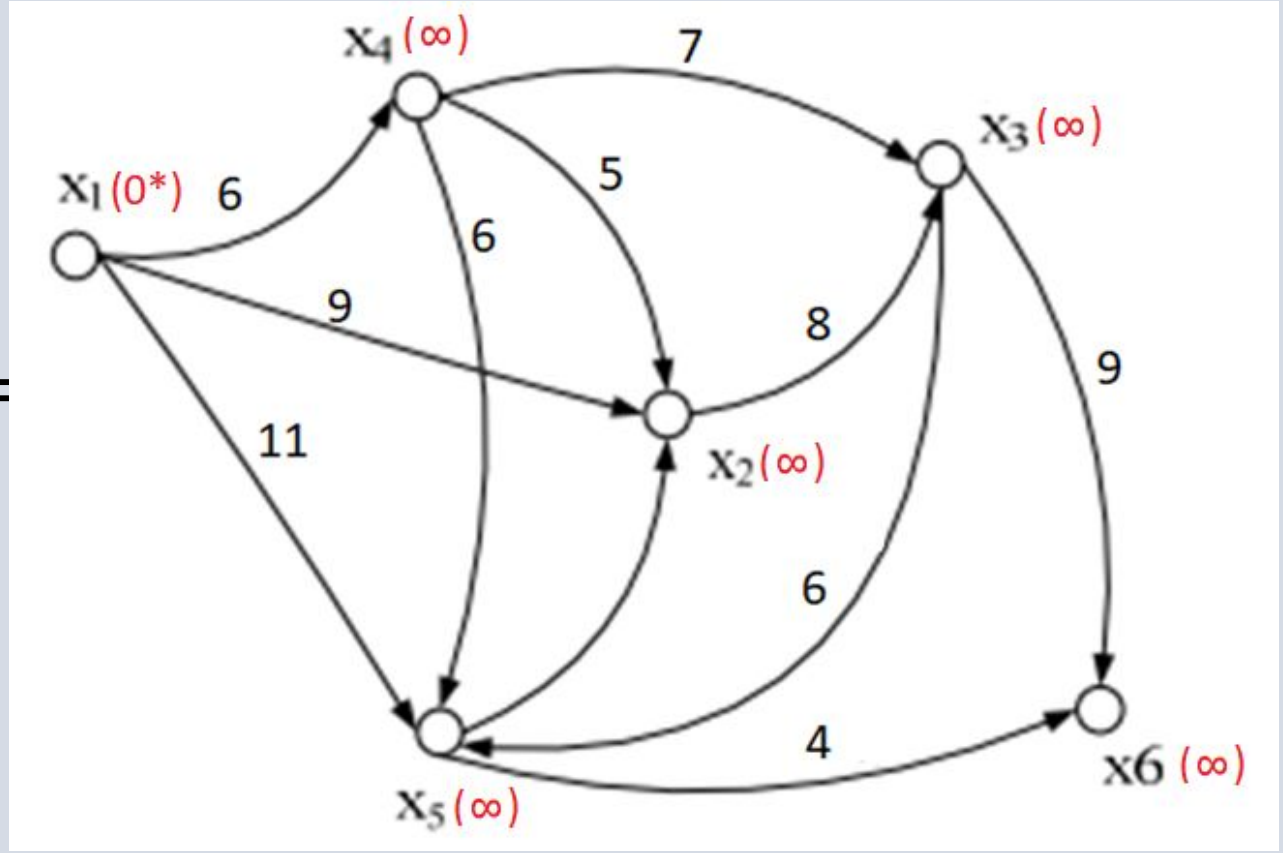
$$\Omega = \begin{array}{c} \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{array} \begin{array}{c} \begin{array}{cccccc} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \end{array} \\ \begin{pmatrix} - & 9 & \infty & 6 & 11 & \infty \\ \infty & - & 8 & \infty & \infty & \infty \\ \infty & \infty & - & \infty & 6 & 9 \\ \infty & 5 & 7 & - & 6 & \infty \\ \infty & 6 & \infty & \infty & - & 4 \\ \infty & \infty & \infty & \infty & \infty & - \end{pmatrix} \end{array}$$



# **Этап 1.** поиск длины кратчайшего пути от источника **s** к стоку **t**

## **Шаг 1.**

Полагаем
d(x1) = 0*, $\tilde{x}$ = x1,
d(x2) = d(x3) = d(x4) = d(x5) =
d(x6) =∞.



# 1-я итерация

**Шаг 2**

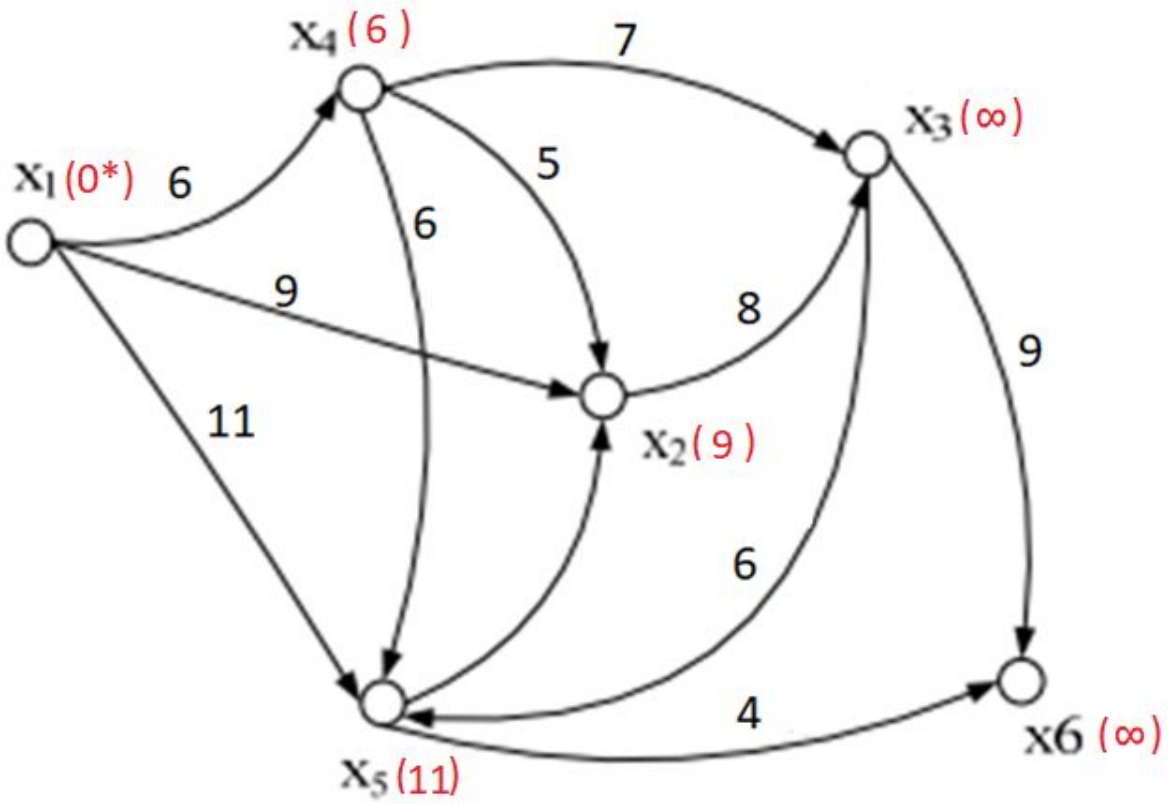
Множество вершин, непосредственно следующих за $x_1$ с временными метками: $S' = \{x_2, x_4, x_5\}$.

Пересчитаем временные метки этих вершин:

$d(x_2) = \min\{\infty, 0^*+9\} = 9$

$d(x_4) = \min\{\infty, 0^*+6\} = 6$

$d(x_5) = \min\{\infty, 0^*+11\} = 11$



# Шаг 3

Одна из временных меток превращается в постоянную:

$\tilde{x}$ = min{9,∞,6,11,∞} = 6* = d($x_4$),

= $x_4$

# Шаг 4

$\tilde{x}$ = $x_4$ ≠t = $x_6$ (т.е. до конечной вершины не дошли) ⇒ возвращение на второй шаг

# 2-я итерация
# Шаг 2

Множество вершин, непосредственно следующих за $\tilde{x} = x_4$:

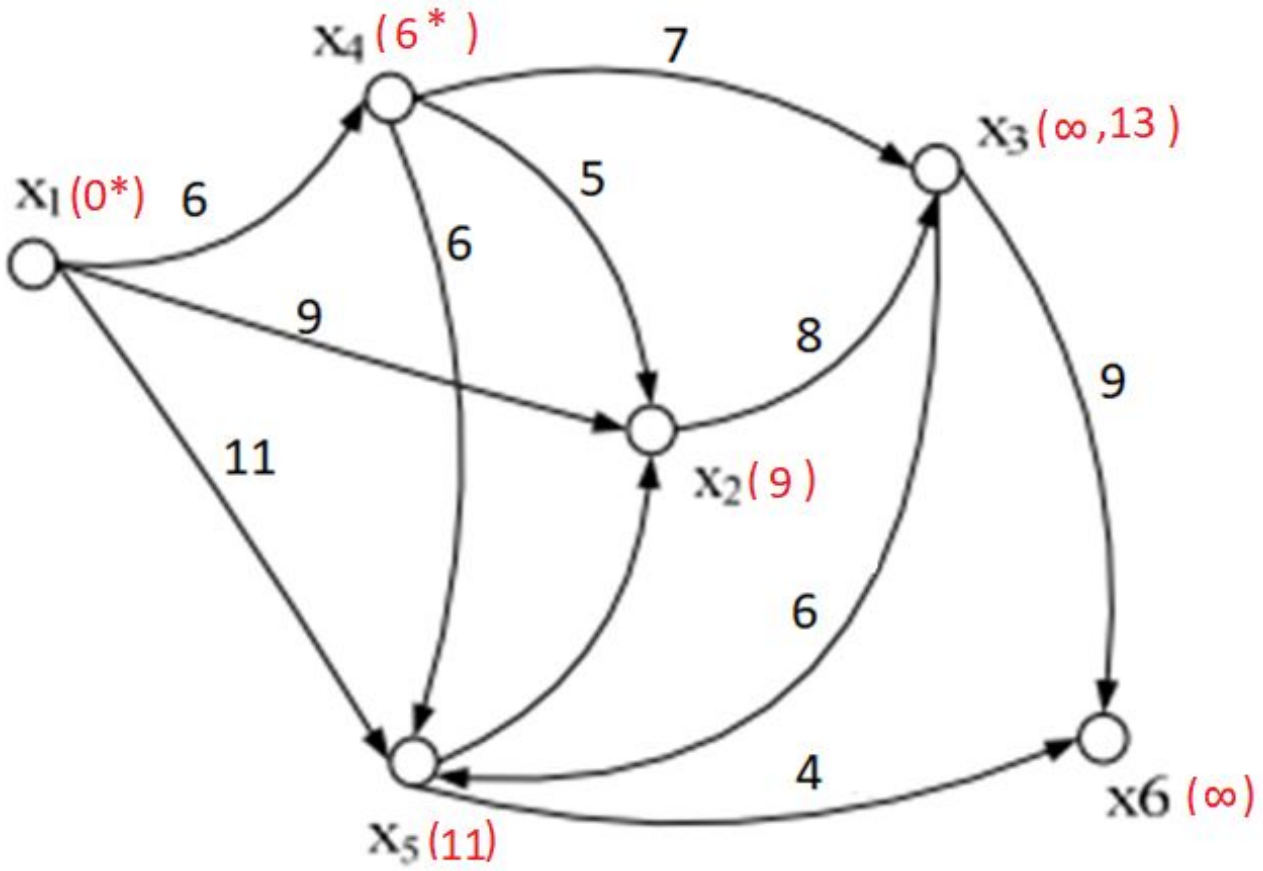
$S' = \{x_2, x_3, x_5\}$

Пересчитаем временные метки этих вершин:

$d(x_2) = \min\{9, 6^*+5\} = 9$

$d(x_3) = \min\{\infty, 6^*+7\} = 13$

$d(x_5) = \min\{11, 6^*+6\} = 11.$



## Шаг 3

$\min\{d(x_2),d(x_3),d(x_5),d(x_6)\} = \min\{9, 13, 11, \infty\}= 9^*= d(x_2)$,

$\widetilde{x} = x_2$.

## Шаг 4

$x_2 \neq x_6 \Rightarrow$ возвращение на шаг 2

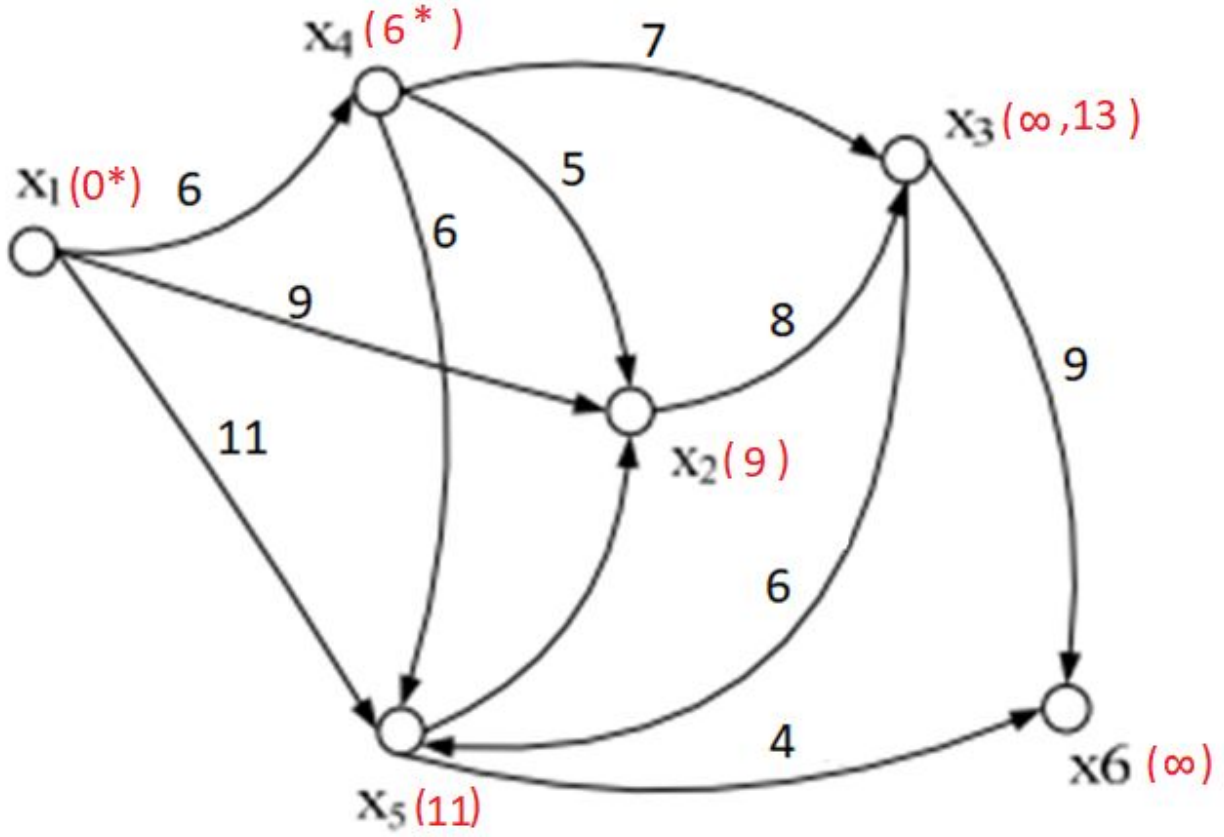
**3-я итерация**
**Шаг 2**

Множество вершин, непосредственно следующих за $\tilde{x} = x_2$

$S' = \{x_3\}$,

Пересчитаем временную метку этой вершины:

$d(x_3) = \min\{13, 9^*+8\} = 13$



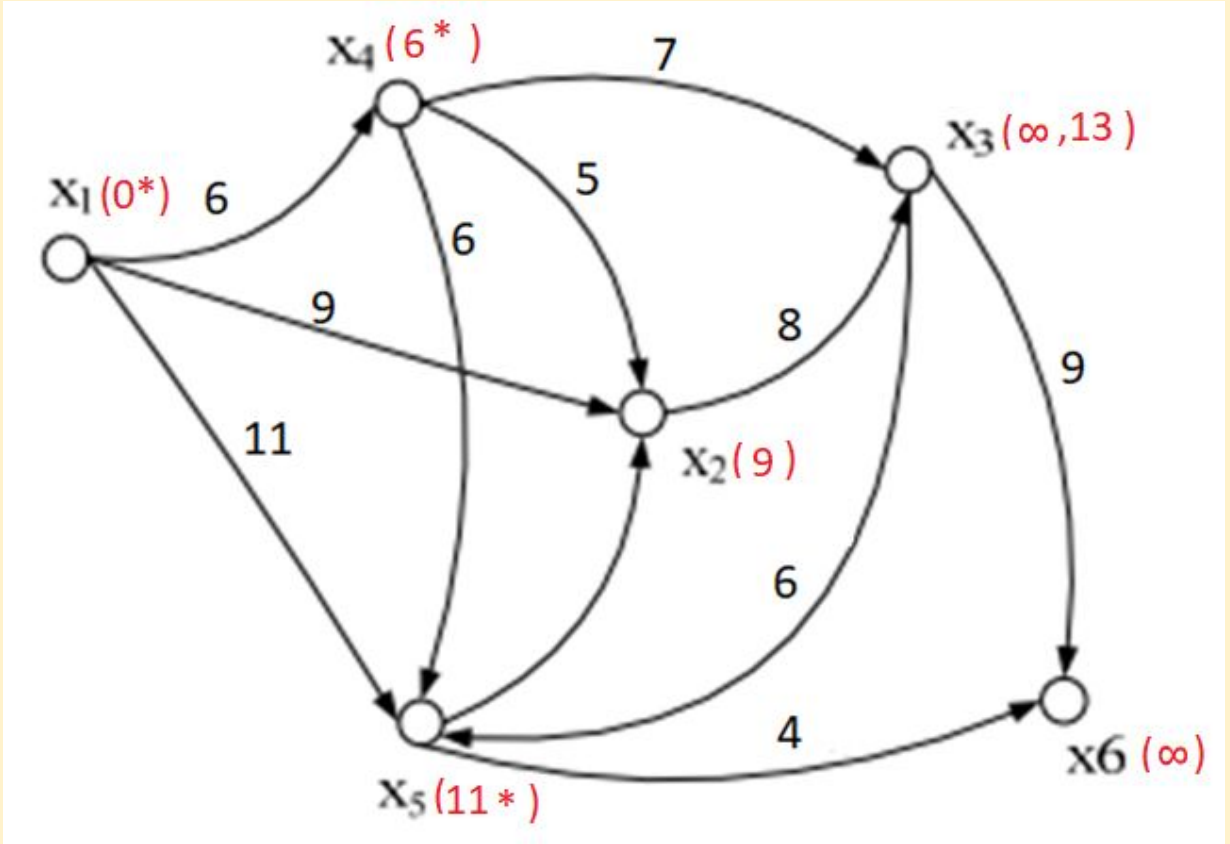
**Шаг 3.**

$\min\{d(x_3),d(x_5),d(x_6)\} = \min\{13,11,\infty\} = 11^* = d(x_5)$,

$\tilde{x} = x_5$.

**Шаг 4**

$x_5 \neq x_6$, возвращение на второй шаг.



# 4-я итерация

**Шаг 2**

S’ = {$x_6$}, d($x_6$) = min{∞,11*+4} = 15.

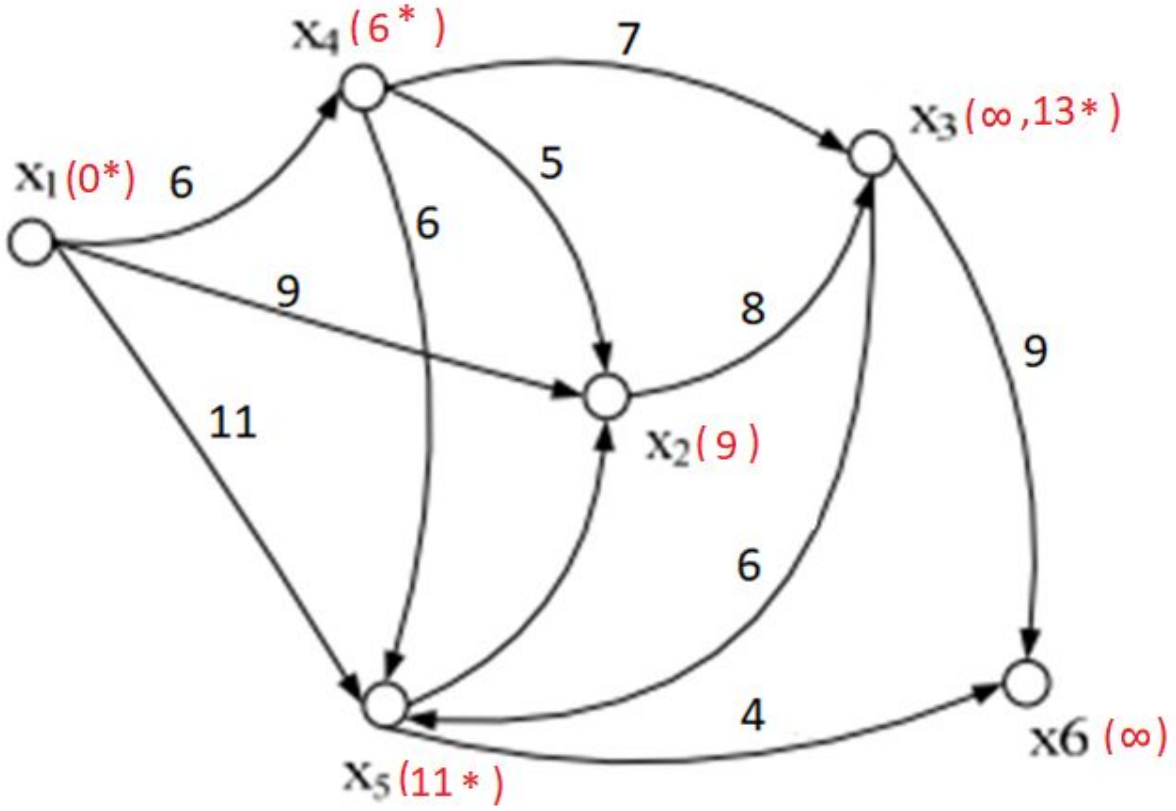
**Шаг 3**

min{d($x_3$),d($x_6$)} = min{13,15}= 13*= d($x_3$),

$\tilde{x}$ = $x_3$.

**Шаг 4**

$x_3 \neq x_6$, возвращение на второй шаг.



# 5-я итерация

## Шаг 2

$S' = \{x_6\}$, $d(x_6) = \min\{15, 13^*+9\} = 15$.

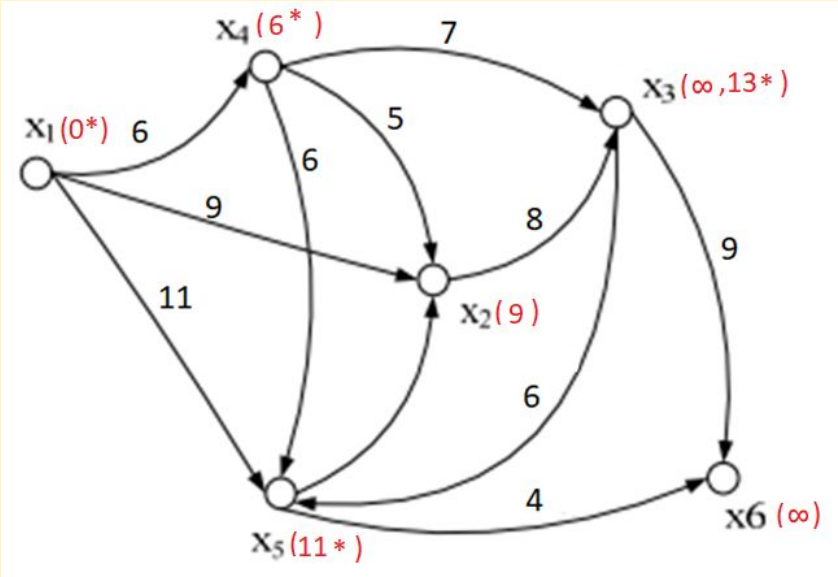
## Шаг 3

$\min\{d(x_6)\} = \min\{15\} = 15^*$, $\tilde{x} = x_6$.

## Шаг 4

$x_6 = t = x_6$, *конец первого этапа*.

