6. Алгоритмы на графах

6.1 Обходы графов

С точки зрения прикладных исследований важнейшими задачами являются задачи глобального анализа графов (систем, моделируемых с помощью графов).

Для решения таких задач необходимо организовывать систематический перебор («обход») всех вершин графа так, чтобы каждая вершина просматривалась ровно один раз.

Такой обход обычно сопровождается нумерацией просматриваемых вершин (и иногда, ребер).

Широко известны две стратегии обхода вершин, называемые ***обходом*** (***поиском***) ***в глубину*** и ***в ширину***.

Поиск в глубину

Поиск в глубину (*depth first search*, *DFS*) – это рекурсивный алгоритм обхода вершин графа.

Существует и нерекурсивный вариант алгоритма поиска в глубину (использование рекурсии заменяется на использование специальной структуры данных – *стека*).

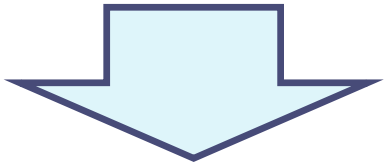
Основные действия:

обработав вершину в некотором ярусе, спускаемся по некоторому ребру на следующий ярус;

после обработки вершины текущего яруса спускаемся снова и т. д.;

если доходим до «тупика» (все вершины, смежные с текущей, уже просмотрены) – возвращаемся наверх на один ярус и ищем ребро, по которому ещё не спускались;

если такого ребра нет, поднимаемся ещё на ярус и т.д.



Основная идея поиска в глубину:

когда возможные пути по ребрам, выходящим из вершин, разветвляются, нужно сначала полностью исследовать одну ветку и только потом переходить к другим веткам.

Описание алгоритма с использованием рекурсивной процедуры DFS.

Вход: граф ***G***(***V***, ***E***), представленный списками смежности вершин (или матрицей смежности);

выход: последовательность вершин обхода.

1. $i = 1$

Глобальная переменная, которая определяет порядок обхода вершин

2. Для всех $v \in V$

   num[v] = 0

Все вершины графа помечаются как не просмотренные

3. Для всех $v \in V$

   если **num**[v] = 0,

   то ***DFS***(v)

Вызов рекурсивной процедуры *DFS*

Описание процедуры *DFS*(*v*):

1. **num**[***v***] = ***i***,

Вершине *v* присваивается очередной порядковый номер

2. ***i*** = ***i***+1

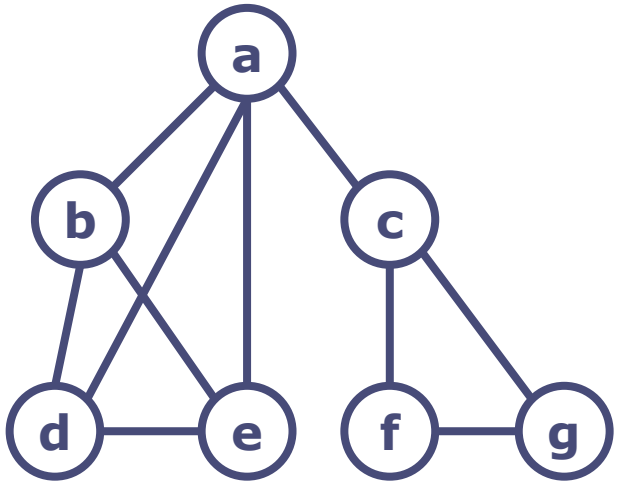
3. Для всех $u \in V$, смежных с вершиной ***v***

если **num**[***u***] = 0

то ***DFS***(***u***)

Если вершины *u* помечена как не просмотренная, то вызов процедуры *DFS* для этой вершины

Пример.



Списки смежности вершин:

- ***a*** $\rightarrow b \rightarrow c \rightarrow d \rightarrow e$;
- ***b*** $\rightarrow a \rightarrow d \rightarrow e$
- ***c*** $\rightarrow a \rightarrow f \rightarrow g$
- ***d*** $\rightarrow a \rightarrow b \rightarrow e$
- ***e*** $\rightarrow a \rightarrow b \rightarrow d$
- ***f*** $\rightarrow c \rightarrow g$
- ***g*** $\rightarrow c \rightarrow f$

Матрица смежности:

$$M = \begin{array}{c} \begin{matrix} a & b & c & d & e & f & g \end{matrix} \\ \begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 \end{pmatrix} \end{array} \begin{matrix} a \\ b \\ c \\ d \\ e \\ f \\ g \end{matrix}$$

Протокол работы алгоритма.

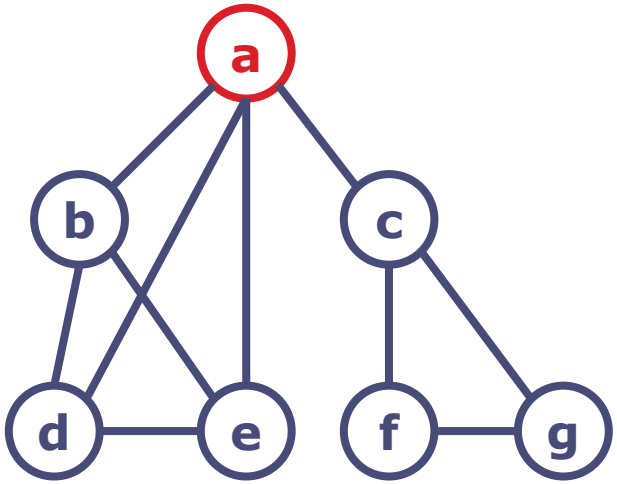
- ***i*** = 1;
- все вершины помечаются как не просмотренные:

	a	***b***	***c***	***d***	***e***	***f***	***g***
num	**0**	**0**	**0**	**0**	**0**	**0**	**0**

- выбирается вершина ***a***: ***v*** = ***a***;
- вызов ***DFS***(***a***);
  - вершине ***a*** присваивается номер 1 (помечается как просмотренная):

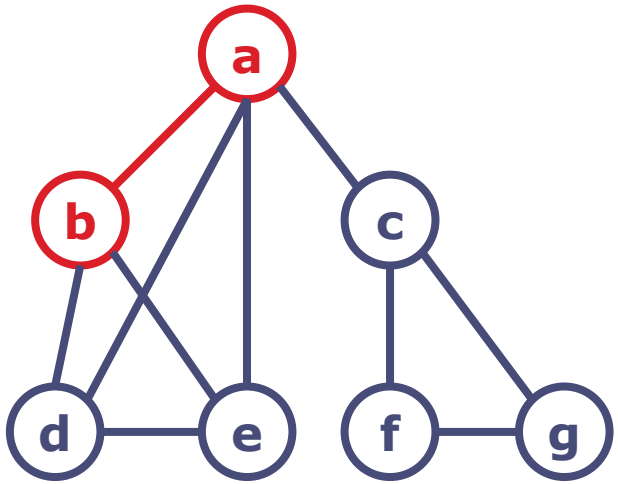
    num[***a***] = ***i*** = 1,

a	*b*	*c*	*d*	*e*	*f*	*g*
1	**0**	**0**	**0**	**0**	**0**	**0**



- ***i*** = 2;
- обход списка смежности вершины ***a***:
  - ***u*** = ***b***;
  - вызов ***DFS***(***b***);
    - вершине ***b*** присваивается номер 2 (помечается как просмотренная): **num**[***b***] = ***i*** = 2;

