УНИВЕРСИТЕТ ИТМО

ГРАФЫ: часть 1

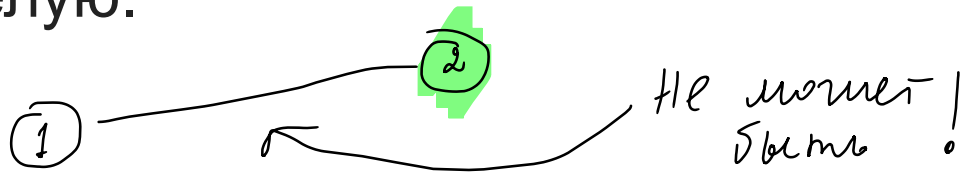
Лекторы
Пермяков Антон Сергеевич
Ткаченко Данил Михайлович

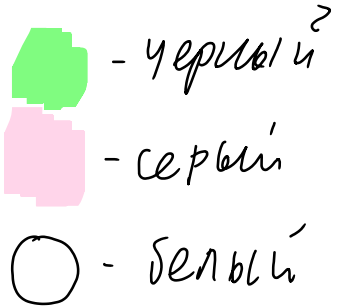
АЛГОРИТМЫ

1. Лемма о белых путях
2. Лемма о кратчайших путях
3. DAG, переборный вариант
4. Наикратчайшие пути
    1. Основные понятия
    2. Виды поиска
    3. Циклы на пути
    4. Релаксация
    5. Массив предков
5. Жадные алгоритмы
6. Алгоритм Дейкстры: суть, реализация, ограничения, оптимизация
7. База алгоритма Уоршала

ЛЕММА о обходе в глубину

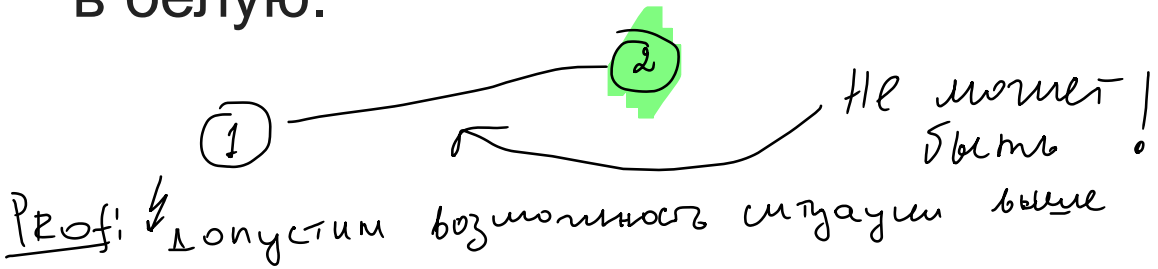
Нет такого момента в процессе выполнения поиска в глубину, когда бы существовало ребро из черной вершины в белую.





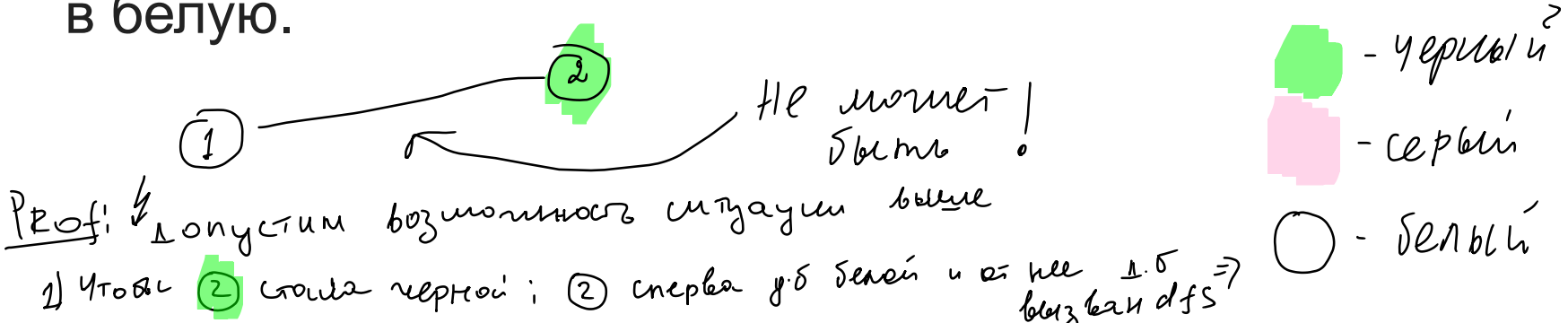
ЛЕММА о обходе в глубину

Нет такого момента в процессе выполнения поиска в глубину, когда бы существовало ребро из черной вершины в белую.



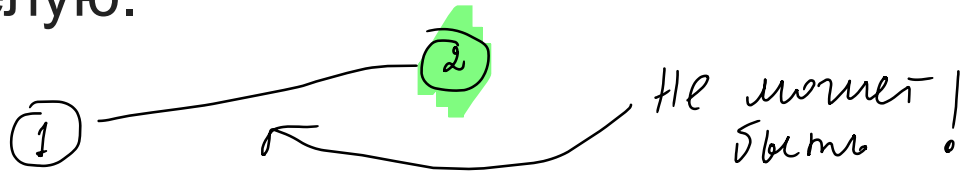
ЛЕММА о обходе в глубину

Нет такого момента в процессе выполнения поиска в глубину, когда бы существовало ребро из черной вершины в белую.



ЛЕММА о обходе в глубину

Нет такого момента в процессе выполнения поиска в глубину, когда бы существовало ребро из черной вершины в белую.



- черный
- серый
- белый

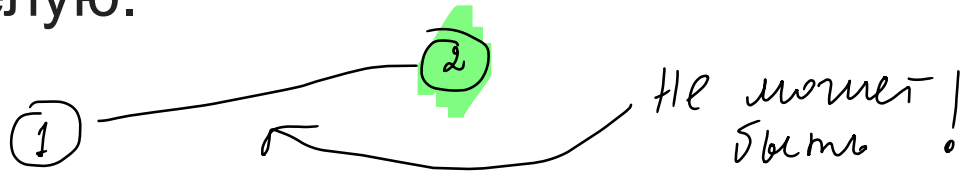
Proof: допустим возможность ситуации выше

1) Чтобы (2) стала черной: (2) сперва ч.б белой и от нее т.б вызван dfs =>

2) dfs(2) => (2) становится серой, а далее смотрим смежные =>

ЛЕММА о обходе в глубину

Нет такого момента в процессе выполнения поиска в глубину, когда бы существовало ребро из черной вершины в белую.



- черный
- серый
- белый

Proof: Допустим возможность ситуации выше

1) Чтобы (2) стала черной; (2) сперва д.б. белой и от нее д.б. вызван dfs =>

2) dfs(2) => (2) становится серой, а далее смотрим смежные =>

3) (1)—(2) значит нужно вызвать dfs(1) => (1)

ЛЕММА о обходе в глубину

Нет такого момента в процессе выполнения поиска в глубину, когда бы существовало ребро из черной вершины в белую.

Не может быть!

- черный
- серый
- белый

Proof: Допустим возможность ситуации выше

1) Чтобы (2) стала черной; (2) сперва д.б белой и от нее д.б вызван dfs =>
2) dfs(2) => (2) становится серой, а далее смотрим смежные =>
3) (1)—(2) значит нужно вызвать dfs(1) => (1) =>
4) (1)—(2) (2) станет черной после (1) согласно работе dfs =>

ЛЕММА О БЕЛЫХ ПУТЯХ

Дан граф **G**. Запустим **dfs(G)**

- остановим выполнение процедуры dfs

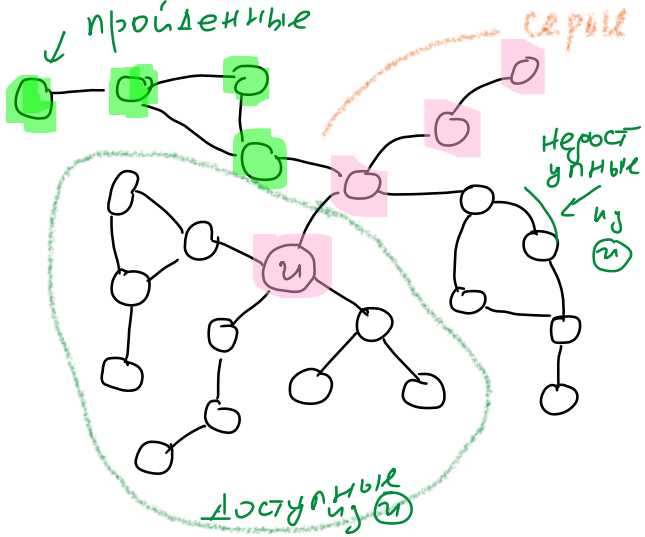
от некоторой серой вершины **u** - первый момент времени **T1**.

- продолжим выполнение dfs(u) пока u не

станет черной - второй момент времени **T2**.

Тогда вершины графа **G\u**, бывшие черными и серыми в первый момент времени, не поменяют свой цвет ко второму моменту времени, а белые вершины либо останутся белыми, либо станут черными, причем черными станут те, что были достижимы от вершины **u** по белым путям.

ЛЕММА О БЕЛЫХ ПУТЯХ



Дан граф **G**. Запустим **dfs(G)**

- остановим выполнение процедуры dfs от некоторой серой вершины **u** - первый момент времени **T1**.
- продолжим выполнение dfs(u) пока u не станет черной - второй момент времени **T2**.

Тогда вершины графа **G\u**, бывшие черными и серыми в первый момент времени, не поменяют свой цвет ко второму моменту времени, а белые вершины либо останутся белыми, либо станут черными, причем черными станут те, что были достижимы от вершины **u** по белым путям.

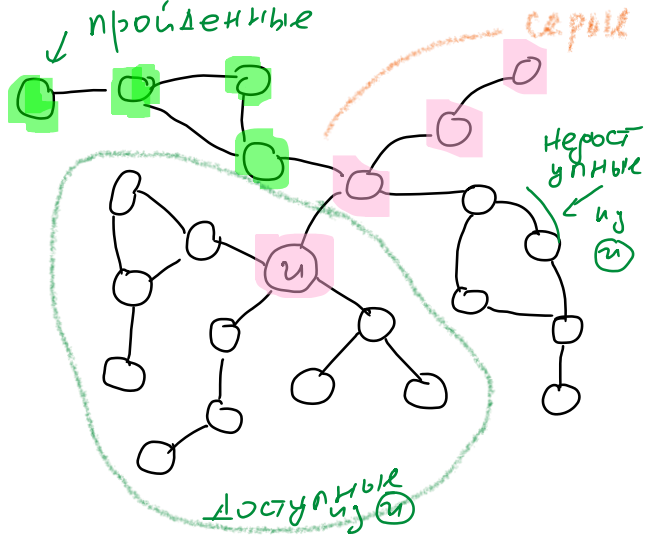
- черный
- серый
- белый

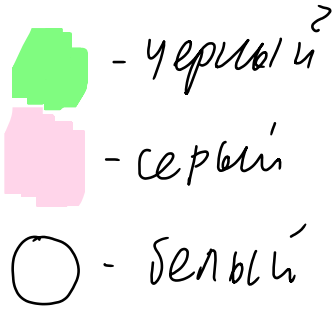
ЛЕММА О БЕЛЫХ ПУТЯХ

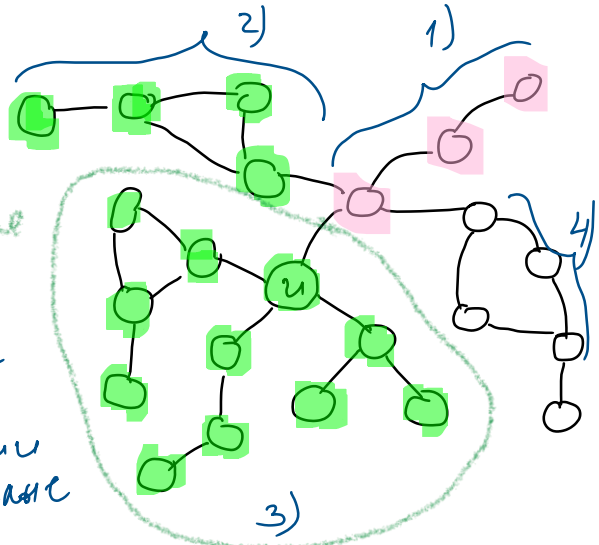
Дан граф **G**. Запустим **dfs(G)**

- остановим выполнение процедуры dfs от некоторой серой вершины **u** - первый момент времени **T1**.
- продолжим выполнение dfs(u) пока u не станет черной - второй момент времени **T2**.

Тогда вершины графа **G\u**, бывшие черными и серыми в первый момент времени, не поменяют свой цвет ко второму моменту времени, а белые вершины либо останутся белыми, либо станут черными, причем черными станут те, что были достижимы от вершины **u** по белым путям.







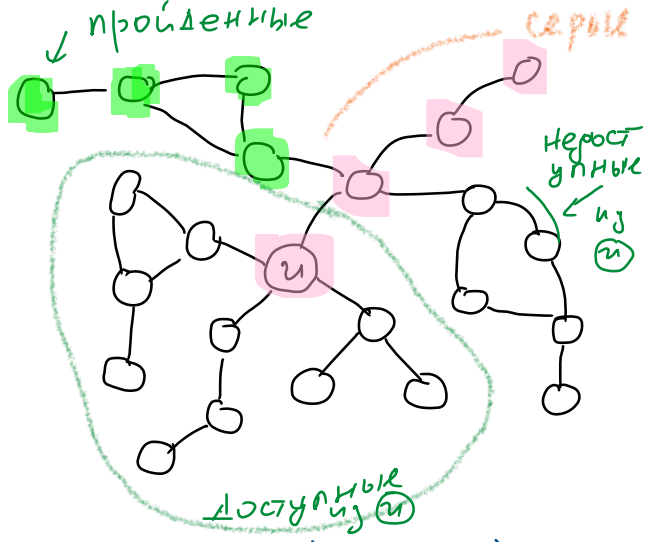
1) серые – серые
2) черные – черные
3) доступные белые – стали черными
4) недоступн. белые – белые

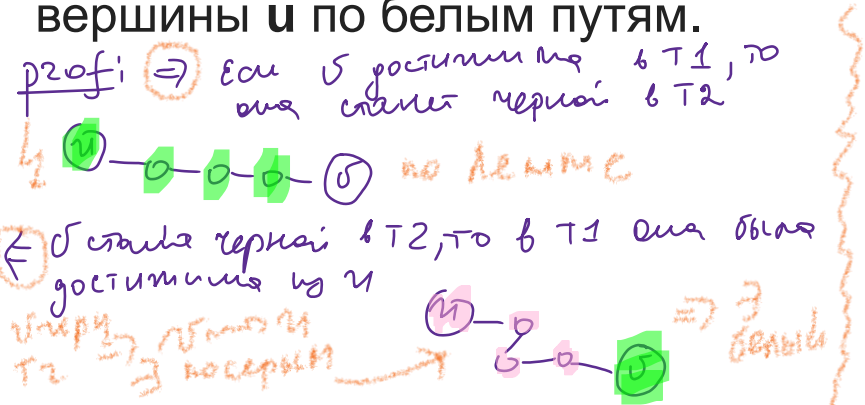
ЛЕММА О БЕЛЫХ ПУТЯХ

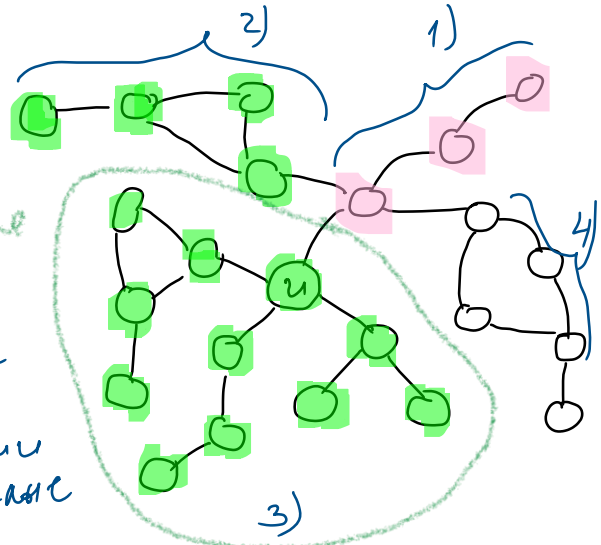
Дан граф **G**. Запустим **dfs(G)**

- остановим выполнение процедуры dfs от некоторой серой вершины **u** - первый момент времени **T1**.
- продолжим выполнение dfs(u) пока u не станет черной - второй момент времени **T2**.

Тогда вершины графа **G\u**, бывшие черными и серыми в первый момент времени, не поменяют свой цвет ко второму моменту времени, а белые вершины либо останутся белыми, либо станут черными, причем черными станут те, что были достижимы от вершины **u** по белым путям.







1) серые – серые
2) черные – черные
3) доступные белые – стали черными
4) недостиж. белые – белые

НАИКРАТЧАЙШИЕ ПУТИ

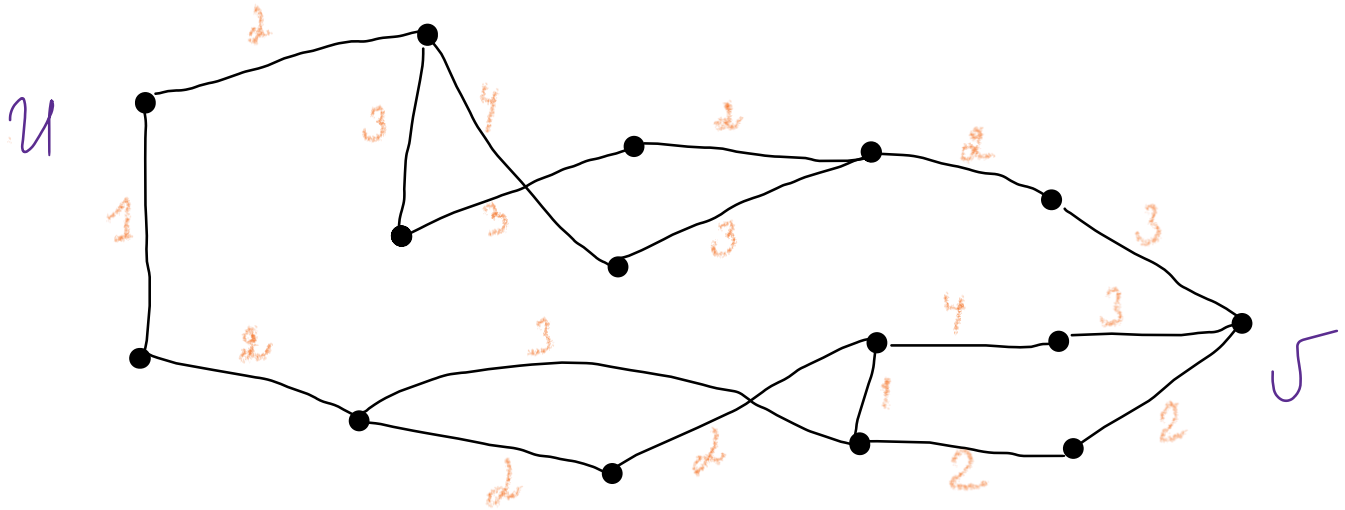
(без ребер): $v_1\ v_2\ v_5\ v_3 \dots v_k$

$=$

$v_1 e_1 v_2 e_4 v_5 e_8 v_3 e_9 \dots v_k$

Путь (маршрут) – последовательность вершин и ребер вида

- длина пути количество ребер в нем (НЕ взвешенный граф) $= k$
- **вес пути** - сумма весом всех ребер пути (взвешенный граф) $= \sum_{i=1}^{k} w[u][v]$



p

$u \rightsquigarrow v$

$\min \rightarrow$

НАИКРАТЧАЙШИЕ ПУТИ

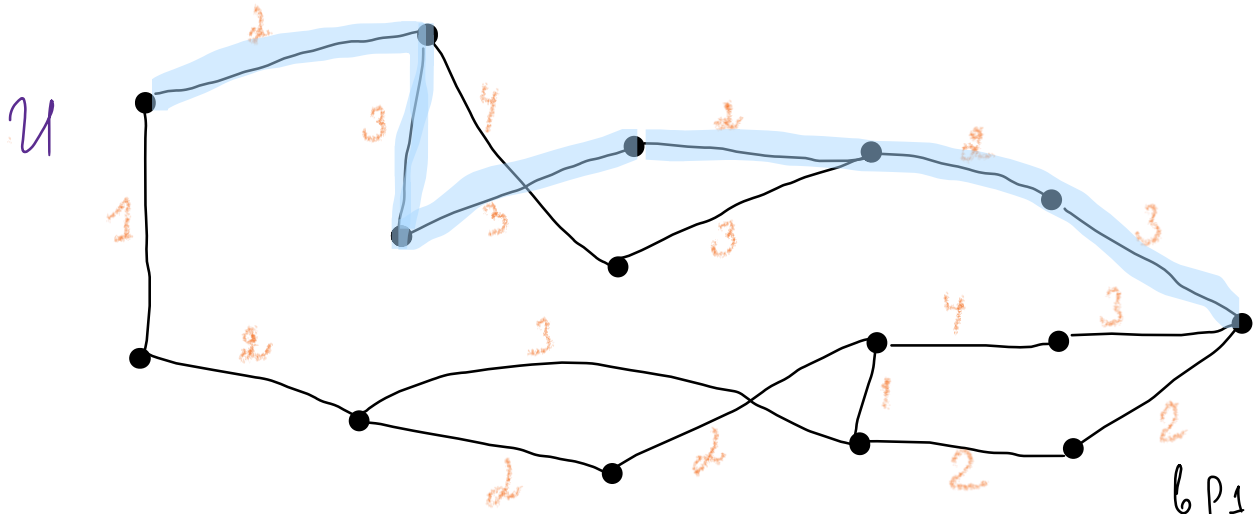
(без ребер): $v_1\ v_2\ v_5\ v_3 \dots v_k$