# Этап 2




## 1-я итерация
## Шаг 5

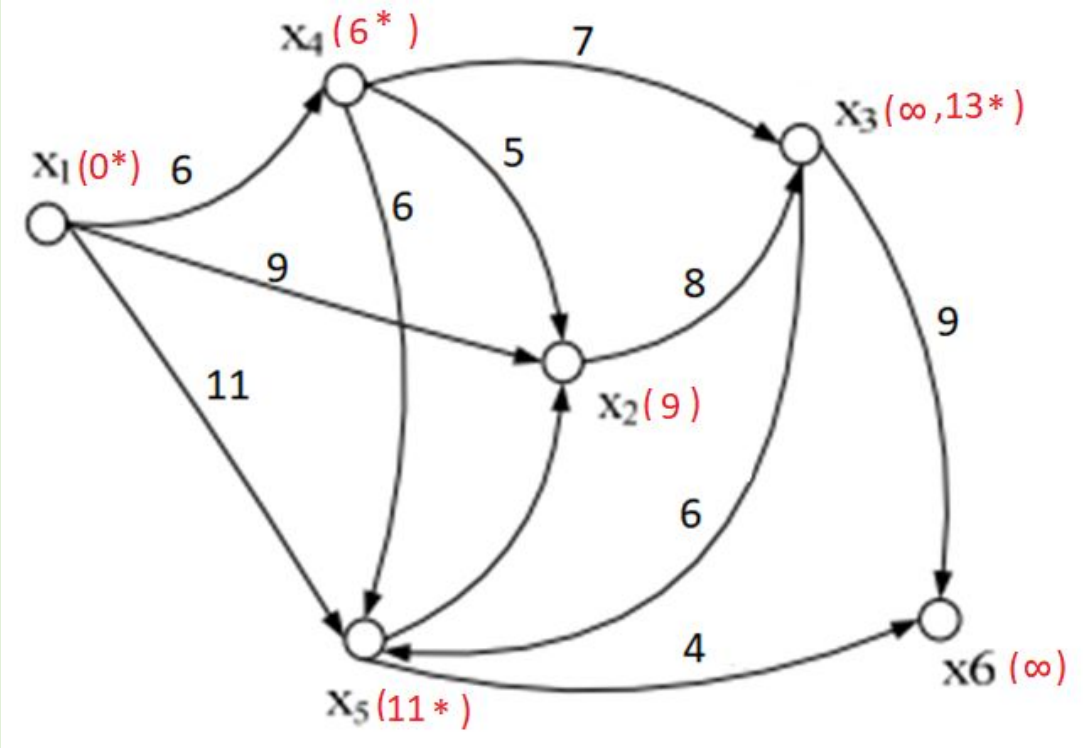
Составим множество вершин, непосредственно предшествующих вершине

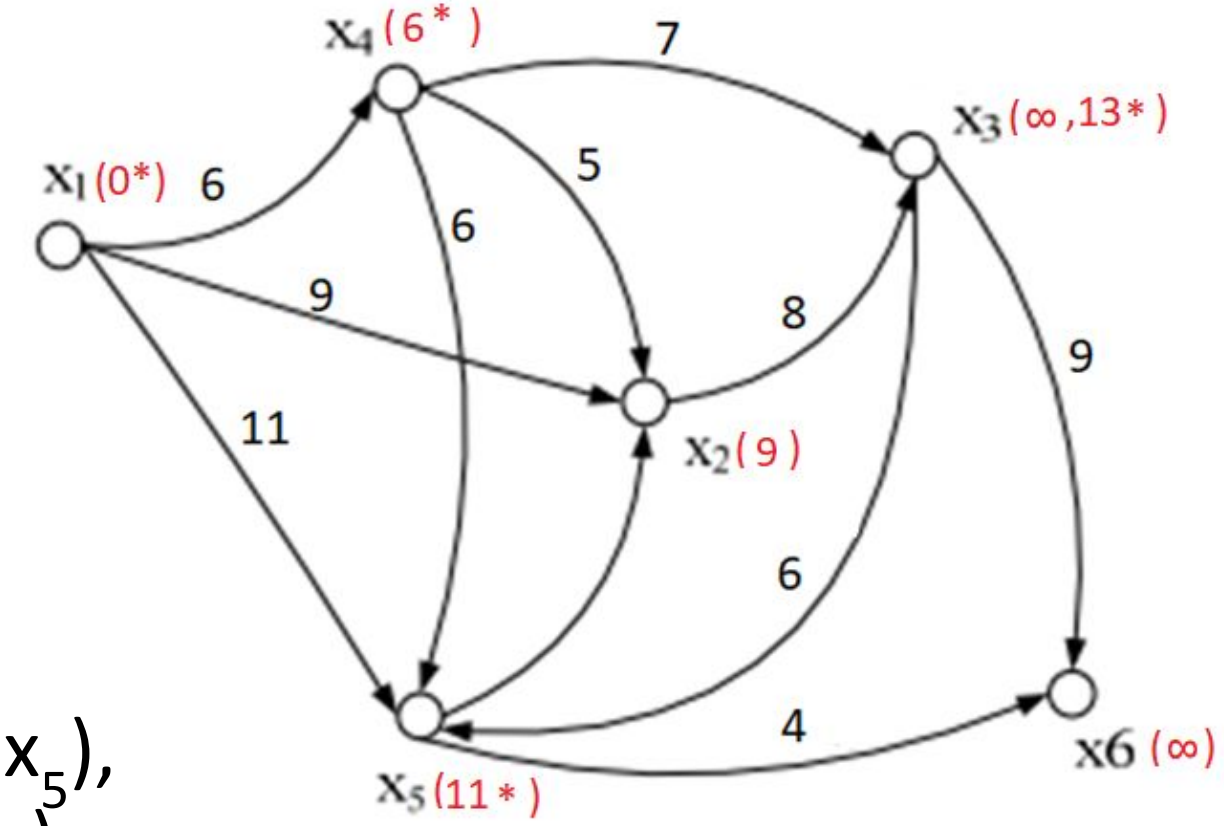
$\tilde{x}$ = $x_6$ с постоянными метками S'= {$x_3$, $x_5$}.

Проверим для этих двух вершин выполнение равенства (*): $d_{нов}(x_i) = \min\{d_{стар}(x_i), \tilde{d}(\quad) + \tilde{\omega}(\quad, x_i)\}$

$d(\tilde{x}) = 15 = 11^* + 4 = d(x_5) + \omega(x_5, x_6)$,
$d(\tilde{x}) = 15 \neq 13^* + 9 = d(x_3) + \omega(x_3, x_6)$.
Включим дугу $(x_5, x_6)$ в кратчайший путь.
$\tilde{x} = x_5$.

**Шаг 6**
$\tilde{x} \neq s = x_1 \Rightarrow$ на шаг 5.






## 2-я итерация

## Шаг 5

S’ = {$x_1$, $x_4$},

d($\widetilde{x}$) = 11=0*+11 = d($x_1$)+ ($x_1$, $x_5$),

d($\widetilde{x}$) = 11≠6*+6 = d($x_4$)+ ($x_4$, $x_5$).

Включаем дугу ($x_1$, $x_5$) в кратчайший путь.

$\widetilde{x}$ = $x_1$.

**Шаг 6**

$\tilde{x}$=s= $x_1$, завершение второго этапа.

Кратчайший путь от вершины $x_1$ до вершины $x_6$:

$\mu = (x_1, x_5) – (x_5, x_6)$

$d_{min} = 11+4=15$

**Ответ:**

$\mu = (x_1, x_5) – (x_5, x_6)$

$d_{min} = 15$

# 3. Алгоритм Беллмана-Мура (поиска кратчайших путей)

Если веса – произвольные числа (в т.ч. отрицательные), то кратчайший путь можно найти по алгоритму Беллмана-Мура (алгоритму корректировки меток).

Как и в алгоритме Дейкстры, всем вершинам приписываются метки.

Однако здесь нет деления на временные и постоянные.

Вместо процедуры превращения временной метки в постоянную формируется очередь вершин.

В процессе работы алгоритма одна и та же вершина может несколько раз вставать в очередь, а затем покидать ее.

# Этапы алгоритма Беллмана-Мура

I этап. Поиск длины кратчайших путей от вершины s до всех остальных вершин. Этап завершается при отсутствии вершин в очереди

II этап. Построение кратчайшего пути от s до t совпадает с соответствующим этапом в алгоритме Дейкстры и выполняется по формуле

$$d(\tilde{x}) = d(x_i) + \omega(x_i, \tilde{x})$$

# I этап. Поиск длины кратчайших путей от вершины s до всех остальных вершин

Шаг 1. Присвоение начальных значений.

$d(s)=0$, $d(x_j)=\infty$, $x_j \in S$, $\tilde{x} = s$

$Q = \{\tilde{x}\}$ - множество вершин в очереди

# Шаг 2. Корректировка меток и очереди

Удалить из очереди Q вершину, находящуюся в самом начале очереди. Для каждой вершины $x_i$, непосредственно достижимой из $\tilde{x}$

**корректируем** ее метку по формуле

$$d_{нов}(x_i) = \min\{d_{стар}(x_i), d(\tilde{x}) + \omega(\tilde{x}, x_i)\}$$

Если при этом $d_{нов}(x_i) < d_{стар}(x_i)$, то корректируем очередь Q вершин.

Иначе продолжаем перебор вершин и корректировку временных меток.

# Шаг 2. Корректировка меток и очереди

## Корректировка очереди

Если вершина $x_i$ не была ранее в очереди и не находится в ней в данный момент, то ставим ее в конец очереди.

Если вершина $x_i$ уже была когда-нибудь ранее в очереди или находится там в данный момент, то переставляем ее в начало очереди

# Шаг 3. Проверка на завершение первого этапа

Если $Q \neq \varnothing$, то возвращаемся к началу шага 2.

Если $Q = \varnothing$, то первый этап завершен, т.е. минимальные расстояния от s до всех вершин графа найдены

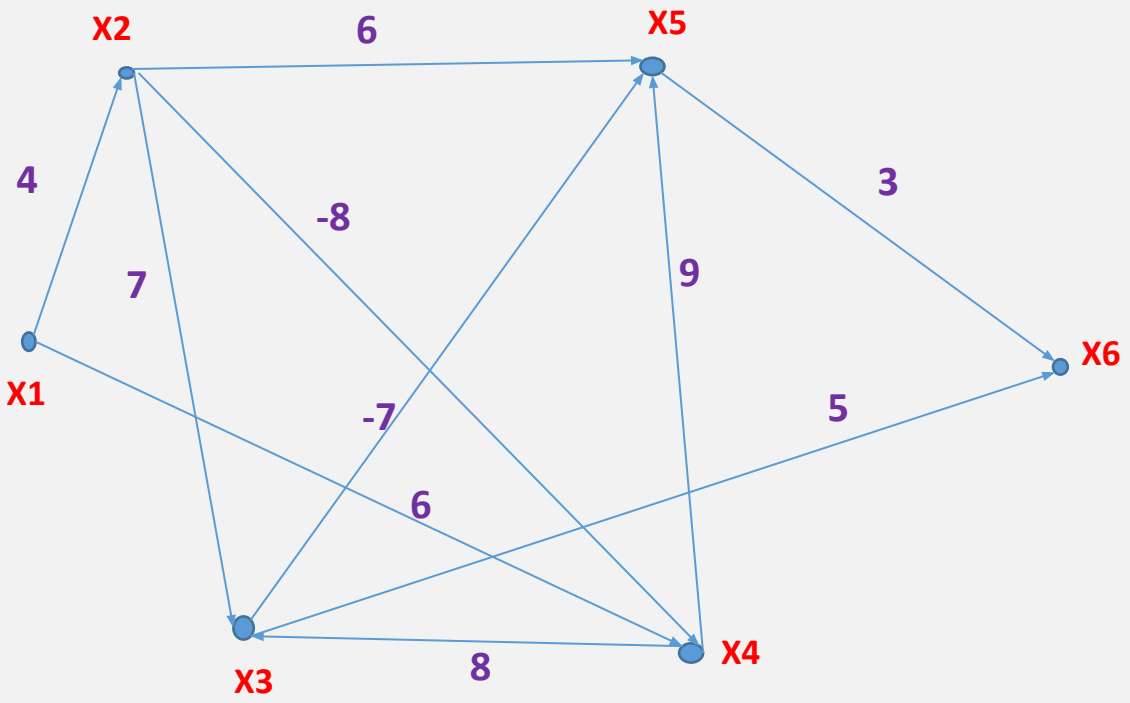
# II этап

Построение кратчайшего пути от s до t совпадает с соответствующим этапом в алгоритме Дейкстры и выполняется по формуле

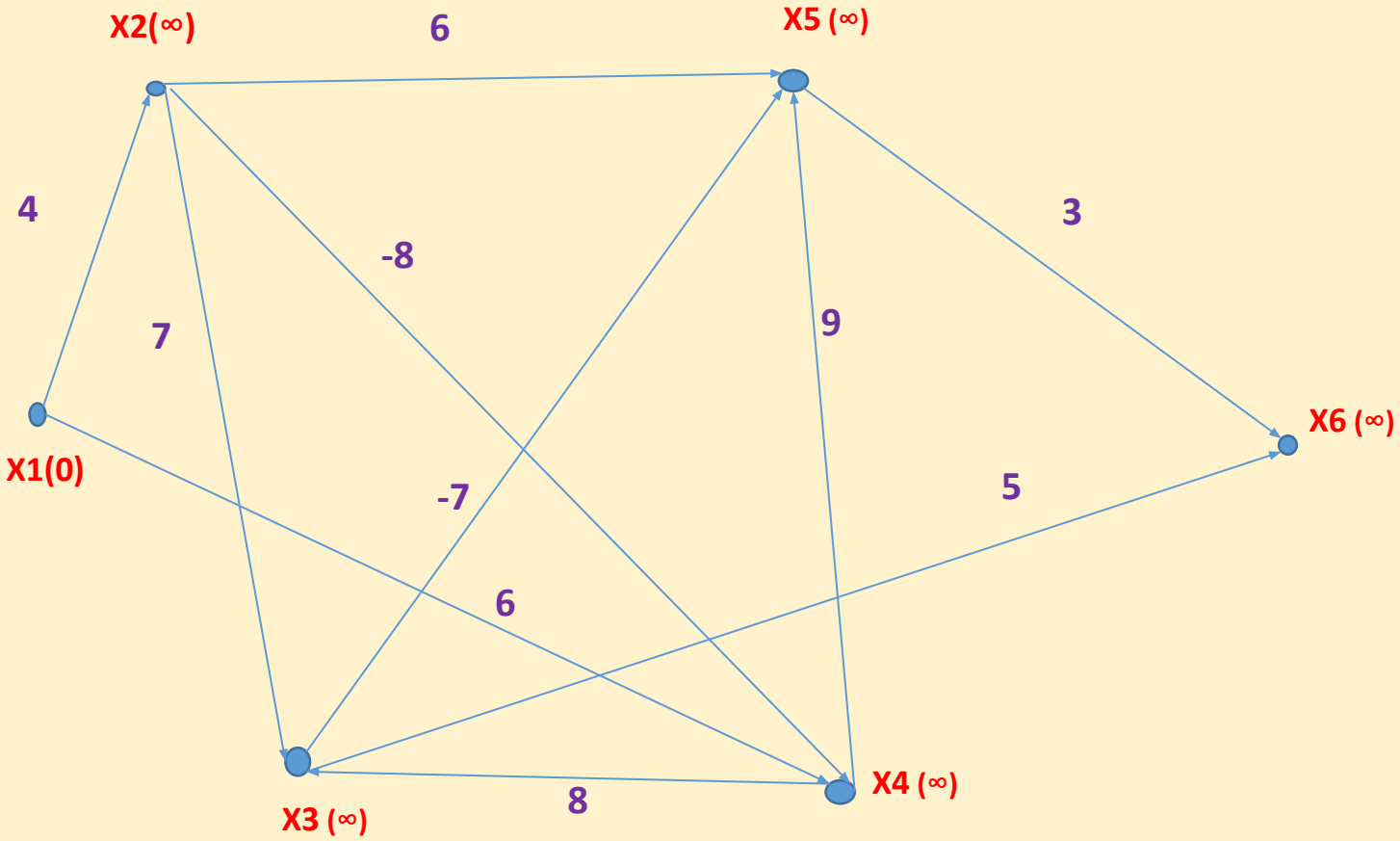
$$d(\tilde{x}) = d(x_i) + \omega(x_i, \tilde{x})$$

# Пример. Граф G задан весовой матрицей Ω

$$\Omega = \begin{array}{c} \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{array} \begin{array}{c} \begin{array}{cccccc} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \end{array} \\ \begin{pmatrix} - & 4 & \infty & 6 & \infty & \infty \\ \infty & - & 7 & -8 & 6 & \infty \\ \infty & \infty & - & \infty & -7 & 5 \\ \infty & \infty & 8 & - & 9 & \infty \\ \infty & \infty & \infty & \infty & - & 3 \\ \infty & \infty & \infty & \infty & \infty & - \end{pmatrix} \end{array}.$$



# Этап I

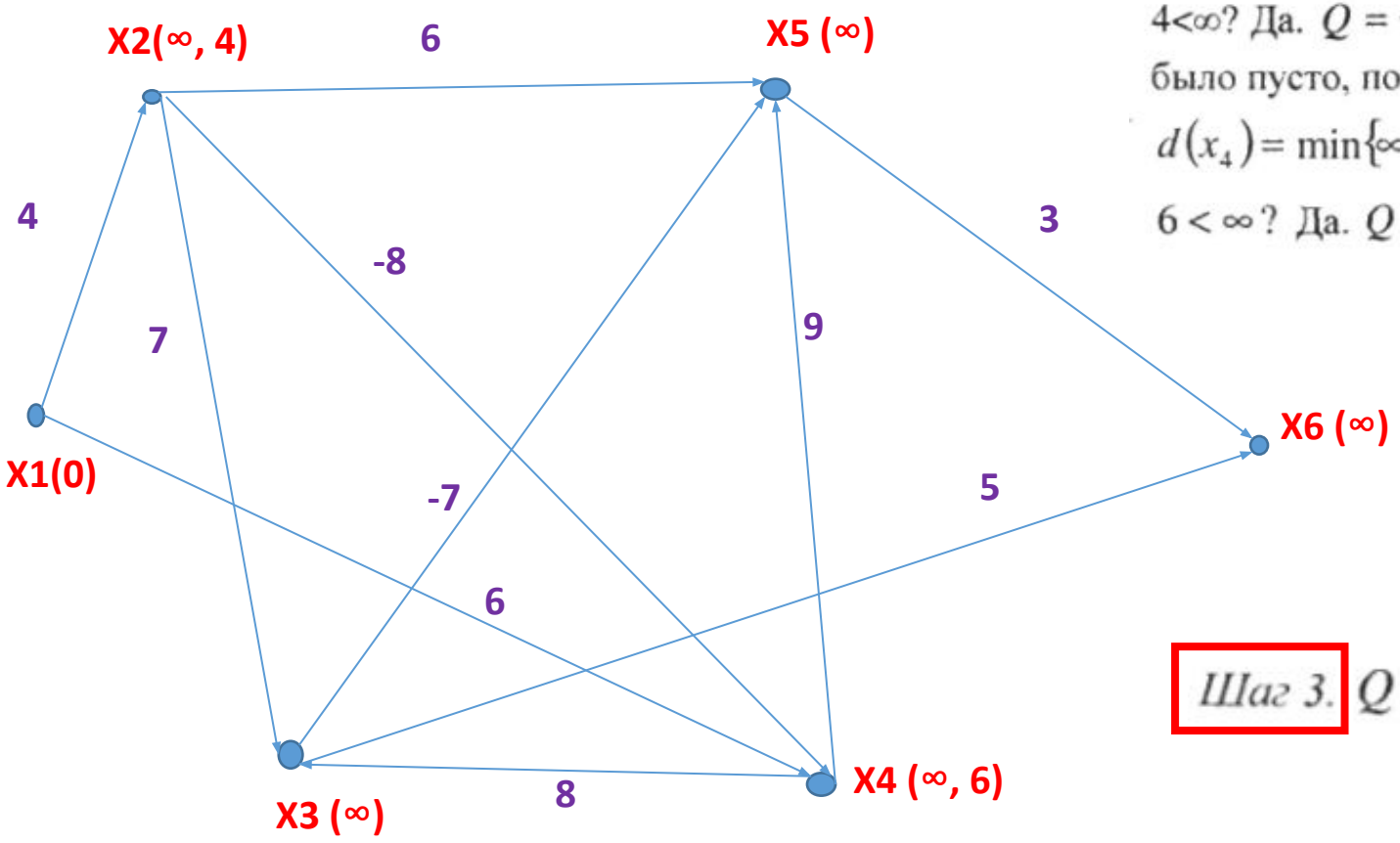



## Шаг 1. $\tilde{x} = x_1$

$d(x_1)=0$, $d(x_2)=\infty$, $d(x_3)=\infty$, $d(x_4)=\infty$, $d(x_5)=\infty$, $d(x_6)=\infty$

$Q=\{x_1\}$ - очередь

# Этап I. 1-я итерация




*Шаг 2.* $\tilde{x} = x_1$, $Q = Q \setminus \{x_1\} = \varnothing$. Пусть $\tilde{S}$ — множество вершин, непосредственно достижимых из вершины $\tilde{x}$.

$\tilde{S} = \{x_2, x_4\}$, $d(x_2) = \min\{\infty, 0+4\} = 4$.

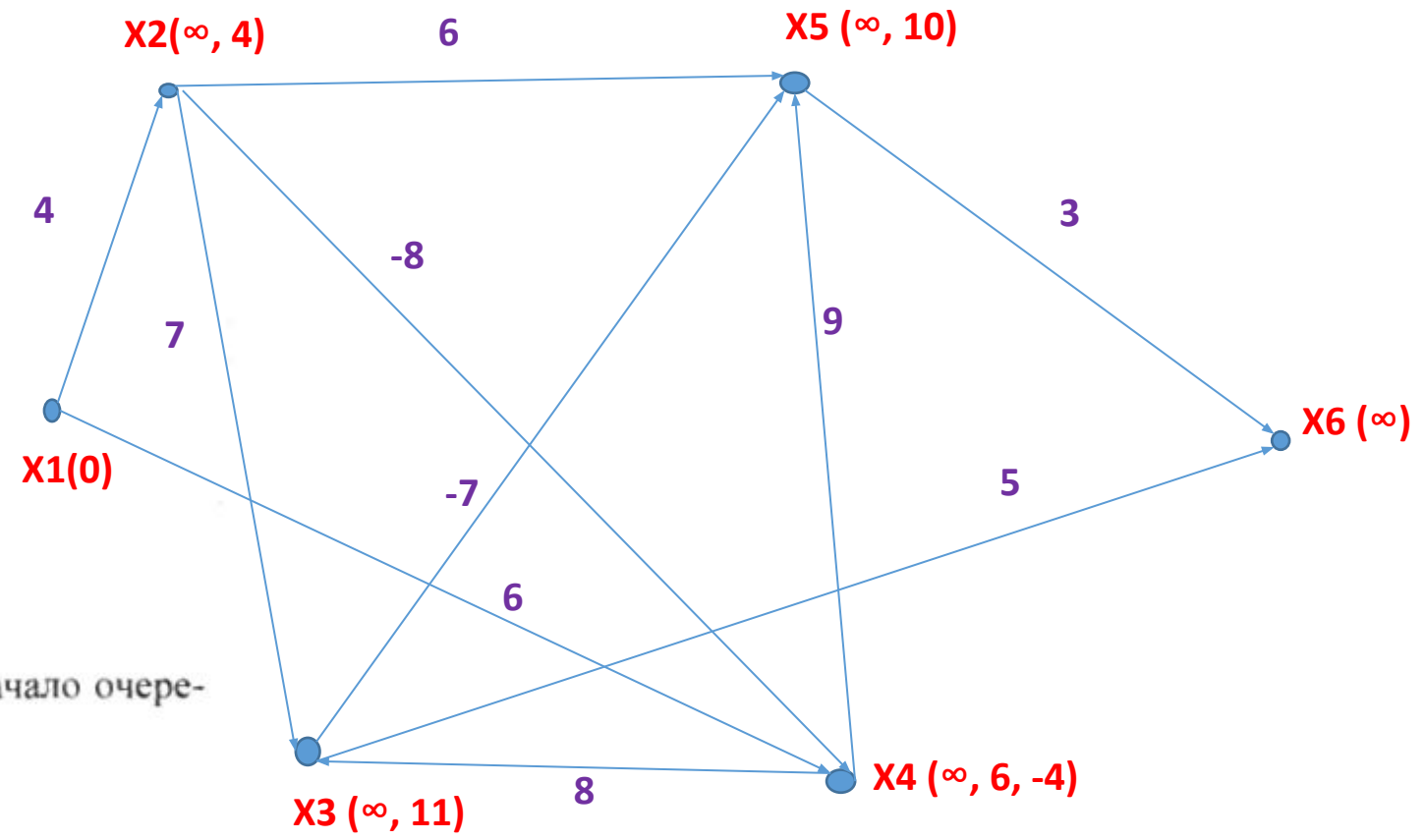
$4<\infty$? Да. $Q = \{x_2\}$. $x_2$ надо было поставить в конец очереди, но $Q$ было пусто, поэтому конец очереди совпал с началом.

$d(x_4) = \min\{\infty, 0+6\} = 6$.

$6 < \infty$? Да. $Q = \{x_2, x_4\}$.

*Шаг 3.* $Q = \varnothing$? Нет. Переход на начало второго шага.

# 2-я итерация




*Шаг 2.* $\tilde{x} = x_2,\ Q = Q \setminus \{\tilde{x}\} = \{x_4\}$,

$\tilde{S} = \{x_3, x_4, x_5\}.\ d(x_3) = \min\{\infty, 4+7\} = 11.$

$11 < \infty$? Да. $Q = \{x_4, x_3\}$.