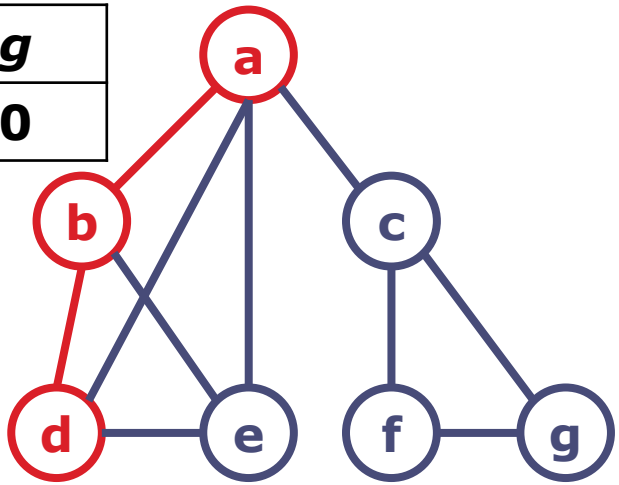
a	*b*	*c*	*d*	*e*	*f*	*g*
1	**2**	**0**	**0**	**0**	**0**	**0**



- ***i*** = 3;
- обход списка смежности вершины ***b***:
  - ***u*** = ***a***;
  - ***u*** = ***d***;
  - вызов ***DFS***(***d***);
    - вершине ***d*** присваивается номер 3 (помечается как просмотренная):

      num[***d***] = ***i*** = 3;

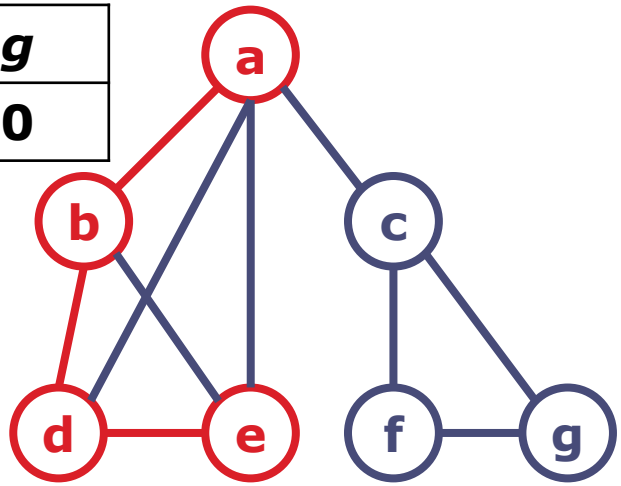
a	***b***	***c***	***d***	***e***	***f***	***g***
1	**2**	**0**	**3**	**0**	**0**	**0**



- ***i*** = 4;
- обход списка смежности вершины ***d***:
  - ***u*** = ***a***;
  - ***u*** = ***b***;
  - ***u*** = ***e***;
  - вызов ***DFS***(***e***);
    - вершине ***e*** присваивается номер 4 (помечается как просмотренная):

      num[***e***] = ***i*** = 4;

a	***b***	***c***	***d***	***e***	***f***	***g***
1	**2**	**0**	**3**	**4**	**0**	**0**

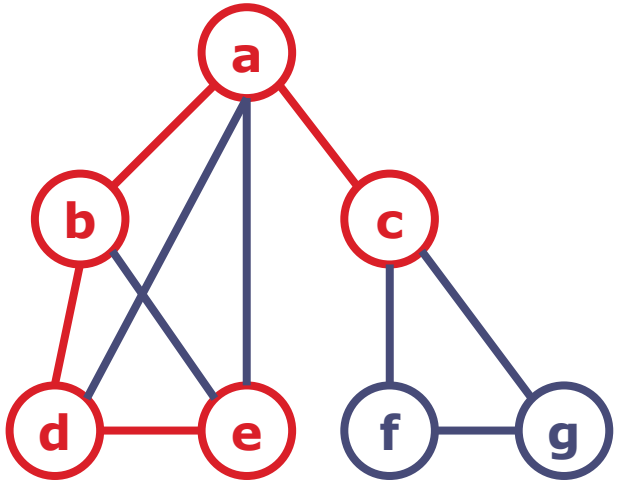


- ***i*** = 5;
- обход списка смежности вершины ***e***:
  - ***u*** = ***a***;
  - ***u*** = ***b***;
  - ***u*** = ***d***;
  - все вершины просмотрены → возвращаемся к списку смежности вершины ***d***;
- все вершины просмотрены → возвращаемся к списку смежности вершины ***b***;
- все вершины просмотрены → возвращаемся к списку смежности вершины ***a***;

- ***u*** = ***c***;
- вызов ***DFS***(***c***);
  - вершине ***c*** присваивается номер 5 (помечается как просмотренная):

    num[***c***] = ***i*** = 5;

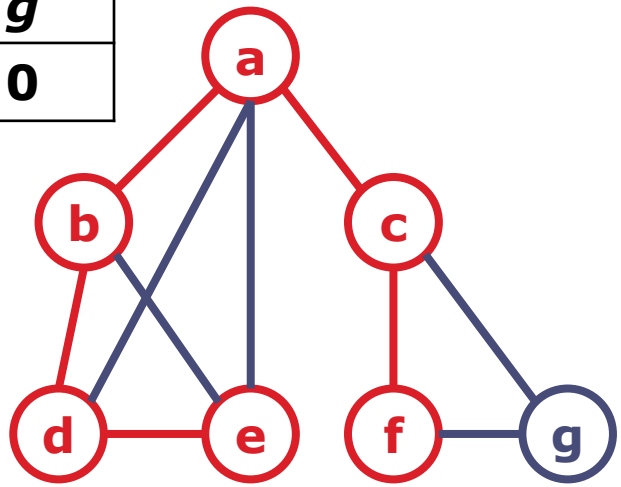
a	*b*	*c*	*d*	*e*	*f*	*g*
1	**2**	**5**	**3**	**4**	**0**	**0**



- ***i*** = 6;
- обход списка смежности вершины ***c***:
  - ***u*** = ***a***;
  - ***u*** = ***f***;
  - вызов ***DFS***(***f***);
    - вершине ***f*** присваивается номер 6 (помечается как просмотренная):

      num[***f***] = ***i*** = 6;

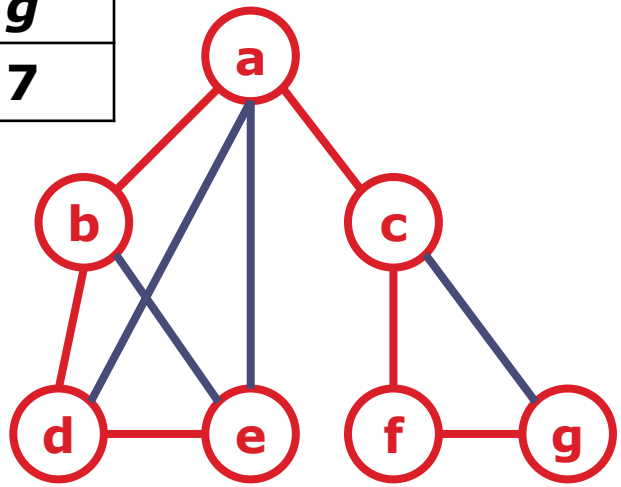
a	***b***	***c***	***d***	***e***	***f***	***g***
1	**2**	**5**	**3**	**4**	**6**	**0**



- ***i*** = 7;
- обход списка смежности вершины ***f***:
  - ***u*** = ***c***;
  - ***u*** = ***g***;
  - вызов ***DFS***(***g***);
    - вершине ***g*** присваивается номер 7 (помечается как просмотренная):

      num[***g***] = ***i*** = 7;

a	***b***	***c***	***d***	***e***	***f***	***g***
1	**2**	**5**	**3**	**4**	**6**	**7**



- ***i*** = 8;
- обход списка смежности вершины ***g***:
  - ***u*** = ***c***;
  - ***u*** = ***f***;
  - все вершины просмотрены → возвращаемся к списку смежности вершины ***f***;
- все вершины просмотрены → возвращаемся к списку смежности вершины ***c***;
- все вершины просмотрены → возвращаемся к списку смежности вершины ***a***;
- все вершины просмотрены → алгоритм заканчивает работу.