$=$

$v_1 e_1 v_2 e_4 v_5 e_8 v_3 e_9 \dots v_k$

Путь (маршрут) – последовательность вершин и ребер вида

- длина пути количество ребер в нем (НЕ взвешенный граф) $= k$
- **вес пути** - сумма весом всех ребер пути (взвешенный граф) $= \sum_{i=1}^{k} w[u][v]$



$u \overset{p}{\rightsquigarrow} v$

$\min \rightarrow$

$p_1 = \sum w_{ij} =$

$= 2+3+3+2+2+3 = 15$

$6\ p_1$; $k = 6$ ребер

НАИКРАТЧАЙШИЕ ПУТИ

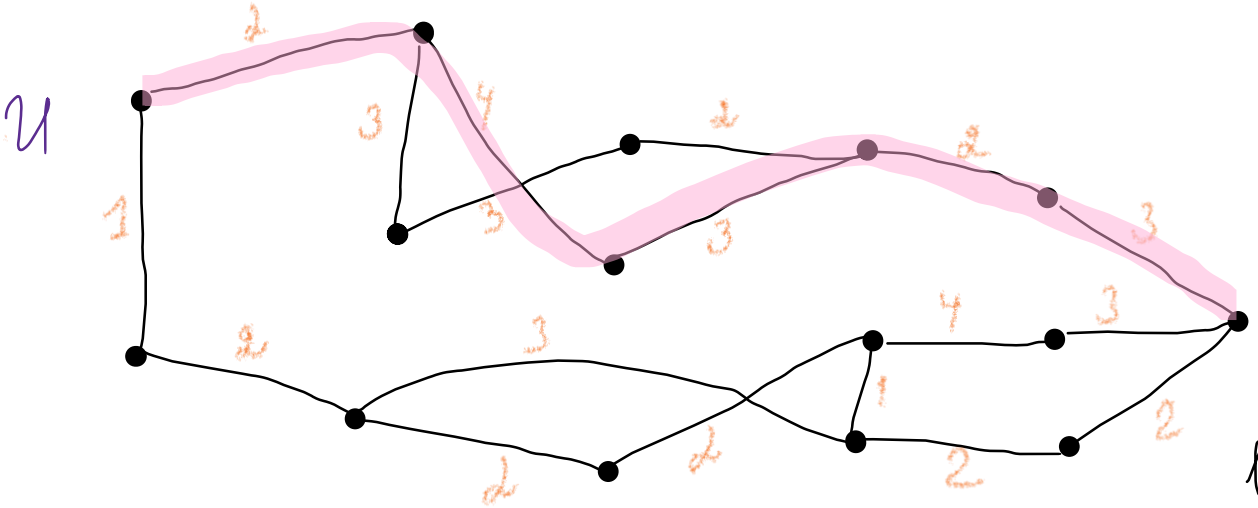
(без ребер): $v_1\ v_2\ v_5\ v_3 \dots v_k$

$=$

$v_1 e_1 v_2 e_4 v_5 e_8 v_3 e_9 \dots v_k$

Путь (маршрут) – последовательность вершин и ребер вида

- длина пути количество ребер в нем (НЕ взвешенный граф) $= k$
- **вес пути** - сумма весом всех ребер пути (взвешенный граф) $= \sum_{i=1}^{k} w[u][v]$



$u \overset{p}{\rightsquigarrow} v$

$\min \rightarrow$

$p_2 = \sum w_{ij} = 2+4+3+2+3 = 14$

в p_2: $k = 5$ ребер

НАИКРАТЧАЙШИЕ ПУТИ

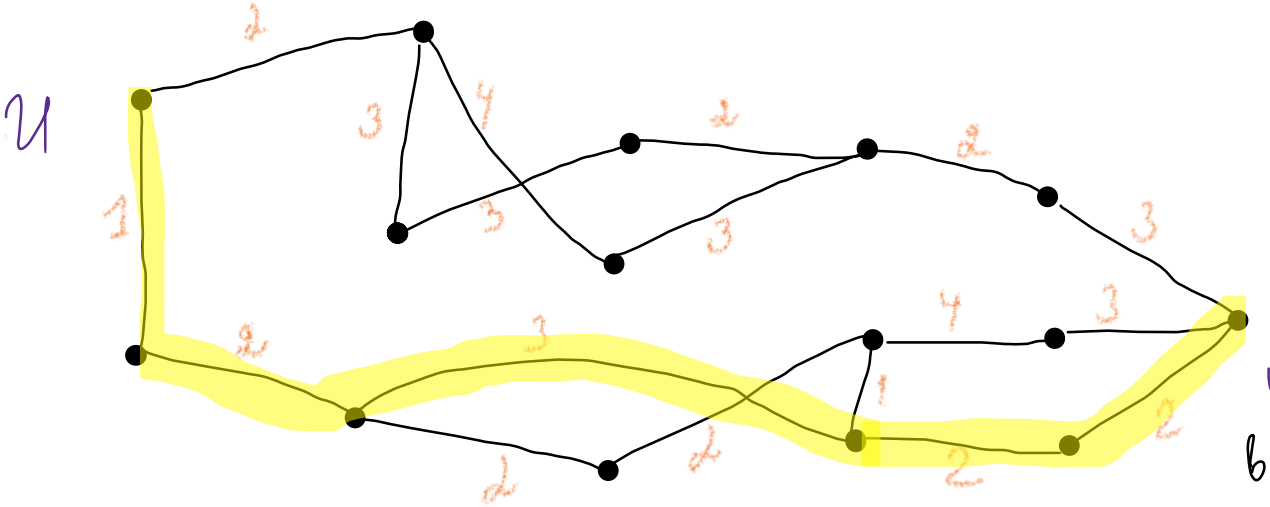
(без ребер): $v_1\ v_2\ v_5\ v_3 \dots v_k$

$=$

$v_1 e_1 v_2 e_4 v_5 e_8 v_3 e_9 \dots v_k$

Путь (маршрут) – последовательность вершин и ребер вида

- длина пути количество ребер в нем (НЕ взвешенный граф) $= k$
- **вес пути** - сумма весом всех ребер пути (взвешенный граф) $= \sum_{i=1}^{k} w[u][v]$



$u \overset{p}{\rightsquigarrow} v$

$\min \rightarrow$

$p_3 = \sum w_j = 1+2+3+2+2 = 10$

v

в p_3: $k = 5$ ребер

НАИКРАТЧАЙШИЕ ПУТИ

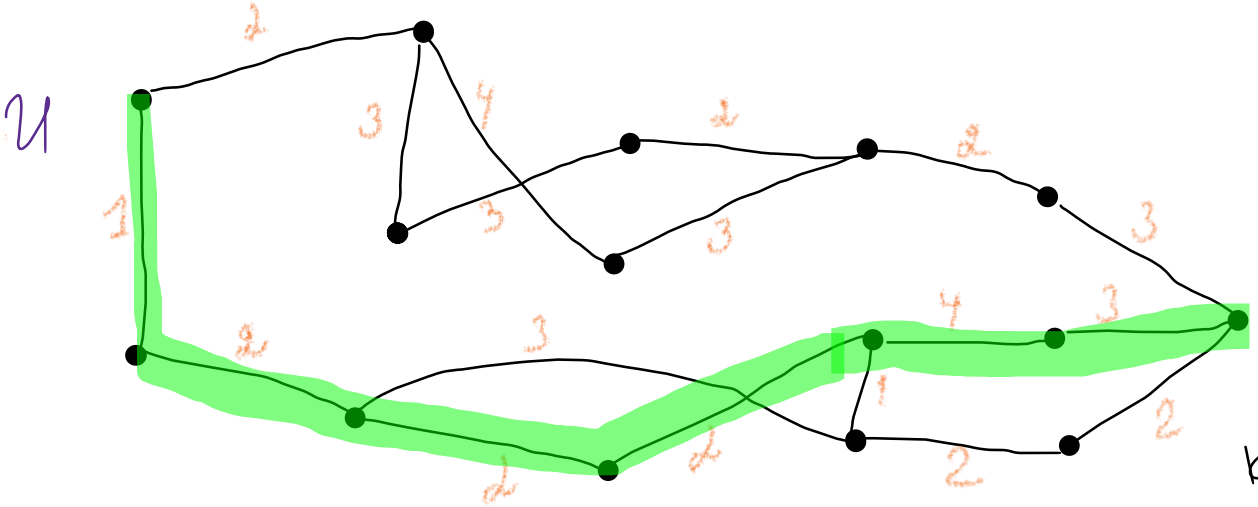
(без ребер): $v_1\ v_2\ v_5\ v_3 \dots v_k$

$=$

$v_1 e_1 v_2 e_4 v_5 e_8 v_3 e_9 \dots v_k$

Путь (маршрут) – последовательность вершин и ребер вида

- длина пути количество ребер в нем (НЕ взвешенный граф) $= k$
- **вес пути** - сумма весом всех ребер пути (взвешенный граф) $= \sum_{i=1}^{k} w[u][v]$



$u \overset{p}{\rightsquigarrow} v$

$\min \rightarrow$

$p_4 = \sum w_{ij} = 1+2+2+2+4+3 =$

$S = 14$

$k = 6$ ребер

НАИКРАТЧАЙШИЕ ПУТИ

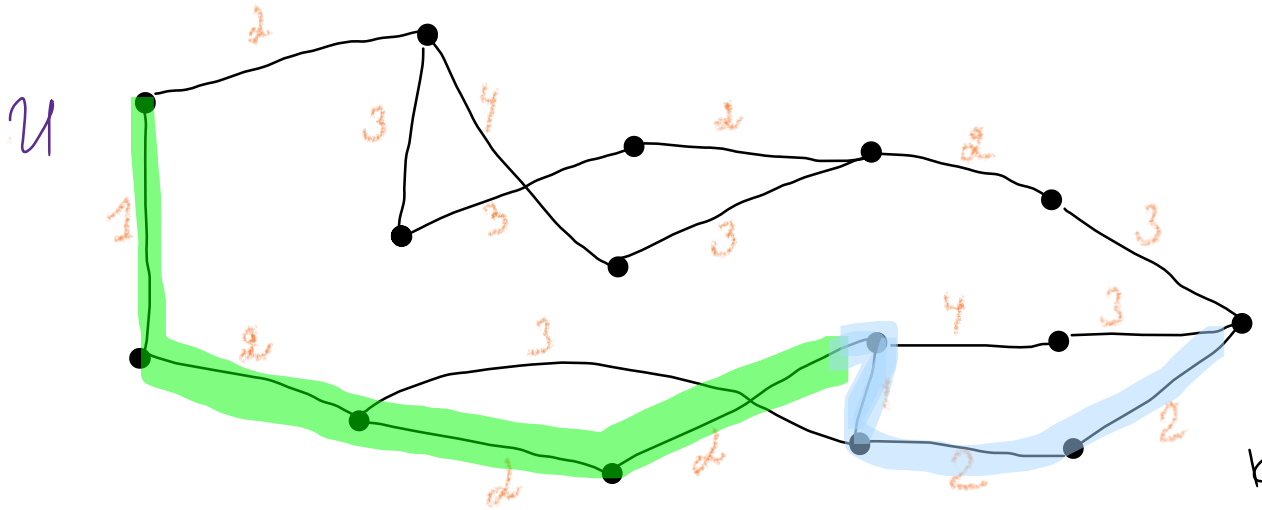
(без ребер): $v_1\ v_2\ v_5\ v_3 \dots v_k$

$=$

$v_1 e_1 v_2 e_4 v_5 e_8 v_3 e_9 \dots v_k$

Путь (маршрут) – последовательность вершин и ребер вида

- длина пути количество ребер в нем (НЕ взвешенный граф) $= k$
- **вес пути** - сумма весом всех ребер пути (взвешенный граф) $= \sum_{i=1}^{k} w[u][v]$



p

$u \rightsquigarrow v$

$\min \rightarrow$

$p_4 = \sum w_{ij} = 1+2+2+2+1+2+2$

$S = 12$

$k = 7$ ребер

НАИКРАТЧАЙШИЕ ПУТИ

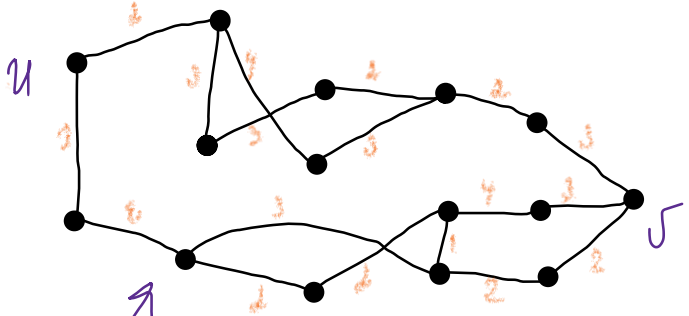
(без ребер): $v_1\ v_2\ v_5\ v_3 \dots v_k$

$=$

$v_1 e_1 v_2 e_4 v_5 e_8 v_3 e_9 \dots v_k$

Путь (маршрут) – последовательность вершин и ребер вида

- длина пути количество ребер в нем (НЕ взвешенный граф) $= k$
- **вес пути** - сумма весом всех ребер пути (взвешенный граф) $= \sum_{i=1}^{k} W[u][v]$



$u \overset{p}{\rightsquigarrow} v$

min →

$p_1 = 15$ $p_2 = 14$ $p_3 = 10$

$p_4 = 14$ $p_5 = 12$

представление G взвешенной матр. смежн.

$W[u][v] =$

	1	2	3	4	…
1	∞	2	3	2	
2	2	∞	∞	2	
3	3	2	∞	3	
4	2	2	3	∞	
…					

НАИКРАТЧАЙШИЕ ПУТИ

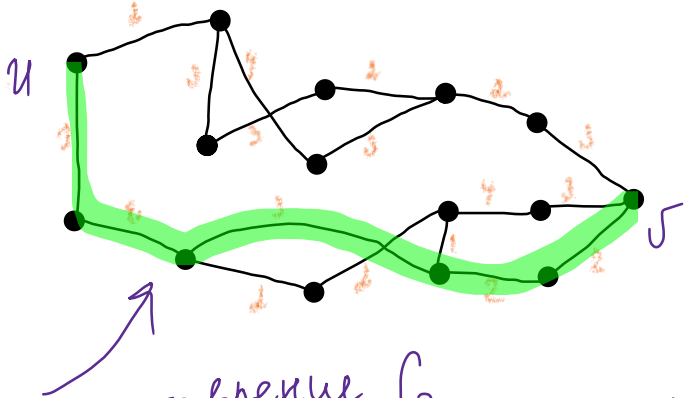
(без ребер): $v_1\ v_2\ v_5\ v_3 \dots v_k$

$=$

$v_1 e_1 v_2 e_4 v_5 e_8 v_3 e_9 \dots v_k$

Путь (маршрут) – последовательность вершин и ребер вида

- длина пути количество ребер в нем (НЕ взвешенный граф) $= k$
- **вес пути** - сумма весом всех ребер пути (взвешенный граф) $= \sum_{i=1}^{k} W[u][v]$



$u \overset{p}{\rightsquigarrow} v$

min →

$p_1 = 15$ $p_2 = 14$ $p_3 = 10$

$p_4 = 14$ $p_5 = 12$

$W(p_3) = 1+2+3+2+2 = 10$

вес пути по $W[u][v]$

представление G взвешенной матр. смежн.

$W[u][v] =$

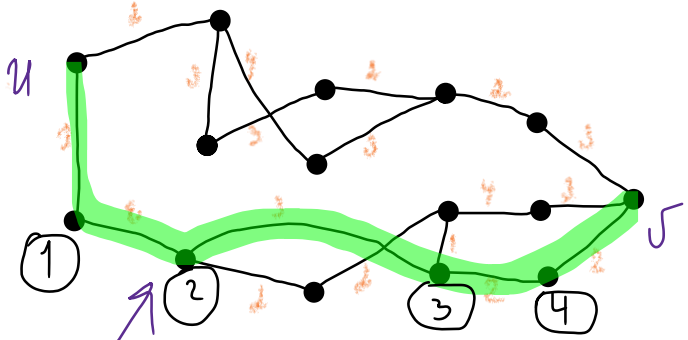
	1	2	3	4	…
1	∞	2	3	2	
2	2	∞	∞	2	
3	3	2	∞	3	
4	2	2	3	∞	
…					

НАИКРАТЧАЙШИЕ ПУТИ

(Seq ребер): $v_1\ v_2\ v_5\ v_3 \dots v_k$
$=$
$v_1 e_1 v_2 e_4 v_5 e_8 v_3 e_9 \dots v_k$

Путь (маршрут) – последовательность вершин и ребер вида

- длина пути количество ребер в нем (НЕ взвешенный граф) $= k$
- **вес пути** - сумма весом всех ребер пути (взвешенный граф) $= \sum_{i=1}^{k} W[u][v]$



представление G взвешенной матр. смежн.

$W[u][v] =$

	1	2	3	4	…
1	∞	2	3	2	
2	2	∞	∞	2	
3	3	2	∞	3	
4	2	2	3	∞	
…					

$u \overset{p}{\rightsquigarrow} v$

min →

$p_1 = 15$ $p_2 = 14$ $p_3 = 10$
$p_4 = 14$ $p_5 = 12$

$w(p_3) = 1+2+3+2+2 = 10$

$= W[u][1] + W[1][2] + W[2][3] + W[3][v]$

$+ W[4][v] = w(p_3 : u \rightsquigarrow v)$

НАИКРАТЧАЙШИЕ ПУТИ

(без ребер): $v_1\ v_2\ v_5\ v_3 \dots v_k$ = $v_1 e_1 v_2 e_4 v_5 e_8 v_3 e_9 \dots v_k$

Путь (маршрут) – последовательность вершин и ребер вида

- длина пути количество ребер в нем (НЕ взвешенный граф) $= k$
- **вес пути** - сумма весом всех ребер пути (взвешенный граф) $= \sum_{i=1}^{k} W[u][v]$

W – взвешанная матрица смежн-ти

Путь: **p = <V0, V1, V2, … Vk>** $|P| = k$

Вес пути **w(p)** $= \sum_{i=1}^{e_i \in P} w(e_i) = \sum_{i=1}^{k} W[v_i][v_{i+1}]$, v_i и $v_{i+1} \in P$

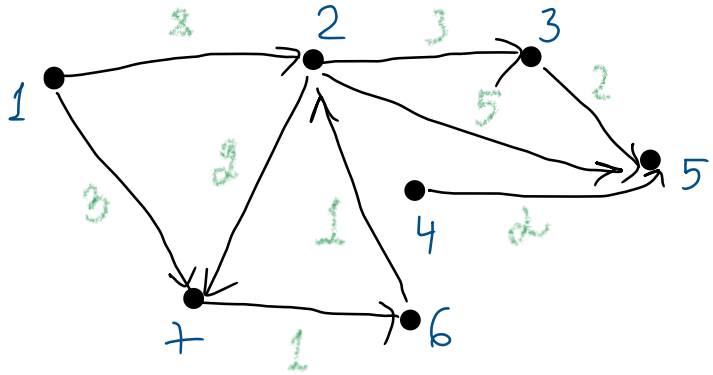
НаиКратчайший путь - путь с наименьшим весом (их может быть несколько разных, но с одним весом)

Примечание: вес наикратчайшего пути **из u в v** будет наименьшим из возможных или равен бесконечности, если пути из **u в v** нет

НАИКРАТЧАЙШИЕ ПУТИ

НаиКратчайший путь - путь с наименьшим весом (их может быть несколько разных, но с одним весом)

Примечание: вес наикратчайшего пути **из u в v** будет наименьшим из возможных или равен бесконечности, если пути из **u в v** нет



W - взвешанная матрица смежн-ти

|P| = k

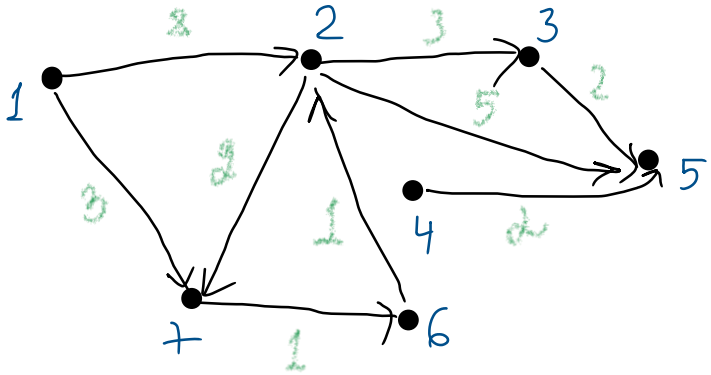
Найдем от (1) до всех:

d	1	2	3	4	5	6	7
	0						

НАИКРАТЧАЙШИЕ ПУТИ

НаиКратчайший путь - путь с наименьшим весом (их может быть несколько разных, но с одним весом)

Примечание: вес наикратчайшего пути **из u в v** будет наименьшим из возможных или равен бесконечности, если пути из **u в v** нет



W+ взвешанная матрица смежн-ти

|P|=K

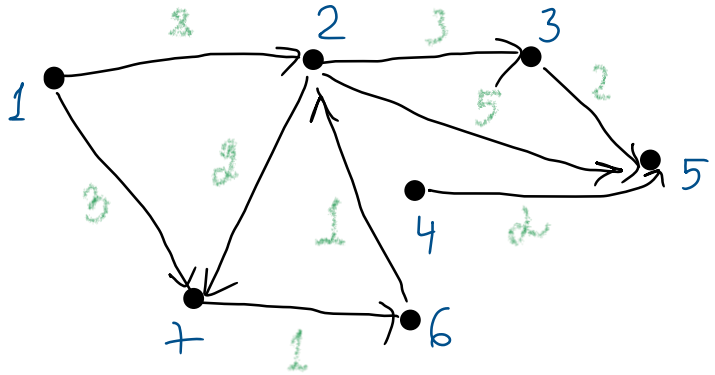
Найдем от (1) до всех:

d	1	2	3	4	5	6	7
	0	5	8	∞	10	4	3

НАИКРАТЧАЙШИЕ ПУТИ

НаиКратчайший путь - путь с наименьшим весом (их может быть несколько разных, но с одним весом)