$d(x_4) = \min\{6, 4-8\} = -4.$

$-4 < 6$? Да. $Q = \{x_4, x_3\}$. Вершину $x_4$ надо поставить в начало очереди, но она уже стоит там.

$d(x_5) = \min\{\infty, 4+6\} = 10.$

$10 < \infty$? Да. $Q = \{x_4, x_3, x_5\}$.

*Шаг 3.* $Q = \varnothing$? Нет, переход на третью итерацию второго шага.

# 3-я итерация

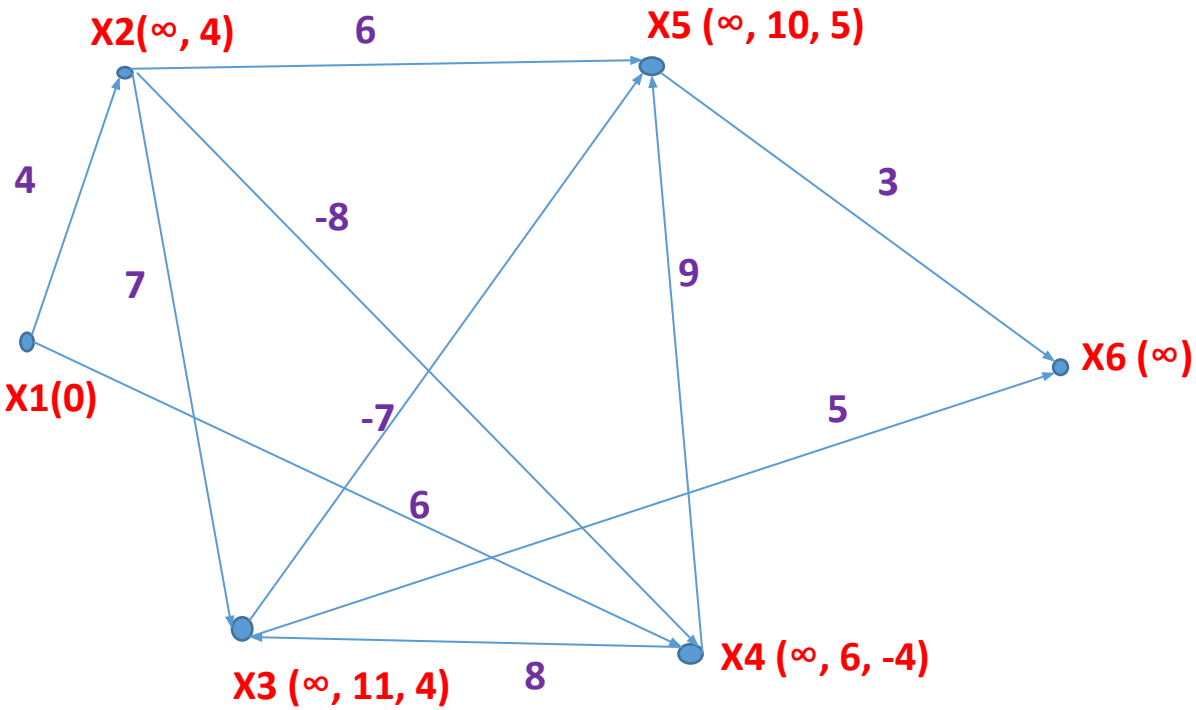
*Шаг 2.* $\tilde{x} = x_4$, $Q = Q \setminus \{\tilde{x}\} = \{x_3, x_5\}$,

$\tilde{S} = \{x_3, x_5\}$ $d(x_3) = \min\{11, -4+8\} = 4$.

4<11? Да. $Q = \{x_3, x_5\}$. Вершину $x_3$ надо поставить в начало очереди, но она уже там.

$d(x_5) = \min\{10, -4+9\} = 5$.

5<10? Да. $Q = \{x_5, x_3\}$. Вершину $x_5$ передвигаем из конца очереди в начало.




*Шаг 3.* $Q = \varnothing$? Нет, переход на четвертую итерацию второго шага.

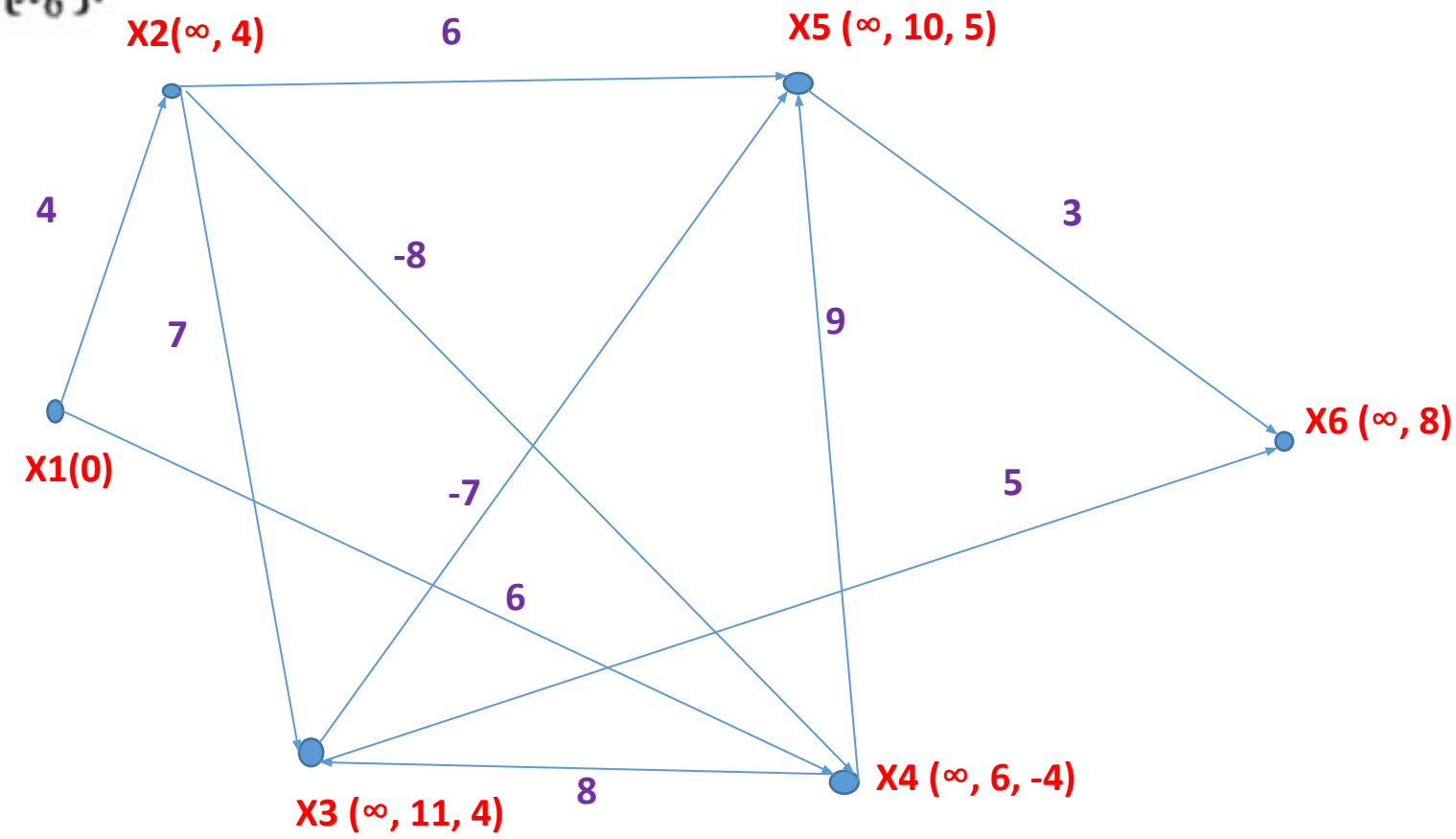
# 4-я итерация

*Шаг 2.* $\tilde{x} = x_5$, $Q = Q \setminus \{\tilde{x}\} = \{x_3\}$, $\tilde{S} = \{x_6\}$.

$d(x_6) = \min\{\infty, 5+3\} = 8.$

$8 < \infty$? Да. $Q = \{x_3, x_6\}$.

*Шаг 3.* $Q = \varnothing$? Нет.



# 5-я итерация

*Шаг 2.* $\tilde{x} = x_3$, $Q = Q \setminus \{\tilde{x}\} = \{x_6\}$, $\tilde{S} = \{x_5, x_6\}$.

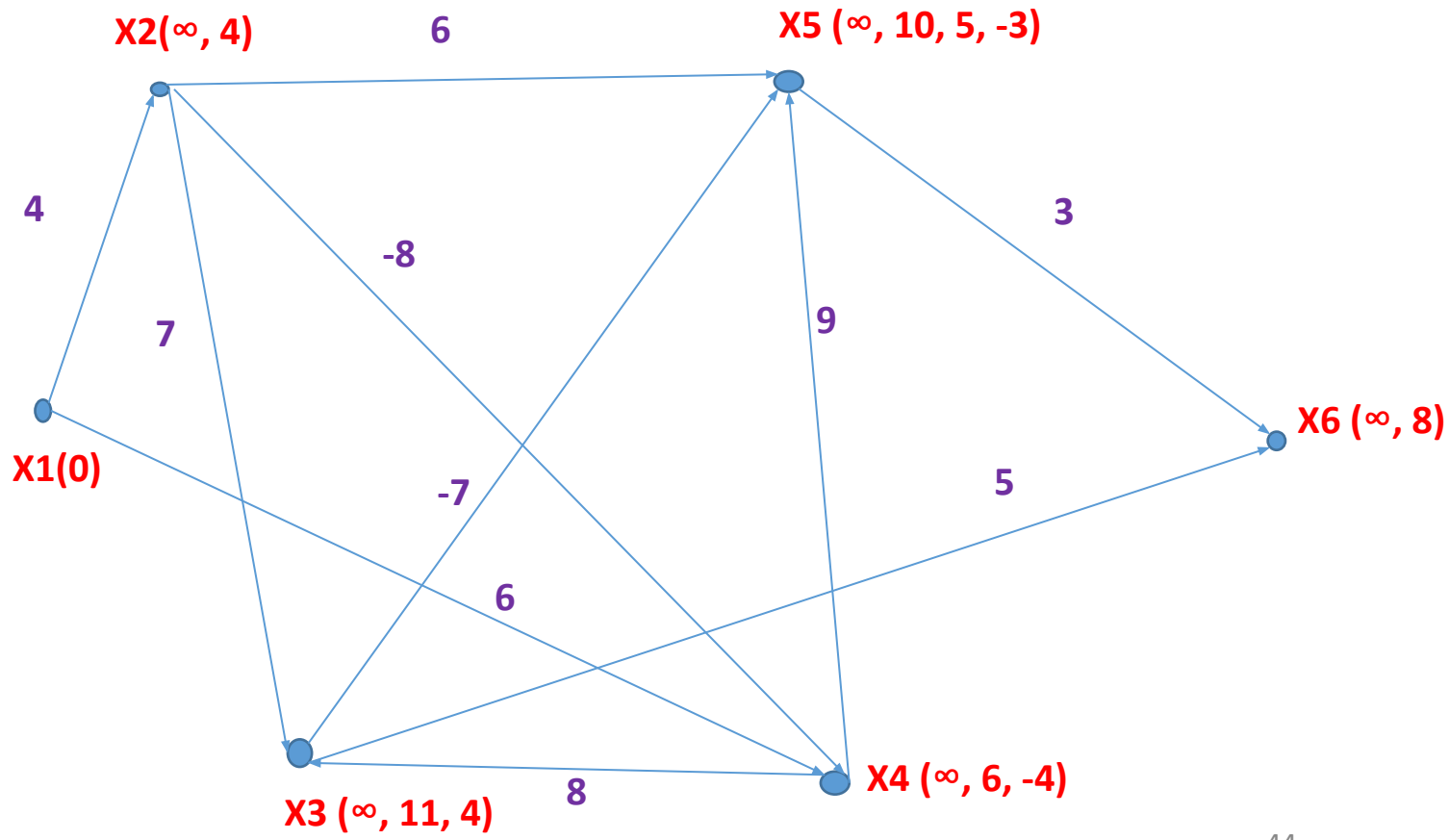
$d(x_5) = \min\{5, 4-7\} = -3.$

-3<5? Да. $Q = \{x_5, x_6\}$.

$d(x_6) = \min\{8, 4+5\} = 8.$

9<8? Нет.

Шаг 3. $Q = \varnothing$? Нет.



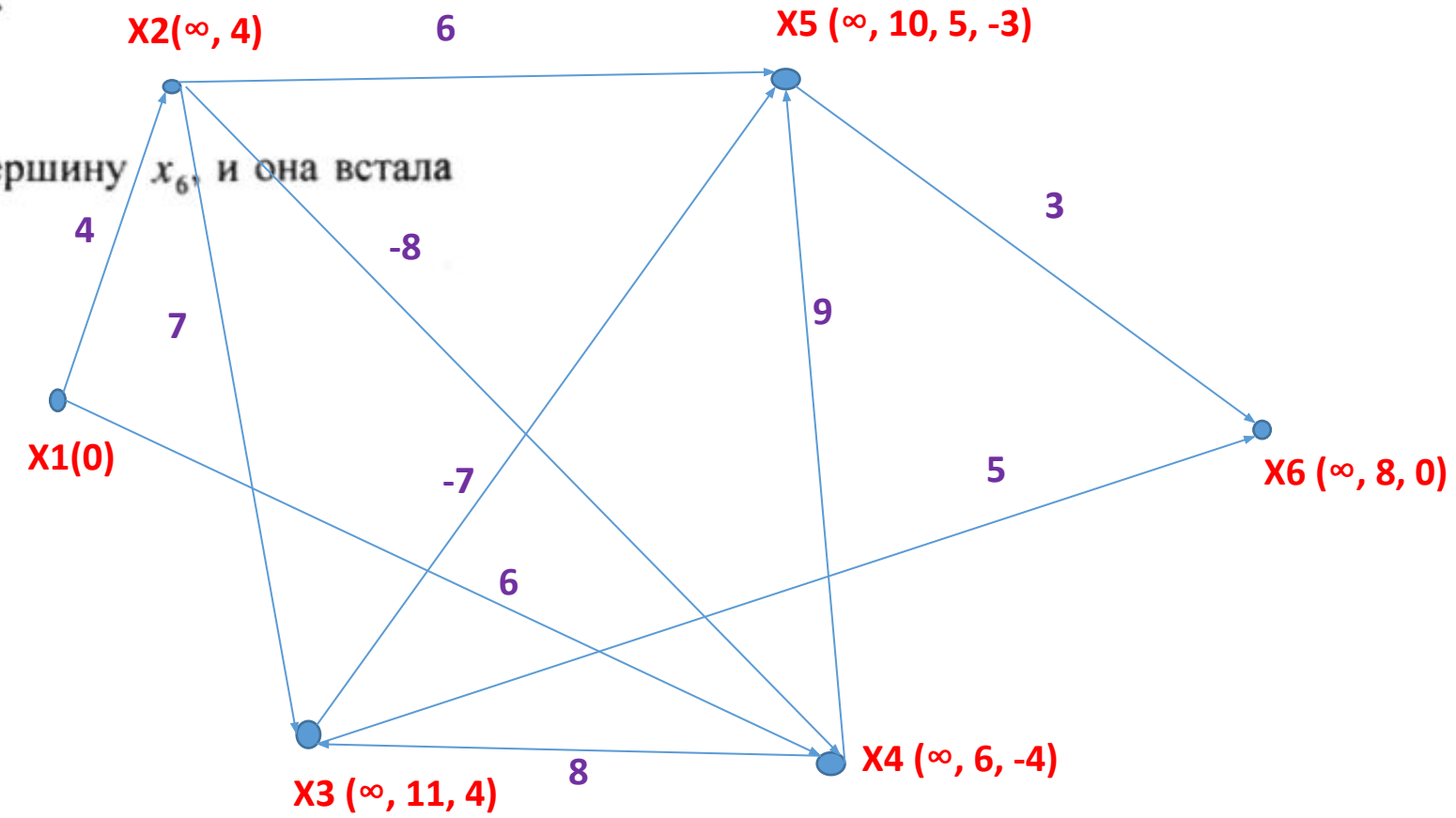
# 6-я итерация

*Шаг 2.* $\tilde{x} = x_5$, $Q = Q \setminus \{\tilde{x}\} = \{x_6\}$, $\tilde{S} = \{x_6\}$.

$d(x_6) = \min\{8, -3+3\} = 0.$

$0<8$? Да. $Q = \{x_6\}$. $Q$ содержало только вершину $x_6$, и она встала в начало очереди.

*Шаг 3.* $Q = \varnothing$? Нет.



# 7-я итерация

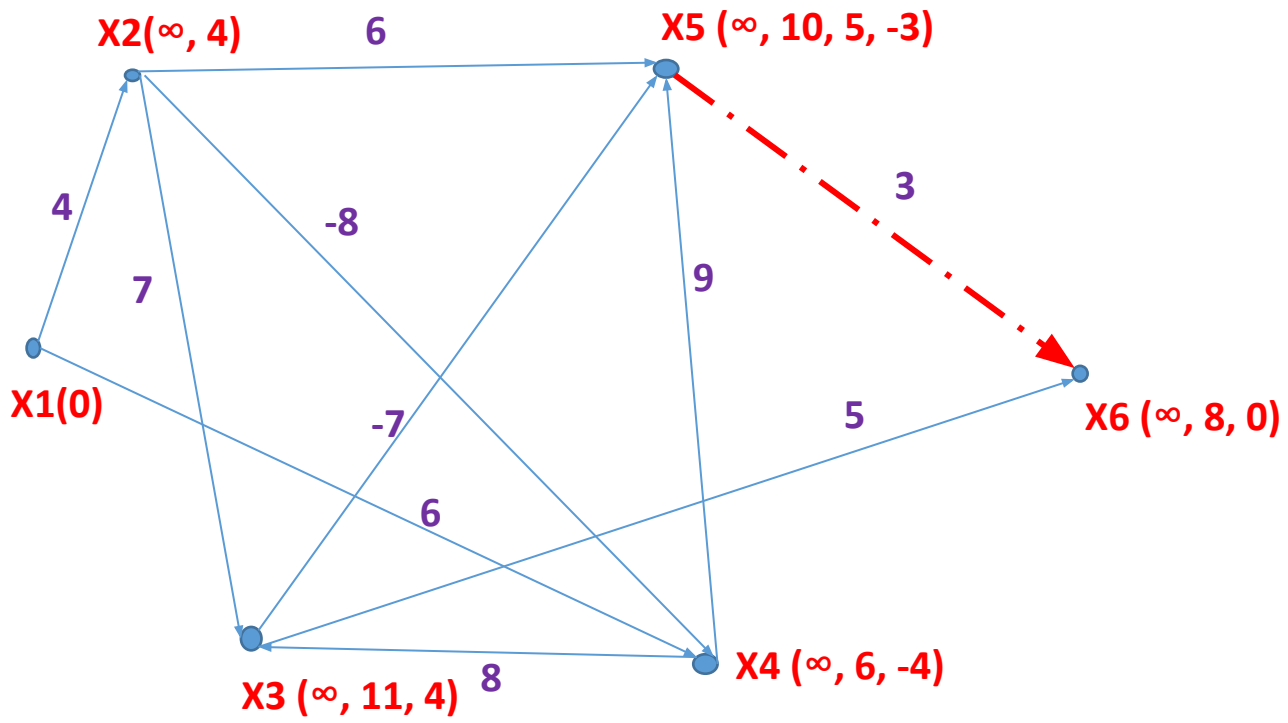
*Шаг 2.* $\tilde{x} = x_6,\ Q = Q \setminus \{\tilde{x}\} = \varnothing.\ \tilde{S} = \varnothing.$

*Шаг 3.* $Q = \varnothing$. Конец первого этапа. Найдены минимальные расстояния до всех вершин от первой. Эти расстояния таковы:
$d(x_2) = 4,\ d(x_3) = 4,\ d(x_4) = -4,\ d(x_5) = -3,\ d(x_6) = 0.$

# Этап II. Шаг 4. 1-я итерация

$$d(\tilde{x}) = d(x_6) = 0 \neq 4 + 5 = d(x_3) + \omega(x_3, x_6)$$ или

$d(\tilde{x}) = d(x_6) = 0 = -3 + 3 = d(x_5) + \omega(x_5, x_6)$. Включаем дугу $(x_5, x_6)$ в кратчайший путь. $\tilde{x} = x_5$. Возвращаемся на четвертый шаг.



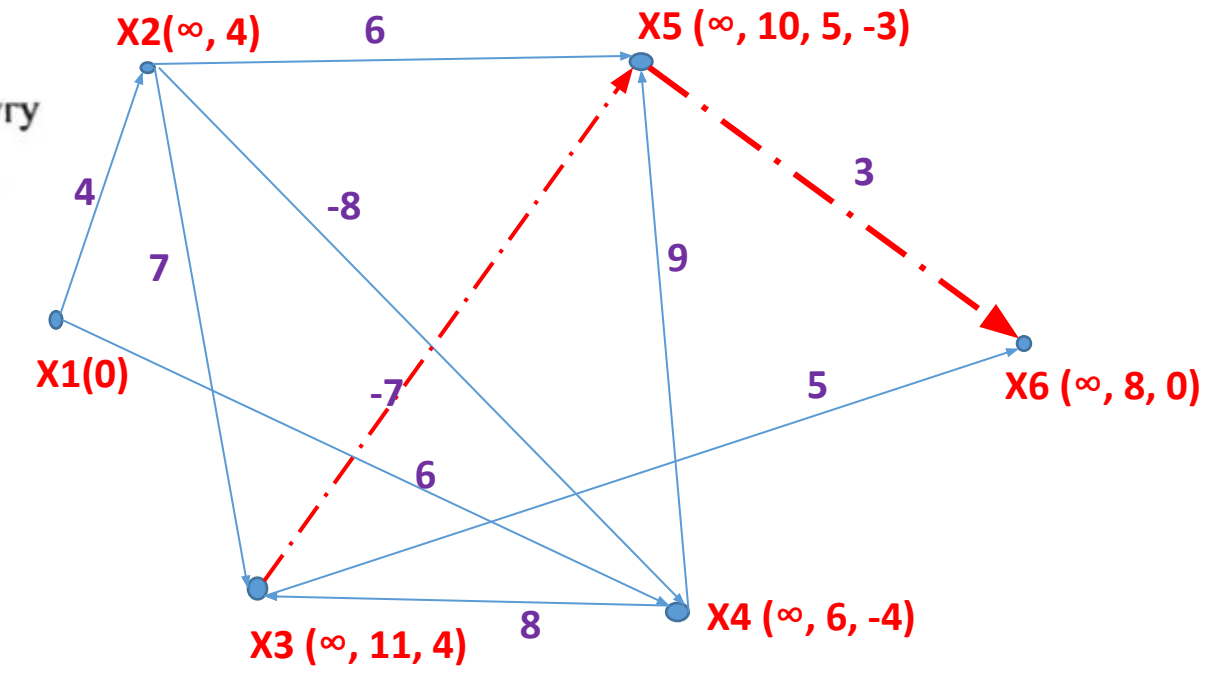
# Этап II. Шаг 4. 2-я итерация

$\tilde{x} = s$ ? Нет.