Свойство ориентированных деревьев (напоминание):

если в свободном дереве любую вершину назначить корнем и задать ориентацию ребер «от корня», то получится ордерево.

Следствие:

алгоритм поиска в глубину строит ордерево с корнем в начальном узле.

Поиск в ширину

Поиск в ширину (*breadth first search*, *BFS*) предполагает реализацию следующих основных действий:

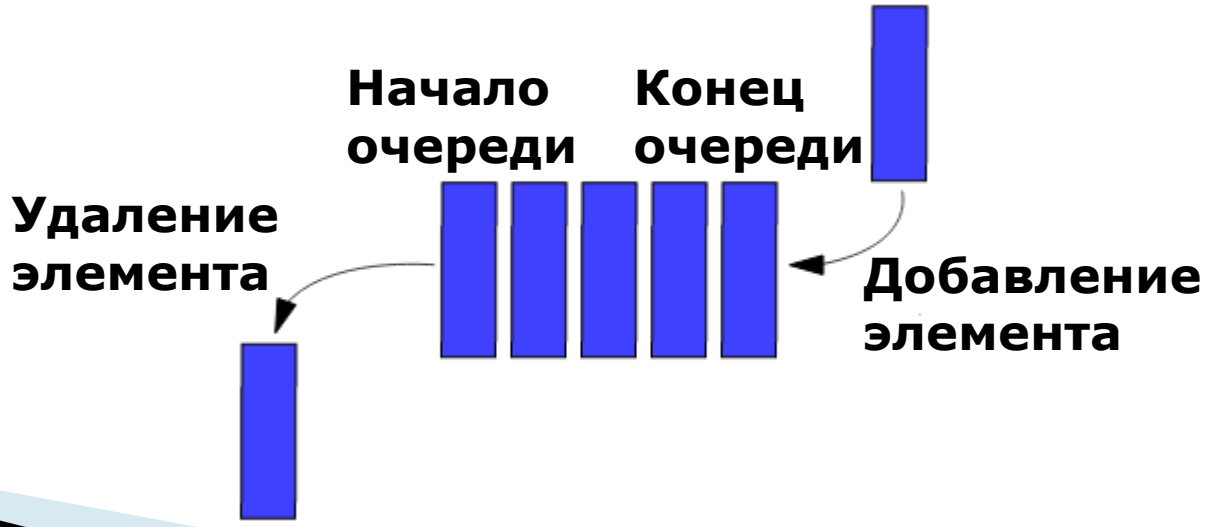
обработав вершину в некотором ярусе, переходим к следующей вершине этого же яруса;

последовательно обрабатываем все вершины текущего яруса, и только после этого переходим к вершинам следующего (т. е. к вершинам, смежным с вершинами текущего яруса);

на каждом новом шаге просматриваются вершины, расстояние от которых до начальной на единицу больше предыдущего.

Для реализации поиска в ширину необходима структура данных, которая называется ***очередью***.

Очередь – это структура данных с дисциплиной доступа к элементам «первый пришёл – первый ушел» (*FIFO*, *First In – First Out*). Добавление элемента возможно лишь в конец очереди, выборка – только из начала очереди, причем выбранный элемент удаляется из очереди.



Поиск начинается с некоторой вершины ***v***.

Эта вершина помещается в очередь, и с этого момента считается просмотренной.

После этого все вершины, смежные с ***v***, помещаются в очередь и получают статус просмотренных, а вершина ***v*** из очереди удаляется.

Если очередь оказывается пустой, то это значит, что совершен обход одной из компонент связности графа.

Если остались непросмотренные вершины (в случае несвязного графа), поиск продолжается из некоторой непросмотренной вершины.

Описание алгоритма с использованием очереди Q.

Вход: граф ***G***(***V***, ***E***), представленный списками смежности вершин (или матрицей смежности);

выход: последовательность вершин обхода.

1. $i = 1$

Глобальная переменная, которая определяет порядок обхода вершин

2. Для всех $v \in V$

   num[v] = 0

Все вершины графа помечаются как не просмотренные

3. Для всех $v \in V$

   если **num**[v] = 0, то

   3.1 Создается пустая очередь ***Q***

   3.2 вершина ***v*** помещается в очередь

   3.3 **num**[v] = i

   3.4 $i = i+1$

Вершине *v* присваивается очередной порядковый номер

   3.5 пока очередь не пуста, выполнять:

3.5.1 из очереди извлекается вершина ***u***

3.5.2 для всех $w \in V$, смежных с вершиной ***u***

если **num**[***w***] = 0, то

Если вершина *w* помечена как не просмотренная

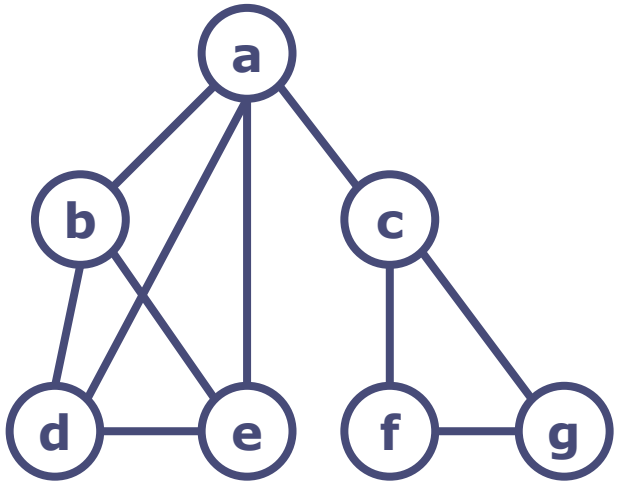
3.5.2.1 вершина ***w*** помещается в очередь

3.5.2.2 **num**[***w***] = ***i***

3.5.3.3 ***i*** = ***i***+1

Вершине *w* присваивается очередной порядковый номер

Пример.



Списки смежности вершин:

- ***a*** $\rightarrow b \rightarrow c \rightarrow d \rightarrow e$;
- ***b*** $\rightarrow a \rightarrow d \rightarrow e$
- ***c*** $\rightarrow a \rightarrow f \rightarrow g$
- ***d*** $\rightarrow a \rightarrow b \rightarrow e$
- ***e*** $\rightarrow a \rightarrow b \rightarrow d$
- ***f*** $\rightarrow c \rightarrow g$
- ***g*** $\rightarrow c \rightarrow f$

Матрица смежности:

$$M = \begin{array}{c} \begin{matrix} a & b & c & d & e & f & g \end{matrix} \\ \begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 \end{pmatrix} \end{array} \begin{matrix} a \\ b \\ c \\ d \\ e \\ f \\ g \end{matrix}$$

Протокол работы алгоритма.