Примечание: вес наикратчайшего пути **из u в v** будет наименьшим из возможных или равен бесконечности, если пути из **u в v** нет



W - взвешанная матрица смежн-ти

|P| = k

Найдем от (1) до всех:

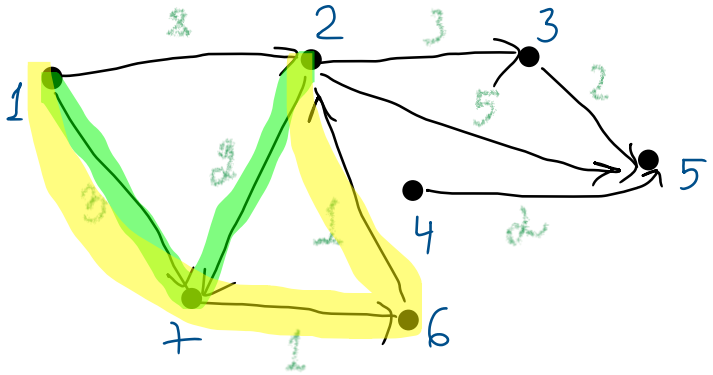
d	1	2	3	4	5	6	7
	0	5	8	∞	10	4	3

рассмотрим p_i : 1 → 2

НАИКРАТЧАЙШИЕ ПУТИ

НаиКратчайший путь - путь с наименьшим весом (их может быть несколько разных, но с одним весом)

Примечание: вес наикратчайшего пути **из u в v** будет наименьшим из возможных или равен бесконечности, если пути из **u в v** нет



|W| взвешанная матрица смежн-ти

|P| = k

Найдем от (1) до всех:

d	1	2	3	4	5	6	7
	0	5	8	∞	10	4	3

рассмотрим p_i : 1 → 2

$w(p_1) = 5$; 1 7 2

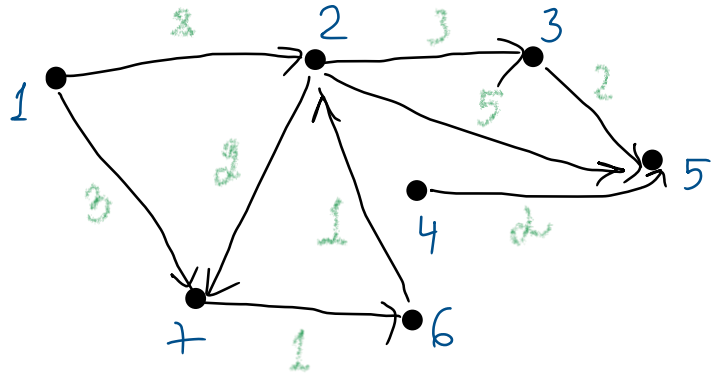
$w(p_2) = 5$; 1 7 6 2

> два наикрат

НАИКРАТЧАЙШИЕ ПУТИ

НаиКратчайший путь - путь с наименьшим весом (их может быть несколько разных, но с одним весом)

Примечание: вес наикратчайшего пути **из u в v** будет наименьшим из возможных или равен бесконечности, если пути из **u в v** нет



W - взвешанная матрица смежн-ти

|P| = k

Найдем от (1) до всех:

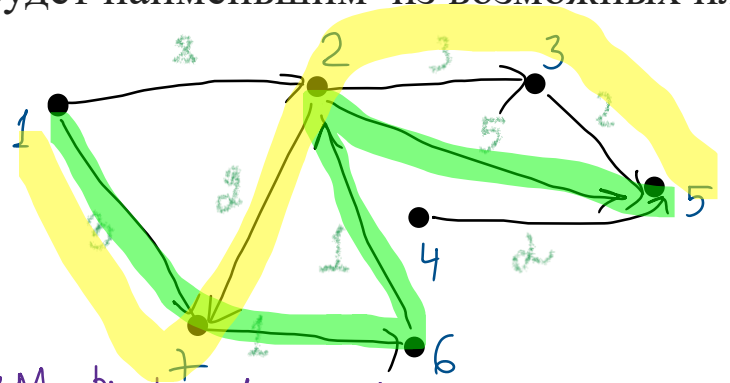
d	1	2	3	4	5	6	7
	0	5	8	∞	10	4	3

рассмотрим p_i : 1 → 5

НАИКРАТЧАЙШИЕ ПУТИ

НаиКратчайший путь - путь с наименьшим весом (их может быть несколько разных, но с одним весом)

Примечание: вес наикратчайшего пути **из u в v** будет наименьшим из возможных или равен бесконечности, если пути из **u в v** нет



$|W|$ взвешанная матрица смежн-ти

$|P| = k$

Найдем от (1) до всех:

d	1	2	3	4	5	6	7
	0	5	8	∞	10	4	3

рассмотрим p_i : 1 → 2

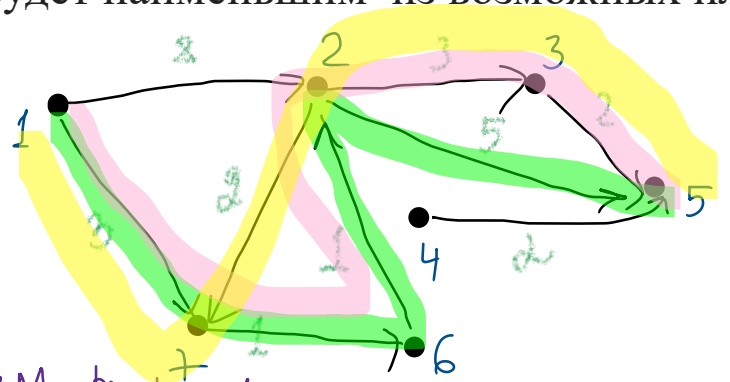
$w(p_1) = 10$	1 7 6 2 3 5	k = 5
$w(p_2) = 10$	1 7 6 2 5	k = 4
$w(p_3) = 10$	1 7 2 3 5	k = 4
$w(p_4) = 10$;	1 7 2 5	k = 3

все наикр

НАИКРАТЧАЙШИЕ ПУТИ

НаиКратчайший путь - путь с наименьшим весом (их может быть несколько разных, но с одним весом)

Примечание: вес наикратчайшего пути **из u в v** будет наименьшим из возможных или равен бесконечности, если пути из **u в v** нет



W – взвешанная матрица смежн-ти

$|P| = k$

Найдем от (1) до всех:

d	1	2	3	4	5	6	7
	0	5	8	∞	10	4	3

рассмотрим p_i : 1 → 2

$w(p_1) = 10$: 1 7 6 2 3 5 — $k = 5$

$w(p_2) = 10$: 1 7 6 2 5 — $k = 4$

$w(p_3) = 10$: 1 7 2 3 5 — $k = 4$

$w(p_4) = 10$: 1 7 2 5 — $k = 3$

все наикр

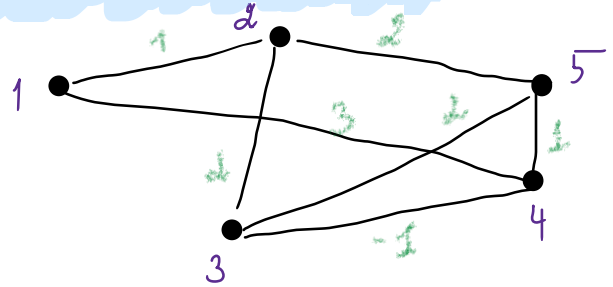
Варианты задачи поиска наикратчайшего пути

Варианты задачи поиска наикратчайшего пути

1. между заданными: **из u в v**
2. от заданной до **u** всех остальных
3. между всеми вершинами (от **всех** до **всех**)

Варианты задачи поиска наикратчайшего пути

1. между заданными: **из u в v**

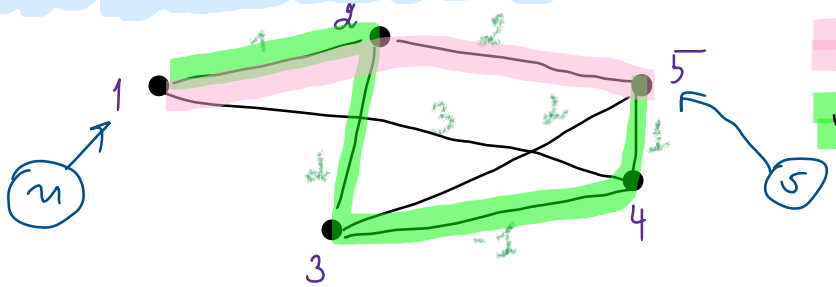


$1 \xrightarrow{p} 5 : \min w(p)$

2. от заданной до **u** всех остальных
3. между всеми вершинами (от **всех** до **всех**)

Варианты задачи поиска наикратчайшего пути

1. между заданными: **из u в v**



$1 \xrightarrow{p} 5 : \min w(p)$

$w(p_1) = 1 + 2 = 3$: 1 – 2 – 5, 2 ребра

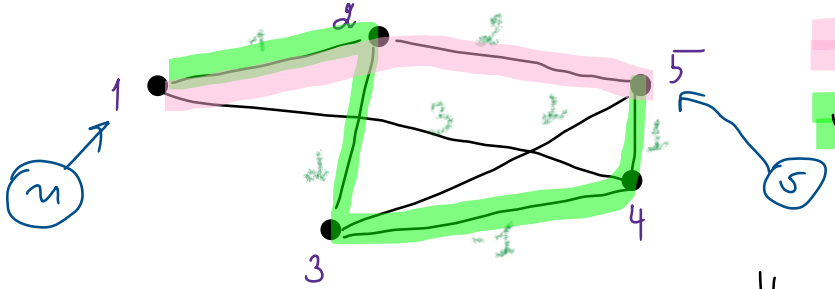
$w(p_2) = 1 + 2 + (-1) + 1 = 3$; 1 – 2 – 3 – 4 – 5, 4 ребра

u ⇝ v – наикратчайший путь ; ∞ – если пути нет (∄) !

2. от заданной до **u** всех остальных
3. между всеми вершинами (от **всех** до **всех**)

Варианты задачи поиска наикратчайшего пути

1. между заданными: **из u в v**

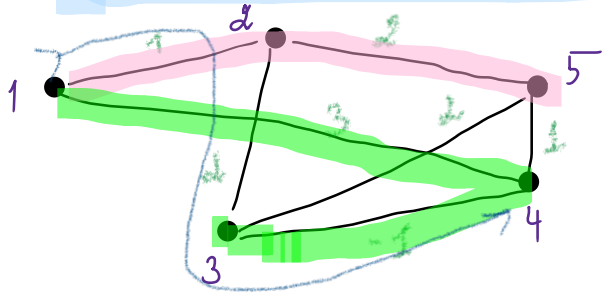


$1 \overset{p}{\rightsquigarrow} 5 : \min w(p)$

$w(p_1) = 1 + 2 = 3$: 1 − 2 − 5, 2 ребра

$w(p_2) = 1 + 2 + (-1) + 1 = 3$: 1 − 2 − 3 − 4 − 5, 4 ребра

2. от заданной до **u** всех остальных



Найдём от ① до всех

$1 \rightsquigarrow 2$: $w(p) = 1$: 1 − 2

$1 \rightsquigarrow 3$: $w(p) = 2$: 1 − 4 − 3

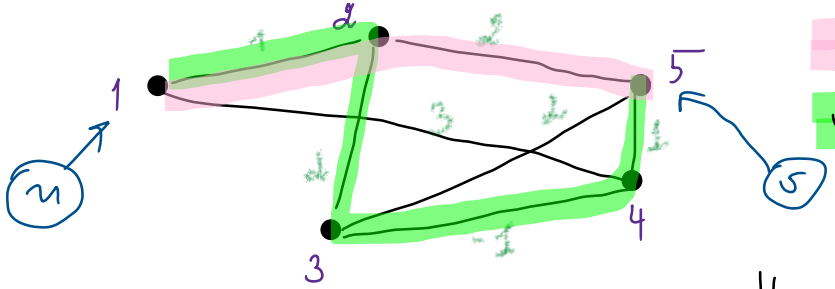
$1 \rightsquigarrow 4$: $w(p) = 2$: 1 − 2 − 3 − 4

$1 \rightsquigarrow 5$: $w(p) = 3$: 1 − 2 − 5

3. между всеми вершинами (от **всех** до **всех**)

Варианты задачи поиска наикратчайшего пути

1. между заданными: **из u в v**

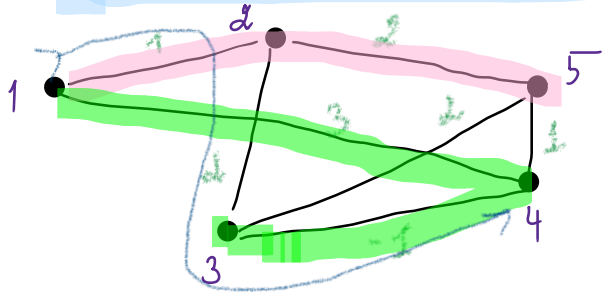


$1 \overset{p}{\rightsquigarrow} 5 : \min w(p)$

$w(p_1) = 1 + 2 = 3$: $1 - 2 - 5$, 2 ребра

$w(p_2) = 1 + 2 + 1 + 1 = 3$: $1 - 2 - 3 - 4 - 5$, 4 ребра

2. от заданной до **u** всех остальных



Найдём от (1) до всех

$1 \rightsquigarrow 2$: $w(p) = 1$: $1 - 2$

$1 \rightsquigarrow 3$: $w(p) = 2$: $1 - 4 - 3$

$1 \rightsquigarrow 4$: $w(p) = 2$: $1 - 2 - 3 - 4$

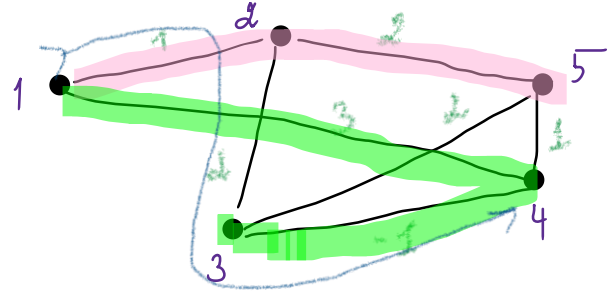
$1 \rightsquigarrow 5$: $w(p) = 3$: $1 - 2 - 5$

$|V| - 1$ раз выполнить задачу (1) ?

3. между всеми вершинами (от **всех** до **всех**)

Варианты задачи поиска наикратчайшего пути

1. между заданными: **из u в v**
2. от заданной до **u** всех остальных



Найдём от ① до всех

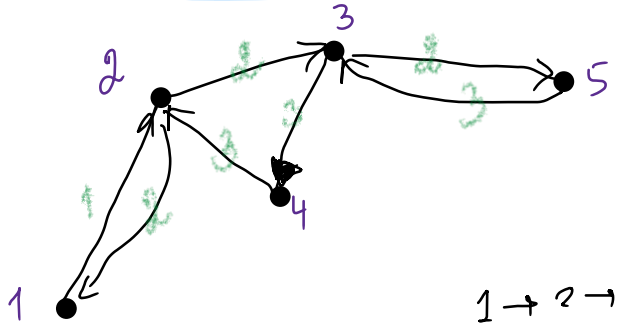
$1 \rightsquigarrow 2$: $w(p) = 1$; $1 - 2$

$1 \rightsquigarrow 3$; $w(p) = 2$; $1 - 4 - 3$

$1 \rightsquigarrow 4$; $w(p) = 2$; $1 - 2 - 3 - 4$

$1 \rightsquigarrow 5$; $w(p) = 3$; $1 - 2 - 5$

3. между всеми вершинами (от **всех** до **всех**)



$1 \rightarrow 2 \rightarrow 3$

$1 \rightsquigarrow 2$

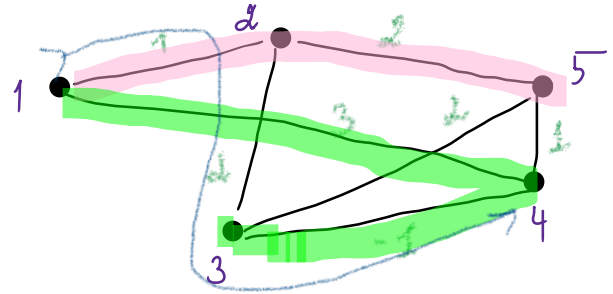
$1 \rightsquigarrow 3$

$1 \rightsquigarrow 4$

$1 \rightsquigarrow 5$

Варианты задачи поиска наикратчайшего пути

1. между заданными: **из u в v**
2. от заданной до **u** всех остальных



Найдём от ① до всех

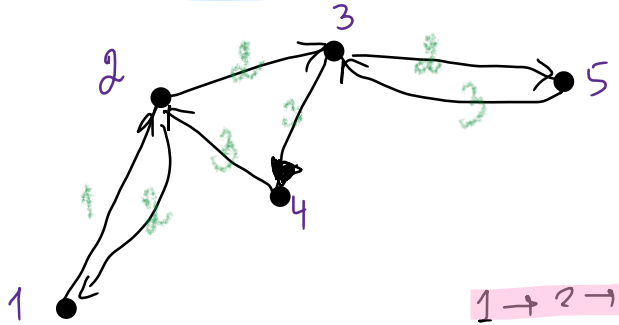
1 ⇝ 2 : w(p) = 1 ; 1 – 2

1 ⇝ 3 ; w(p) = 2 ; 1 – 4 – 3

1 ⇝ 4 ; w(p) = 2 ; 1 – 2 – 3 – 4

1 ⇝ 5 ; w(p) = 3 ; 1 – 2 – 5

3. между всеми вершинами (от **всех** до **всех**)



1 ⇝ 2
1 ⇝ 3
1 ⇝ 4
1 ⇝ 5

2 ⇝ 1
2 ⇝ 3
2 ⇝ 4
2 ⇝ 5

3 ⇝ 1
3 ⇝ 2
3 ⇝ 4
3 ⇝ 5

4 ⇝ 1
4 ⇝ 2
4 ⇝ 3
4 ⇝ 5

5 ⇝ 1
5 ⇝ 2
5 ⇝ 3
5 ⇝ 4

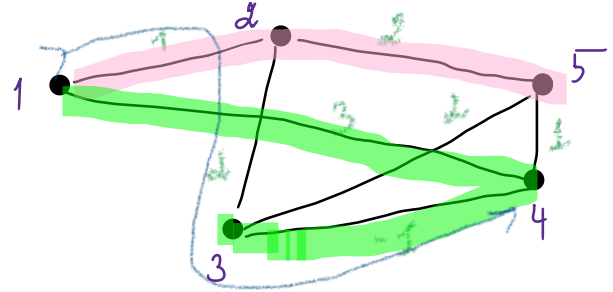
1 → 2 → 3
3 → 4 → 2 → 1

1 → 2 → 3 → 5
5 → 3 → 4 → 2 → 1

] разные

Варианты задачи поиска наикратчайшего пути

1. между заданными: **из u в v**
2. от заданной до **u** всех остальных



Найдём от ① до всех

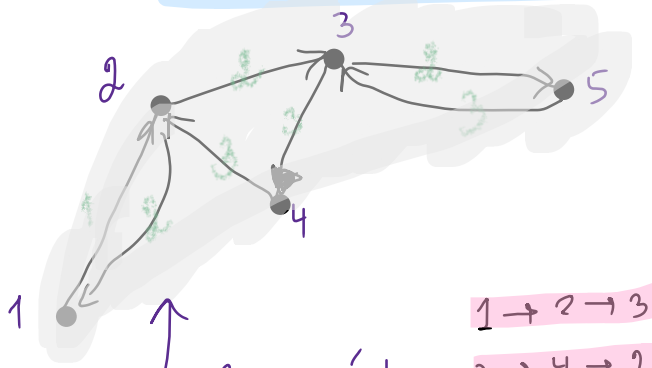
$1 \rightsquigarrow 2$: $w(p) = 1$; $1 - 2$

$1 \rightsquigarrow 3$; $w(p) = 2$; $1 - 4 - 3$

$1 \rightsquigarrow 4$; $w(p) = 2$; $1 - 2 - 3 - 4$

$1 \rightsquigarrow 5$; $w(p) = 3$; $1 - 2 - 5$

3. между всеми вершинами (от **всех** до **всех**)



сильно связный!

$1 \rightsquigarrow 2$

$1 \rightsquigarrow 3$

$1 \rightsquigarrow 4$

$1 \rightsquigarrow 5$

$2 \rightsquigarrow 1$

$2 \rightsquigarrow 3$

$2 \rightsquigarrow 4$

$2 \rightsquigarrow 5$

$3 \rightsquigarrow 1$

$3 \rightsquigarrow 2$

$3 \rightsquigarrow 4$

$3 \rightsquigarrow 5$

$4 \rightsquigarrow 1$

$4 \rightsquigarrow 2$

$4 \rightsquigarrow 3$

$4 \rightsquigarrow 5$

$5 \rightsquigarrow 1$

$5 \rightsquigarrow 2$

$5 \rightsquigarrow 3$

$5 \rightsquigarrow 4$

$1 \to 2 \to 3$

$3 \to 4 \to 2 \to 1$

$1 \to 2 \to 3 \to 5$

$5 \to 3 \to 4 \to 2 \to 1$

] разные

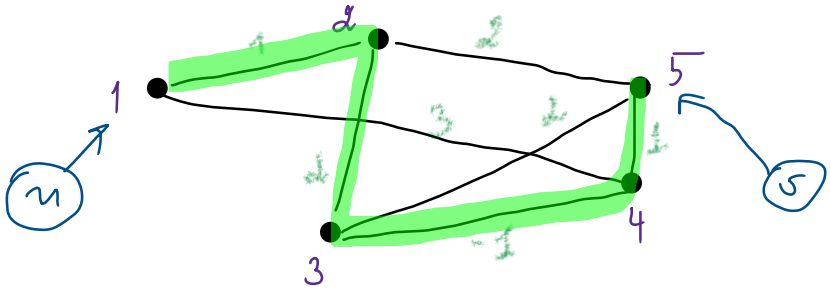
Что может содержать НАИКРАТЧАЙШИЙ путь?