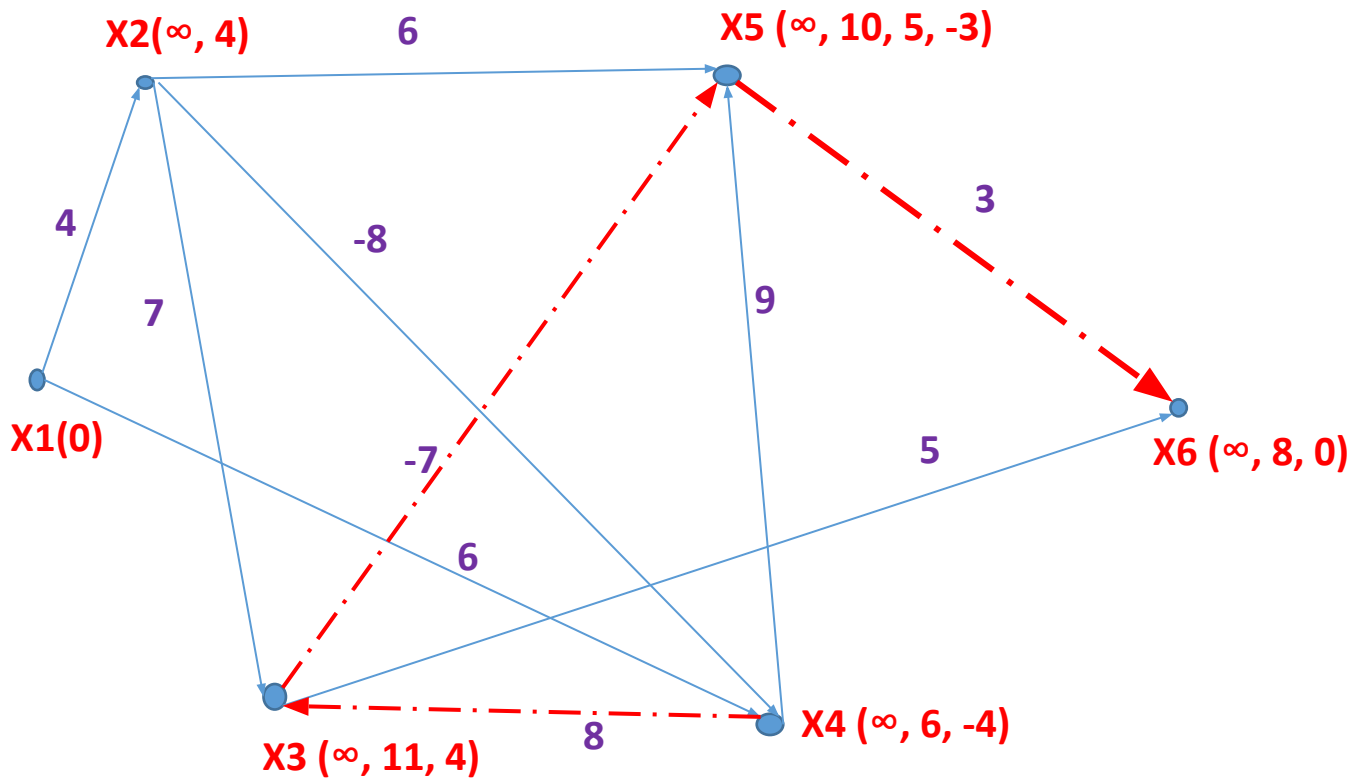
$\tilde{S} = \{x_2, x_3, x_4\}$. $d(\tilde{x}) = d(x_5) = -3 = 4 - 7 = d(x_3) + \omega(x_3, x_5)$,

$d(\tilde{x}) = d(x_5) = -3 \neq 4 + 6 = d(x_2) + \omega(x_2, x_5)$,

$d(\tilde{x}) = d(x_5) = -3 \neq -4 + 9 = d(x_4) + \omega(x_4, x_5)$. Включаем дугу $(x_3, x_5)$ в кратчайший путь. $\tilde{x} = x_3$. Возвращаемся на четвертый шаг.



# Этап II. Шаг 4. 3-я итерация

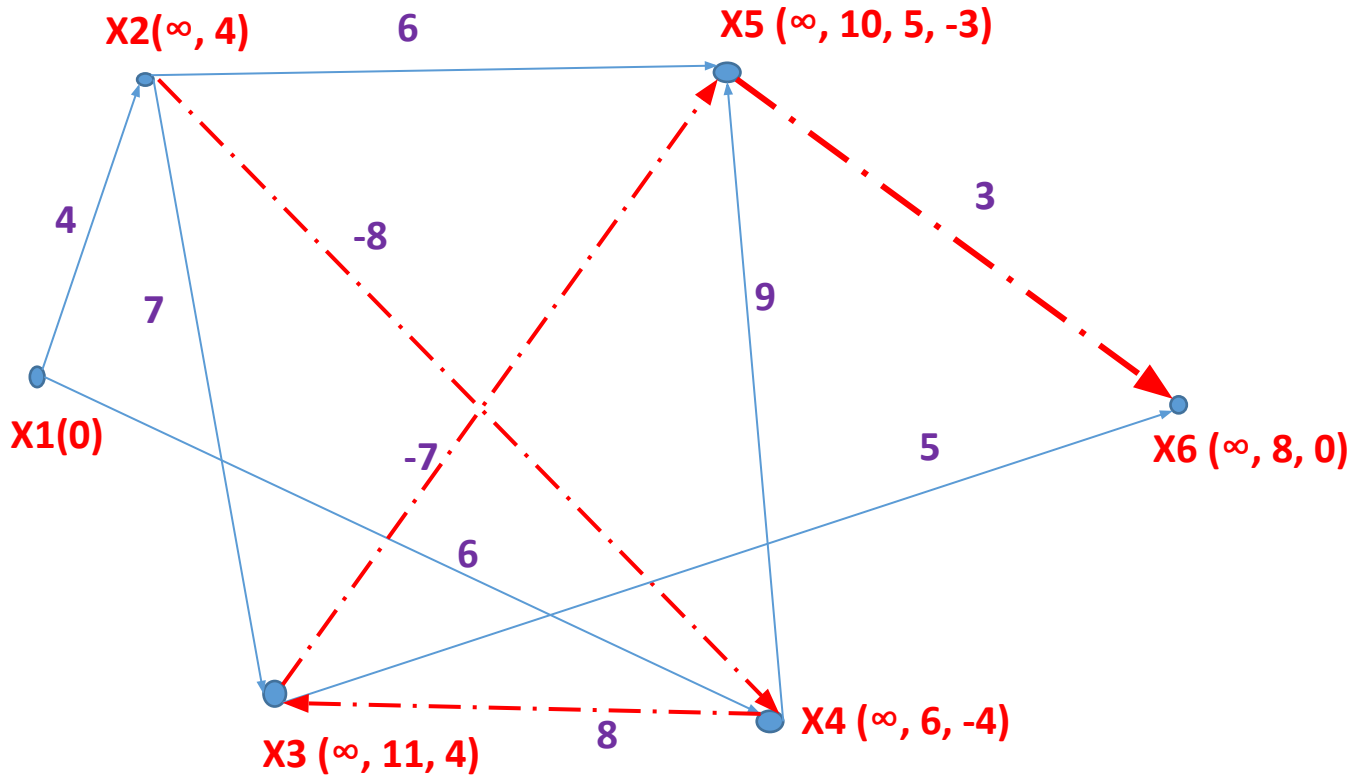



$\tilde{x} = s$ ? Нет. $\tilde{S} = \{x_2, x_4\}$.

$d(\tilde{x}) = d(x_3) = 4 \neq 4 + 7 = d(x_2) + \omega(x_2, x_3)$,

$d(\tilde{x}) = d(x_3) = 4 = -4 + 8 = d(x_4) + \omega(x_3, x_4)$. Включаем дугу $(x_3, x_4)$ в кратчайший путь. $\tilde{x} = x_4$. Возвращаемся на четвертый шаг.

# Этап II. Шаг 4. 4-я итерация

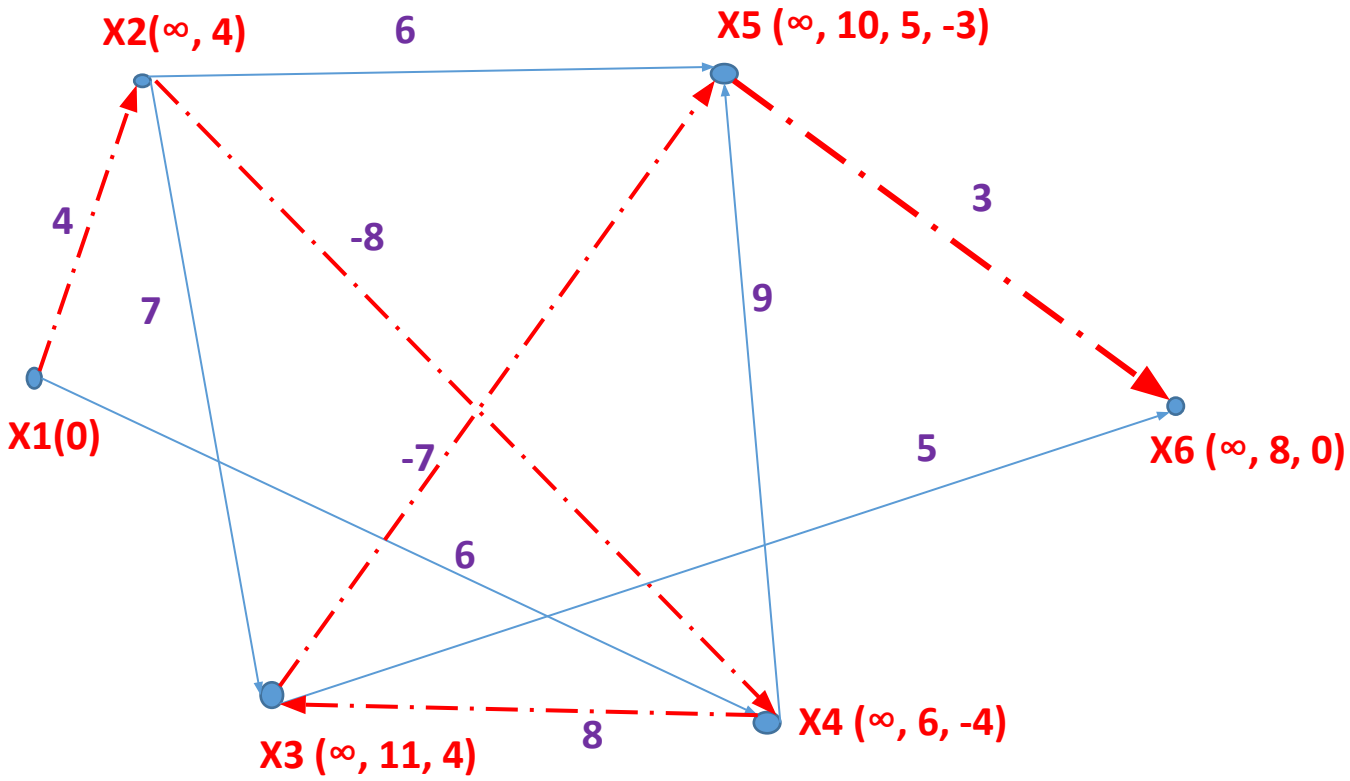



$\tilde{x} = s$? Нет. $\tilde{S} = \{x_1, x_2\}$.

$d(\tilde{x}) = d(x_4) = -4 \neq 0 + 6 = d(x_1) + \omega(x_1, x_4)$,

$d(\tilde{x}) = d(x_4) = -4 = 4 - 8 = d(x_2) + \omega(x_2, x_4)$. Включаем дугу $(x_2, x_4)$ в кратчайший путь. $\tilde{x} = x_2$. Возвращаемся на четвертый шаг.

# Этап II. Шаг 4. 5-я итерация




$\tilde{x} = s$? Нет. $\tilde{S} = \{x_1\}$.

$d(\tilde{x}) = d(x_2) = 4 = 0 + 4 = d(x_1) + \omega(x_1, x_2)$. Включаем дугу $(x_1, x_2)$ в кратчайший путь. $\tilde{x} = x_1$. Возвращаемся на четвертый шаг.

# Этап II. Шаг 4. 6-я итерация

$$\tilde{x} = s = x_1 = ?$$

Да. Задача решена.

Искомый кратчайший путь от вершины x1 до $x_6$ имеет вес

d=4-8+8-7+3=0

и состоит из дуг $(x_1, x_2)$ - $(x_2, x_4)$ – $(x_4, x_3)$ – $(x_3, x_5)$ – $(x_5, x_6)$

**Ответ:**

$d_{min}$=0, μmin=$(x_1, x_2)$ - $(x_2, x_4)$ – $(x_4, x_3)$ – $(x_3, x_5)$ – $(x_5, x_6)$

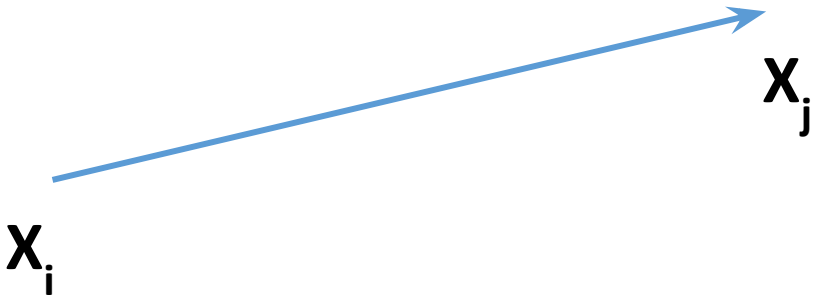
# 4. Алгоритм нахождения максимального пути

**Замечания**

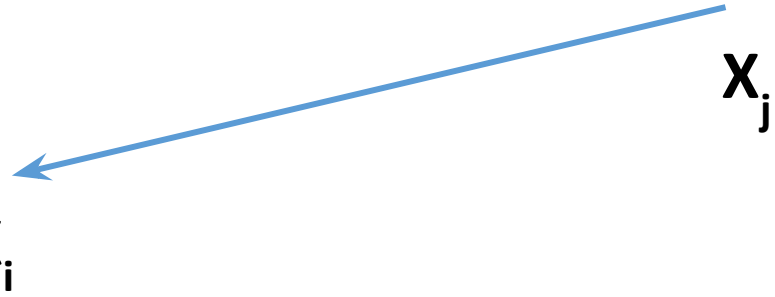
1. Граф G (сеть) должен быть **ациклическим**.

Если G - ациклический граф, то для любых двух вершин $x_i \neq x_j$ выполняется одно из условий:

1) $x_i$ предшествует $x_j$; $x_i \in S_{предш}(x_j)$;



2) $x_i$ следует за $x_j$; $x_i \in S_{след}(x_j)$;




3) нет пути между $x_i$ и $x_j$.

2. Упорядочить вершины по алгоритму Фалкерсона.

# Этап 1. Поиск длины максимального пути

- Пусть $d_j$ - длина максимального пути от вершины $x_1$ до вершины $x_j$

$$\begin{cases} d_1 = 0, \\ d_j = \max\left(d_i + \omega_{ij} \mid x_i \in S_{pred}(x_j), j = 2,3,\ldots,k+1\right), \\ d_j = \infty, j = k+2, k+3,\ldots,n. \end{cases} \quad (**)$$

# Этап 2. Построение максимальных путей

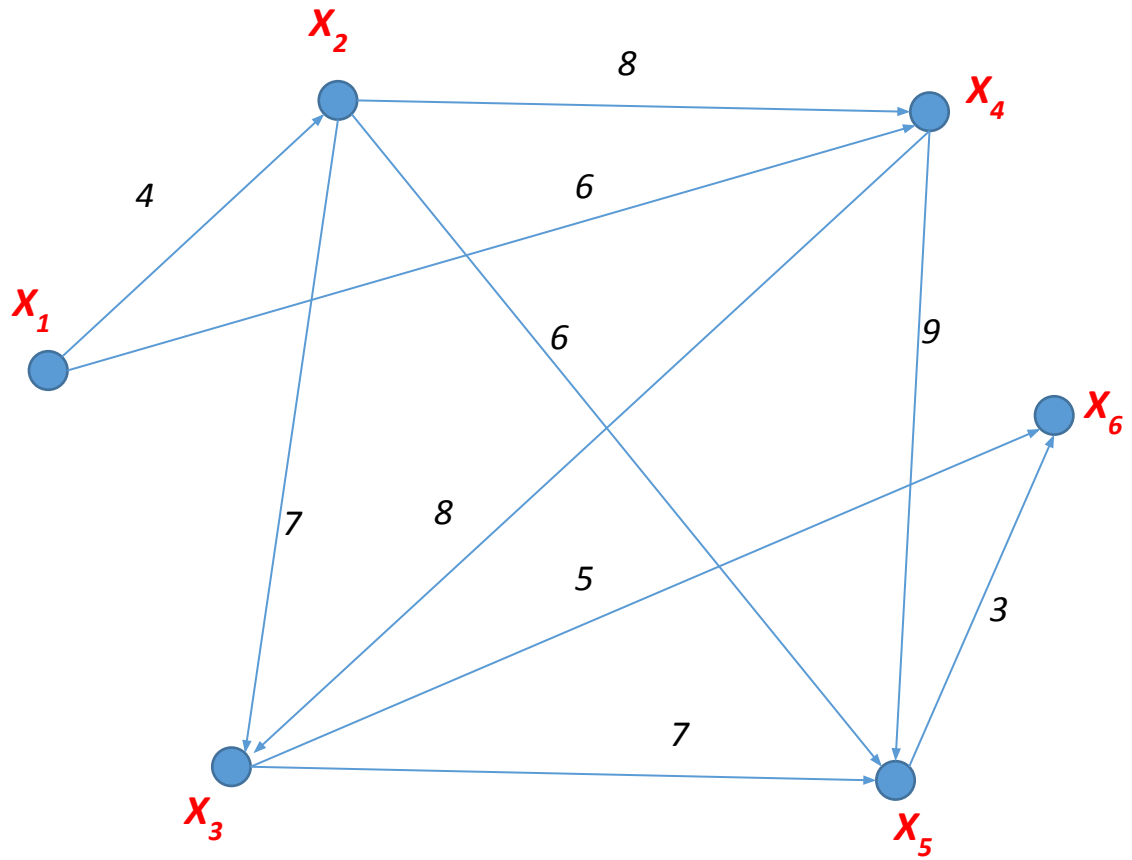
- Максимальные пути можно построить методом последовательного возвращения (второй этап алгоритма Дейкстры).

# Пример

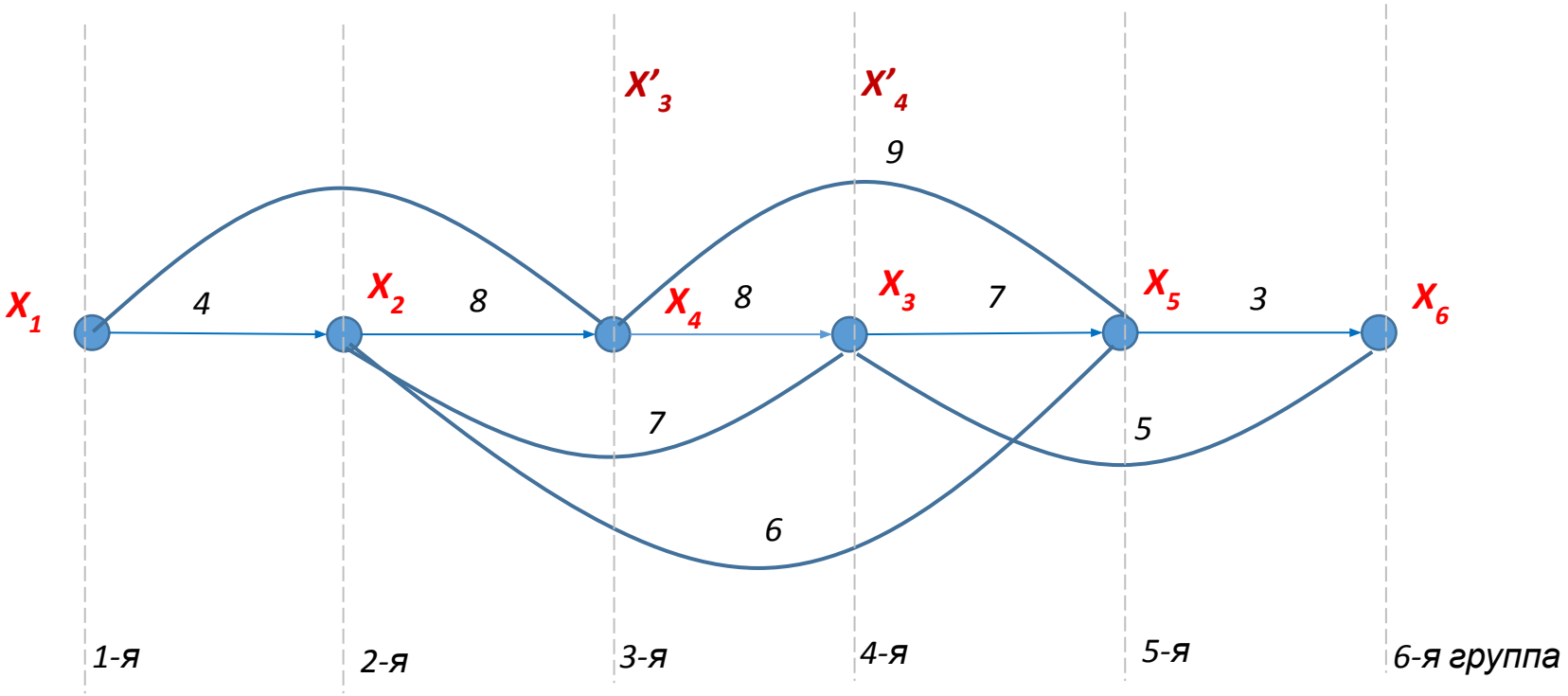
- Граф задан матрицей весов Ω. Построить максимальный путь из вершины $x_1$ в $x_6$. Найти длину максимального пути из вершины $x_1$ в $x_6$ .

$$\Omega = \begin{array}{c} \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{array} \begin{array}{c} \begin{array}{cccccc} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \end{array} \\ \begin{pmatrix} - & 4 & \infty & 6 & \infty & \infty \\ \infty & - & 7 & 8 & 6 & \infty \\ \infty & \infty & - & \infty & 7 & 5 \\ \infty & \infty & 8 & - & 9 & \infty \\ \infty & \infty & \infty & \infty & - & 3 \\ \infty & \infty & \infty & \infty & \infty & - \end{pmatrix} \end{array}.$$

$$\Omega = \begin{array}{c} \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{array} \begin{array}{c} \begin{array}{cccccc} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \end{array} \\ \begin{pmatrix} - & 4 & \infty & 6 & \infty & \infty \\ \infty & - & 7 & 8 & 6 & \infty \\ \infty & \infty & - & \infty & 7 & 5 \\ \infty & \infty & 8 & - & 9 & \infty \\ \infty & \infty & \infty & \infty & - & 3 \\ \infty & \infty & \infty & \infty & \infty & - \end{pmatrix} \end{array}.$$



# Граф ациклический, упорядочим вершины по алгоритму Фалкерсона



# Этап 1

$d_1 = 0$,

$d_2 = \max (d_1 + 4) = 4$,

$d'_3 = \max (d_1 + 6, d_2 + 8) = \max (0 + 6, 4 + 8) = 12$,

$d'_4 = \max (d'_3 + 8, d_2 + 7) = \max (12 + 8, 4 + 7) = 20$,

$d_5 = \max (d'_4 + 7, d'_3 + 9, d_2 + 6) = \max (20 + 7, 12 + 9, 4 + 6) = 27$,

$d_6 = \max (d_5 + 3, d'_4 + 5) = \max (27 + 3, 20 + 5) = 30$.

- **Вывод:** $d_{max} = 30$ - длина максимального пути из $x_1$ в $x_6$.

# Этап 2

- $x_6$ : $d_6 = 30 = d_5 + 3 = 27 + 3 \Rightarrow$ включим дугу $(x_5, x_6)$ в максимальный путь
- $x_6$ : $d_6 = 30 \neq d'_4 + 5 = 20 + 5$
- $x_5$ : $d_5 = 27 = d'_4 + 7 = 20 + 7 \Rightarrow$ включим дугу $(x'_4, x_5)$ в максимальный путь,
- $x_5$ : $d_5 = 27 \neq d'_3 + 9 = 12 + 9$
- $x_5$ : $d_5 = 27 \neq d_2 + 6 = 4 + 6$

- $x'_4$: $d = 20 = d + 8 = 12 + 8 \Rightarrow$ включим дугу $(x'_3, x'_4)$ в максимальный путь
- $x'_4$: $d'_4 = 20 \neq d_2 + 7 = 4 + 7$
- $x'_3$: $d = 12 = d_2 + 8 = 4 + 8 \Rightarrow$ включим дугу $(x_2, x'_3)$ в максимальный путь
- $x'_3$: $d = 12 \neq d_1 + 6 = 0 + 6$
- $x_2$ : $d_2 = 4 = d_1 + 4 = 0 + 4 \Rightarrow$ включим дугу $(x_1, x_2)$ в максимальный путь.
- **Вывод:** искомый путь $\mu_{max} = (x_1, x_2)\text{-}(x_2, x'_3)\text{-}(x'_3, x'_4)\text{-}(x'_4, x_5)\text{-}(x_5, x_6)$ или в первоначальных обозначениях $\mu_{max} = (x_1, x_2)\text{-}(x_2, x_4)\text{-}(x_4, x_3)\text{-}(x_3, x_5)\text{-}(x_5, x_6)$
- **Ответ:** $d_{max} = 30$, $\mu_{max} = (x_1, x_2)\text{-}(x_2, x_4)\text{-}(x_4, x_3)\text{-}(x_3, x_5)\text{-}(x_5, x_6)$

# 5. Теорема Форда-Фалкерсона (задача о максимальном потоке и минимальном разрезе )

- Большинство физически реализованных сетей - носители систем потоков.
- Объекты текут, движутся, транспортируются по системе каналов (дуг сети) **ограниченной** пропускной способности.

**Примеры потоков:**

- автомобильный транспорт по сети автодорог,
- грузы по железнодорожной сети,
- вода в сети водоснабжения,
- электрический ток в электросети,
- сообщения по каналам связи.

# Замечания

- **G(S, U, Ω)** – связный граф без петель
- существует ровно **одна** вершина **s** (источник, source), не имеющая предшествующих
- существует ровно **одна** вершина **t** (сток), не имеющая последующих
- каждой дуге $(x_i, x_j) \in U$ соответствует пропускная способность дуги - число $c(x_i,x_j) \in \Omega$, $c(x_i,x_j) \geq 0$.

Функция $\phi(x_i, x_j)$, определенная на множестве дуг сети $G=(S, U, \Omega)$,
называется **потоком**, если $0 \le \phi(x_i, x_j) \le c(x_i, x_j)$, $\forall (x_i, x_j) \in U$ и

$$\sum_{x_i \in S_{pred}(x_j)} \varphi(x_i, x_j) = \sum_{x_j \in S_{sled}(x_i)} \varphi(x_i, x_j) \quad (***)$$

$\forall x_i \in S$ и $x_i \notin \{s, t\}$.
Формула (***) - условие сохранения потока – **в промежуточных вершинах потоки не создаются и не исчезают**.