- ***i*** = 1;
- все вершины помечаются как не просмотренные:

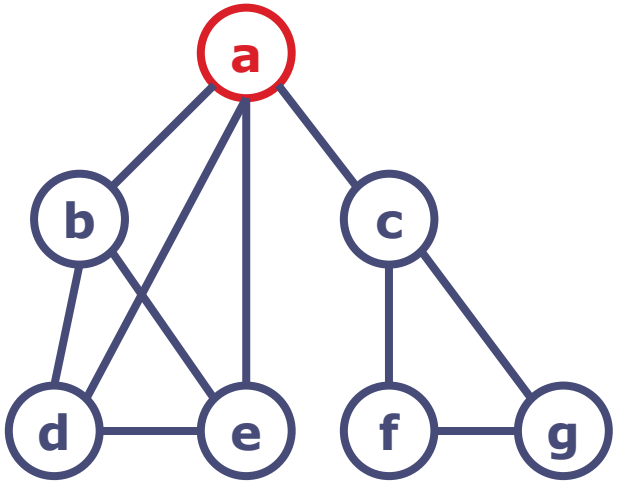
	a	***b***	***c***	***d***	***e***	***f***	***g***
num	**0**	**0**	**0**	**0**	**0**	**0**	**0**

- выбирается вершина ***a***: ***v*** = ***a***;
  - создается пустая очередь;
  - вершина ***a*** помещается в очередь; ей присваивается номер 1 (помечается как просмотренная):

    num[***a***] = ***i*** = 1,

a	*b*	*c*	*d*	*e*	*f*	*g*
1	0	0	0	0	0	0

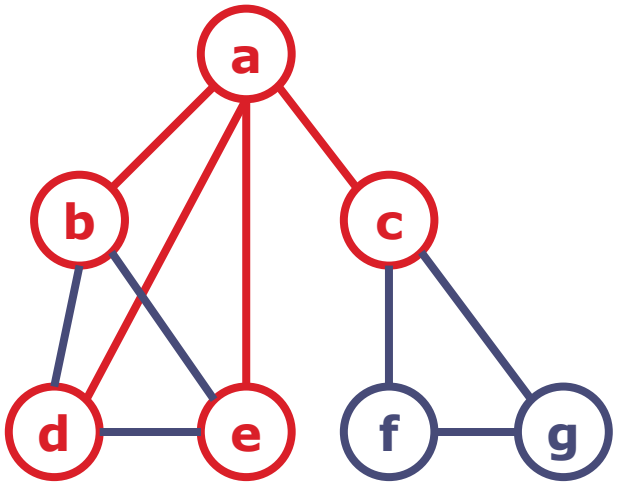
Q | *a* |



- ***i*** = 2;
  - из очереди извлекается вершина ***a***: ***u*** = ***a***;
  - обход списка смежности вершины ***a***:
    - ***w*** = ***b***, вершина ***b*** помещается в очередь, и ей присваивается номер 2: **num**[***b***] = ***i*** = 2;
    - ***i*** = 3;
    - ***w*** = ***c***, вершина ***c*** помещается в очередь, и ей присваивается номер 3: **num**[***c***] = ***i*** = 3;
    - ***i*** = 4;
    - ***w*** = ***d***, вершина ***d*** помещается в очередь, и ей присваивается номер 4: **num**[***d***] = ***i*** = 4;
    - ***i*** = 5;
    - ***w*** = ***e***, вершина ***e*** помещается в очередь, и ей присваивается номер 5: **num**[***e***] = ***i*** = 5;
    - ***i*** = 6;

a	*b*	*c*	*d*	*e*	*f*	*g*
1	2	3	4	5	0	0

Q	*b*	*c*	*d*	*e*



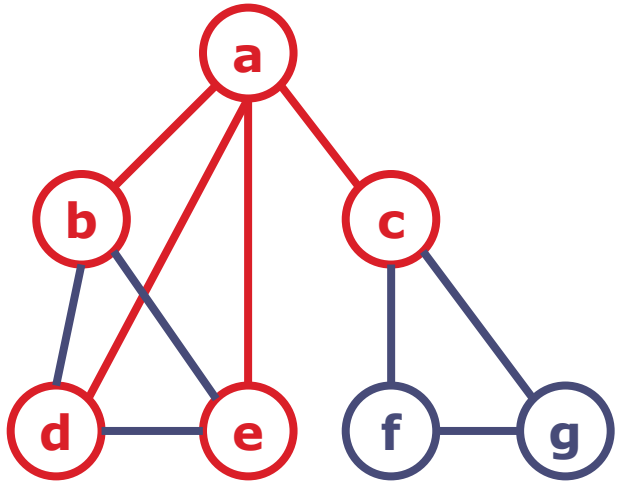
- из очереди извлекается вершина ***b***: ***u*** = ***b***;
- обход списка смежности вершины ***b***:
  - ***w*** = ***a***,
  - ***w*** = ***d***,
  - ***w*** = ***e***,

  все вершины просмотрены;

a	***b***	***c***	***d***	***e***	***f***	***g***
1	**2**	**3**	**4**	**5**	**0**	**0**

Q

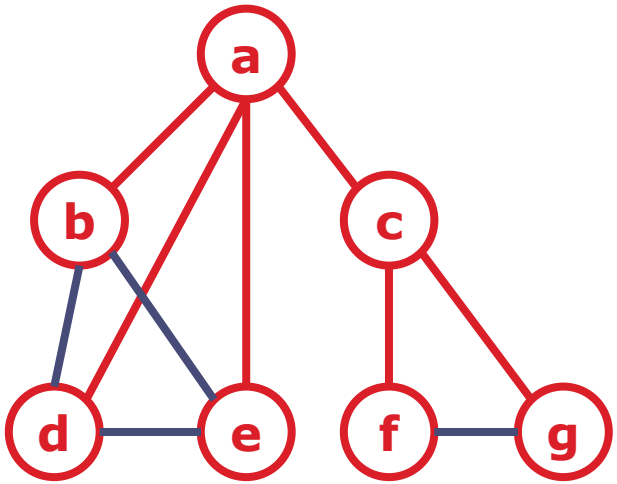
c	***d***	***e***



- из очереди извлекается вершина ***c***: ***u*** = ***c***;
- обход списка смежности вершины ***c***:
  - ***w*** = ***a***,
  - ***w*** = ***f***, вершина ***f*** помещается в очередь, и ей присваивается номер 6: **num**[***f***] = ***i*** = 6;
  - ***i*** = 7;
  - ***w*** = ***g***, вершина ***g*** помещается в очередь, и ей присваивается номер 7: **num**[***g***] = ***i*** = 7;
  - ***i*** = 8;

a	*b*	*c*	*d*	*e*	*f*	*g*
1	2	3	4	5	6	7

Q	*d*	*e*	*f*	*g*



- из очереди извлекается вершина ***d***: ***u*** = ***d***;
- обход списка смежности вершины ***d***:
  - ***w*** = ***a***,
  - ***w*** = ***b***,
  - ***w*** = ***e***,

  все вершины просмотрены;

Q | ***e*** | ***f*** | ***g***

- из очереди извлекается вершина ***e***: ***u*** = ***e***;
- обход списка смежности вершины ***e***:
  - ***w*** = ***a***,
  - ***w*** = ***b***,
  - ***w*** = ***d***,

  все вершины просмотрены;

Q | ***f*** | ***g***

- из очереди извлекается вершина ***f***: ***u*** = ***f***;
- обход списка смежности вершины ***f***:
  - ***w*** = ***c***,
  - ***w*** = ***g***,

    все вершины просмотрены;

Q | ***g*** |

- из очереди извлекается вершина ***g***: ***u*** = ***g***;
- обход списка смежности вершины ***g***:
  - ***w*** = ***c***,
  - ***w*** = ***f***,

    все вершины просмотрены;

Q |

  - очередь пуста;
- все вершины просмотрены → алгоритм заканчивает работу.

Замечание.

Нерекурсивный вариант алгоритма поиска в глубину, фактически, может быть получен из рассмотренного алгоритма поиска в ширину путем замены очереди на стек.

Теорема.