Что может содержать НАИКРАТЧАЙШИЙ путь?

1. Ребра отрицательного веса
2. Циклы отрицательного веса
3. Циклы нулевого веса
4. Циклы положительного веса

Что может содержать НАИКРАТЧАЙШИЙ путь?

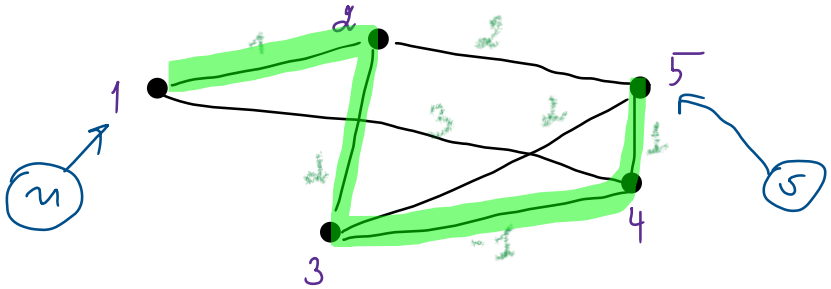
1. Ребра отрицательного веса



2. Циклы отрицательного веса
3. Циклы нулевого веса
4. Циклы положительного веса

Что может содержать НАИКРАТЧАЙШИЙ путь?

1. Ребра отрицательного веса

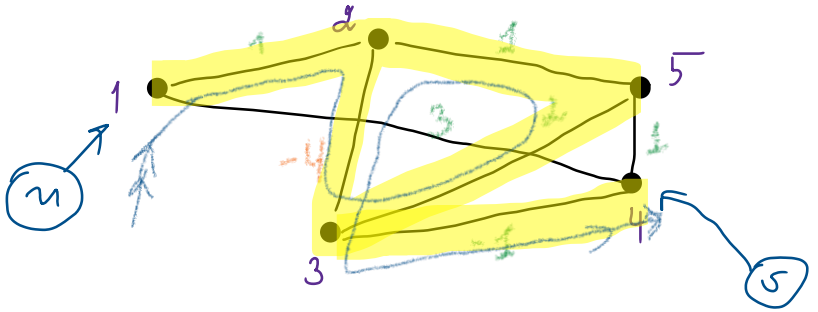


может содержать!

2. Циклы отрицательного веса
3. Циклы нулевого веса
4. Циклы положительного веса

Что может содержать НАИКРАТЧАЙШИЙ путь?

1. Ребра отрицательного веса
2. Циклы отрицательного веса

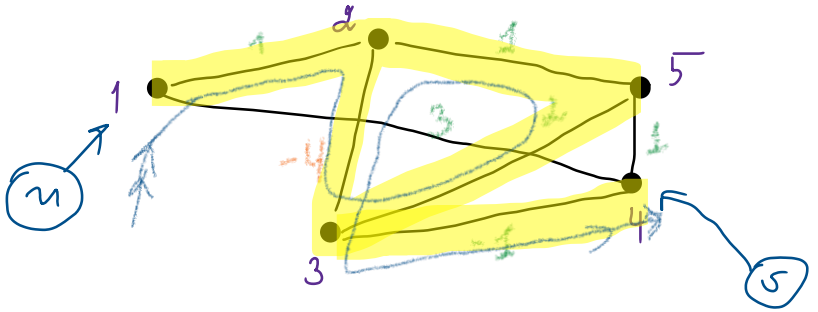


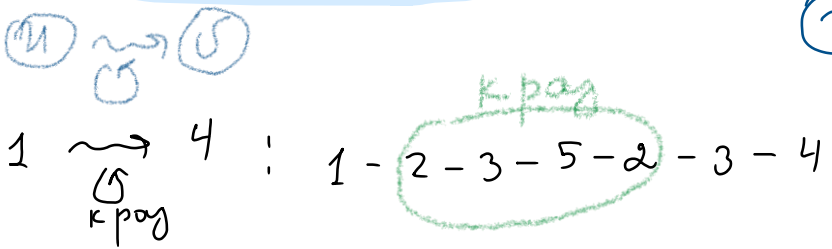
цикл: 2-5-3-2; $\omega(p) = -1$

3. Циклы нулевого веса
4. Циклы положительного веса

Что может содержать НАИКРАТЧАЙШИЙ путь?

1. Ребра отрицательного веса
2. Циклы отрицательного веса

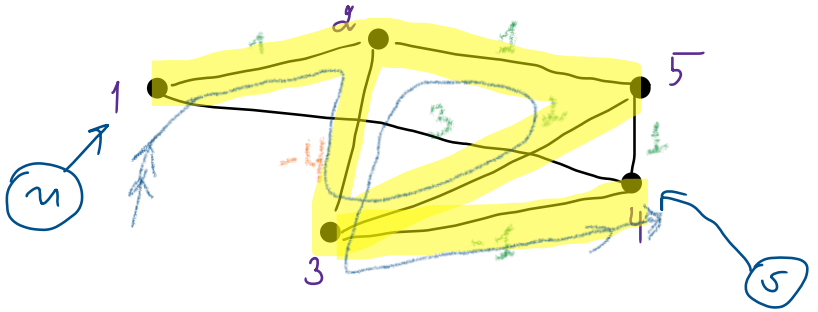


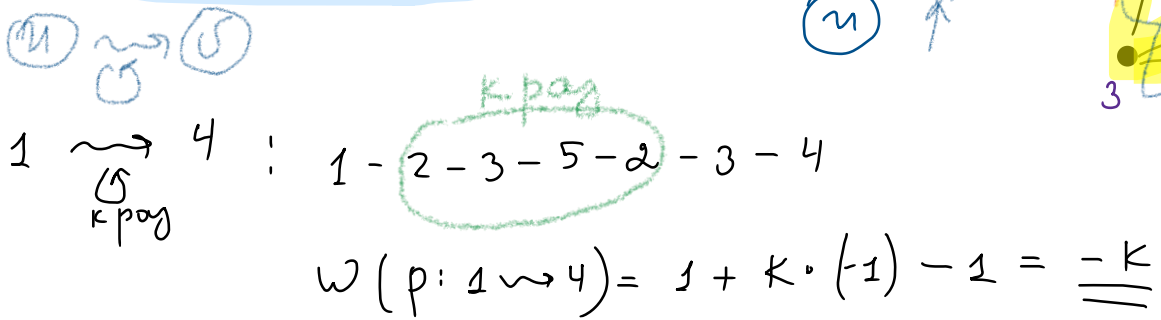


3. Циклы нулевого веса
4. Циклы положительного веса

Что может содержать НАИКРАТЧАЙШИЙ путь?

1. Ребра отрицательного веса
2. Циклы отрицательного веса

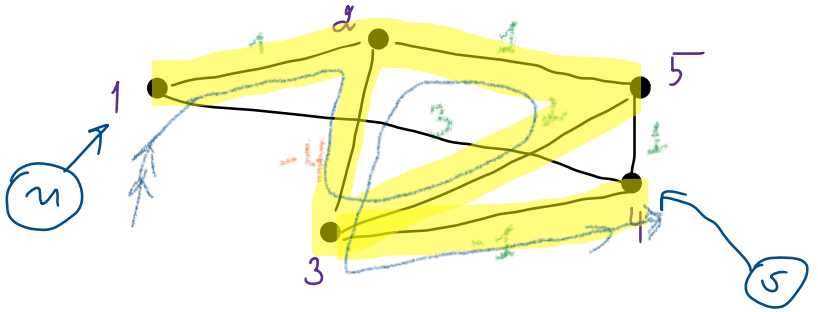




3. Циклы нулевого веса
4. Циклы положительного веса

Что может содержать НАИКРАТЧАЙШИЙ путь?

1. Ребра отрицательного веса
2. Циклы отрицательного веса



$1 \rightsquigarrow 4$: $1 - 2 - 3 - 5 - 2 - 3 - 4$ (k раз)

$w(p: 1 \rightsquigarrow 4) = 1 + k \cdot (-1) - 1 = -k$

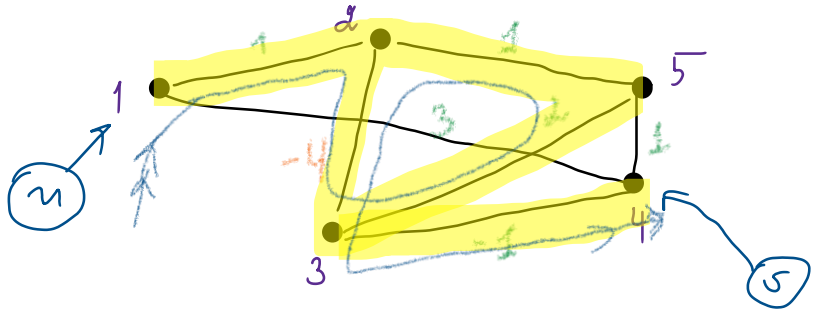
$k \to \infty$

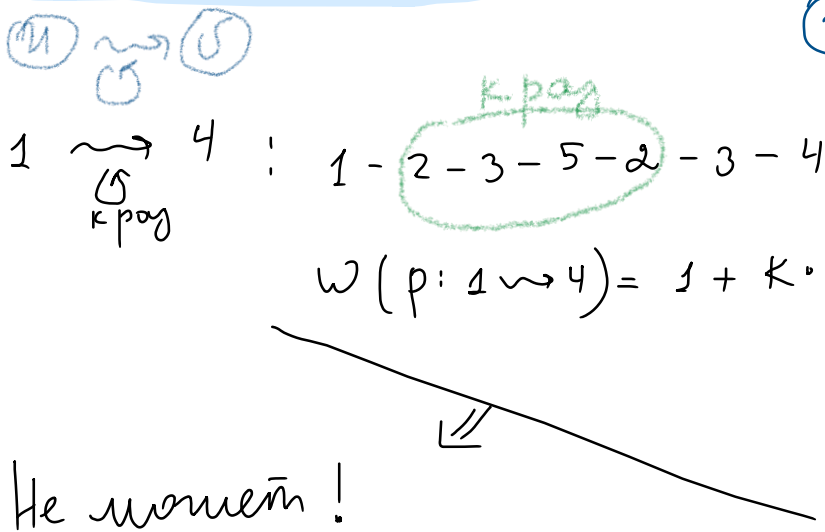
$w(p: 1 \rightsquigarrow 4) \to -\infty$

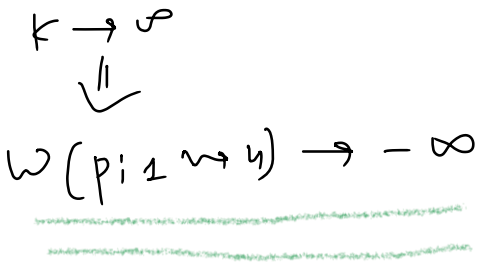
3. Циклы нулевого веса
4. Циклы положительного веса

Что может содержать НАИКРАТЧАЙШИЙ путь?

1. Ребра отрицательного веса
2. Циклы отрицательного веса



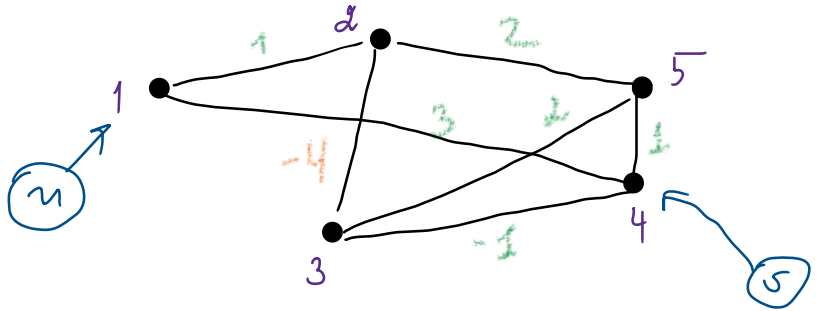




3. Циклы нулевого веса
4. Циклы положительного веса

Что может содержать НАИКРАТЧАЙШИЙ путь?

1. Ребра отрицательного веса
2. Циклы отрицательного веса
3. Циклы нулевого веса

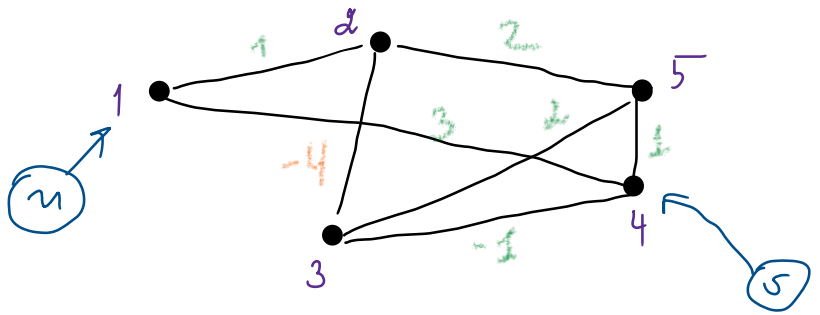


Ц: 2 - 5 - 3 - 2 : w(p) = 0

4. Циклы положительного веса

Что может содержать НАИКРАТЧАЙШИЙ путь?

1. Ребра отрицательного веса
2. Циклы отрицательного веса
3. Циклы нулевого веса



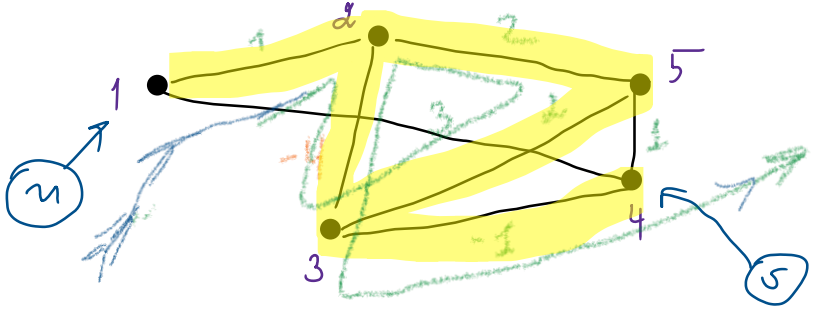
Q: 2 - 5 - 3 - 2 : w(P) = 0

1 ⇝ 4 : 1 - 2 - 3 - 5 - 2 - 3 - 4

4. Циклы положительного веса

Что может содержать НАИКРАТЧАЙШИЙ путь?

1. Ребра отрицательного веса
2. Циклы отрицательного веса
3. Циклы нулевого веса



Q: 2 - 5 - 3 - 2 : w(P) = 0

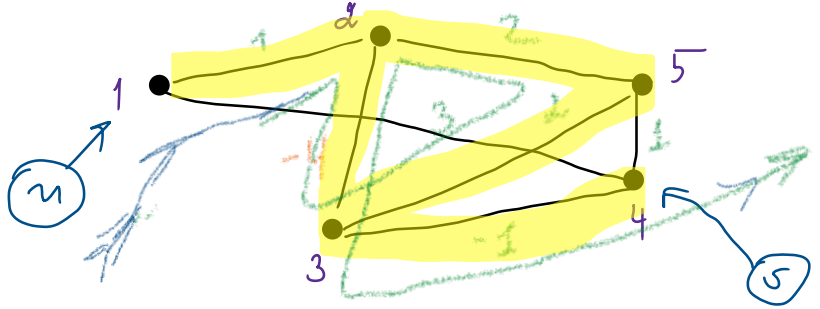
1 ⇝ 4 : 1 - 2 - 3 - 5 - 2 - 3 - 4 = 1 + k·0 - 4 - 1 = -4

k раз

4. Циклы положительного веса

Что может содержать НАИКРАТЧАЙШИЙ путь?

1. Ребра отрицательного веса
2. Циклы отрицательного веса
3. Циклы нулевого веса



Ц: 2 - 5 - 3 - 2 : $w(P) = 0$

$1 \rightsquigarrow 4$: 1 - 2 - 3 - 5 - 2 - 3 - 4 = $1 + k \cdot 0 - 4 - 1 = -4$ (k раз 3)

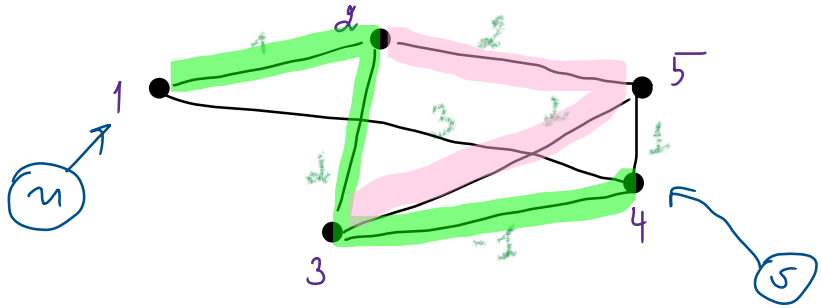
⇓

Не нужен т.к. ничего не меняет ⇒ зацикливание!

4. Циклы положительного веса

Что может содержать НАИКРАТЧАЙШИЙ путь?

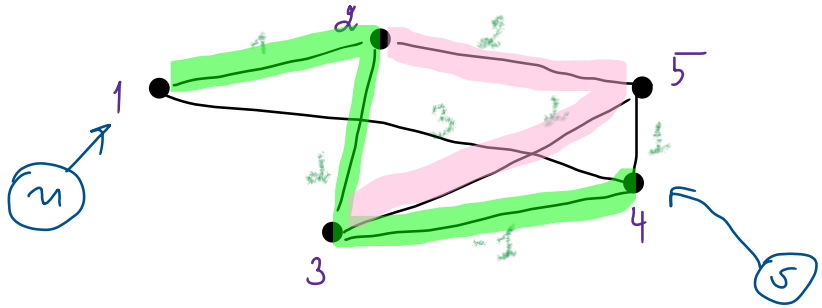
1. Ребра отрицательного веса
2. Циклы отрицательного веса
3. Циклы нулевого веса
4. Циклы положительного веса



$u \rightsquigarrow v : 1 \overset{P}{\rightsquigarrow} 4 : 1 - 2 - 3 - 4 = w(P) = 2$

Что может содержать НАИКРАТЧАЙШИЙ путь?

1. Ребра отрицательного веса
2. Циклы отрицательного веса
3. Циклы нулевого веса
4. Циклы положительного веса



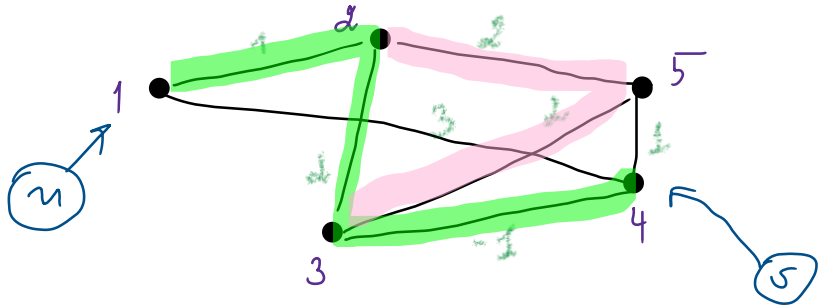
$u \rightsquigarrow v$: $1 \overset{P}{\rightsquigarrow} 4$: $1 - 2 - 3 - 4 = w(p) = 2$

если $+ \partial = \underbrace{2+2+2}_{6} = 2 - 3 - 5 - 2 \Rightarrow$ $w(p') = 2 + 6$

это не min

Что может содержать НАИКРАТЧАЙШИЙ путь?

1. Ребра отрицательного веса
2. Циклы отрицательного веса
3. Циклы нулевого веса
4. Циклы положительного веса



$u \rightsquigarrow v$: $1 \overset{P}{\rightsquigarrow} 4$: $1 - 2 - 3 - 4 = w(P) = 2$

если $+\partial$: $2+2+2 = 2-3-5-2 \Rightarrow$ $w(P') = 2 + 6$

это не min

не может содержать!

Что может содержать НАИКРАТЧАЙШИЙ путь?

1. Ребра отрицательного веса
2. Циклы отрицательного веса
3. Циклы нулевого веса
4. Циклы положительного веса

ВЫВОДЫ

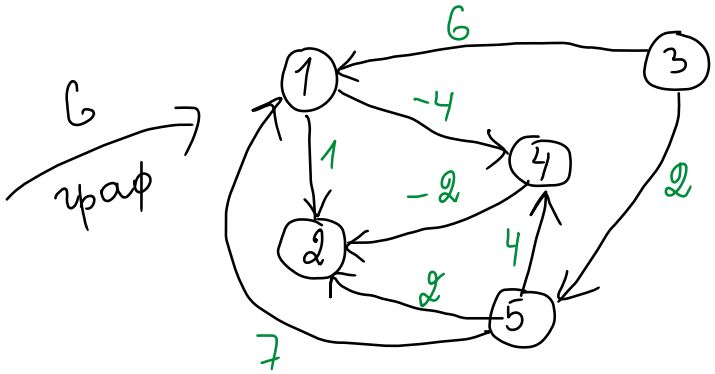
НаиКратчайший путь - путь с наименьшим весом (их может быть несколько разных, но с одним весом):

- Ацикличный
- Содержит не более V вершин и V-1 ребер
- Это ПРОСТАЯ ЦЕПЬ!