**Остаточная пропускная способность дуги** $(x_i, x_j)$:

$\Delta(x_i, x_j) = c(x_i, x_j) - \phi(x_i, x_j)$

Если $c(x_i, x_j) = \phi(x_i, x_j)$, то дуга **насыщенная**.

**Разрез** – это множество дуг, исключение которых из сети отделило бы некоторое множество узлов остальной сети.

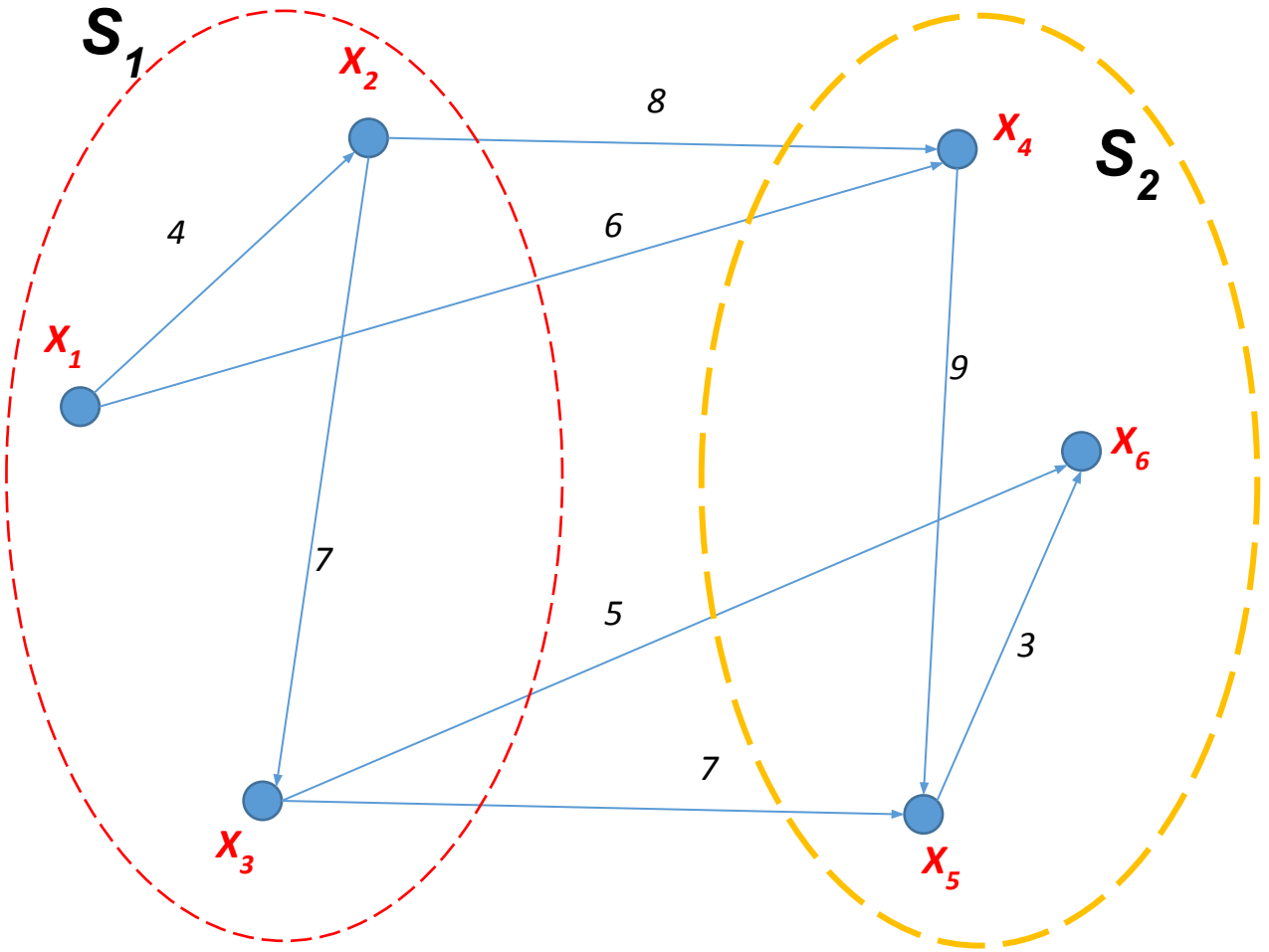
$S = S1 \cup S2$

$S1 \cap S2 = \varnothing$

Множество дуг, начала которых лежат в S1, а концы в S2, называется **ориентированным разрезом** и обозначается (S1 →S2):

$(S1 \rightarrow S2)=\{(x_i, x_j) \mid x_i \in S1, x_j \in S2\}$

Пропускная способность (величина) ориентированного разреза равна сумме пропускных способностей образующих его дуг




$(S1 \rightarrow S2)=\{(x_2, x_4), (x_1, x_4), (x_3, x_6), (x_3, x_5)\}$

$c(S1 \rightarrow S2)=8+6+5+7=26$

# Теорема Форда-Фалкерсона

Для любой сети с одним источником и одним стоком величина максимального потока в сети от источника к стоку равна пропускной способности минимального разреза.

# Алгоритм

а) ищем любую цепь из истока графа в сток;

б) каждой дуге приписываем возможный больший поток из истока в сток (записываем его через дробь с весом дуги; при этом поток не может превысить вес дуги, но может быть ему равен);

в) если поток становится равен весу дуги, то эта дуга является насыщенной, то есть через нее нельзя пройти при рассмотрении цепей в графе;

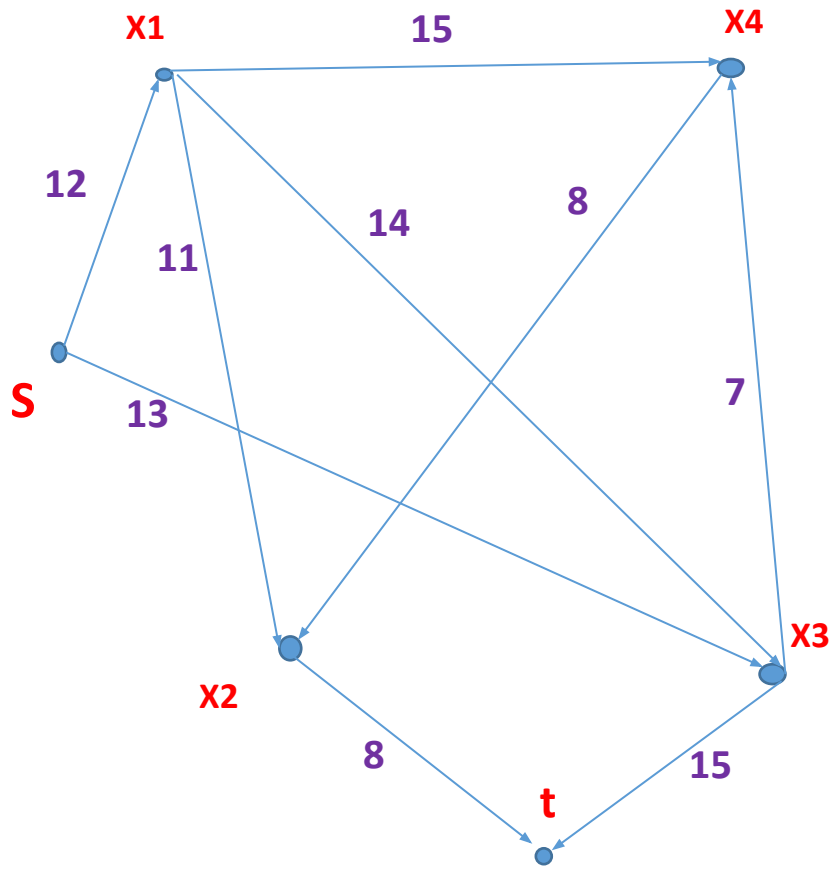
г) так перебираем все возможные цепи, пока станет невозможно попасть из истока в сток;

д) поток  в сети будет равен сумме потоков всех дуг, инцидентных стоку графа (следует заметить, что сумма потоков всех дуг, инцидентных стоку графа равна сумме потоков всех дуг, инцидентных истоку графа).

# Пример

Пропускные способности дуг заданы следующей матрицей. Построить минимальный поток из s в t и указать минимальный разрез, отделяющий s от t.

$$\Omega = \begin{array}{c} \\ s \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ t \end{array} \begin{array}{c} \begin{array}{cccccc} s & x_1 & x_2 & x_3 & x_4 & t \end{array} \\ \begin{pmatrix} - & 12 & - & 13 & - & - \\ - & - & 11 & 14 & 15 & - \\ - & - & - & - & - & 8 \\ - & - & - & - & 7 & 15 \\ - & - & 8 & - & - & - \\ - & - & - & - & - & - \end{pmatrix} \end{array}.$$



# Этап 1

- Рассмотрим какой-либо путь от s до t, например,

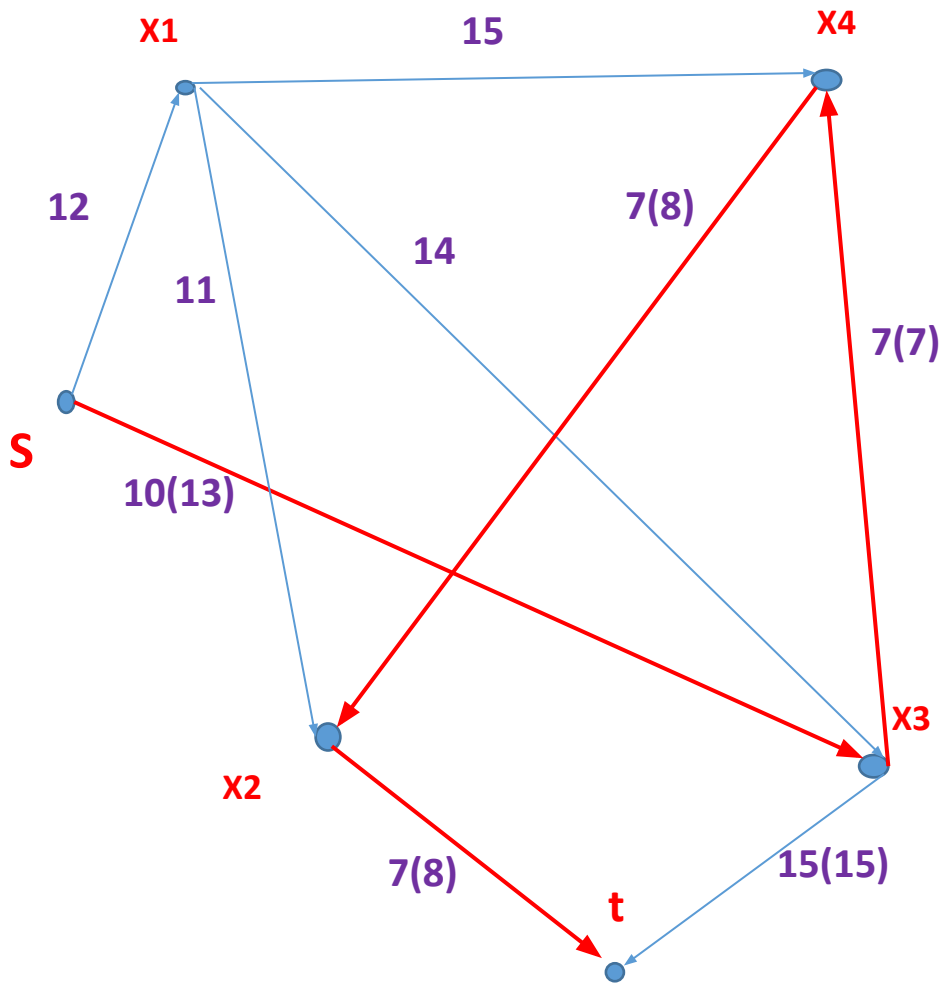
$$s \xrightarrow{12} x_1 \xrightarrow{14} x_3 \xrightarrow{15} t$$

- δ = min(12; 14; 15)=12
- Увеличим по этому пути поток до 12 единиц ⇒ ребро (s, x1) станет насыщенным.
- Поставим величину потока на дугах (x1,x3) и (x3,t)

- Рассмотрим путь $s \xrightarrow{13} x_3 \xrightarrow{12(15)} t$
- δ = min(13; 15 – 12)=3 ⇒ Поток можно увеличить на 3 единицы ⇒ дуга ($x_3$, t) станет насыщенной.
- Рассмотрим путь $s \xrightarrow{3(13)} x_3 \xrightarrow{7} x_4 \xrightarrow{8} x_2 \xrightarrow{8} t$
- δ = min(13-3; 7; 8; 8)=7 ⇒ Поток можно увеличить поток на 7 единиц ⇒ дуга ($x_3$, $x_4$) станет насыщенной

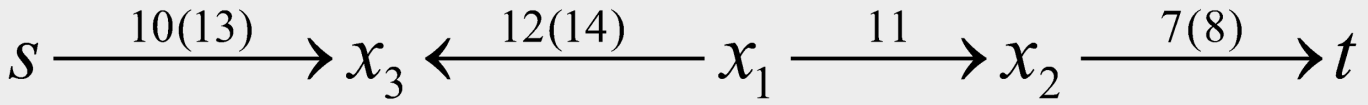
$$s \xrightarrow{10(13)} x_3 \xrightarrow{7(7)} x_4 \xrightarrow{7(8)} x_2 \xrightarrow{7(8)} t$$

Больше путей нет,
конец первого этапа.

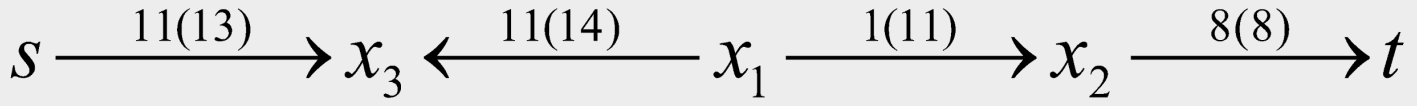


# Этап 2

- Рассмотрим маршруты, содержащие противоположные дуги.

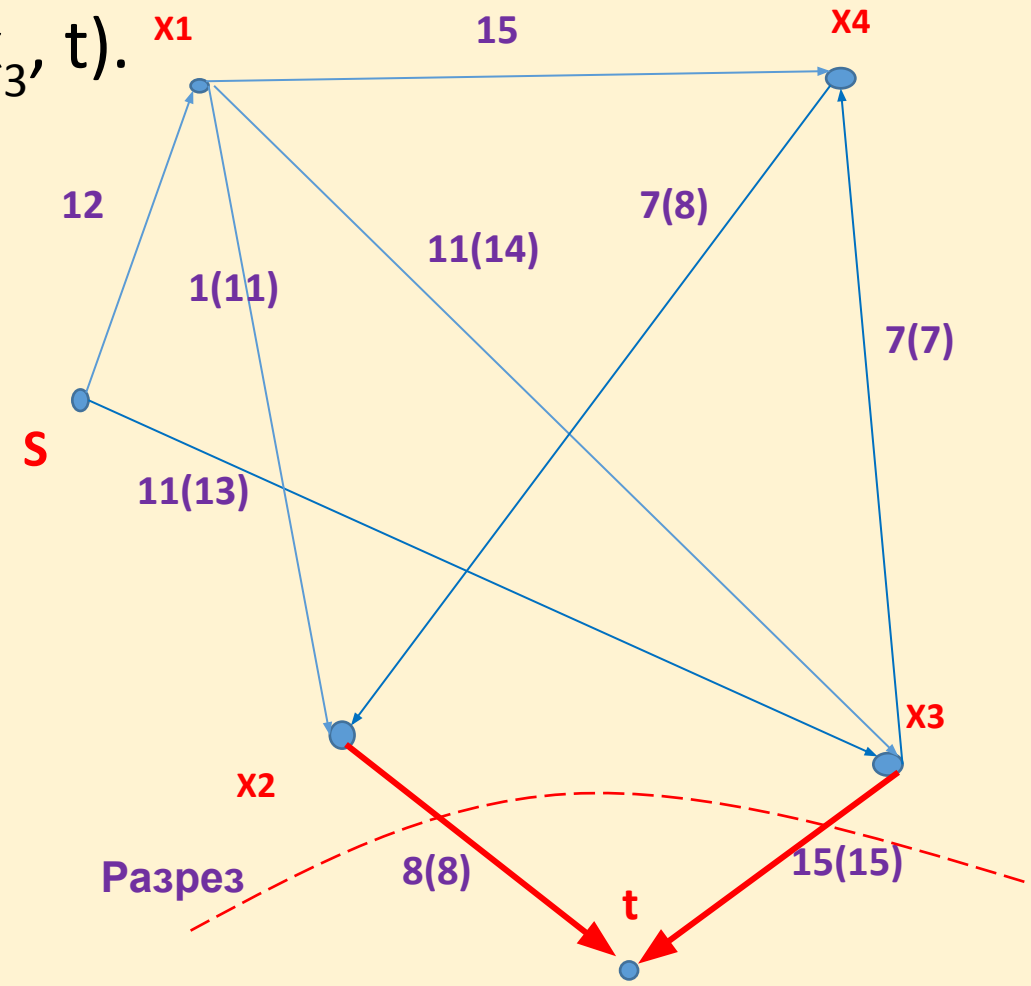



- δ = min(13-10; 14-12; 11; 8-7)=1
- Поток по дуге ($x_2$; t) можно увеличить на 1




- дуга (x2; t) насыщена

- Больше маршрутов нет. Поток максимален.
- Проведем разрез вокруг t по насыщенным дугам
- Величина разреза 15+8=23 единицы.
- **Ответ:** $\phi_{max}$=23 ед., $\mu_{min}$= ($x_2$, t) ($x_3$, t).



# 6. Система ПЕРТ

Program (Project) Evaluation and Review Technique (**PERT**) — метод оценки и анализа проектов, который используется в управлении проектами. Предназначен для масштабных, единовременных, сложных, нерутинных проектов.

Метод был разработан в 1958 году консалтинговой фирмой «Буз, Ален и Гамильтон» совместно с корпорацией «Локхид» по заказу Подразделения специальных проектов ВМС США в составе Министерства Обороны США для проекта создания ракетной системы «Поларис» (Polaris).

Проект «Поларис» был ответом на кризис, наступивший после запуска Советским Союзом первого космического спутника.

# Некоторые термины

**Событие PERT** (PERT event) - момент, отмечающий начало или окончание одной или нескольких задач. Событие не имеет длительности и не потребляет ресурсы. В случае, если событие отмечает завершение нескольких задач, оно не «наступает» (не происходит) до того, пока все задачи, приводящие к событию, не будут выполнены.

**Предшествующее событие** (predecessor event) - событие, которое предшествует некоторому другому событию непосредственно, без промежуточных событий. Любое событие может иметь несколько предшествующих событий и может быть предшественником для нескольких событий.

**Последующее событие** (successor event) — событие, которое следует за некоторым событием непосредственно, без промежуточных событий. Любое событие может иметь несколько последующих событий и может быть последователем нескольких событий.

**Задача PERT** (PERT activity) — конкретная работа (задача), которое имеет длительность и требует ресурсов для выполнения. Примеры ресурсов: исполнители, сырьё, пространство, оборудование, техника и т.д. Невозможно начать выполнение задачи PERT, пока не наступили все предшествующие ей события.

**Проскальзывание** или **провисание** (float, slack) — мера дополнительного времени и ресурсов, доступных для выполнения работы. Время, на которое выполнение задачи может быть сдвинуто без задержки любых последующих задач (свободное проскальзывание) или всего проекта (общее проскальзывание). Позитивное провисание показывает опережение расписания, негативное провисание показывает отставание, и нулевое провисание показывает соответствие расписанию.

**Критический путь** (critical path) — длиннейший маршрут на пути от начального до финального события. Критический путь определяет минимальное время, требуемое для выполнения всего проекта, и, таким образом, любые задержки на критическом пути соответственно задерживают достижение финального события.

**Критическая задача** (critical activity) — задача, проскальзывание которой равно нулю. Задача с нулевым проскальзыванием не обязательно должна находиться на критическом пути, но все задачи на критических путях имеют нулевое проскальзывание.

# Пример

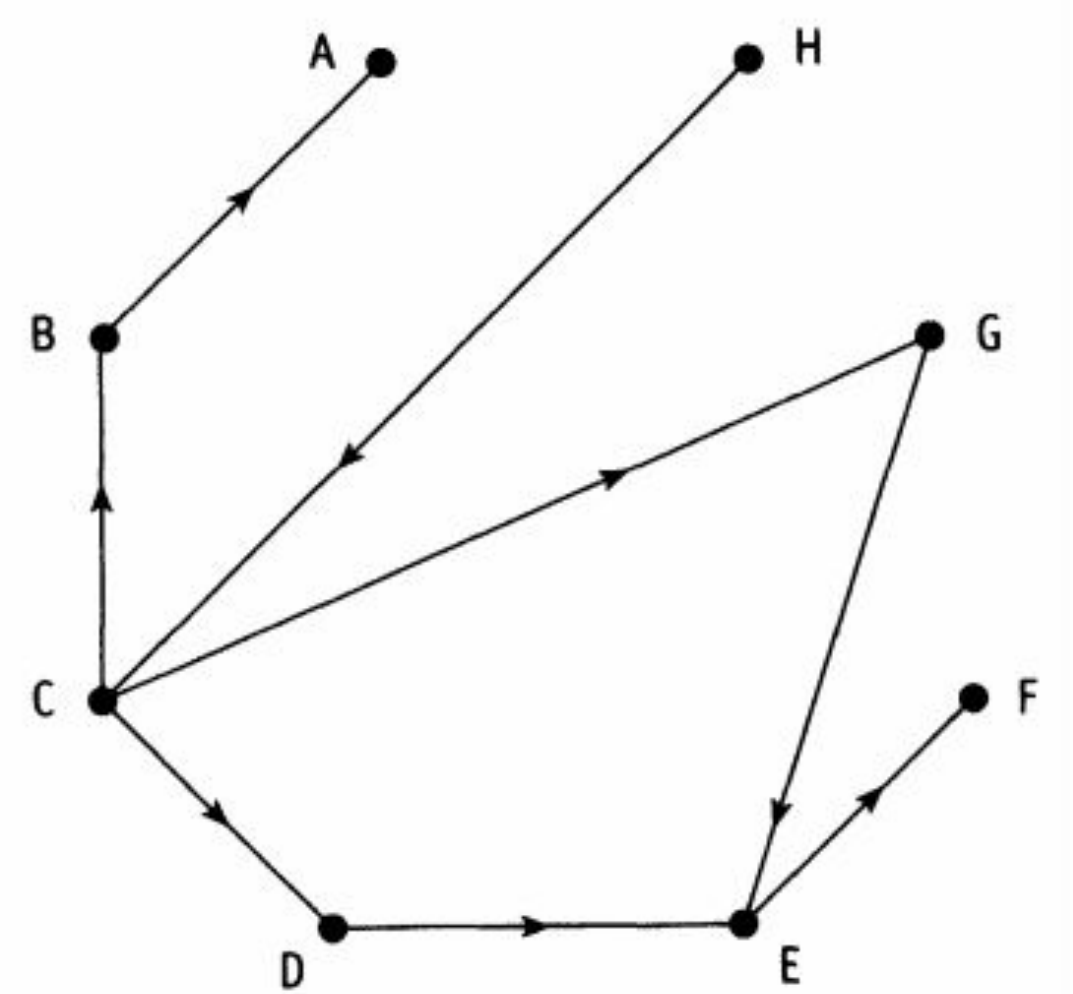
Студенту университета необходимо прослушать восемь курсов,
которые некоторым образом зависят друг от друга. Эта зависимость представлена в таблице. Изобразите систему ПЕРТ, иллюстрирующую приоритетную структуру курсов.

| | | Предварительные курсы |
|---|---|---|
| (A) | Биотехнология | B |
| (B) | Начальный курс биотехнологии | C |
| (C) | Цитология | H |
| (D) | Структура ДНК | C |
| (E) | Энзимология | D, G |
| (F) | Диетология | E |
| (G) | Генная инженерия | C |
| (H) | Биология человека | Никаких требований |

Система ПЕРТ представляет собой орграф, описывающий данную приоритетную структуру.

Вершины орграфа — восемь курсов.

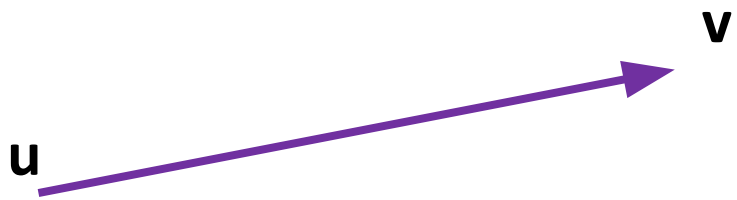
Дуги орграфа отражают представленные в таблице требования, необходимые для усвоения данного курса.