Пусть ***G***(***V***, ***E***) – связный (***n***, ***m***)-граф. Тогда поиск в ширину и поиск в глубину просматривает каждую вершину ровно 1 раз.

Идеи алгоритмов поиска в глубину и в ширину лежат в основе огромного числа конкретных алгоритмов на графах.

Как правило, эти идеи предполагаются заранее известными, и изложение того или иного алгоритма сразу погружается в технические детали применения общей концепции поиска в ширину и в глубину для решения конкретной задачи.

6.2 Кратчайшие пути

Применение рассмотренного ранее алгоритма Уоршалла позволяет получить ответ на вопрос: достижима ли вершина ***v*** из вершины ***u***

(т. е. существует ли цепь $\langle u, v \rangle$)?

В прикладных задачах обычно требуется не только определить, существует ли цепь, соединяющая данные вершины, но и найти эту цепь, а также ее длину.

Типичная задача – найти кратчайшую цепь (путь), соединяющую заданные вершины.

Длина пути в нагруженном графе

Напоминание:

в нагруженном графе $G(V, E)$ ребрам (дугам) приписаны некоторые числа c_{ij} (*веса* или *длины* дуг).

Такой граф может быть представлен *матрицей весов* (*длин*) C:

$$c_{ij} = \begin{cases} 0, & \text{если } i = j, \\ \text{вес ребра } (v_i, v_j), & \text{если } (v_i, v_j) \in E, \\ \infty, & \text{если } (v_i, v_j) \notin E. \end{cases}$$

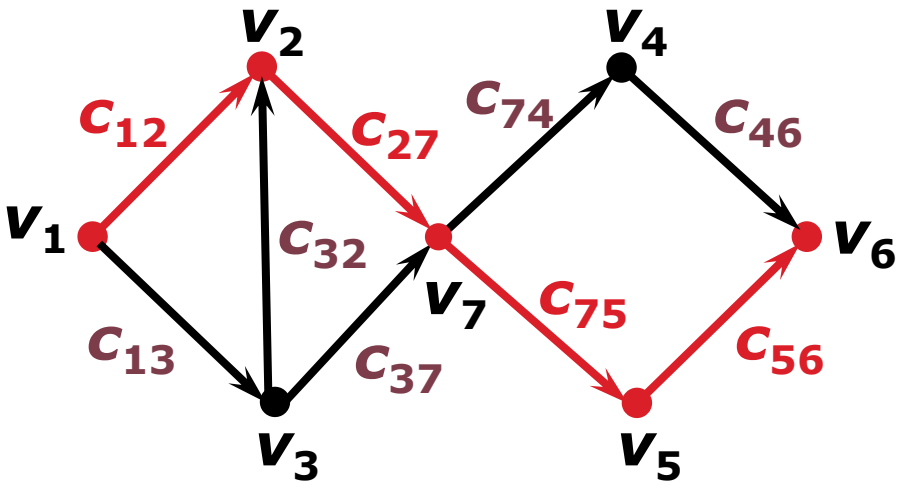
Длиной пути в нагруженном графе называется сумма длин дуг, входящих в этот путь.

Пример.

Пусть

$$M = v_1, v_2, v_7, v_5, v_6,$$

$$|M| = c_{12} + c_{27} + c_{75} + c_{56}.$$



Для решения ***задачи отыскания кратчайшего пути в графе*** существует несколько известных алгоритмов.

Рассматриваемые далее алгоритмы применимы как к неорграфам, так и к орграфам.

Алгоритм Флойда

Вход: ***C*** – матрица весов (длин) дуг (квадратная матрица порядка ***n***);

выход: ***T*** – матрица длин кратчайших путей (квадратная матрица порядка ***n***);

H – матрица кратчайших путей (квадратная матрица порядка ***n***):

$$h_{ij} = \begin{cases} k, & \text{если } v_k - \text{первая вершина на кратчайшем пути } \langle \overrightarrow{v_i, v_j} \rangle, \\ 0, & \text{если не существует пути } \langle \overrightarrow{v_i, v_j} \rangle. \end{cases}$$

Описание алгоритма:

1. Для $i = 1, 2, \ldots, n$

для $j = 1, 2, \ldots, n$

$t_{ij} = c_{ij}$;

если $c_{ij} = \infty$, то

$h_{ij} = 0$

иначе

$h_{ij} = j$

Инициализация

Нет дуги (v_i, v_j)

Есть дуга (v_i, v_j)

2. Для $i = 1, 2, \ldots, n$

2.1 Для $j = 1, 2, \ldots, n$

для $k = 1, 2, \ldots, n$

если $i \neq j \;\&\; t_{ji} \neq \infty \;\&\; i \neq k \;\&\; t_{ik} \neq \infty \;\&\; \&\; \left(t_{jk} = \infty \vee t_{jk} > t_{ji} + t_{ik}\right)$

то

$h_{jk} = h_{ji}$;

Запоминаем новый путь

$t_{jk} = t_{ji} + t_{ik}$

Запоминаем длину нового пути

конец цикла по k

конец цикла по j

2.2 Для всех $j = 1, 2, \ldots, n$

если $t_{jj} < 0$

то стоп

конец цикла по i

Нет решения: вершина v_j входит в цикл отрицательной длины (на таком цикле можно получить сколь угодно короткий путь)

Замечание.

Элемент h_{ij} матрицы H хранит номер вершины, являющейся первой в последовательности вершин кратчайшего пути $\langle \overrightarrow{v_i, v_j} \rangle$ (если такой путь существует).

Весь путь (номера вершин) может быть восстановлен следующим простым алгоритмом:

$w = i$

вывод w

пока $w \neq j$

выполнять:

$w = h_{wj}$;

вывод w

Такой способ хранения кратчайших путей является более экономным

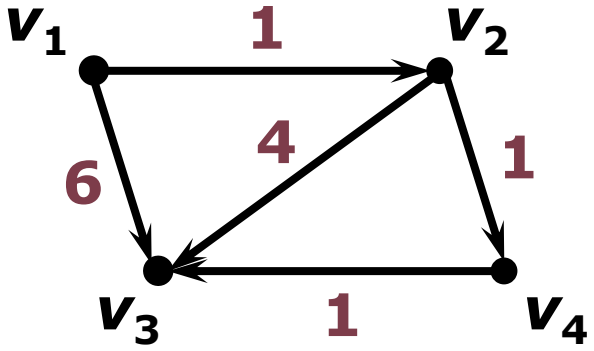
(требуемый объем памяти составляет $O(n^2)$)

по сравнению с непосредственным представлением всех путей

(требуемый объем памяти – $O(n^3)$).

Пример.