вспомним теорему: ∃ цикл ⇒ ∃ простые !!! цепь

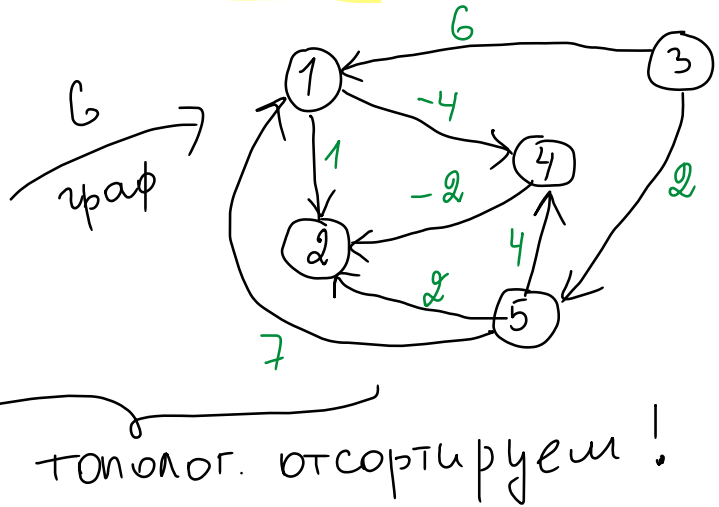
DAG: кратчайшие пути в ациклическом ориентированном графе

G	1	2	3	4	5
1	inf	1	inf	-1	inf
2	inf	inf	inf	inf	inf
3	6	inf	inf	inf	2
4	inf	-2	inf	inf	inf
5	7	2	inf	4	inf



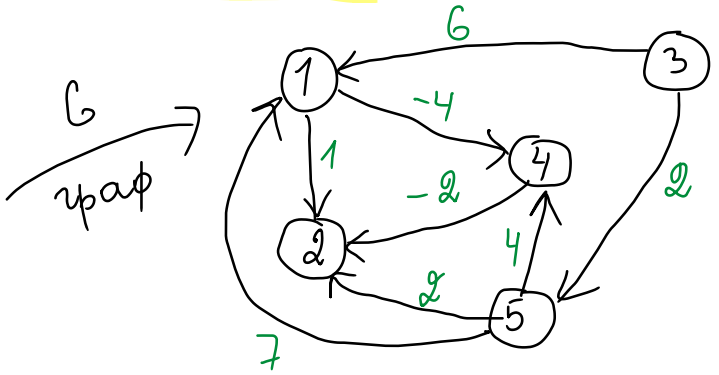
DAG: кратчайшие пути в ациклическом ориентированном графе

G	1	2	3	4	5
1	inf	1	inf	-1	inf
2	inf	inf	inf	inf	inf
3	6	inf	inf	inf	2
4	inf	-2	inf	inf	inf
5	7	2	inf	4	inf



DAG: кратчайшие пути в ациклическом ориентированном графе

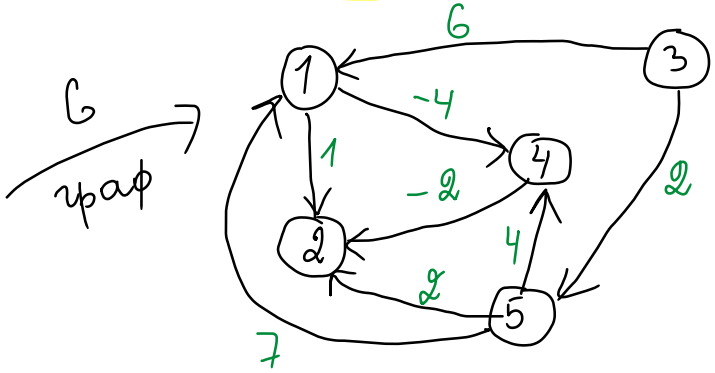
G	1	2	3	4	5
1	inf	1	inf	-1	inf
2	inf	inf	inf	inf	inf
3	6	inf	inf	inf	2
4	inf	-2	inf	inf	inf
5	7	2	inf	4	inf



тополог. отсортируем! 3 5 1 4 2

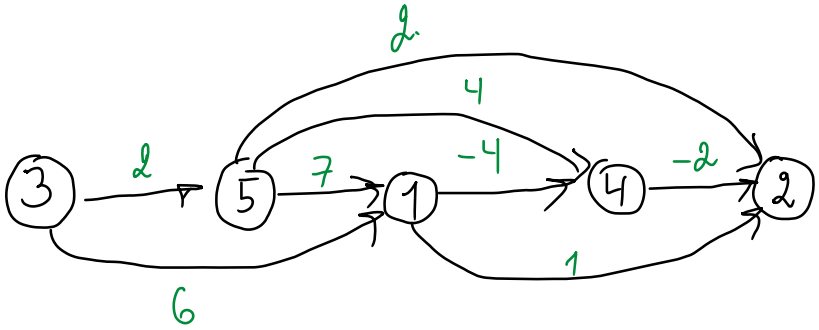
DAG: кратчайшие пути в ациклическом ориентированном графе

G	1	2	3	4	5
1	inf	1	inf	-1	inf
2	inf	inf	inf	inf	inf
3	6	inf	inf	inf	2
4	inf	-2	inf	inf	inf
5	7	2	inf	4	inf



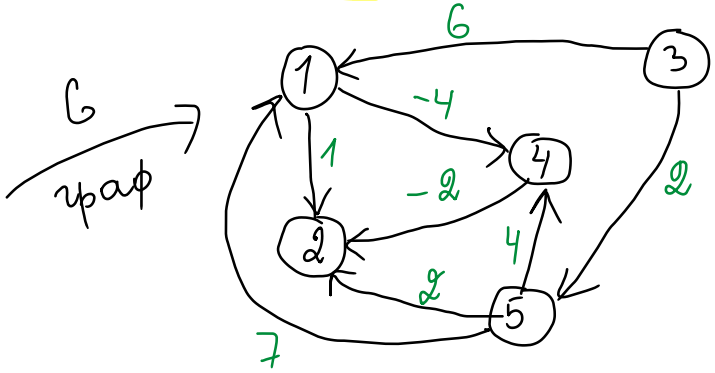
тополог. отсортируем! 3 5 1 4 2

граф G



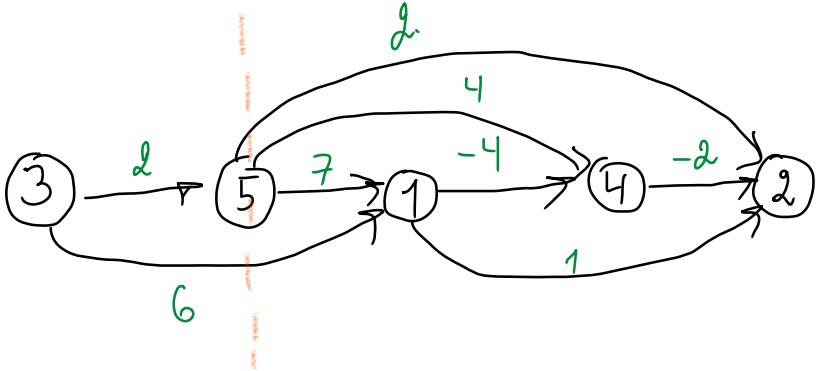
DAG: кратчайшие пути в ациклическом ориентированном графе

G	1	2	3	4	5
1	inf	1	inf	-1	inf
2	inf	inf	inf	inf	inf
3	6	inf	inf	inf	2
4	inf	-2	inf	inf	inf
5	7	2	inf	4	inf



тополог. отсортируем! 3 5 1 4 2

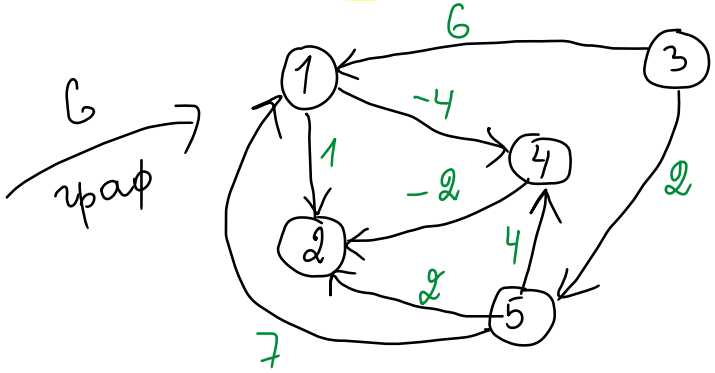
граф G



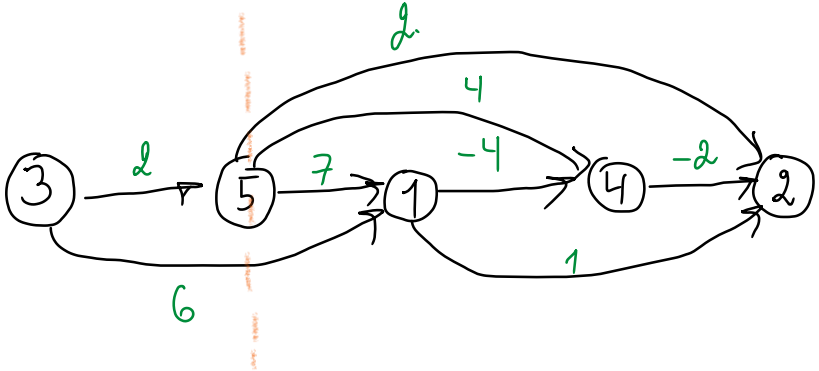
?Найти пути от 5 до всех?

DAG: кратчайшие пути в ациклическом ориентированном графе

G	1	2	3	4	5
1	inf	1	inf	-1	inf
2	inf	inf	inf	inf	inf
3	6	inf	inf	inf	2
4	inf	-2	inf	inf	inf
5	7	2	inf	4	inf



тополог. отсортируем! 3 5 1 4 2

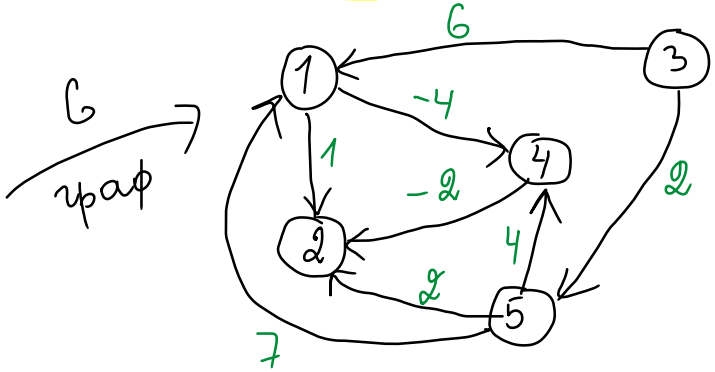


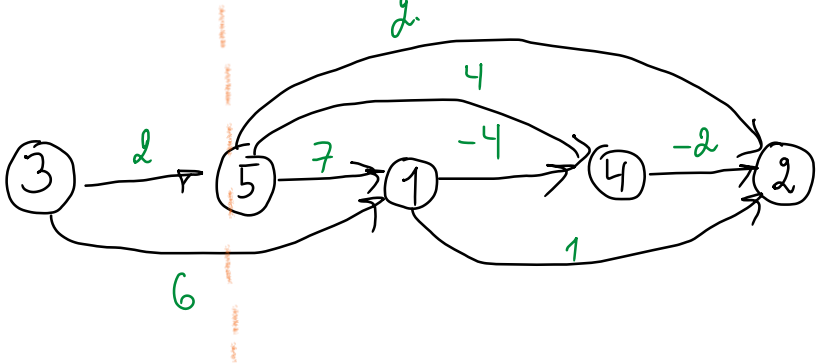
?Найти пути от 5 до всех?

все вершины до (5)ки в топ сорт недостижимы из (5)ки!

DAG: кратчайшие пути в ациклическом ориентированном графе

G	1	2	3	4	5
1	inf	1	inf	-1	inf
2	inf	inf	inf	inf	inf
3	6	inf	inf	inf	2
4	inf	-2	inf	inf	inf
5	7	2	inf	4	inf



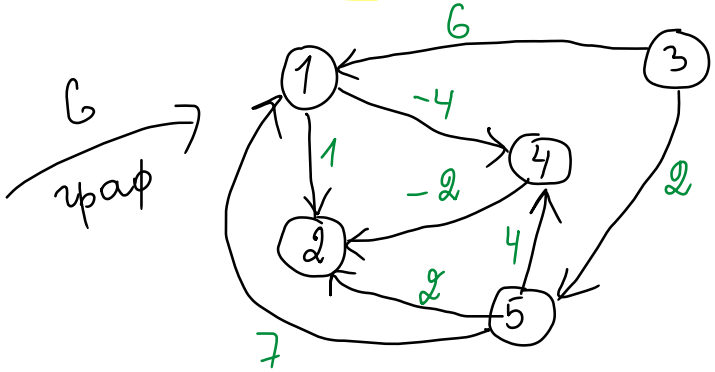


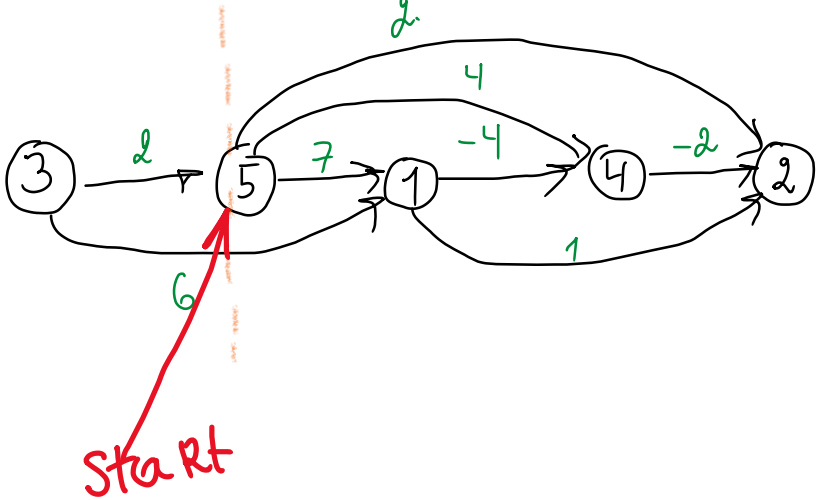
?Найти пути от 5 до всех?

d	1	2	3	4	5	← старт
	∞	∞	∞	∞	∞	

DAG: кратчайшие пути в ациклическом ориентированном графе

G	1	2	3	4	5
1	inf	1	inf	-1	inf
2	inf	inf	inf	inf	inf
3	6	inf	inf	inf	2
4	inf	-2	inf	inf	inf
5	7	2	inf	4	inf





?Найти пути от 5 до всех?

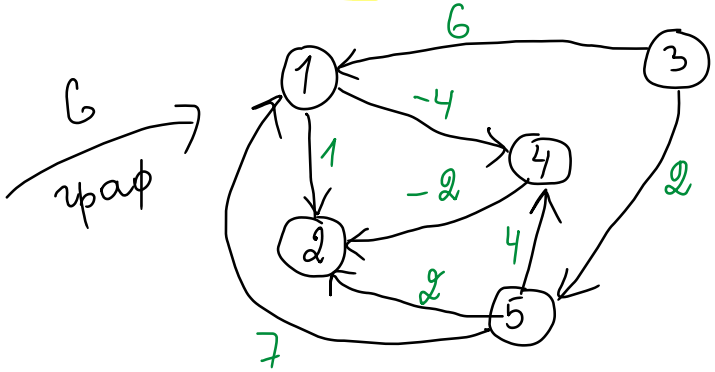
1: из (5) все смежные

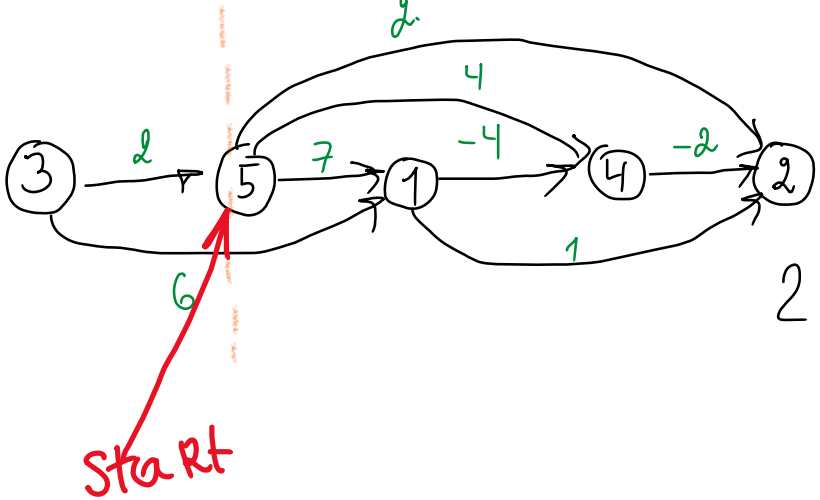
1	2	3	4	5
∞	∞	∞	∞	∞
7	2		4	0

start

DAG: кратчайшие пути в ациклическом ориентированном графе

G	1	2	3	4	5
1	inf	1	inf	-1	inf
2	inf	inf	inf	inf	inf
3	6	inf	inf	inf	2
4	inf	-2	inf	inf	inf
5	7	2	inf	4	inf





? Найти пути от 5 до всех ?

1: из (5) все смежные

1	2	3	4	5
∞	∞	∞	∞	0
7	2		4	

start

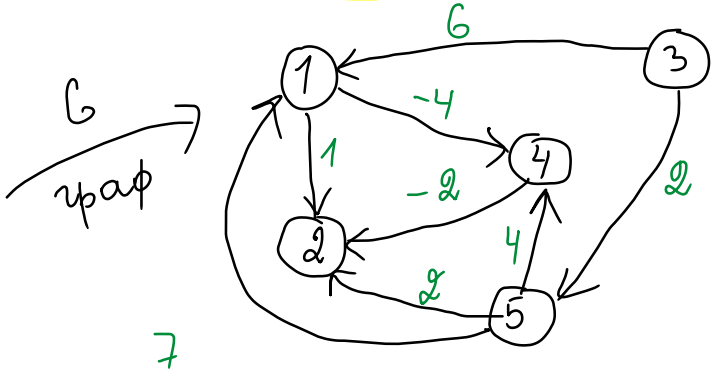
2: из (1) все смежные если в (1) ≠ ∞

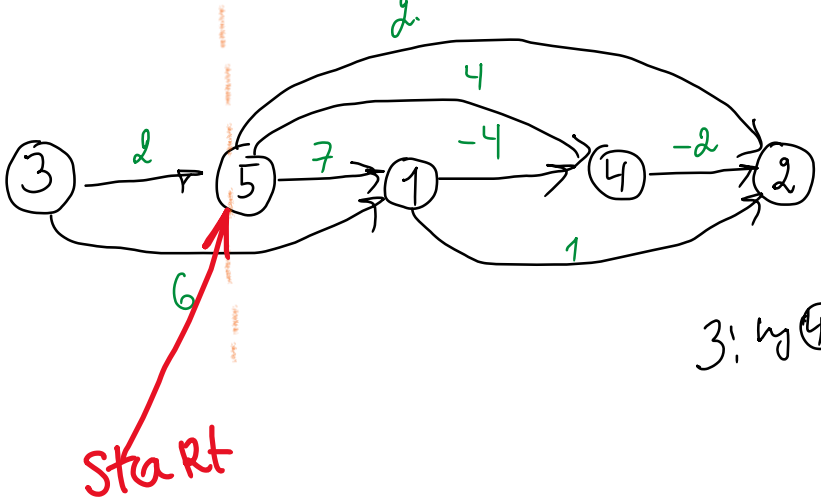
1	2	3	4	5
7	2	∞	4	0
			3	

8 3-4=3

DAG: кратчайшие пути в ациклическом ориентированном графе

G	1	2	3	4	5
1	inf	1	inf	-1	inf
2	inf	inf	inf	inf	inf
3	6	inf	inf	inf	2
4	inf	-2	inf	inf	inf
5	7	2	inf	4	inf





1: из (5) все смежные

2: из (1) все смежные если в (1) ≠ ∞

3: из (4) смежные, если в (4) ≠ ∞

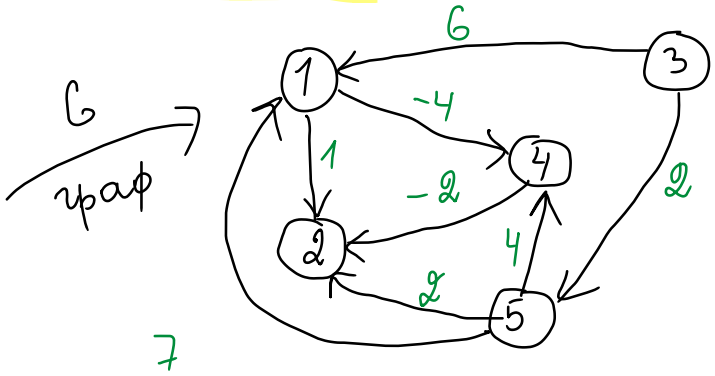
1	2	3	4	5
∞	∞	∞	∞	0

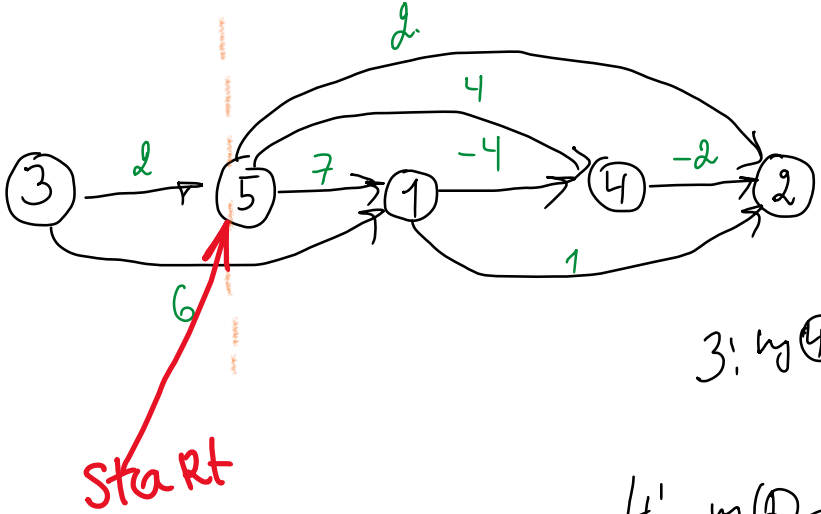
1	2	3	4	5
7	2	∞	4	0

1	2	3	4	5
7	2	∞	3	0

DAG: кратчайшие пути в ациклическом ориентированном графе

G	1	2	3	4	5
1	inf	1	inf	-1	inf
2	inf	inf	inf	inf	inf
3	6	inf	inf	inf	2
4	inf	-2	inf	inf	inf
5	7	2	inf	4	inf





1: из (5) все смежные

2: из (1) все смежные если в (1) ≠ ∞

3: из (4) смежные, если в (4) ≠ ∞

4: из (2) – нет смежных!