Определить порядок изучения курсов можно, например, по *алгоритму топологической сортировки.*

Алгоритм создает *последовательность согласованных меток для* вершин бесконтурного орграфа (не имеющего циклов) таким образом, что если 1, 2, 3, . . . , **n** — метки вершин и ***uv*** — дуга орграфа, идущая от вершины ***u*** *с* меткой ***i*** к вершине ***v*** *с* меткой **j**, то **i < *j***

Если ***uv*** — дуга орграфа, то ***u*** называют ***антецедентом v*** (лат. antecedens — «предшествующее»)*: u=A(v)*



# Алгоритм топологической сортировки

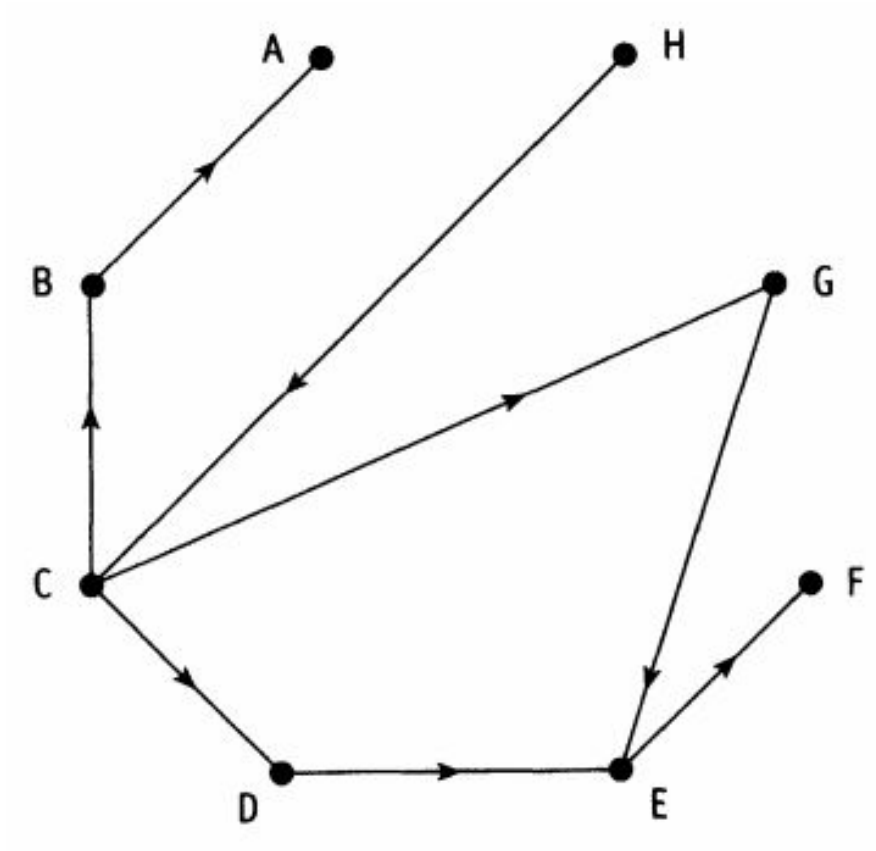
Алгоритм генерирует последовательность согласованных меток для вершин бесконтурного орграфа ***G(V, E)***. В самом начале работы алгоритма антецеденты каждой вершины ***v*** записываются в м

```
begin
    for v ∈ V do
      вычислить A(v);
      label := 0;
      while остаются неотмеченные вершины, для которых
            A(v) = ∅ do
          begin
              label := 1;
              u := вершина с A(u) = ∅;
              присвоить метку вершине u;
              for каждой неотмеченной вершины v ∈ V do;
              A(v) := A(v) \ {u};
          end
end
```

Алгоритм успешно присваивает метки вершинам. Каждая вершина получает очередную метку в том случае, если у нее нет неотмеченных антецедентов.

# Пример

Найдите последовательность меток для орграфа, изображенного на рисунке



# Шаг 0

Множество антецедентов:

**A(A) = {B},**

**A(B) = {C},**

**A(C) = {H},**

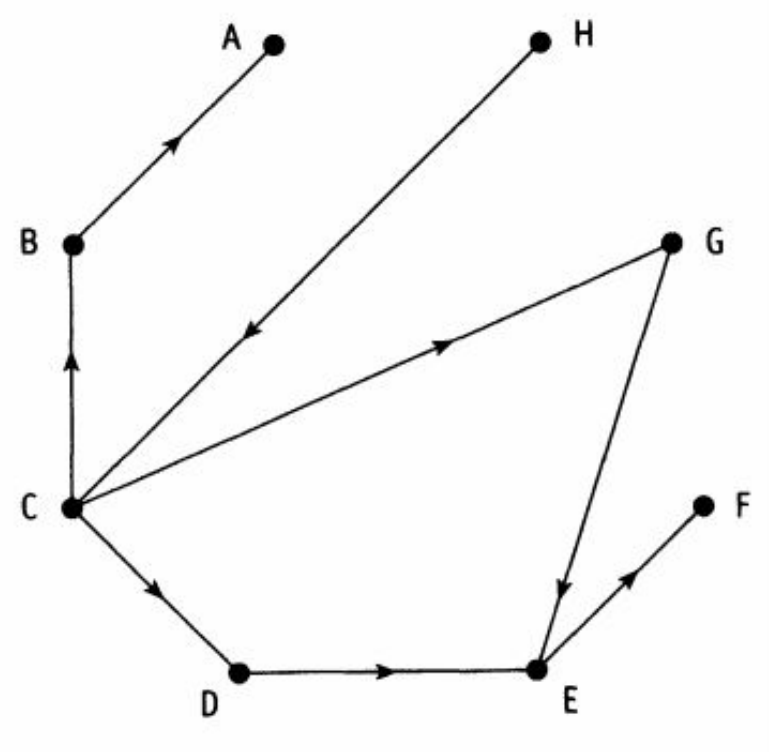
**A(D) = {C},**

**A(E) = {D, G},**

**A(F) = {E},**

**A(G) = {C}**

**A(H) = ∅**



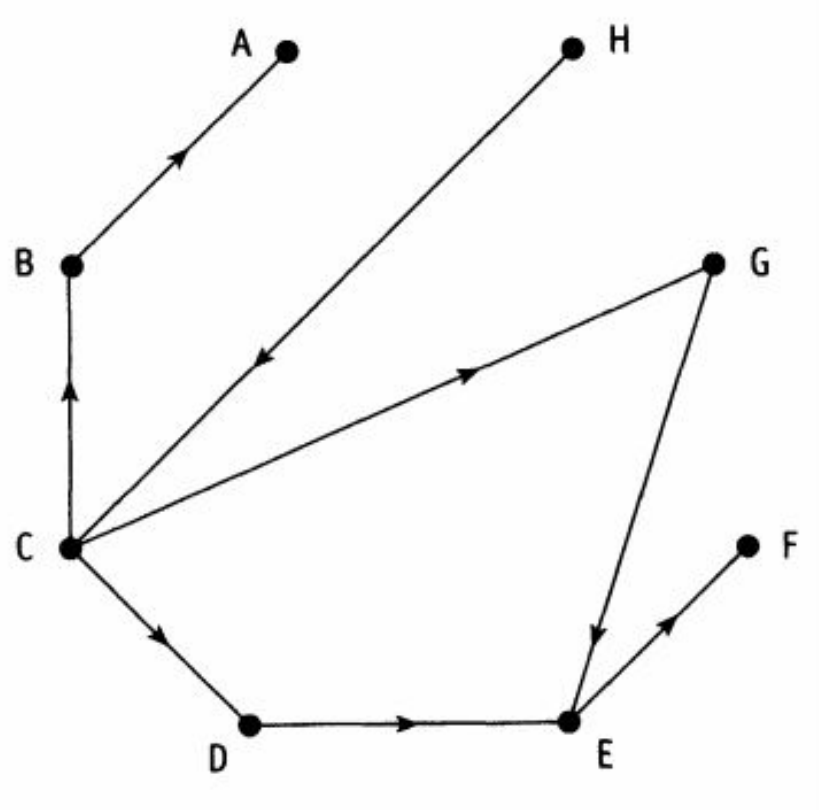
# Шаг 1

Первый проход цикла **while**. Назначить метку 1 вершине **H** и удалить вершину **H** из всех оставшихся множеств $A(v)$.
$A(\mathbf{A}) = \{\mathbf{B}\}$, $A(\mathbf{B}) = \{\mathbf{C}\}$, $A(\mathbf{C}) = \varnothing$, $A(\mathbf{D}) = \{\mathbf{C}\}$,
$A(\mathbf{E}) = \{\mathbf{D}, \mathbf{G}\}$, $A(\mathbf{F}) = \{\mathbf{E}\}$ и $A(\mathbf{G}) = \{\mathbf{C}\}$.

# Шаг 2

Второй проход цикла **while.** Назначить метку 2 вершине C и удалить вершину C из всех оставшихся множеств *A(v).*

A(A) = {B}, A(B) = ∅, A(D) = ∅, A (E) = {D, G},
A(F) = {E} и A(G) = ∅.



# Шаг 3

Третий проход цикла **while**. Теперь у нас появился выбор: какой вершине присвоить очередную метку? В зависимости от нашего выбора, получатся разные последовательности меток. Присвоим, например, метку 3 вершине **B**, и удалим **B** из множеств $A(v)$. $A(\mathbf{A}) = \varnothing$, $A(\mathbf{D}) = \varnothing$, $A(\mathbf{E}) = \{\mathbf{D}, \mathbf{G}\}$, $A(\mathbf{F}) = \{\mathbf{E}\}$ и $A(\mathbf{G}) = \varnothing$.

# **Шаг** 4

Четвертый проход цикла **while.** Мы снова стоим перед выбором. Назначим метку 4 вершине A и удалим вершину A из *A(v),*

A(D) = ∅, A(E) = {D, G}, A(F) = {E} и
A(G) = ∅.

# **Шаг** 5

Пятый проход цикла while.

Назначим метку 5 вершине D и удалим вершину D из *A(v).*

A(E) = {G}, A(F) = {E} и A(G) = ∅

# **Шаг** 6

Шестой проход цикла **while.**

Назначим метку 6 вершине G и удалим вершину G из *A(v).*

A(E) = ∅, A(F) = ∅.

# **Шаг** 7

Седьмой проход цикла **while.**

Назначаем метку 7 вершине E и удаляем E из списка *A(v).*

Останется только A(F) = ∅.

# **Шаг** 8

Последний проход цикла while. Назначаем метку 8 вершине F.

Получили один из возможных приоритетных списков:

H, C, B, A, D, G, E, F, который дает порядок изучения курсов, соблюдая должную последовательность.

# Задача

Приведен список действий по приготовлению блюда из мяса цыпленка с расставленными приоритетами. Упорядочите список
согласно приоритета

| Задания | | Предварительные действия |
|---|---|---|
| А | Добавить лук к цыпленку | И |
| Б | Вымыть салат-латук | Л |
| В | Приготовить салатную заправку | Л |
| Г | Перемешать жаркое | К |
| Д | Перемешать салат | Б, В |
| Е | Разрезать цыпленка | никаких |
| Ж | Растереть имбирь | И |
| З | Подать готовое блюдо | И |
| И | Замариновать цыпленка | Е |
| К | Поставить казанок на огонь | А, Ж, З, Л |
| Л | Приготовить рис | никаких |

# 7. Коммуникационные сети

**Коммуникационная сеть** - это соединение определенным образом участвующих в коммуникационном процессе индивидов (узлов, вершин) с помощью информационных потоков.

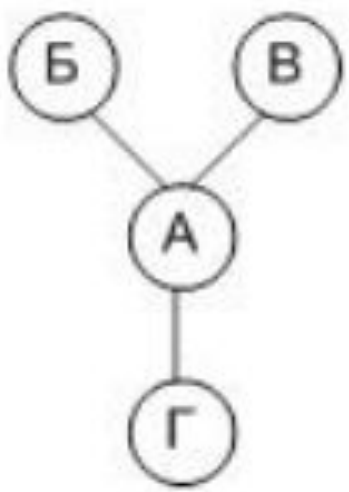
*рассматриваются не индивиды как таковые, а коммуникационные отношения между ними*


виды коммуникационных сетей
**открытые** (движение команды или информации может быть остановлено: 1) тупик в конце канала; 2) препятствие в виде посредника или контролера, которого нельзя обойти)
**замкнутые** - тупики и контролеры либо отсутствуют, либо могут быть обойдены
**комбинированные** - сочетают оба вида

# Простой вид открытой коммуникационной сети – линейная (змея),

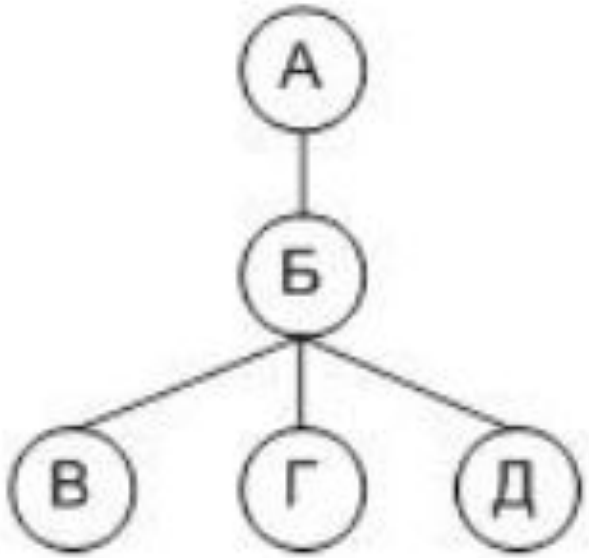

- А и Б – тупики
- В – посредник или контролер
- работники одного уровня управления

## Звезда



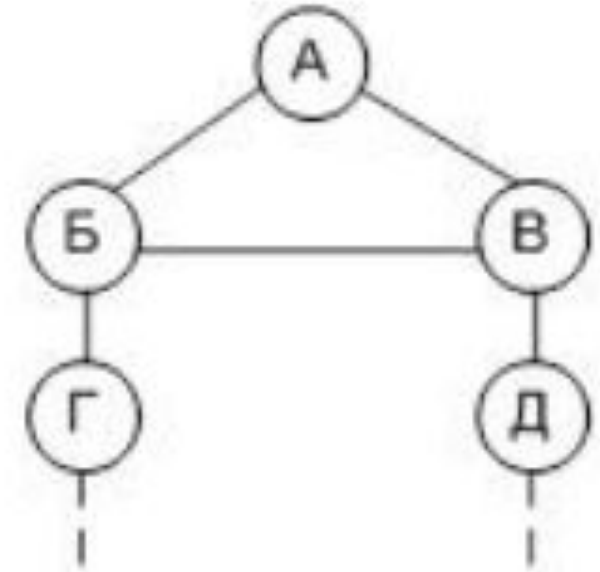

- А – центральный узел (руководитель)
- исполнители Б, В, Г
- Звену А легко поддерживать порядок в управлении (отсутствуют посредники)
- Характерна для небольших управленческих структур

**Шпора**

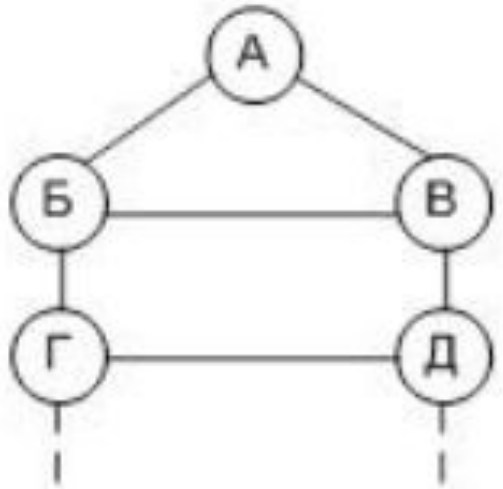



- Характерна для крупных управленческих структур.
- Центральное звено А не в состоянии вырабатывать самостоятельно все решения и доводить их до исполнителей.
- Требуется помощник (посредник) Б, конкретизирующий команды и распределяющий информацию между исполнителями В, Г, Д.
- Помощник Б получает огромную власть, так как контролирует информацию и может навязывать свою волю первому лицу

## Палатка



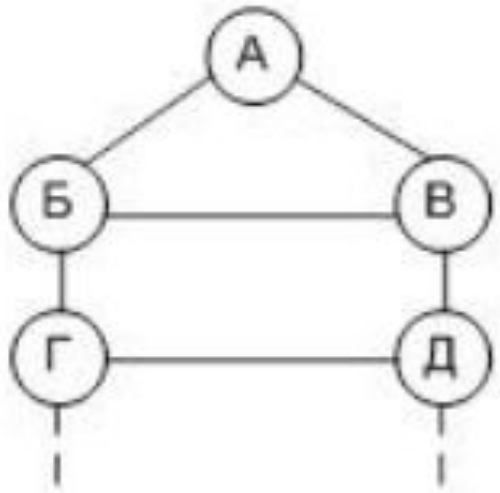

## Дом




- Характерны для крупных многопрофильных функциональных структур
- Наряду с вертикальными коммуникационными каналами официально допускаются горизонтальные каналы
- Посредством горизонтальных каналов подчиненные могут напрямую самостоятельно решать многие второстепенные проблемы, что позволяет руководству не отвлекаться на них

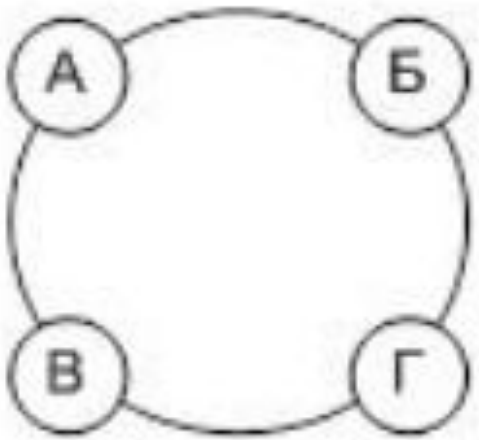
- В «палатке» допускается один уровень горизонтальной коммуникации - между вторыми лицами;
- в «доме» же такие каналы возможны на всех уровнях управленческой структуры, что придает ему характер замкнутой сети.
- Практика показывает, что здесь могут возникать целенаправленные деформации, с помощью которых отдельные субъекты управленческой структуры могут быть сначала выключены из системы коммуникаций, а затем удалены из нее.

- Например, на основе предварительной договоренности субъект Д может направлять информацию для А через Б и Г, минуя В, что должен делать в соответствии с формальными предписаниями. Через некоторое время будет нетрудно доказать принципиальную ненужность В и возможность исключения его из управленческой структуры.



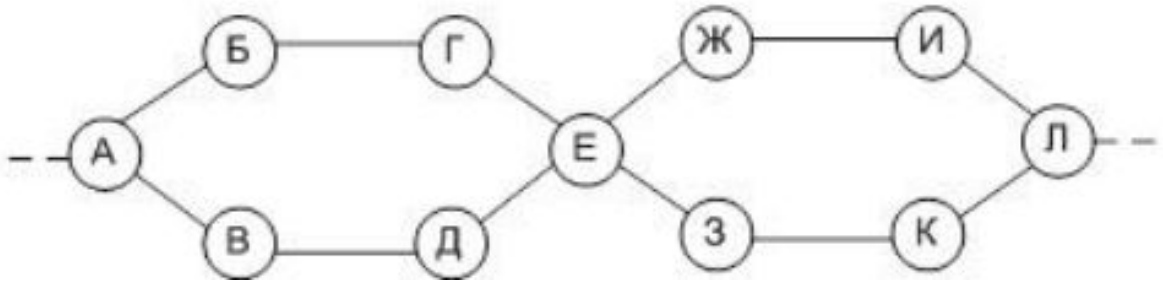
- В целом открытые коммуникационные структуры присущи бюрократическим структурам (жесткое подчинение одних звеньев другим, преобладают формальные связи)
- Однако могут существовать и гибкие структуры – консультационные и совещательные (комитеты, комиссии, специальные творческие группы), которые основаны преимущественно на неформальных или полуформальных внутренних связях и принципах самоуправления. Коммуникации здесь осуществляются посредством замкнутых сетей, в которых посредники.

**Круг**




- Круг характерен для структур с благоприятным морально-психологическим климатом.
- Помогает объединять людей, облегчать обмен информацией и идеями, стимулирует творческие процессы.

## Соты (комбинированный вид)




- На крупных предприятиях творческие группы могут быть связаны друг с другом

**Рассмотрим пример коммуникационной сети** - модель компьютерной сети - ориентированный граф

Вершины — компьютерные компоненты,

дуги — коммуникационные линии связи.

Каждая дуга снабжена весом (пропускная способность)

# Пример простой сети из семи узлов

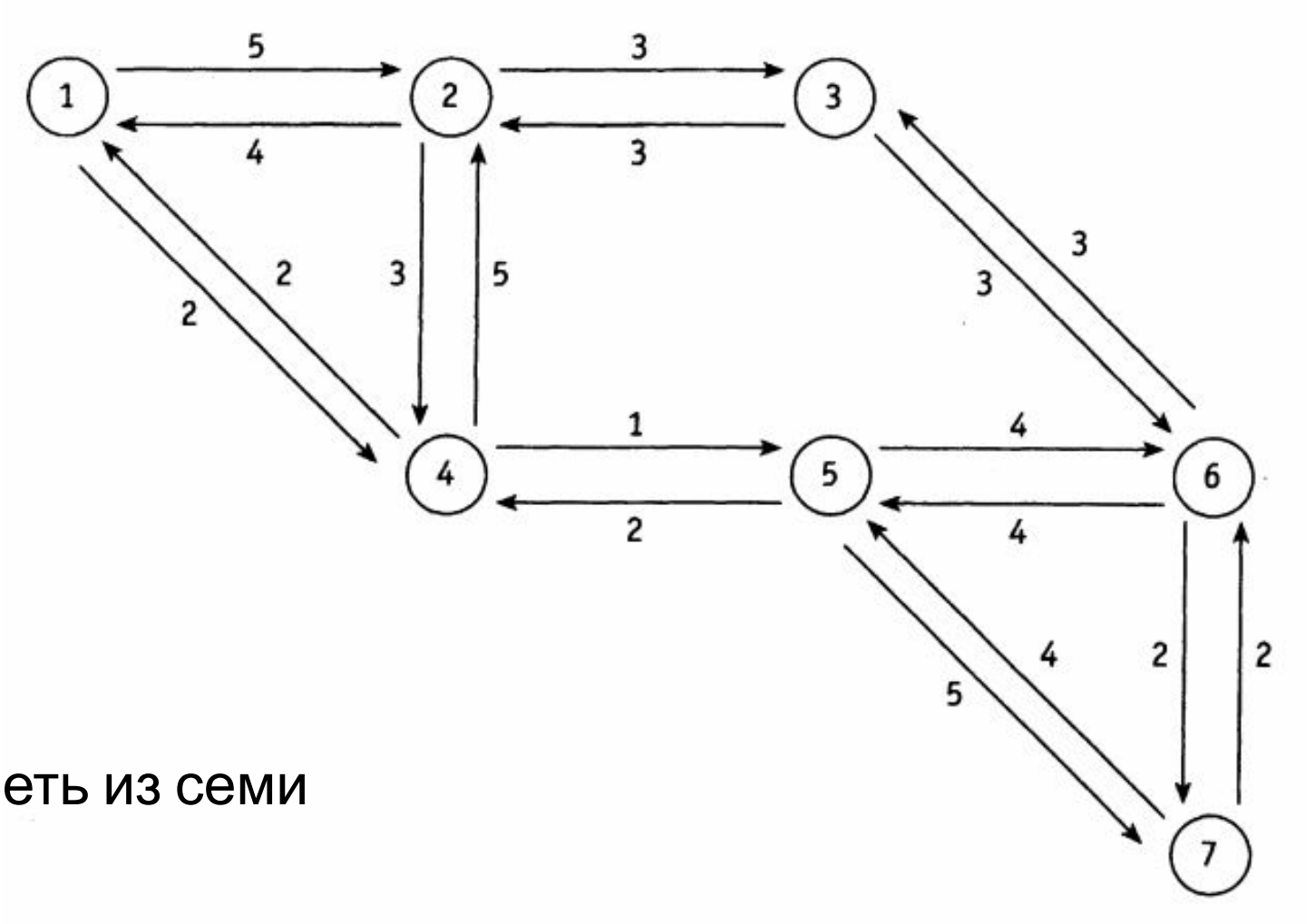



Рис. 1. Сеть из семи узлов

После ввода в действие сети возникает вопрос: как передавать сообщения между несмежными узлами?

Процедура ***статической маршрутизации*** учитывает информацию о пропускной способности линий *для* определения фиксированного пути передач между узлами.

Для оптимизации таких путей в сети применяют алгоритм, подобный алгоритму Дейкстры.

Однако при этом подходе могут возникать сбои в линиях или узлах сети.

Задержки передач могут происходить, когда превышается пропускная
способность линии

Процедура ***динамической маршрутизации*** постоянно корректирует пропускную способность линий с учетом потребности.

Чтобы узлам «решать», когда и куда передавать новую информацию, разработан ***протокол*** (множество правил).

Каждый узел поддерживает свою таблицу путей, поэтому задача оптимизации передачи сообщений *рассредоточена* по всей
сети.