Рассмотрим граф:



Матрица весов (длин) имеет вид

$$C = \begin{pmatrix} 0 & 1 & 6 & \infty \\ \infty & 0 & 4 & 1 \\ \infty & \infty & 0 & \infty \\ \infty & \infty & 1 & 0 \end{pmatrix}$$

После первого шага алгоритма Флойда

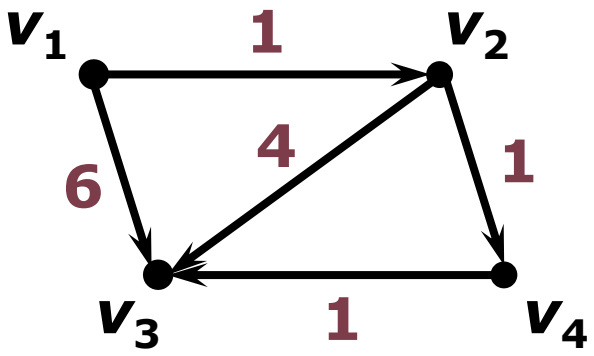
$$T = C, \qquad H = \begin{pmatrix} 1 & 2 & 3 & 0 \\ 0 & 2 & 3 & 4 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 3 & 4 \end{pmatrix}$$

Шаг 2:

$i = 1$

условие $i \neq j \ \& \ t_{ji} \neq \infty$

не выполняется ни при каком j



$i = 2$

$j = 1$

$k = 3$

$$i \neq j \ \& \ t_{ji} \neq \infty \ \& \ i \neq k \ \& \ t_{ik} \neq \infty \ \& \ t_{jk} > t_{ji} + t_{ik}$$

$$h_{13} = h_{12} = 2$$

$$t_{13} = t_{12} + t_{23} = 1 + 4 = 5$$

$k = 4$

$$i \neq j \ \& \ t_{ji} \neq \infty \ \& \ i \neq k \ \& \ t_{ik} \neq \infty \ \& \ t_{jk} = \infty$$

$$h_{14} = h_{12} = 2$$

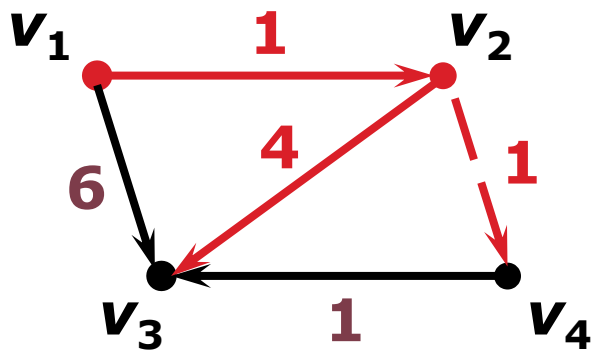
$$t_{14} = t_{12} + t_{24} = 1 + 1 = 2$$

Итоги после рассмотрения $i = 1, 2$:

$h_{13} = 2$, $h_{14} = 2$,

$t_{13} = 5$, $t_{14} = 2$

$$T = \begin{pmatrix} 0 & 1 & 5 & 2 \\ \infty & 0 & 4 & 1 \\ \infty & \infty & 0 & \infty \\ \infty & \infty & 1 & 0 \end{pmatrix} \qquad H = \begin{pmatrix} 1 & 2 & 2 & 2 \\ 0 & 2 & 3 & 4 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 3 & 4 \end{pmatrix}$$



Шаг 2 (продолжение):

$i = 3$

условие $i \neq k \;\&\; t_{ik} \neq \infty$

не выполняется ни при каком k

$$T = \begin{pmatrix} 0 & 1 & 5 & 2 \\ \infty & 0 & 4 & 1 \\ \infty & \infty & 0 & \infty \\ \infty & \infty & 1 & 0 \end{pmatrix}$$

$i = 4$

$j = 1$

$k = 3$

$$i \neq j \;\&\; t_{ji} \neq \infty \;\&\; i \neq k \;\&\; t_{ik} \neq \infty \;\&\; t_{jk} > t_{ji} + t_{ik}$$

$$h_{13} = h_{14} = 2$$

$$t_{13} = t_{14} + t_{43} = 2 + 1 = 3$$

$j = 2$

$k = 3$

$$i \neq j \;\&\; t_{ji} \neq \infty \;\&\; i \neq k \;\&\; t_{ik} \neq \infty \;\&\; t_{jk} > t_{ji} + t_{ik}$$

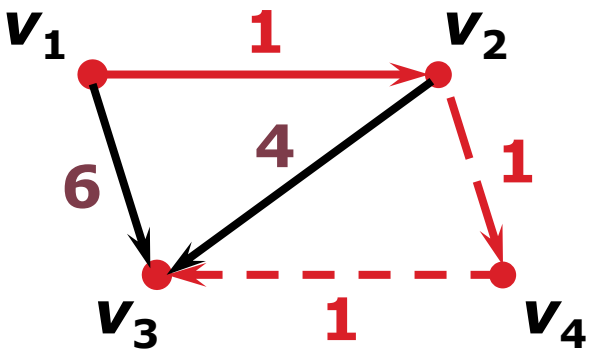
$$h_{23} = h_{24} = 4$$

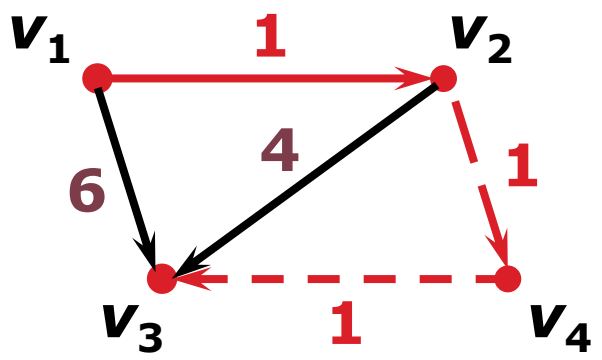
$$t_{23} = t_{24} + t_{43} = 1 + 1 = 2$$

Итоги после завершения шага 2:

$h_{23} = 4,$
$t_{23} = 2$

$$T = \begin{pmatrix} 0 & 1 & 3 & 2 \\ \infty & 0 & 2 & 1 \\ \infty & \infty & 0 & \infty \\ \infty & \infty & 1 & 0 \end{pmatrix} \qquad H = \begin{pmatrix} 1 & 2 & 2 & 2 \\ 0 & 2 & 4 & 4 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 3 & 4 \end{pmatrix}$$





$$H = \begin{pmatrix} 1 & 2 & 2 & 2 \\ 0 & 2 & 4 & 4 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 3 & 4 \end{pmatrix}$$

Используя полученную матрицу ***H*** и алгоритм, описанный в замечании выше, можно получить кратчайший путь, соединяющий заданные вершины.

Восстановим, например, $\langle \overrightarrow{v_1, v_3} \rangle$:

$w = 1$

$w = h_{wj} = h_{13} = 2$

$w = h_{wj} = h_{23} = 4$

$w = h_{wj} = h_{43} = 3$

Искомый путь:

v_1, v_2, v_4, v_3

Если веса (длины) всех ребер (дуг) графа (орграфа) **неотрицательны**, то для нахождения кратчайшего пути между заданными вершинами графа может применяться рассматриваемый далее алгоритм Дейкстры.

Алгоритм Дейкстры

Вход: ***C*** – матрица весов (длин) дуг (квадратная матрица порядка ***n***);

s, ***t*** – номера заданных вершин графа.

выход: ***T*** – вектор длин кратчайших путей (одномерный массив размерности ***n***):

$$t_i = \begin{cases} \text{длина кратчайшего пути } \left\langle \overrightarrow{v_s, v_i} \right\rangle, \text{ если вершина } v_i \\ \qquad\qquad \text{лежит на кратчайшем пути } \left\langle \overrightarrow{v_s, v_t} \right\rangle, \\ \infty \quad \text{в противном случае.} \end{cases}$$

H – вектор кратчайших путей (одномерный массив размерности ***n***):

h_i – номер вершины, непосредственно предшествующей v_i на кратчайшем пути.