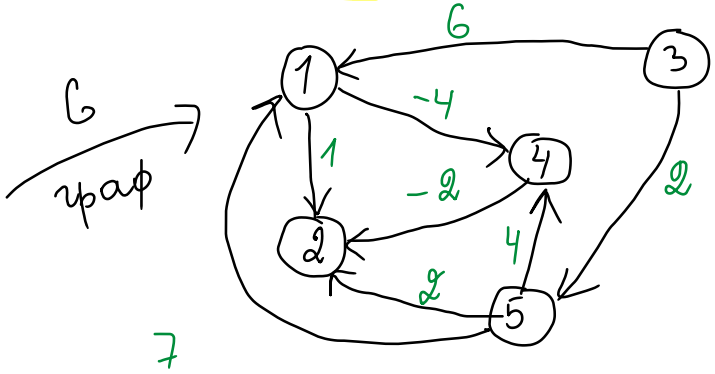
1	2	3	4	5
∞	∞	∞	∞	0

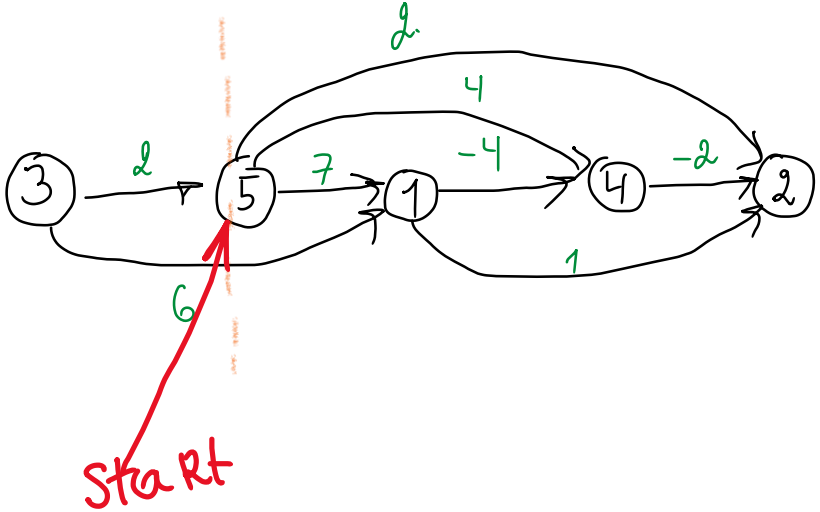
1	2	3	4	5
7	2	∞	4	0

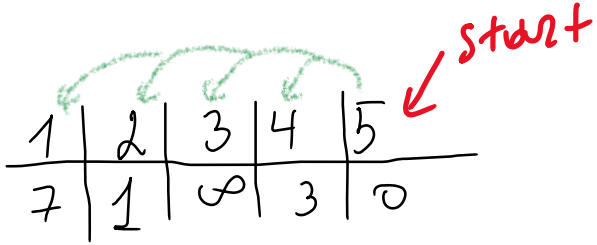
1	2	3	4	5
7	2	∞	3	0

DAG: кратчайшие пути в ациклическом ориентированном графе

G	1	2	3	4	5
1	inf	1	inf	-1	inf
2	inf	inf	inf	inf	inf
3	6	inf	inf	inf	2
4	inf	-2	inf	inf	inf
5	7	2	inf	4	inf







Как будем искать НАИКРАТЧАЙШИЙ путь?

Как будем искать НАИКРАТЧАЙШИЙ путь?

- Понадобится массив **distance [v] или d[v]** - вес кратчайшего пути от **s** до **v**

Как будем искать НАИКРАТЧАЙШИЙ путь?

- Понадобится массив **distance [v] или d[v]** - вес кратчайшего пути от **s** до **v:**
  - на каждом шаге это оценка веса кратчайшего пути сверху (на старте считаем бесконечностями или что пути такого нет)

- Массив предков: **`predki[v] или p[v]`** – предшествующая вершина на кратчайшем пути или null
  - При каждом улучшении пути записываем через какую вершину данное улучшение было получено, а именно запоминаем предков

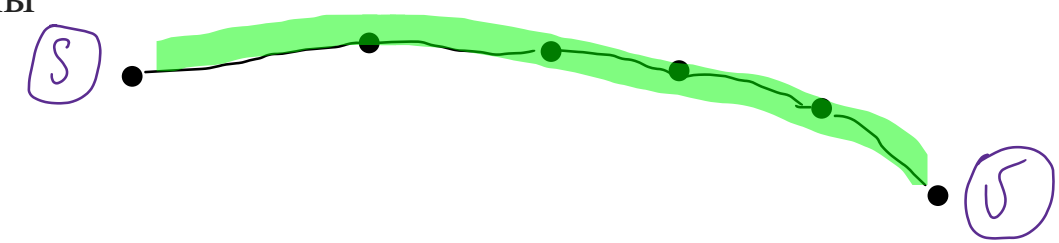
Как будем искать НАИКРАТЧАЙШИЙ путь?

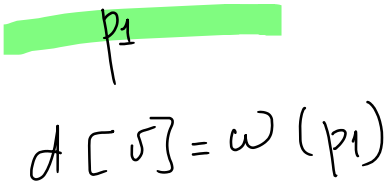
- Понадобится массив **distance [v] или d[v]** - вес кратчайшего пути от **s** до **v:**
  - на каждом шаге это оценка веса кратчайшего пути сверху (на старте считаем бесконечностями или что пути такого нет)
- Массив предков: **`predki[v]` `или` `p[v]`** – предшествующая вершина на кратчайшем пути или null
  - При каждом улучшении пути записываем через какую вершину данное улучшение было получено, а именно запоминаем предков

- **Ослабление ребра (релаксация: Relax)** - попытка улучшить найденный путь до вершины

Как будем искать НАИКРАТЧАЙШИЙ путь?

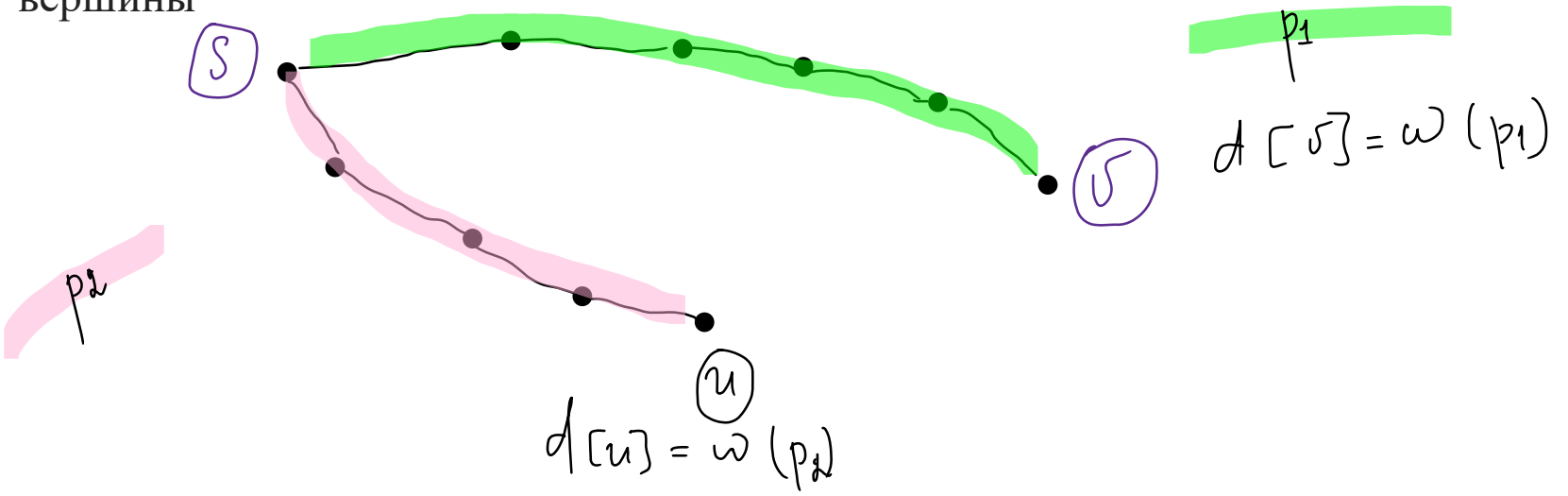
Ослабление ребра (релаксация: Relax) - попытка улучшить найденный путь до вершины





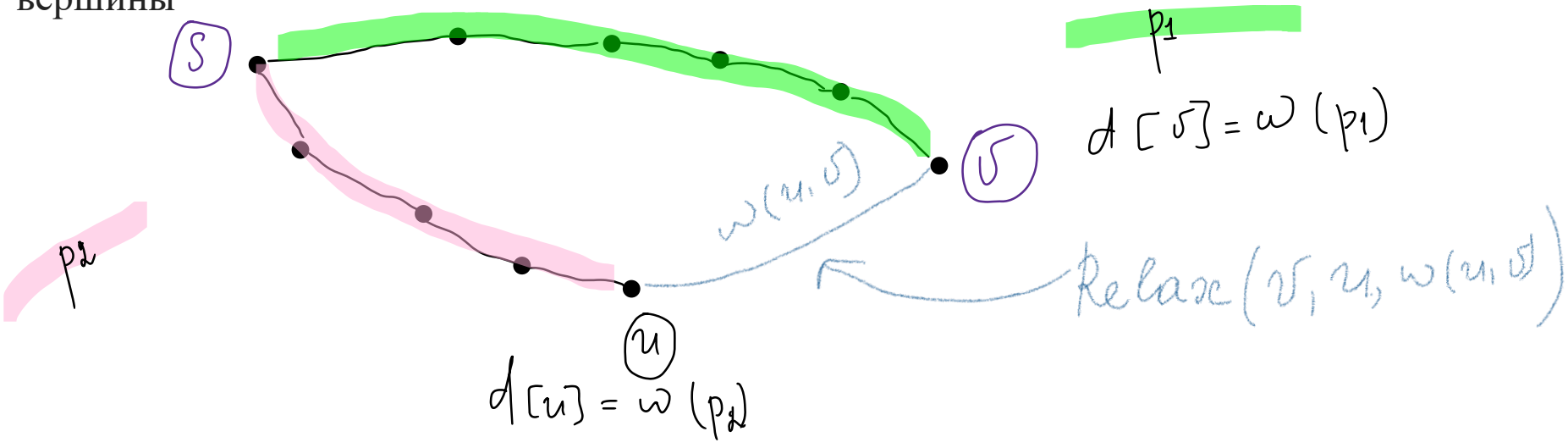
Как будем искать НАИКРАТЧАЙШИЙ путь?

Ослабление ребра (релаксация: Relax) - попытка улучшить найденный путь до вершины



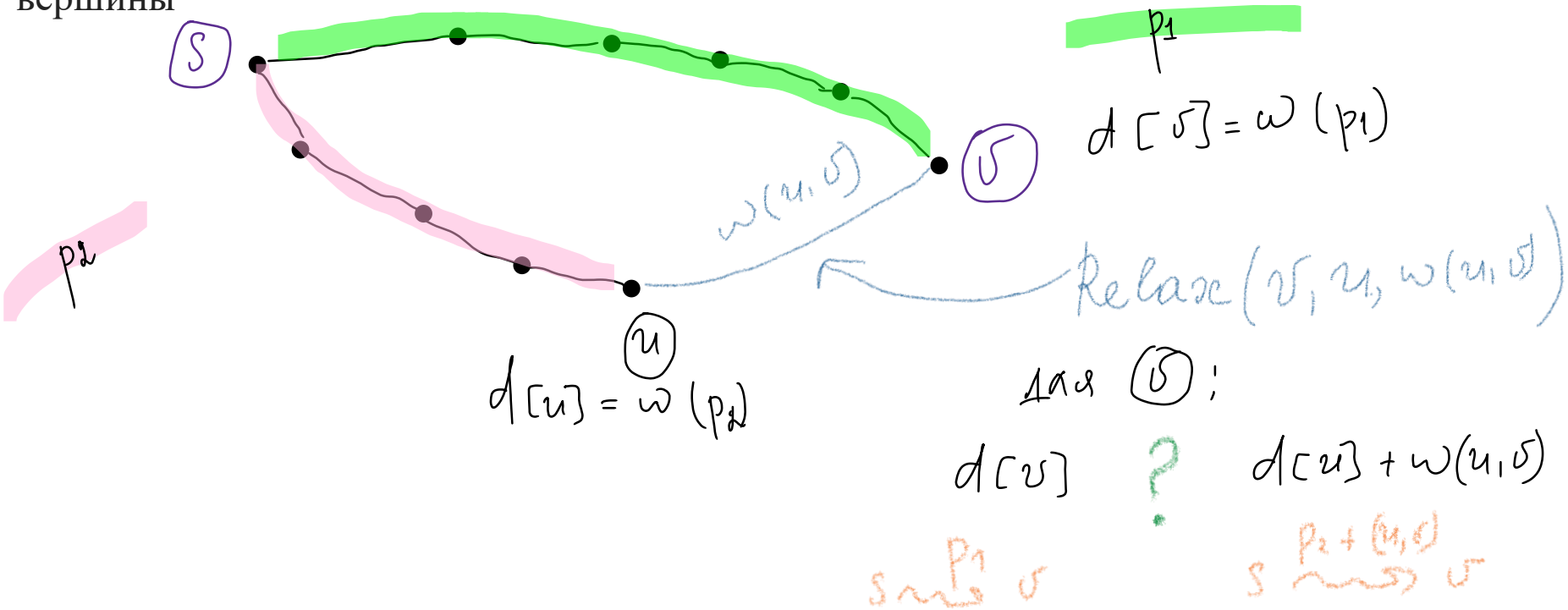
Как будем искать НАИКРАТЧАЙШИЙ путь?

Ослабление ребра (релаксация: Relax) - попытка улучшить найденный путь до вершины



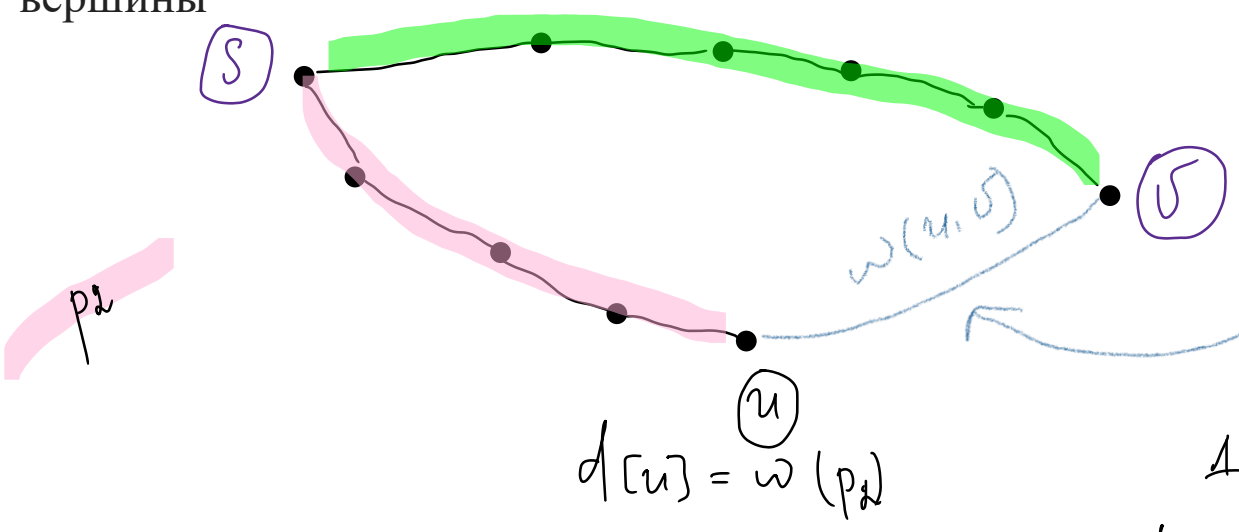
Как будем искать НАИКРАТЧАЙШИЙ путь?

Ослабление ребра (релаксация: Relax) - попытка улучшить найденный путь до вершины



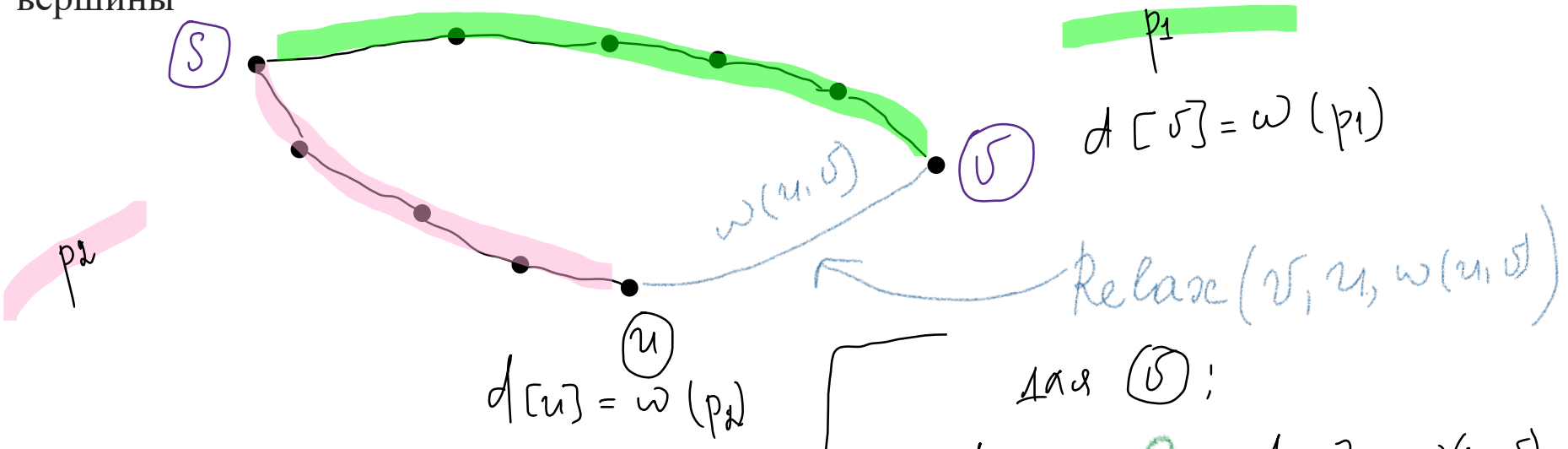
Как будем искать НАИКРАТЧАЙШИЙ путь?

Ослабление ребра (релаксация: Relax) - попытка улучшить найденный путь до вершины



Как будем искать НАИКРАТЧАЙШИЙ путь?

Ослабление ребра (релаксация: Relax) - попытка улучшить найденный путь до вершины




```
Relax(u, v, w[u,v]):
if (d[v] > d[u] + w(u,v) )
    d[v] = d[u] + w(u,v)
    predki[v] = u
```

для (v):

d[v] ? d[u] + w(u,v)

s ~p1~ v

s ~p2 + (u,v)~ v

Выбираем MIN !