- Каждый узел сети на рис. 1 «прогоняет» алгоритм Дейкстры для определения наилучших путей к другим узлам и распространяет эту информацию по дереву, чей корень соответствует «домашнему узлу».
- Например, для узла 1 соответствующее дерево показано на рис. 2

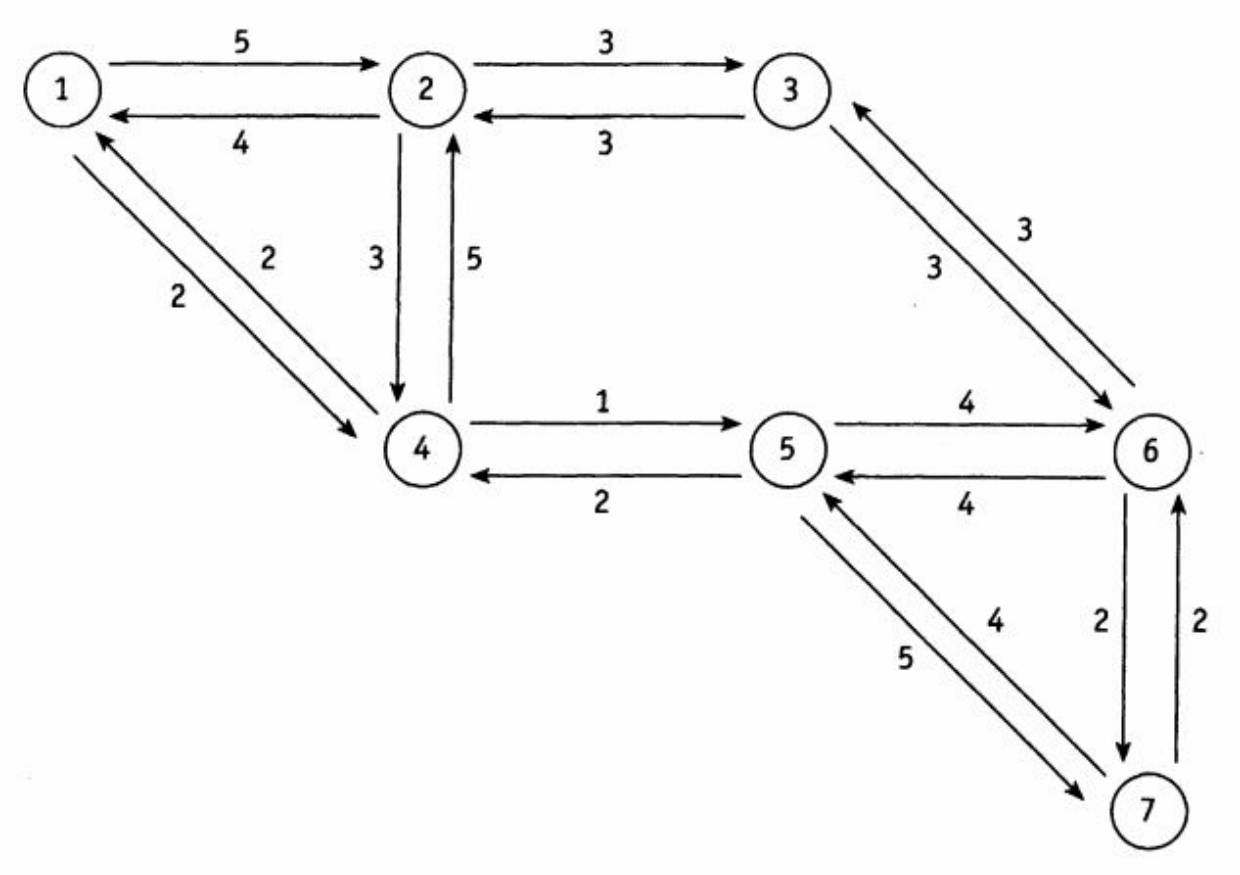



**Рис. 1. Сеть из семи узлов**

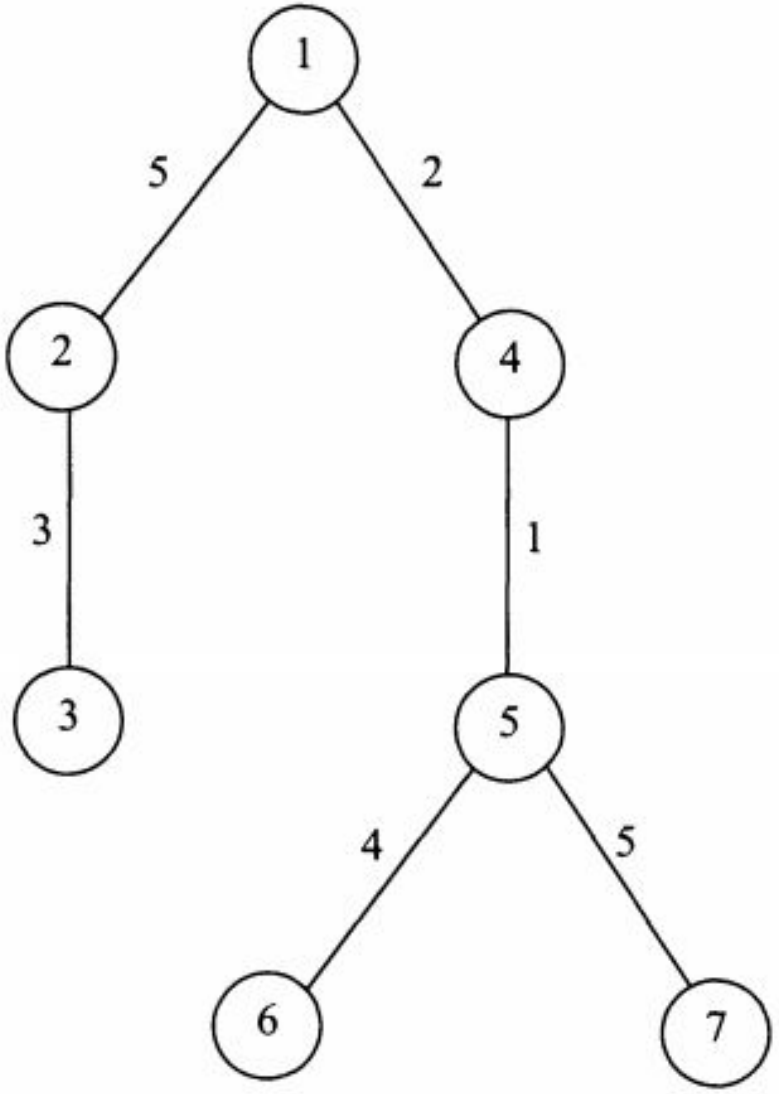
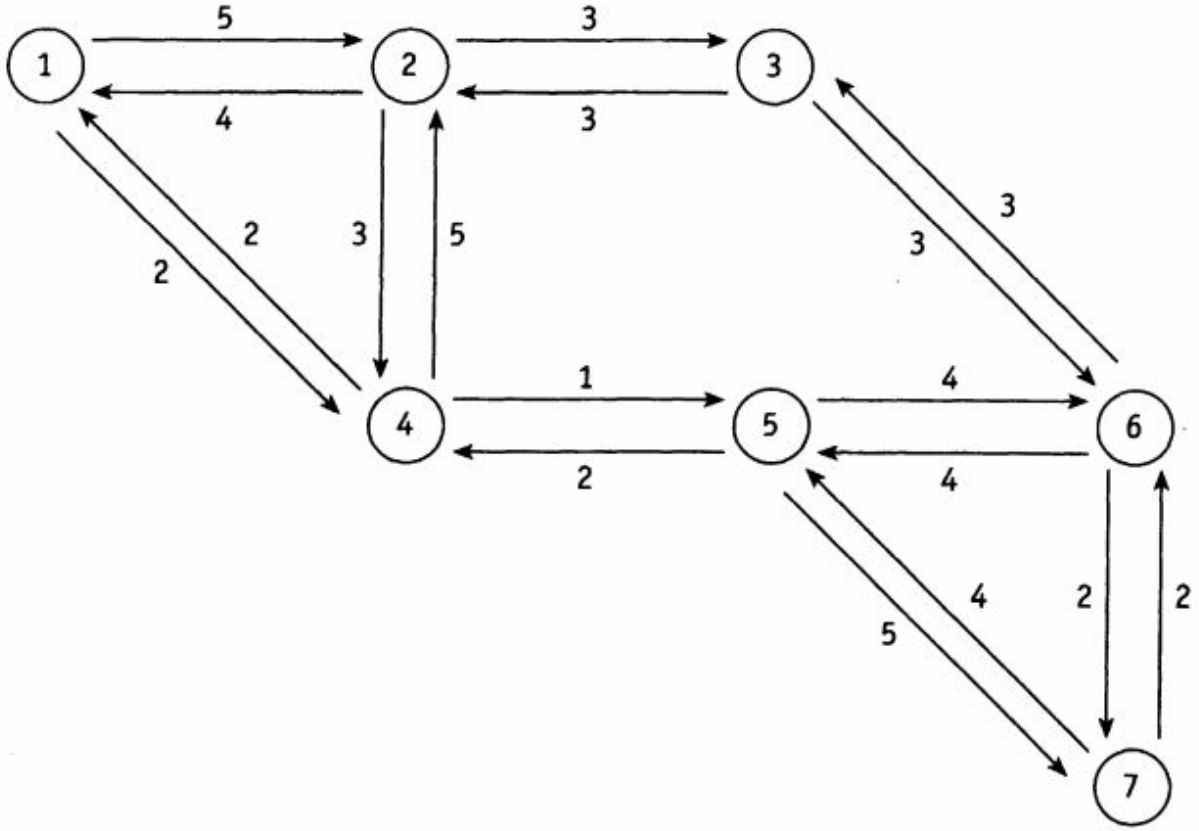



**Рис. 2. Дерево передачи информации узла 1**

- Для передачи сообщений любому узлу требуется таблица, в которой указаны ближайшие соседи для передачи сообщения тому или иному адресату.
- Таблица ближайших соседей (маршрутов) для узла 1:



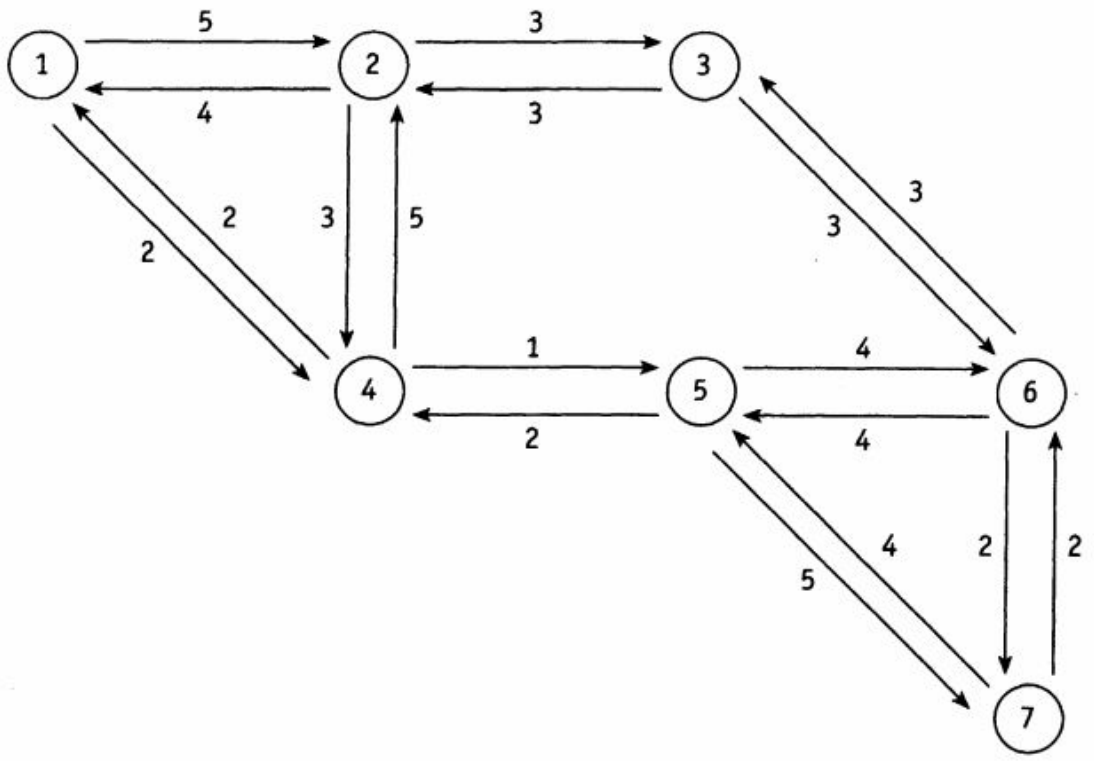

| Адресат | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|
| Следующий узел | 2 | 2 | 4 | 4 | 4 | 4 |

**З а д а ч а 1.** Используя алгоритм Дейкстры, найти кратчайшие пути от узла 2 к любому другому по сети на рис. 1. Показать дерево кратчайших путей и заполнить таблицу маршрутов для узла 2.

# Шаг 0

| Отмеченные вершины | Расстояние до вершины | | | | | | | Неотмеченные вершины |
|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | |
| 2 | 4 | 0 | **3** | 3 | ∞ | ∞ | ∞ | 1, 3, 4, 5, 6, 7 |

Начинаем от вершины 2, отмечаем ее и используем первую строку весовой матрицы *W* для определения начальных значений *d[v].*




Наименьшие числа из всех *d[v]* для неотмеченных вершин —
это *d[3]* = 3 и *d[4]* = 3 .

# Шаг 1

| Отмеченные вершины | Расстояние до вершины | | | | | | | Неотмеченные вершины |
|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | |
| 2 | 4 | 0 | **3** | 3 | ∞ | ∞ | ∞ | 1, 3, 4, 5, 6, 7 |
| 3 | 4 | 0 | 3 | **3** | ∞ | 6 | ∞ | 1, 4, 5, 6, 7 |

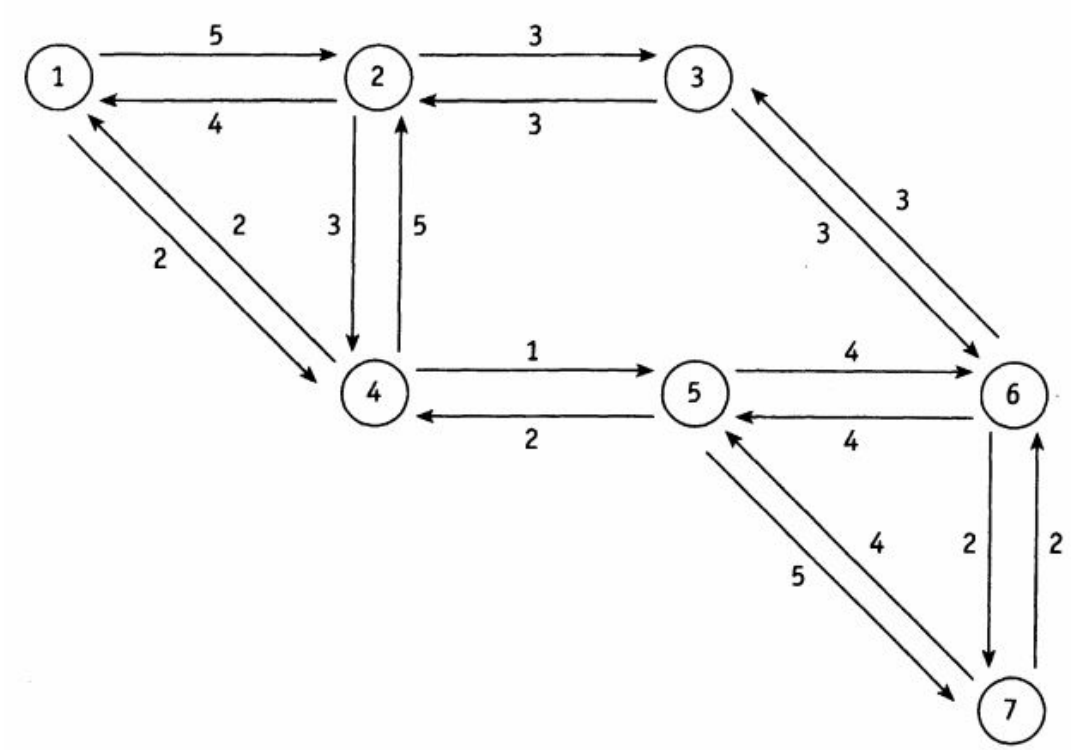
Ближайшие к вершине 2 – это вершины 3 и 4.

Отметим вершину ***3***.
Вычислим длины путей, ведущих от ***2*** к неотмеченным вершинам через вершину ***3***.

Если новые значения ***d[v]*** оказываются меньше старых, то меняем их на новые.

Получим: путь *2→3 → 6* имеет вес 6,
(старое расстояние было равно ∝).

Заполняя вторую строку таблицы, заменим *d[6]* на 6.



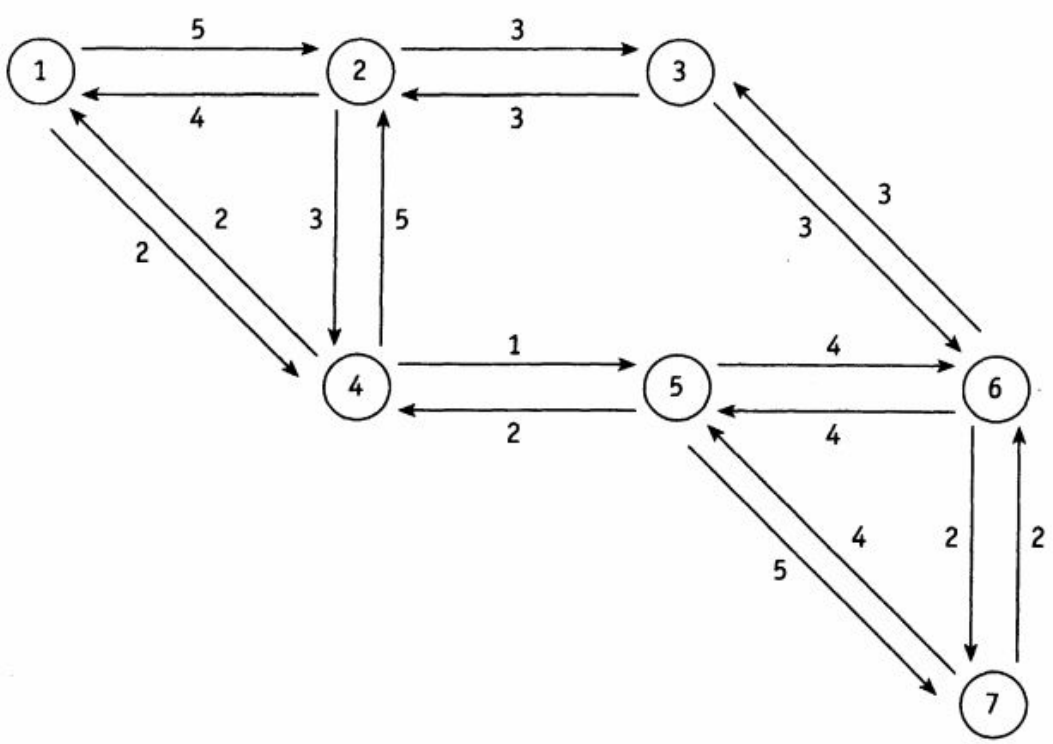
# Шаг 2

| Отмеченные вершины | Расстояние до вершины 1 | 2 | 3 | 4 | 5 | 6 | 7 | Неотмеченные вершины |
|---|---|---|---|---|---|---|---|---|
| 2 | 4 | 0 | **3** | 3 | ∞ | ∞ | ∞ | 1, 3, 4, 5, 6, 7 |
| 3 | 4 | 0 | 3 | **3** | ∞ | 6 | ∞ | 1, 4, 5, 6, 7 |
| 4 | 4 | 0 | 3 | 3 | **4** | 6 | ∞ | 1, 5, 6, 7 |

Из оставшихся неотмеченными, вершина ***5*** находится ближе всех к ***1.***

Так как длина пути ***2***→ ***4***→ *5* равна 4, то текущее значение ***d[5]*** следует уменьшить до 4.

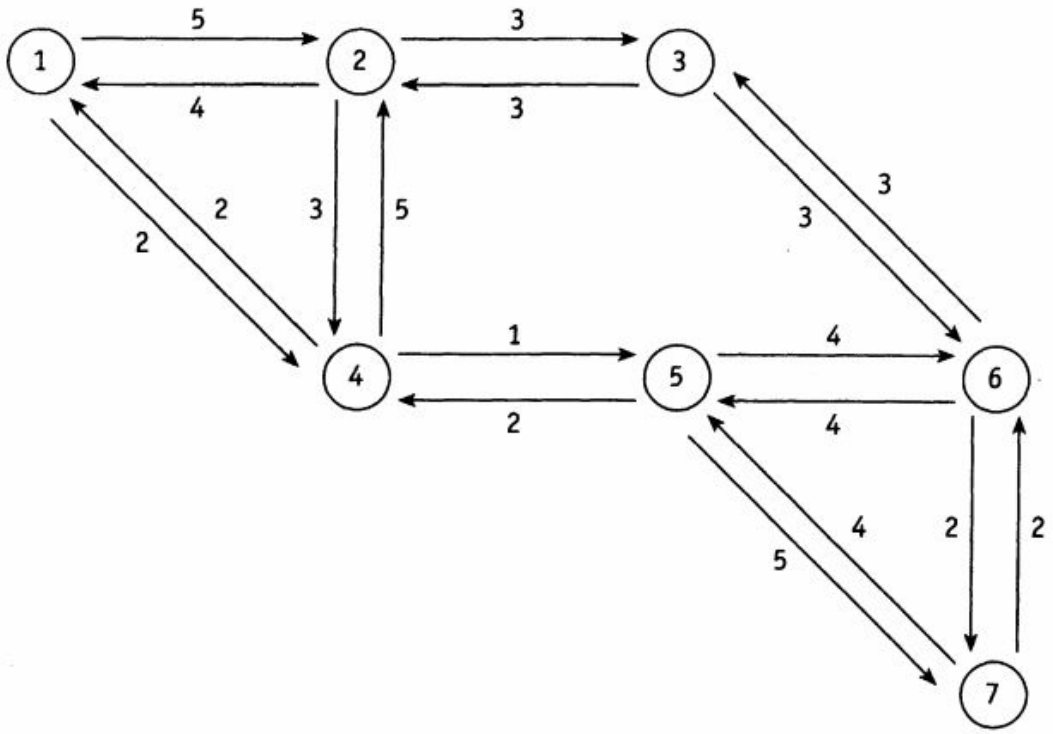
Заполним третью строчку таблицы. Наименьшее значение ***d[v]*** среди неотмеченных вершин оказывается у вершины ***5***.