Описание алгоритма:

1. Для $i = 1, 2, \ldots, n$

   $t_i = \infty$; **Кратчайший путь неизвестен**

   $x_i = 0$ **Ни одна вершина не отмечена**

2. $h_s = 0$ **Вершине s не предшествует никакая другая вершина**

   $t_s = 0$ **Длина кратчайшего пути $<v_s, v_s>$ равна 0**

   $x_s = 1$ **Вершина s помечена**

   $v = s$ **Текущая вершина – вершина s**

3. Для всех $u \in \Gamma(v)$

если $x_u = 0 \ \& \ t_u > t_v + c_{vu}$

то

$t_u = t_v + c_{vu}$;

$h_u = v$

Найден более короткий путь из *s* в *u* через *v*

Запоминаем новый путь

4. **4.1** $m = \infty$; $v = 0$

4.2 Для $u = 1, 2, \ldots, n$

Поиск конца кратчайшего пути

если $x_u = 0 \ \& \ t_u < m$

то

$v = u$;

$m = t_u$

Вершина *v* заканчивает кратчайший путь из *s*

конец цикла по u

4.3 Если $v = 0$

то

стоп

Нет пути из *s* в *t*

4.4 Если $v = t$

то

стоп

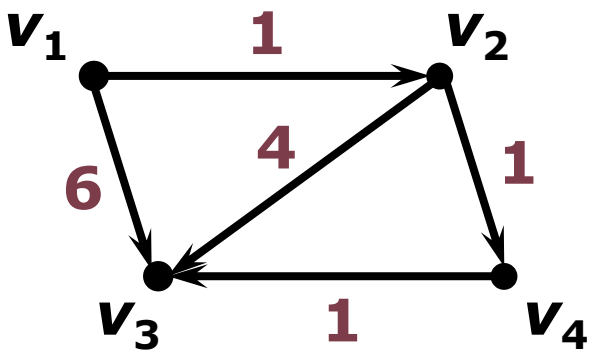
Найден кратчайший путь из *s* в *t*

4.5 $x_v = 1$;

переход на шаг 3

Пример.

Рассмотрим граф:



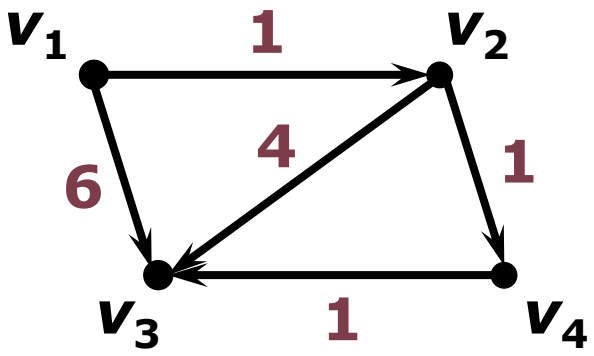
Матрица весов (длин) имеет вид

$$C = \begin{pmatrix} 0 & 1 & 6 & \infty \\ \infty & 0 & 4 & 1 \\ \infty & \infty & 0 & \infty \\ \infty & \infty & 1 & 0 \end{pmatrix}$$

Найдем кратчайший путь из вершины v_1 в v_3.

$s = 1$, $t = 3$.

1. $T = (\infty, \infty, \infty, \infty)$;
   $X = (0, 0, 0, 0)$.



2. $h_s = h_1 = 0$, $t_s = t_1 = 0$,
   $x_s = x_1 = 1$, $\underline{v = 1}$,
   $T = (0, \infty, \infty, \infty)$;
   $X = (1, 0, 0, 0)$; $H = (0$, не опр., не опр., не опр.$)$.

3. $\Gamma(v) = \{v_2, v_3\}$.
   $u = 2$
   $x_2 = 0$ & $t_2 > t_1 + c_{12} = 0 + 1 = 1$
   $t_2 = t_1 + c_{12} = 1$, $h_2 = 1$
   $u = 3$
   $x_3 = 0$ & $t_3 > t_1 + c_{13} = 0 + 6 = 6$
   $t_3 = t_1 + c_{13} = 6$, $h_3 = 1$

$\boldsymbol{T}$ = (0, 1, 6, ∞); $\boldsymbol{X}$ = (1, 0, 0, 0);

$\boldsymbol{H}$ = (0, 1, 1, не опр.).

4. **4.1** $\boldsymbol{m} = \infty$; $\underline{\boldsymbol{v} = 0}$

4.2 $\boldsymbol{u} = 1$

$\boldsymbol{x_1} \neq 0$

$\boldsymbol{u} = 2$

$\boldsymbol{x_2} = 0 \;\&\; \boldsymbol{t_2} < \infty$

$\underline{\boldsymbol{v} = 2}$;

$\boldsymbol{m} = \boldsymbol{t_2} = 1$

$\boldsymbol{u} = 3$

$\boldsymbol{t_3} > \boldsymbol{m}$

$\boldsymbol{u} = 4$

$\boldsymbol{t_4} > \boldsymbol{m}$

$x_u = 0 \;\&\; t_u < m$

4.3 $\boldsymbol{v} \neq 0$

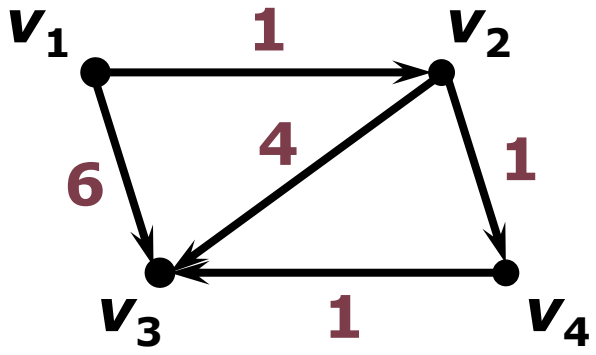
4.4 $\boldsymbol{v} \neq 3$

4.5 $\boldsymbol{x_v} = \boldsymbol{x_2} = 1$;

$\boldsymbol{T} = (0, 1, 6, \infty)$; $\boldsymbol{X} = (1, 1, 0, 0)$;

$\boldsymbol{H} = (0, 1, 1,$ не опр.).

Переход на шаг 3.



3. $\Gamma(v) = \{v_3, v_4\}$.

$u = 3$

$x_3 = 0$ & $t_3 > t_2 + c_{23} = 1 + 4 = 5$

$t_3 = t_2 + c_{23} = 5$, $h_3 = 2$

$u = 4$

$x_4 = 0$ & $t_4 > t_2 + c_{24} = 1 + 1 = 2$

$t_4 = t_1 + c_{24} = 2$, $h_4 = 2$

$T = (0, 1, 5, 2)$; $X = (1, 1, 0, 0)$;

$H = (0, 1, 2, 2)$.

4. 4.1 $m = \infty$; $\underline{v = 0}$

4.2 $u = 1$

$x_1 \neq 0$

$u = 2$

$x_2 \neq 0$

$u = 3$

$x_3 = 0 \;\&\; t_3 < \infty$

$\underline{v = 3}$;

$m = t_3 = 5$

$u = 4$

$x_4 = 0 \;\&\; t_4 < 5$

$\underline{v = 4}$;

$m = t_4 = 2$

$x_u = 0 \;\&\; t_u < m$

4.3 $\boldsymbol{v} \neq 0$

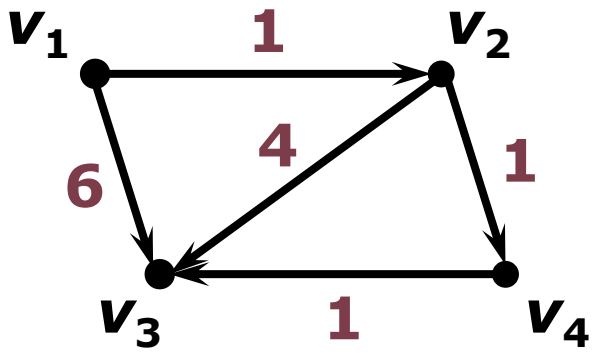
4.4 $\boldsymbol{v} \neq 3$

4.5 $\boldsymbol{x_v} = \boldsymbol{x_4} = 1$;

$\boldsymbol{T} = (0, 1, 5, 2)$; $\boldsymbol{X} = (1, 1, 0, 1)$;

$\boldsymbol{H} = (0, 1, 2, 2)$.