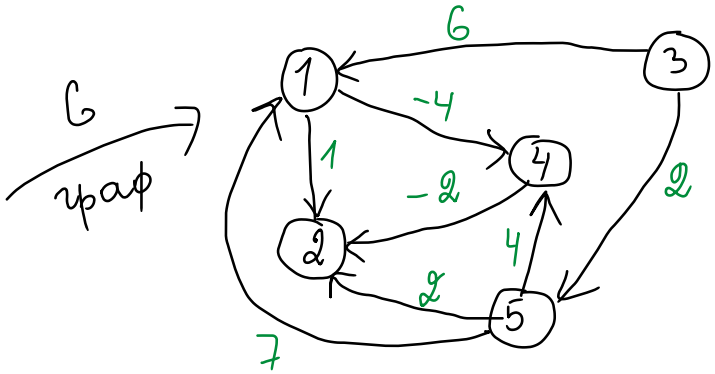
Инициализация дополнительных структур

- Нужно заполнить distance [v] бесконечностями, так как на старте нет найденных кратчайших путей
- Массив предков заполнить null, так как предков нет на старте ни для какой вершины

```
InitSource(G, s):
      for v ∈ V
            d[v] = infinity
            predki[v] = null
      d[s] = 0
```

DAG: кратчайшие пути в ациклическом ориентированном графе

G	1	2	3	4	5
1	inf	1	inf	-1	inf
2	inf	inf	inf	inf	inf
3	6	inf	inf	inf	2
4	inf	-2	inf	inf	inf
5	7	2	inf	4	inf

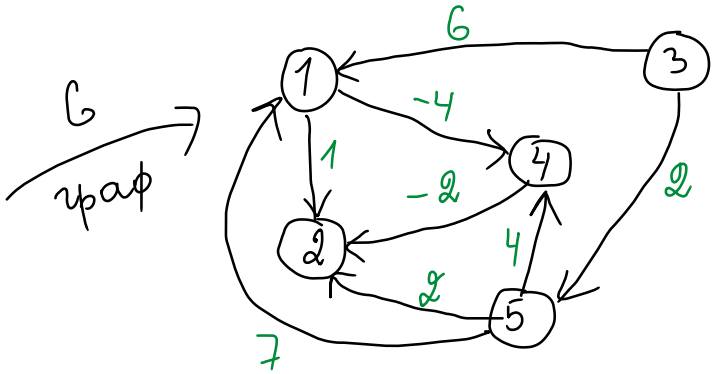


Суть: ослабление ребер в порядке топологической сортировки

```
DAG(G, s):
        TS = TopSort(G)
        InitSource(G, s)
        for u ∈ TS
              for v ∈ W[u]
                    Relax(u, v, W[u, v])
```

DAG: кратчайшие пути в ациклическом ориентированном графе

G	1	2	3	4	5
1	inf	1	inf	-1	inf
2	inf	inf	inf	inf	inf
3	6	inf	inf	inf	2
4	inf	-2	inf	inf	inf
5	7	2	inf	4	inf



топсорт!
3 5 1 4 2

Суть: ослабление ребер в порядке топологической сортировки

```
DAG(G, s):
        TS = TopSort(G)
        InitSource(G, s)
        for u ∈ TS
            for v ∈ W[u]
                Relax(u, v, W[u, v])
```

predki

№	1	2	3	4	5
0	\	\	\	\	\
1	3	\	\	\	3
2	3	5	\	5	3
3	3	5	\	1	3
4	3	4	\	1	3

d

start

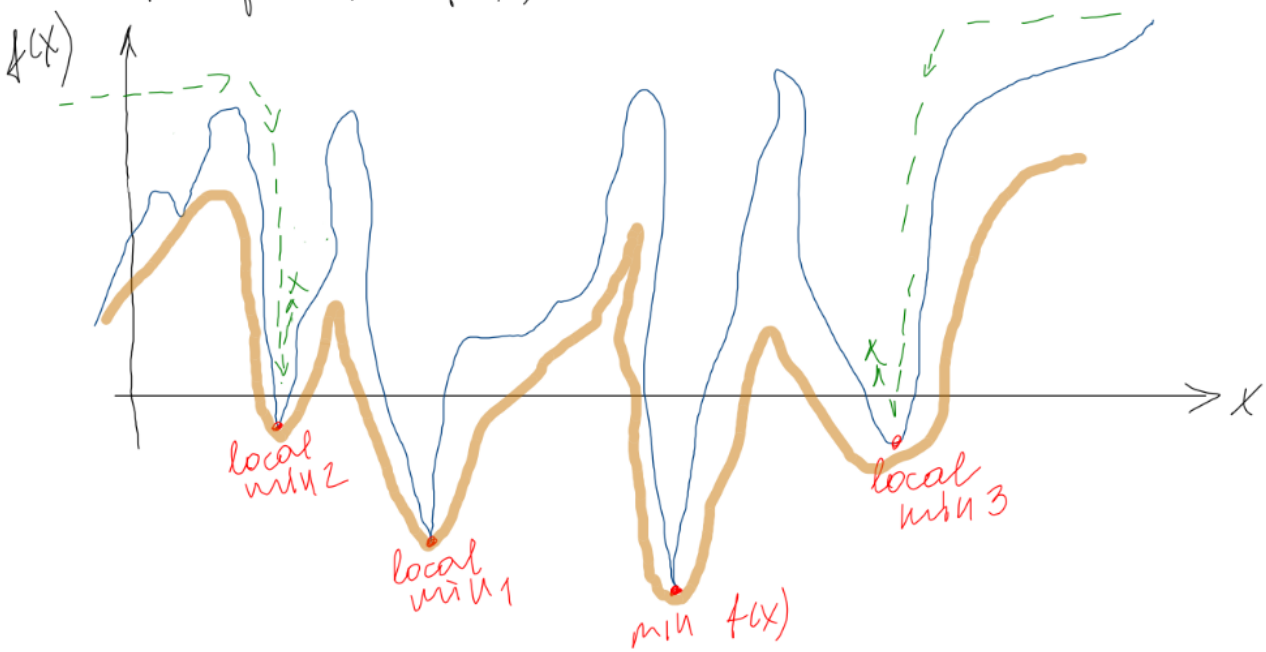
1	2	3	4	5
inf	inf	0	inf	inf
6	inf	0	inf	2
6	4	0	6	2
6	4	0	5	2
6	3	0	5	2

ЖАДНЫЙ алгоритм

ЖАДНЫЙ алгоритм - алгоритм, заключающийся в принятии локально оптимальных решений на каждом этапе, допуская, что конечное решение также окажется оптимальным.

иллюстрирующий пример

на примере f(x) и её min и локальных min



время выполнения для нахождения min f(x) практически всего f(x) => долго

время работы алгоритма, кт нашел min но локальный, но кт отражает перепады функции что позволяет принять верное решение

ЖАДНЫЙ алгоритм

ЖАДНЫЙ алгоритм - алгоритм, заключающийся в принятии локально оптимальных решений на каждом этапе, допуская, что конечное решение также окажется оптимальным.

Рассмотрим алгоритм, который без перебора всех возможных путей позволит найти самое оптимальное решение!

ЖАДНЫЙ алгоритм

ЖАДНЫЙ алгоритм - алгоритм, заключающийся в принятии локально оптимальных решений на каждом этапе, допуская, что конечное решение также окажется оптимальным.

Рассмотрим алгоритм, который без перебора всех возможных путей позволит найти самое оптимальное решение!

Стремимся за один обход графа найти наикратчайшие пути!

ЖАДНЫЙ алгоритм

ЖАДНЫЙ алгоритм - алгоритм, заключающийся в принятии локально оптимальных решений на каждом этапе, допуская, что конечное решение также окажется оптимальным.

Рассмотрим алгоритм, который без перебора всех возможных путей позволит найти самое оптимальное решение!

Стремимся за один обход графа найти наикратчайшие пути!

АЛГОРИТМ ДЕЙКСТРЫ!

ЖАДНЫЙ алгоритм

ЖАДНЫЙ алгоритм - алгоритм, заключающийся в принятии локально оптимальных решений на каждом этапе, допуская, что конечное решение также окажется оптимальным.

Лемма

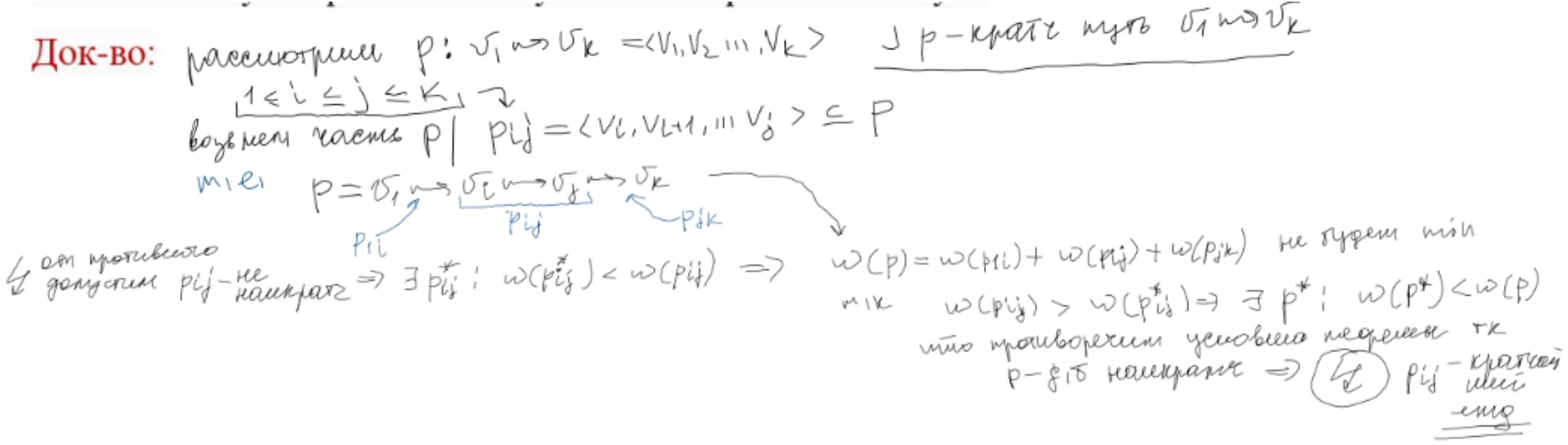
Частичные пути кратчайших путей тоже кратчайшие пути

ЖАДНЫЙ алгоритм

ЖАДНЫЙ алгоритм - алгоритм, заключающийся в принятии локально оптимальных решений на каждом этапе, допуская, что конечное решение также окажется оптимальным.

Лемма

Частичные пути кратчайших путей тоже кратчайшие пути



АЛГОРИТМ ДЕЙКСТРЫ

Суть:

1. Каждую итерацию выбираем вершину, до которой кратчайший путь и она не помечена

2. Помечаем выбранную вершину и просматриваем все ребра из нее (каждым пытаем ослабить кратчайшие пути)

АЛГОРИТМ ДЕЙКСТРЫ

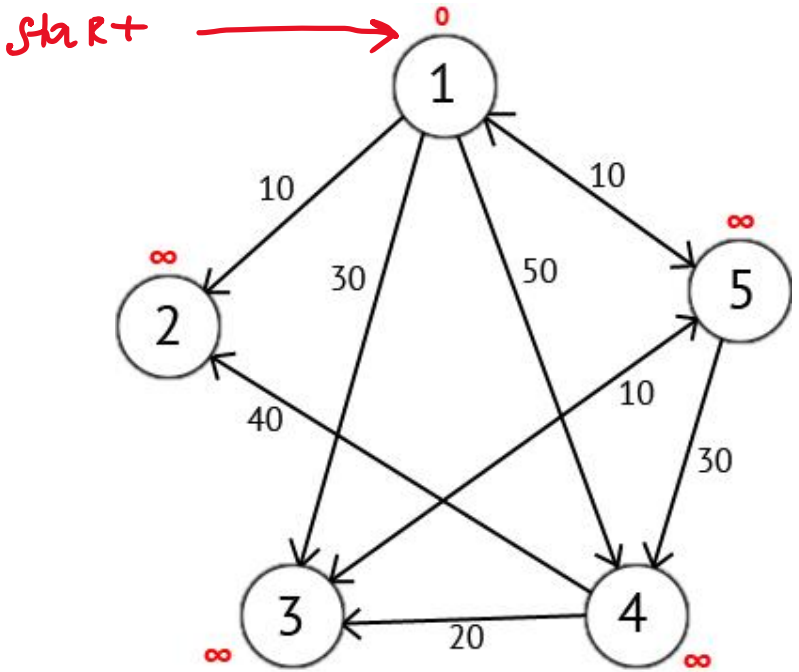
Суть:

1. Каждую итерацию выбираем вершину, до которой кратчайший путь и она не помечена

2. Помечаем выбранную вершину и просматриваем все ребра из нее (каждым пытаем ослабить кратчайшие пути)

От (1) до всех!

	1	2	3	4	5
distance					
predki					



АЛГОРИТМ ДЕЙКСТРЫ

Суть:

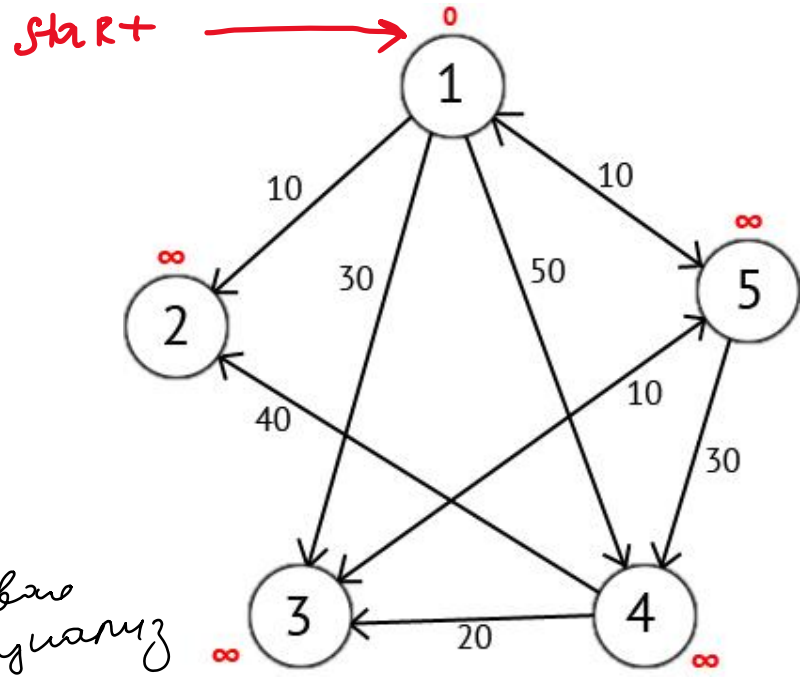
1. Каждую итерацию выбираем вершину, до которой кратчайший путь и она не помечена

2. Помечаем выбранную вершину и просматриваем все ребра из нее (каждым пытаем ослабить кратчайшие пути)

От (1) до всех!

	1	2	3	4	5
distance	0	∞	∞	∞	∞
predki	null	null	null	null	null

стартовая инициализ



АЛГОРИТМ ДЕЙКСТРЫ

Суть:

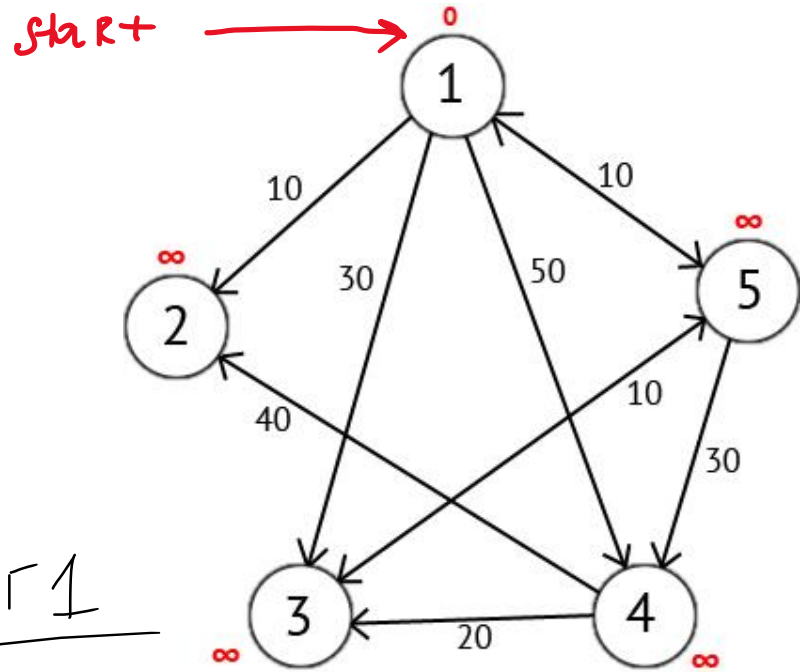
1. Каждую итерацию выбираем вершину, до которой кратчайший путь и она не помечена

2. Помечаем выбранную вершину и просматриваем все ребра из нее (каждым пытаем ослабить кратчайшие пути)

От (1) до всех!

	1	2	3	4	5
distance	0	∞	∞	∞	∞
predki	null	null	null	null	null

ШАГ 1



АЛГОРИТМ ДЕЙКСТРЫ

Суть:

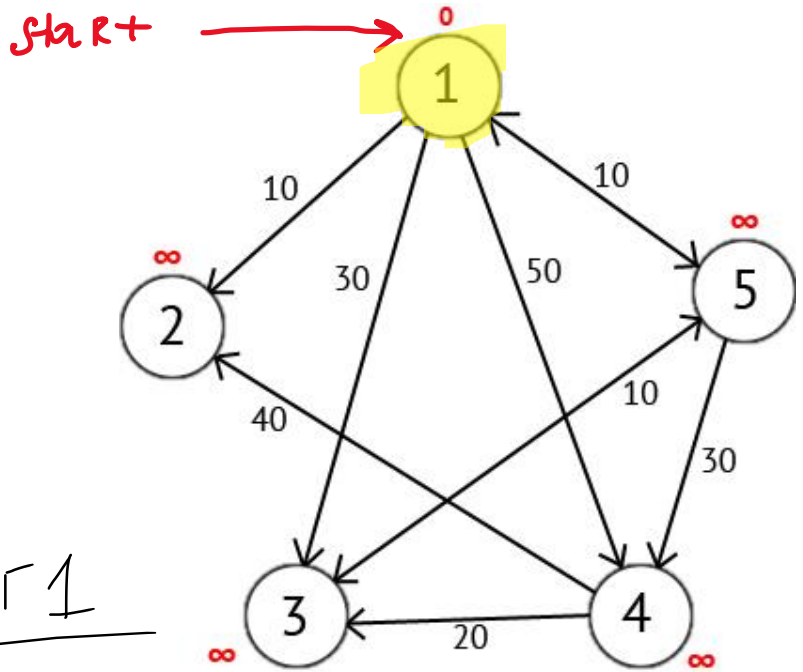
1. Каждую итерацию выбираем вершину, до которой кратчайший путь и она не помечена

2. Помечаем выбранную вершину и просматриваем все ребра из нее (каждым пытаем ослабить кратчайшие пути)

От (1) до всех!

	1	2	3	4	5
distance	0	∞	∞	∞	∞
predki	null	null	null	null	null
used	0	0	0	0	0

ШАГ 1

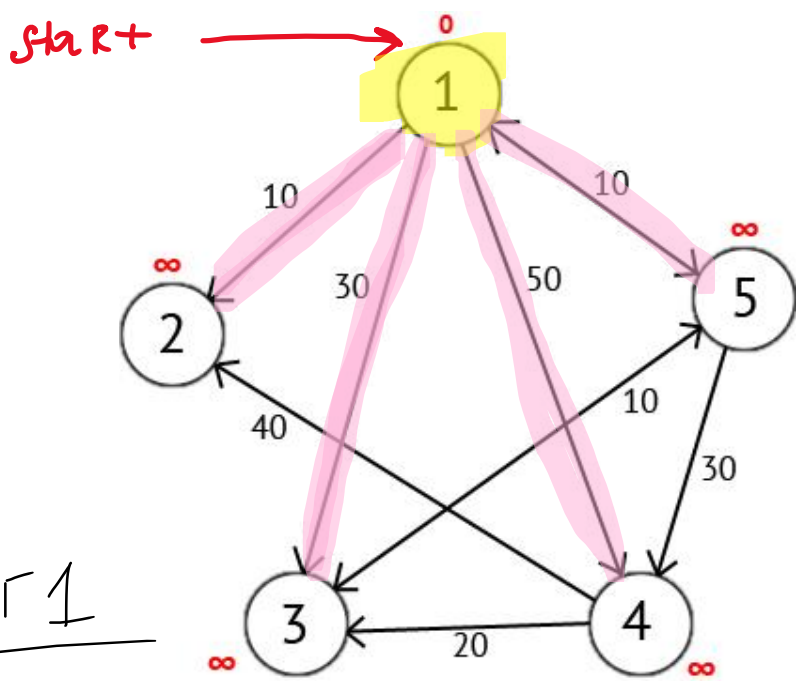


АЛГОРИТМ ДЕЙКСТРЫ

Суть:

1. Каждую итерацию выбираем вершину, до которой кратчайший путь и она не помечена

2. Помечаем выбранную вершину и просматриваем все ребра из нее (каждым пытаем ослабить кратчайшие пути)



От (1) до всех!

	1	2	3	4	5
distance	0	∞	∞	∞	∞
predki	null	null	null	null	null
used	1	0	0	0	0

ШАГ 1

Relax(5, 1, 10)

Relax(2, 1, 10) => ∞ > d[1] + 10

Relax(3, 1, 30) => ∞ > d[1] + 30

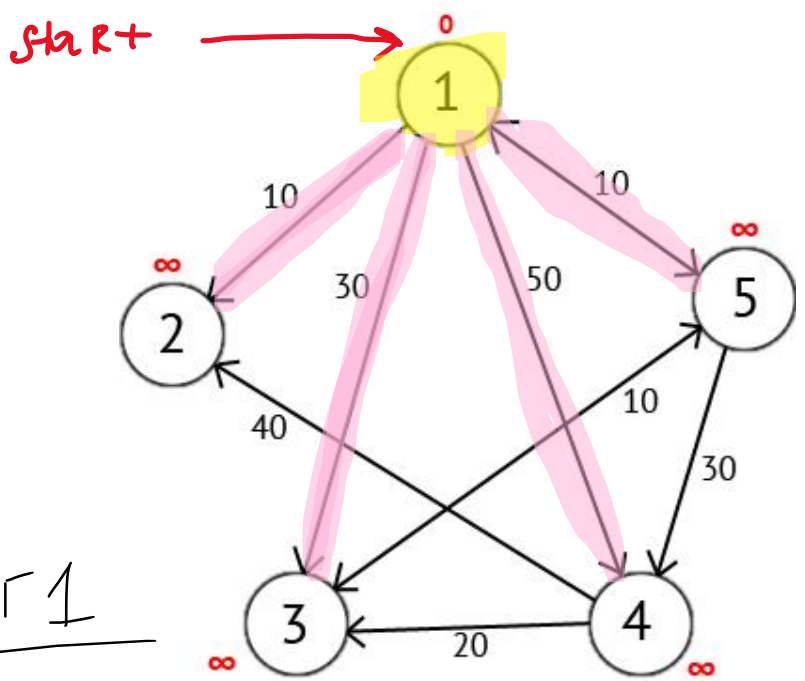
Relax(4, 1, 50) => ∞ > d[1] + 50

АЛГОРИТМ ДЕЙКСТРЫ

Суть:

1. Каждую итерацию выбираем вершину, до которой кратчайший путь и она не помечена

2. Помечаем выбранную вершину и просматриваем все ребра из нее (каждым пытаем ослабить кратчайшие пути)



От (1) до всех:

	1	2	3	4	5
distance	0	10	30	50	10
predki	null	null	null	null	null
used	1	0	0	0	0

ШАГ 1

Relax (5, 1, 10) ⇒ d[5] = 10

Relax (2, 1, 10) ⇒ ∞ > d[1] + 10 ⇒ d[2] = 10

Relax (3, 1, 30) ⇒ ∞ > d[1] + 30 ⇒ d[3] = 30

Relax (4, 1, 50) ⇒ ∞ > d[1] + 50 ⇒ d[4] = 50

АЛГОРИТМ ДЕЙКСТРЫ

Суть:

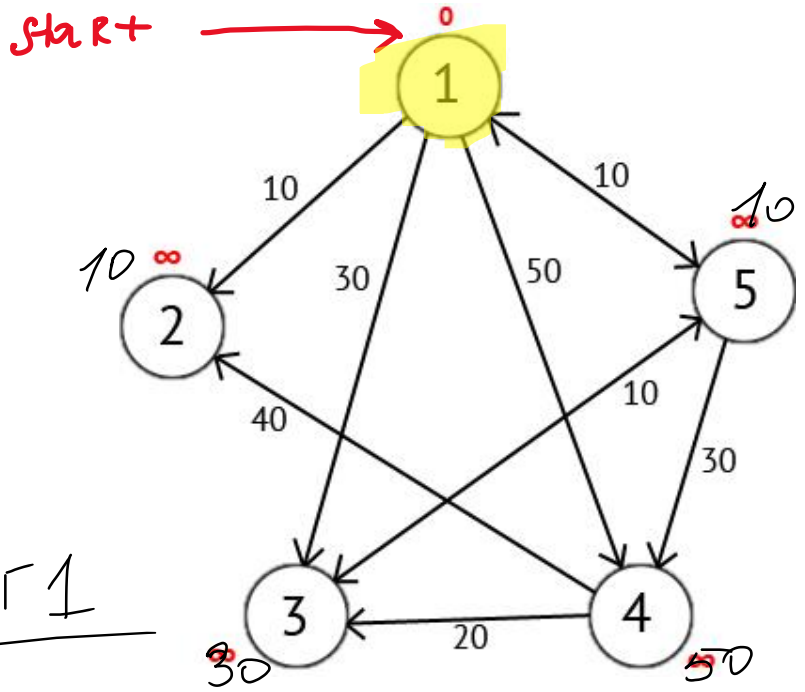
1. Каждую итерацию выбираем вершину, до которой кратчайший путь и она не помечена

2. Помечаем выбранную вершину и просматриваем все ребра из нее (каждым пытаем ослабить кратчайшие пути)

От 1 до всех!