# Шаг 3

| Отмеченные вершины | Расстояние до вершины | | | | | | | Неотмеченные вершины |
|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | |
| 2 | 4 | 0 | **3** | 3 | ∞ | ∞ | ∞ | 1, 3, 4, 5, 6, 7 |
| 3 | 4 | 0 | 3 | **3** | ∞ | 6 | ∞ | 1, 4, 5, 6, 7 |
| 4 | 4 | 0 | 3 | 3 | **4** | 6 | ∞ | 1, 5, 6, 7 |
| 5 | **4** | 0 | 3 | 3 | 4 | 6 | 9 | 1, 6, 7 |

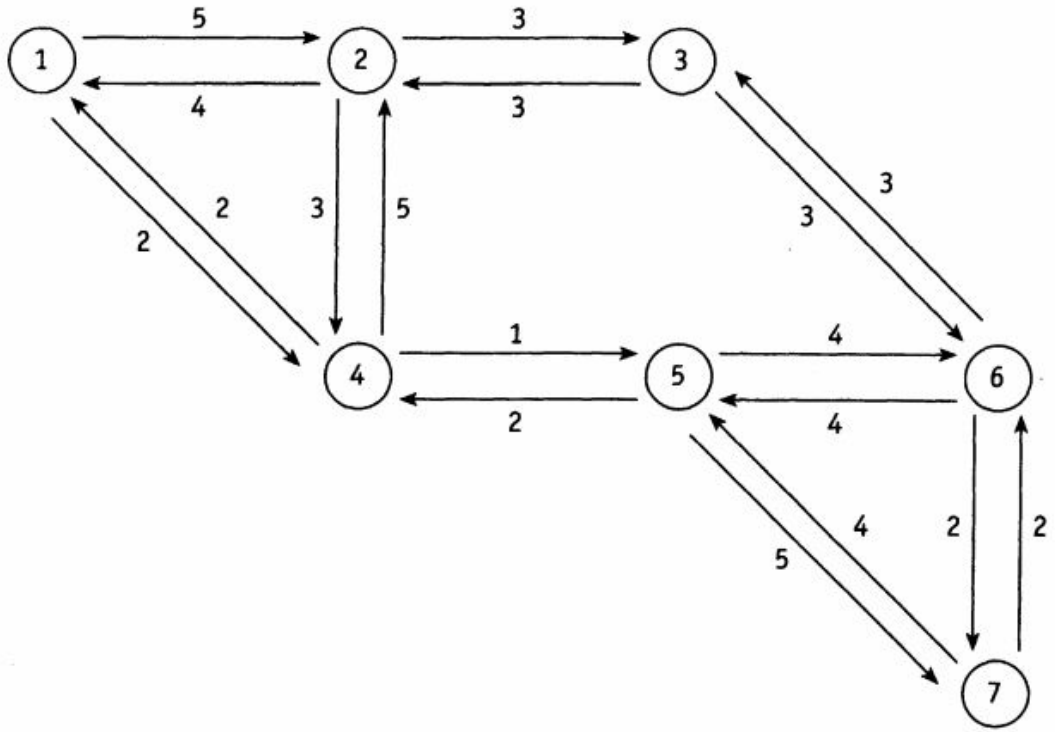
- Отметим вершину ***5*** и подправим значения ***d[v]***.
- Можно дойти и до вершины ***7***, следуя путем ***2*** → ***4*** → ***5*** → ***7***.
- Его длина, т.е. ***d[7]***= 9.
- Минимальное значение равно 4 (среди неотмеченных вершин), соответствующее вершине 1



# Шаг 4

| Отмеченные вершины | Расстояние до вершины 1 | 2 | 3 | 4 | 5 | 6 | 7 | Неотмеченные вершины |
|---|---|---|---|---|---|---|---|---|
| 2 | 4 | 0 | **3** | 3 | ∞ | ∞ | ∞ | 1, 3, 4, 5, 6, 7 |
| 3 | 4 | 0 | 3 | **3** | ∞ | 6 | ∞ | 1, 4, 5, 6, 7 |
| 4 | 4 | 0 | 3 | 3 | **4** | 6 | ∞ | 1, 5, 6, 7 |
| 5 | **4** | 0 | 3 | 3 | 4 | 6 | 9 | 1, 6, 7 |
| 1 | 4 | 0 | 3 | 3 | 4 | **6** | 9 | 6, 7 |

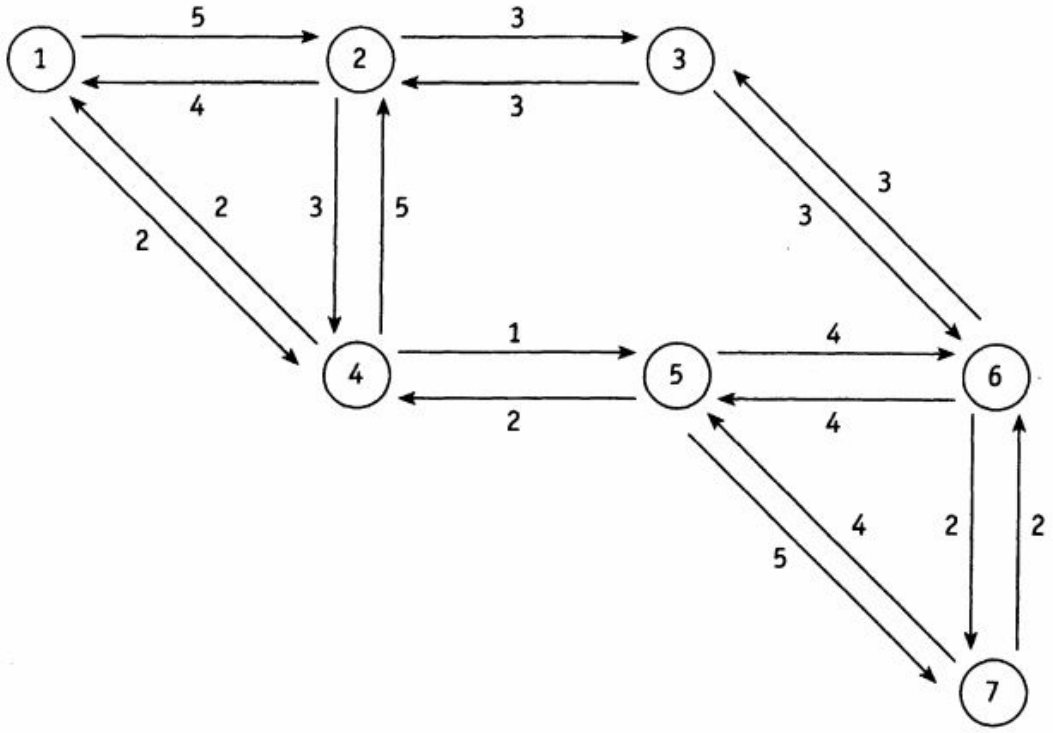
- Отметим вершину ***1***
- Из неотмеченных вершин минимальное значение соответствует вершине 6.
- Его длина, т.е. ***d[6]***= 6.



# Шаг 5

| Отмеченные вершины | Расстояние до вершины 1 | 2 | 3 | 4 | 5 | 6 | 7 | Неотмеченные вершины |
|---|---|---|---|---|---|---|---|---|
| 2 | 4 | 0 | **3** | 3 | ∞ | ∞ | ∞ | 1, 3, 4, 5, 6, 7 |
| 3 | 4 | 0 | 3 | **3** | ∞ | 6 | ∞ | 1, 4, 5, 6, 7 |
| 4 | 4 | 0 | 3 | 3 | **4** | 6 | ∞ | 1, 5, 6, 7 |
| 5 | **4** | 0 | 3 | 3 | 4 | 6 | 9 | 1, 6, 7 |
| 1 | 4 | 0 | 3 | 3 | 4 | **6** | 9 | 6, 7 |
| 6 | 4 | 0 | 3 | 3 | 4 | 6 | **8** | 7 |

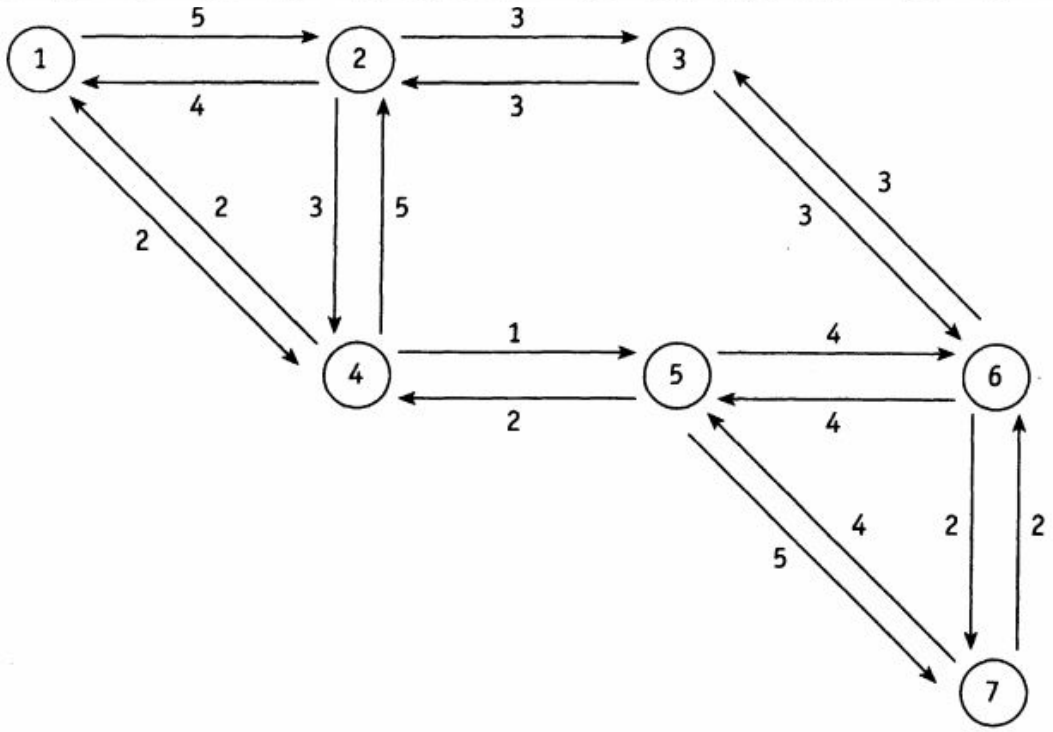
- Отметим вершину ***6***
- Из неотмеченных вершин минимальное осталась только вершина 7.
- длина, т.е. ***d[7]***= 8.



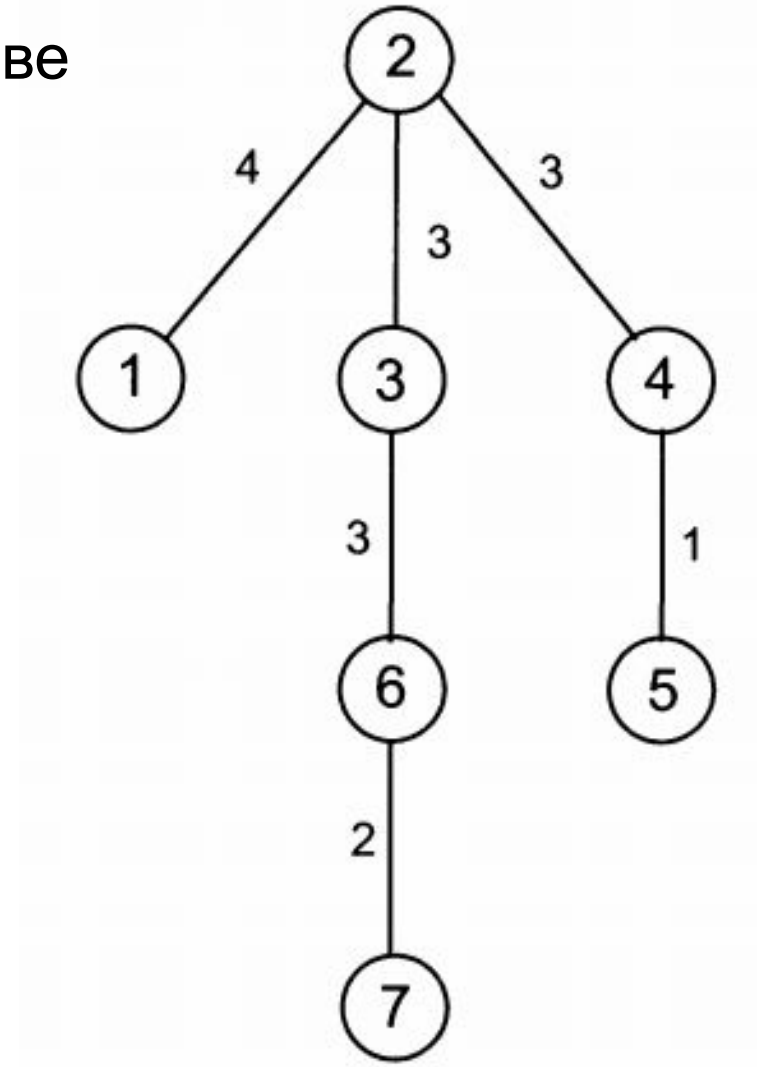
# Шаг 6

| Отмеченные вершины | Расстояние до вершины 1 | 2 | 3 | 4 | 5 | 6 | 7 | Неотмеченные вершины |
|---|---|---|---|---|---|---|---|---|
| 2 | 4 | 0 | **3** | 3 | ∞ | ∞ | ∞ | 1, 3, 4, 5, 6, 7 |
| 3 | 4 | 0 | 3 | **3** | ∞ | 6 | ∞ | 1, 4, 5, 6, 7 |
| 4 | 4 | 0 | 3 | 3 | **4** | 6 | ∞ | 1, 5, 6, 7 |
| 5 | **4** | 0 | 3 | 3 | 4 | 6 | 9 | 1, 6, 7 |
| 1 | 4 | 0 | 3 | 3 | 4 | **6** | 9 | 6, 7 |
| 6 | 4 | 0 | 3 | 3 | 4 | 6 | **8** | 7 |
| 7 | 4 | 0 | 3 | 3 | 4 | 6 | 8 | |

- Отметим последнюю вершину 7



Ответ:




| Отмеченные вершины | Расстояние до вершины 1 | 2 | 3 | 4 | 5 | 6 | 7 | Неотмеченные вершины |
|---|---|---|---|---|---|---|---|---|
| 2 | 4 | 0 | **3** | 3 | ∞ | ∞ | ∞ | 1, 3, 4, 5, 6, 7 |
| 3 | 4 | 0 | 3 | **3** | ∞ | 6 | ∞ | 1, 4, 5, 6, 7 |
| 4 | 4 | 0 | 3 | 3 | **4** | 6 | ∞ | 1, 5, 6, 7 |
| 5 | **4** | 0 | 3 | 3 | 4 | 6 | 9 | 1, 6, 7 |
| 1 | 4 | 0 | 3 | 3 | 4 | **6** | 9 | 6, 7 |
| 6 | 4 | 0 | 3 | 3 | 4 | 6 | **8** | 7 |
| 7 | 4 | 0 | 3 | 3 | 4 | 6 | 8 | |

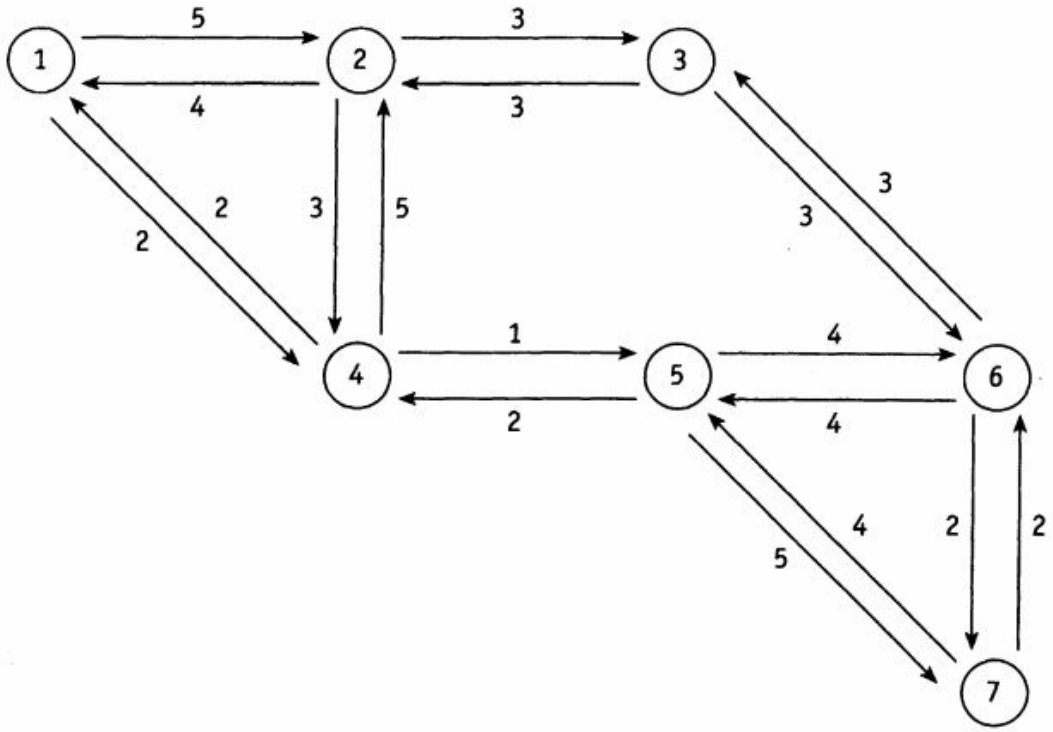
**Рис. 3. Дерево кратчайших путей от вершины 2 к любой другой**

# Ответ:

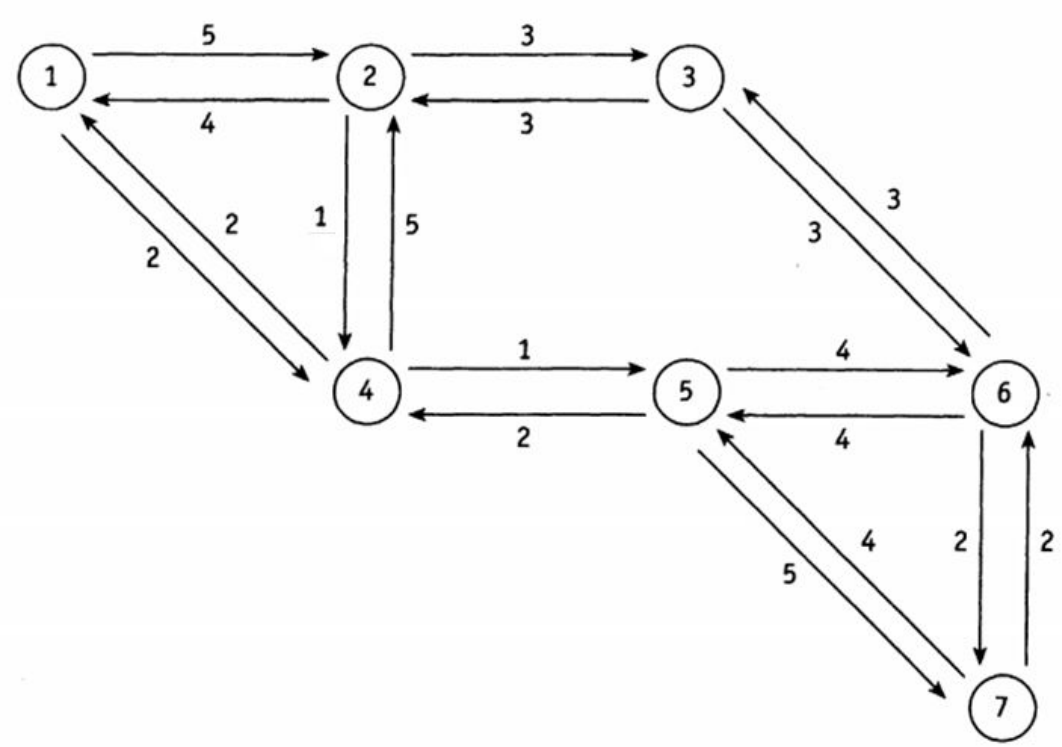
**таблица маршрутов для узла 2**

| Адресат | 1 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|
| Следующий узел | 1 | 3 | 4 | 4 | 3 | 3 |

| Отмеченные вершины | Расстояние до вершины | | | | | | | Неотмеченные вершины |
|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | |
| 2 | 4 | 0 | **3** | 3 | ∞ | ∞ | ∞ | 1, 3, 4, 5, 6, 7 |
| 3 | 4 | 0 | 3 | **3** | ∞ | 6 | ∞ | 1, 4, 5, 6, 7 |
| 4 | 4 | 0 | 3 | 3 | **4** | 6 | ∞ | 1, 5, 6, 7 |
| 5 | **4** | 0 | 3 | 3 | 4 | 6 | 9 | 1, 6, 7 |
| 1 | 4 | 0 | 3 | 3 | 4 | **6** | 9 | 6, 7 |
| 6 | 4 | 0 | 3 | 3 | 4 | 6 | **8** | 7 |
| 7 | 4 | 0 | 3 | 3 | 4 | 6 | 8 | |



- **Задача 2.** Предположим, что временная задержка передачи от узла 2 к узлу 4 уменьшилась с 3 до 1. Как изменятся при этом дерево кратчайших путей и таблица маршрутов для узла 2?



# Перезапустим алгоритм Дейкстры

**Рис. 4. Дерево кратчайших путей от узла 2**

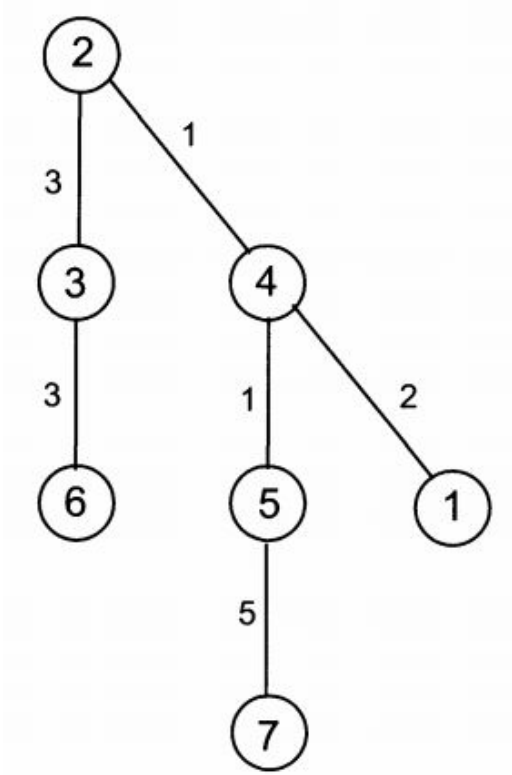
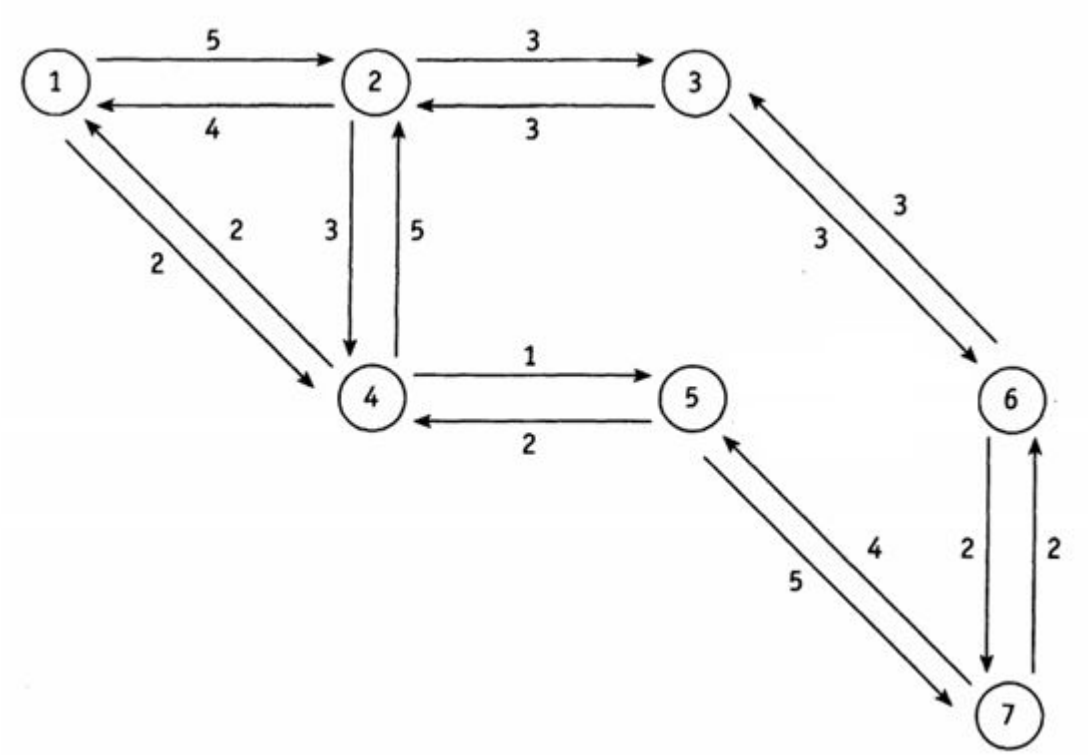



**таблица маршрутов для узла 2**

| Адресат | 1 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|
| Следующий узел | 4 | 3 | 4 | 4 | 3 | 4 |

- **Задача 3.** Какими будут дерево кратчайших путей и таблица маршрутов для узлов 1 и 2, если удалить линии между узлами 5 и 6?



- **Решение.**
- Поскольку линия **5** → **6** не задействована при передаче информации от узла 2, то его дерево кратчайших путей и таблица
маршрутов, найденные в задаче 1, останутся без изменений.

- Что касается узла 1, то мы можем ограничиться поиском кратчайшего пути от узла 1 к узлу 6.
- Алгоритм Дейкстры найдет следующий путь: $1 \rightarrow 4 \rightarrow 5 \rightarrow 7 \rightarrow 6$.

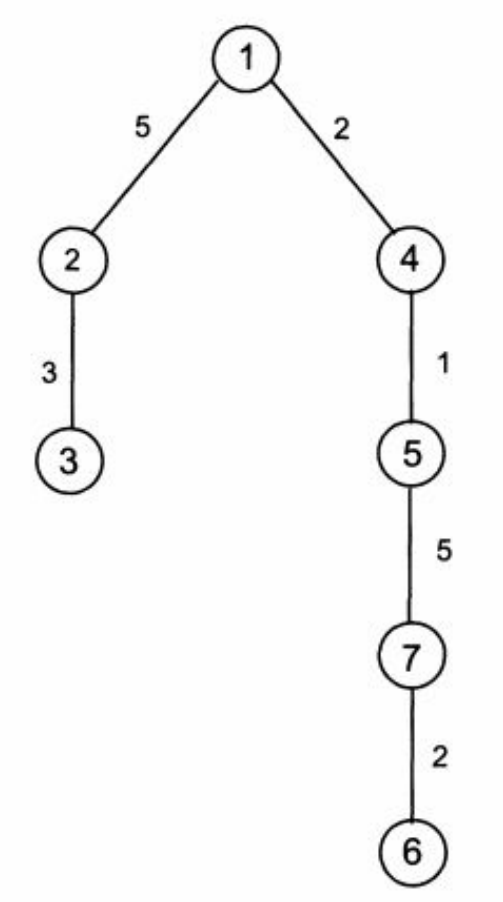



**Рис. 5. Новое дерево кратчайших путей**

## Таблица маршрутов

| Адресат | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|
| Следующий узел | 2 | 2 | 4 | 4 | 4 | 4 |