Переход на шаг 3.



3. $\Gamma(v) = \{v_3\}$.

$u = 3$

$x_3 = 0$ & $t_3 > t_4 + c_{43} = 2 + 1 = 3$

$t_3 = t_4 + c_{43} = 3$, $h_3 = 4$

$T = (0, 1, 3, 2)$; $X = (1, 1, 0, 1)$;

$H = (0, 1, 4, 2)$.

4. 4.1 $m = \infty$; $\underline{v = 0}$

4.2 $u = 1$

$x_1 \neq 0$

$u = 2$

$x_2 \neq 0$

$u = 3$

$x_3 = 0 \;\&\; t_3 < \infty$

$\underline{v = 3}$;

$m = t_3 = 3$

$u = 4$

$x_4 \neq 0$

$x_u = 0 \;\&\; t_u < m$

4.3 $v \neq 0$

4.4 $v = 3$

стоп.

Итоги:

$T = (0, 1, 3, 2)$;

$H = (0, 1, 4, 2)$.

Длины кратчайших путей от вершины v_1 до v_1, v_2, v_3 и v_4

Кратчайший путь от v_1 до v_3: v_1, v_2, v_4, v_3

Известно:

применение алгоритма Флойда в среднем примерно вдвое менее трудоемко, чем применение алгоритма Дейкстры для всех пар вершин.

Построение дерева кратчайших путей в бесконтурном орграфе

Как отмечалось выше, алгоритм Дейкстры применим в случае, когда длины дуг неотрицательны.

Далее рассматривается эффективный алгоритм, позволяющий найти дерево кратчайших путей в орграфе (от узла-источника).
Этот алгоритм может применяться даже если длины дуг отрицательны, но при этом известно, что орграф не содержит контуров.

Лемма.

В произвольном бесконтурном орграфе ***G***(***V***, ***E***) узлы можно перенумеровать так, что

$$\forall (v_i, v_j) \in E \quad i < j.$$

Каждая дуга ведет из узла с меньшим номером в узел с большим номером

Обоснование:

отношение достижимости в бесконтурном орграфе есть строгое частичное упорядочение на конечном множестве узлов.

Применяя алгоритм топологической сортировки можно получить требуемую нумерацию узлов.

Предположим:

узлы в бесконтурном орграфе перенумерованы так, что каждая дуга ведет из узла с меньшим номером в узел с большим номером;

узел-источник, из которого нужно построить дерево кратчайших путей, имеет номер 1.

Следующий алгоритм находит кратчайшие пути от узла 1 до всех достижимых из него узлов.

Вход: ***C*** – матрица весов (длин) дуг (квадратная матрица порядка ***n***);

Γ **–1** – списки предшествующих узлов:

$$\Gamma^{-1}(v) \stackrel{\text{def}}{=} \{u \mid v \in \Gamma(u)\}.$$

выход: ***T*** – вектор длин кратчайших путей от источника
(одномерный массив размерности ***n***).

Описание алгоритма:

1. Для $i = 1, 2, \ldots, n$

$$t_i = c_{1i}$$

Начальные значения длин путей

2. Для $i = 2, 3, \ldots, n$

для всех $j \in \Gamma^{-1}(i)$

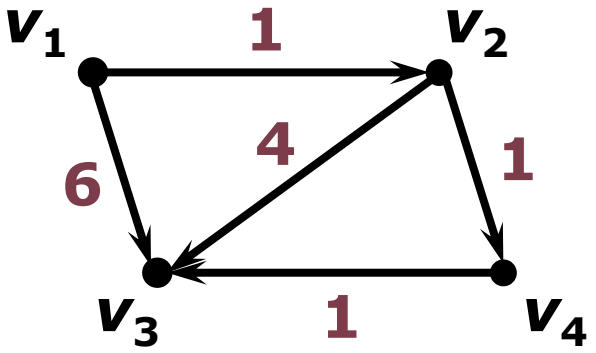
$$t_i = \min \{t_i, t_j + c_{ji}\}$$

Определяется номерами строк $j \neq i$, для которых c_{ji} конечны

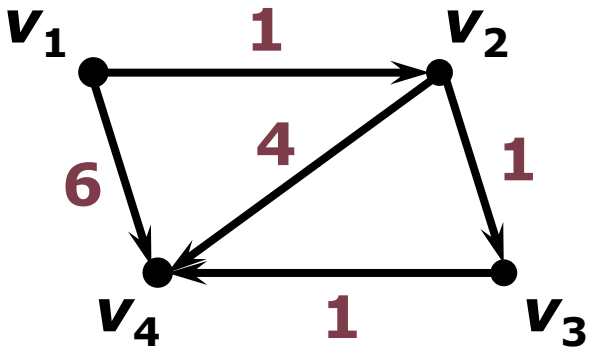
Пересчет оценки длины пути

Пример.

Рассмотрим граф
из предыдущего примера.



Перенумеруем узлы:



$$C = \begin{pmatrix} 0 & 1 & \infty & 6 \\ \infty & 0 & 1 & 4 \\ \infty & \infty & 0 & 1 \\ \infty & \infty & \infty & 0 \end{pmatrix}$$

Найдем кратчайшие пути от узла v_1 до всех достижимых узлов.

1. $\boldsymbol{T} = (0, 1, \infty, 6)$

$$C = \begin{pmatrix} \mathbf{0} & \mathbf{1} & \infty & \mathbf{6} \\ \infty & \mathbf{0} & \mathbf{1} & \mathbf{4} \\ \infty & \infty & \mathbf{0} & \mathbf{1} \\ \infty & \infty & \infty & \mathbf{0} \end{pmatrix}$$

2. $\boldsymbol{i} = 2$

$\boldsymbol{\Gamma^{-1}}(2) = \{\boldsymbol{v_1}\}$

$\boldsymbol{j} = 1$

$\boldsymbol{t_2} = \mathbf{min}\ \{\boldsymbol{t_2}, \boldsymbol{t_1} + \boldsymbol{c_{12}}\} = \mathbf{min}\ \{1, 0+1\} = 1$

$\boldsymbol{T} = (0, 1, \infty, 6)$

$\boldsymbol{i} = 3$

$\boldsymbol{\Gamma^{-1}}(3) = \{\boldsymbol{v_2}\}$

$\boldsymbol{j} = 2$

$\boldsymbol{t_3} = \mathbf{min}\ \{\boldsymbol{t_3}, \boldsymbol{t_2} + \boldsymbol{c_{23}}\} = \mathbf{min}\ \{\infty, 1+1\} = 2$

$\boldsymbol{T} = (0, 1, 2, 6)$

$$C = \begin{pmatrix} 0 & 1 & \infty & 6 \\ \infty & 0 & 1 & 4 \\ \infty & \infty & 0 & 1 \\ \infty & \infty & \infty & 0 \end{pmatrix}$$

$i = 4$

$\Gamma^{-1}(4) = \{v_1, v_2, v_3\}$

$j = 1$

$t_4 = \min\{t_4, t_1 + c_{14}\} = \min\{6, 0+6\} = 6$

$j = 2$

$t_4 = \min\{t_4, t_2 + c_{24}\} = \min\{6, 1+4\} = 5$

$j = 3$

$t_4 = \min\{t_4, t_3 + c_{34}\} = \min\{5, 2+1\} = 3$

$T = (0, 1, 2, 3)$

Вектор кратчайших путей от узла v_1