	1	2	3	4	5
distance	0	10	30	50	10
predki	null	1	1	1	1
used	1	0	0	0	0

ШАГ 1



АЛГОРИТМ ДЕЙКСТРЫ

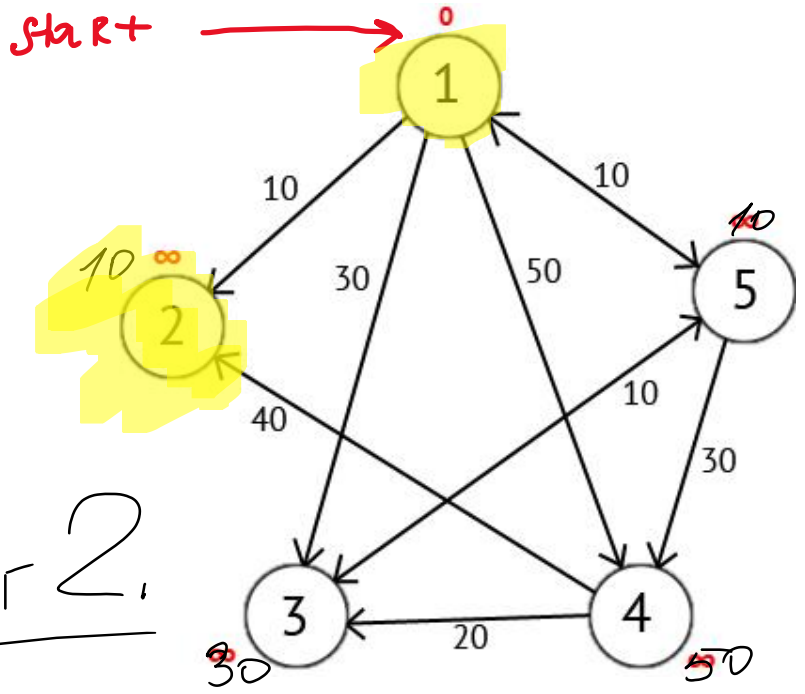
Суть:

1. Каждую итерацию выбираем вершину, до которой кратчайший путь и она не помечена

2. Помечаем выбранную вершину и просматриваем все ребра из нее (каждым пытаем ослабить кратчайшие пути)

От (1) до всех!

	1	2	3	4	5
distance	0	10	30	50	10
predki	null	1	1	1	1
used	1	0	0	0	0



Шаг 2.

Выбираем min d[2] = 10

АЛГОРИТМ ДЕЙКСТРЫ

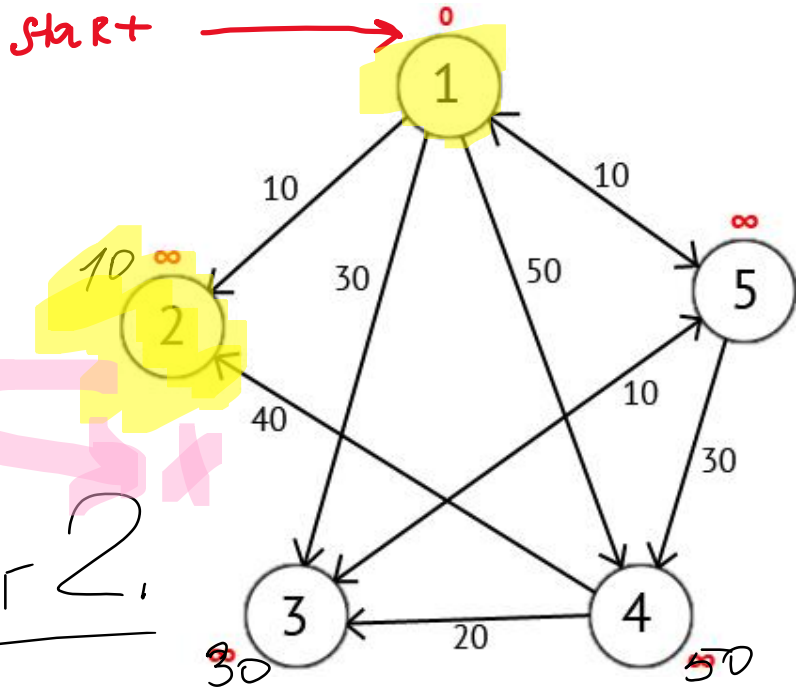
Суть:

1. Каждую итерацию выбираем вершину, до которой кратчайший путь и она не помечена

2. Помечаем выбранную вершину и просматриваем все ребра из нее (каждым пытаем ослабить кратчайшие пути)

От (1) до всех:

	1	2	3	4	5
distance	0	10	30	50	10
predki	null	1	1	1	1
used	1	1	0	0	0



Шаг 2.

Выбираем min d[2] = 10

↳ Relax = ∅

Т.к нет ребер из 2

АЛГОРИТМ ДЕЙКСТРЫ

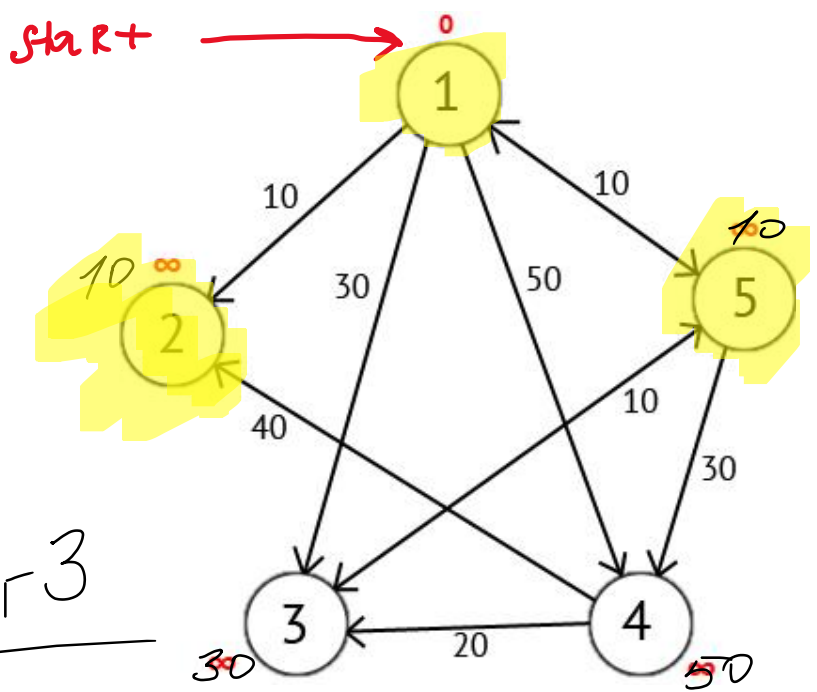
Суть:

1. Каждую итерацию выбираем вершину, до которой кратчайший путь и она не помечена

2. Помечаем выбранную вершину и просматриваем все ребра из нее (каждым пытаем ослабить кратчайшие пути)

От (1) до всех:

	1	2	3	4	5
distance	0	10	30	50	10
predki	null	1	1	1	1
used	1	1	0	0	1



Шаг 3

Выбираем min d[5] = 10

Relax(3, 5, 10) => 30 > d[5]+10

Relax(4, 5, 30) => 50 > d[5]+30

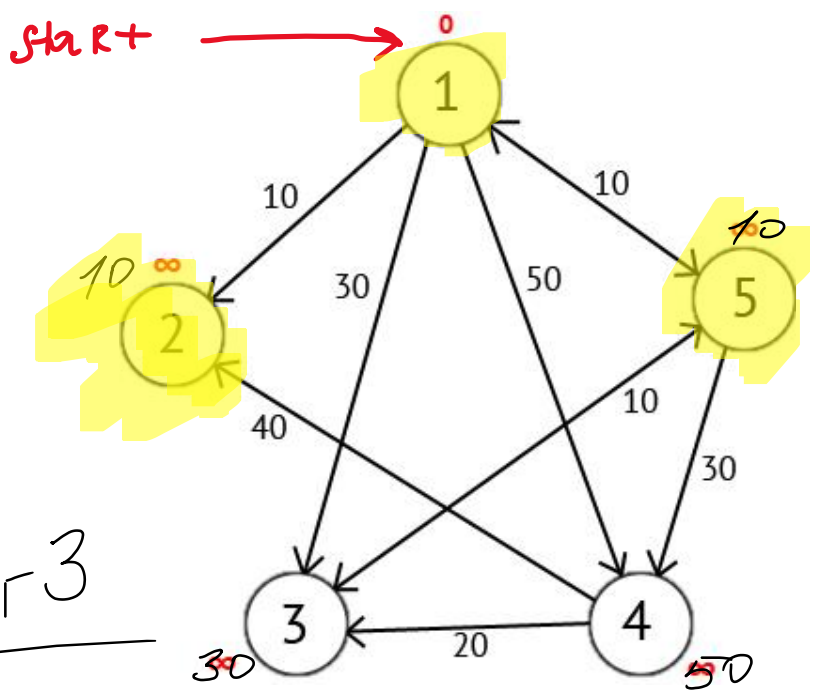
(1) – помечена

АЛГОРИТМ ДЕЙКСТРЫ

Суть:

1. Каждую итерацию выбираем вершину, до которой кратчайший путь и она не помечена

2. Помечаем выбранную вершину и просматриваем все ребра из нее (каждым пытаем ослабить кратчайшие пути)



От (1) до всех:

	1	2	3	4	5
distance	0	10	30	50	10
predki	null	1	1	1	1
used	1	1	0	0	1

Шаг 3

Выбираем min d[5] = 10

Relax(3, 5, 10) => 30 > d[5]+10 => d[3] = 20

Relax(4, 5, 30) => 50 > d[5]+30 => d[4] = 40

(1) – помечена

АЛГОРИТМ ДЕЙКСТРЫ

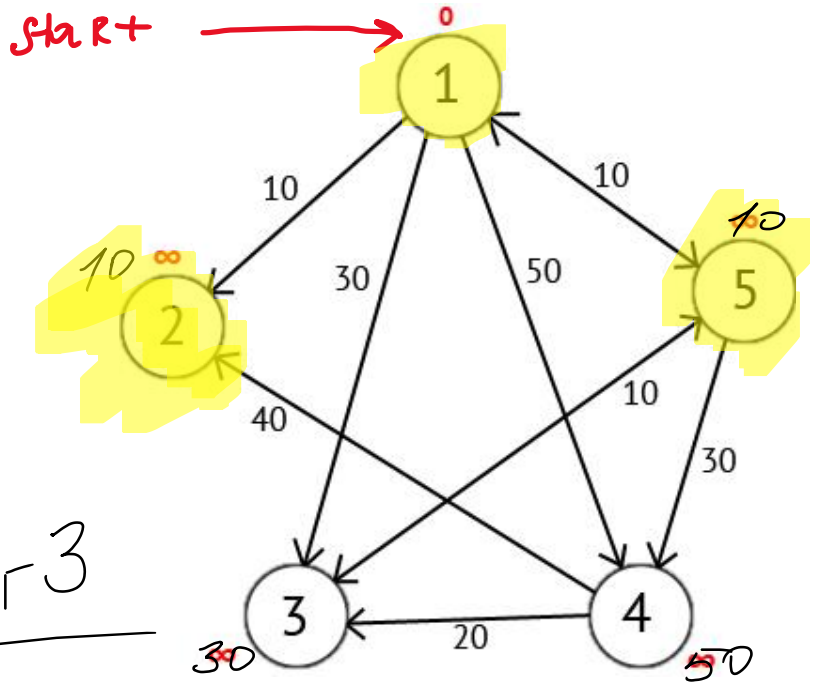
Суть:

1. Каждую итерацию выбираем вершину, до которой кратчайший путь и она не помечена

2. Помечаем выбранную вершину и просматриваем все ребра из нее (каждым пытаем ослабить кратчайшие пути)

От (1) до всех!

	1	2	3	4	5
distance	0	10	20	40	10
predki	null	1	5	5	1
used	1	1	0	0	1



Шаг 3

АЛГОРИТМ ДЕЙКСТРЫ

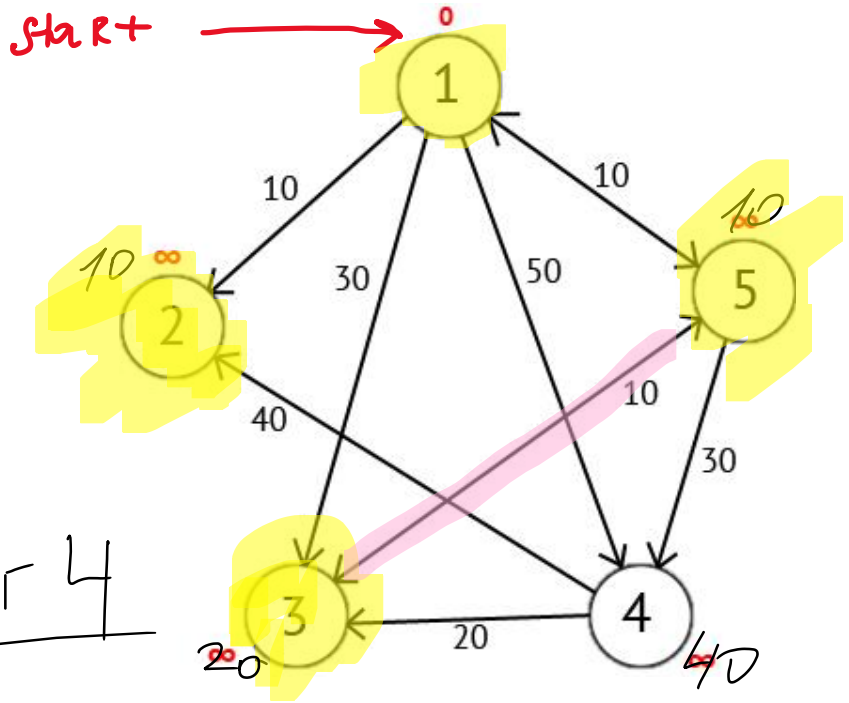
Суть:

1. Каждую итерацию выбираем вершину, до которой кратчайший путь и она не помечена

2. Помечаем выбранную вершину и просматриваем все ребра из нее (каждым пытаем ослабить кратчайшие пути)

От (1) до всех!

	1	2	3	4	5
distance	0	10	20	40	10
predki	null	1	5	5	1
used	1	1	1	0	1



Шаг 4

Выбираем min d[3] = 20

Relax (5, 3, 10) т.к 5 помечена!

АЛГОРИТМ ДЕЙКСТРЫ

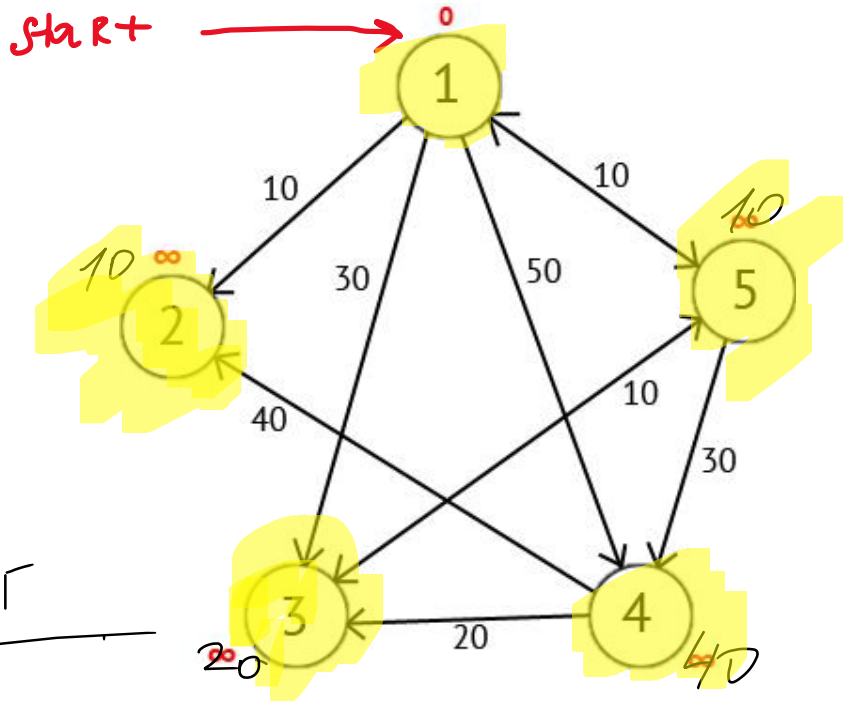
Суть:

1. Каждую итерацию выбираем вершину, до которой кратчайший путь и она не помечена

2. Помечаем выбранную вершину и просматриваем все ребра из нее (каждым пытаем ослабить кратчайшие пути)

От 1 до всех!

	1	2	3	4	5
distance	0	10	20	40	10
predki	null	1	5	5	1
used	1	1	1	1	1



ШАГ 1

Выбираем min d[4] = 40

АЛГОРИТМ ДЕЙКСТРЫ

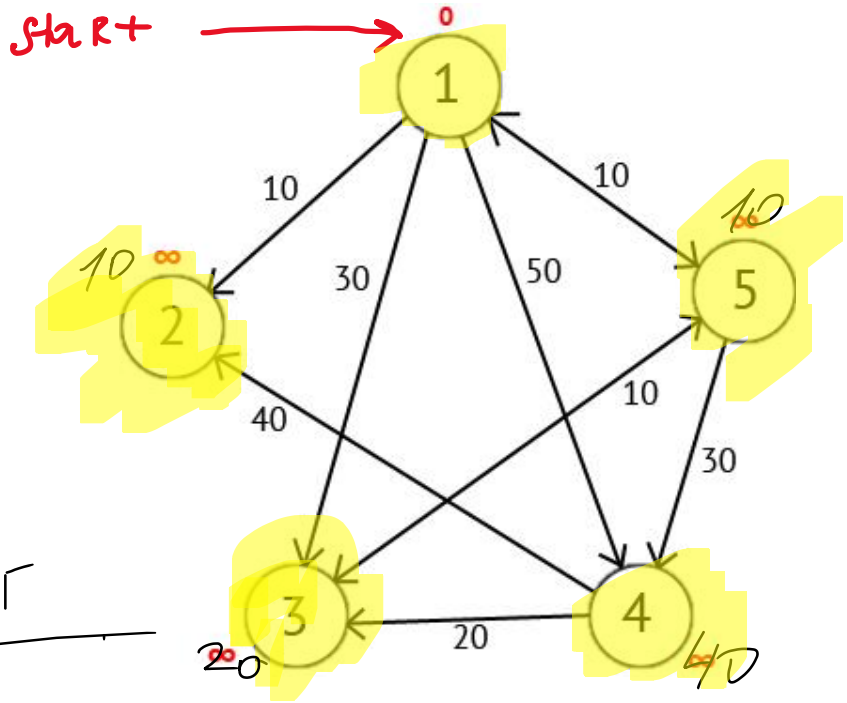
Суть:

1. Каждую итерацию выбираем вершину, до которой кратчайший путь и она не помечена

2. Помечаем выбранную вершину и просматриваем все ребра из нее (каждым пытаем ослабить кратчайшие пути)

От (1) до всех!

	1	2	3	4	5
distance	0	10	20	40	10
predki	null	1	5	5	1
used	1	1	1	1	1



ШАГ

Выбираем min d[4] = 40

→ relax (X)

т.к. все помечены

АЛГОРИТМ ДЕЙКСТРЫ

Суть:

1. Каждую итерацию выбираем вершину, до которой кратчайший путь и она не помечена

2. Помечаем выбранную вершину и просматриваем все ребра из нее (каждым пытаем ослабить кратчайшие пути)

Какой обход в основе?

Как хранить граф ?

Какие нужны структуры вспомогательные?

АЛГОРИТМ ДЕЙКСТРЫ

Суть:

1. Каждую итерацию выбираем вершину, до которой кратчайший путь и она не помечена

2. Помечаем выбранную вершину и просматриваем все ребра из нее (каждым пытаем ослабить кратчайшие пути)

Какой обход в основе?

- обход в ширину

Как хранить граф ?

- взвешенная матрица смежности

Какие нужны структуры вспомогательные?

- массивы: меток вершин, расстояний, предков

АЛГОРИТМ ДЕЙКСТРЫ

Суть:

1. Каждую итерацию выбираем вершину, до которой кратчайший путь и она не помечена

2. Помечаем выбранную вершину и просматриваем все ребра из нее (каждым пытаем ослабить кратчайшие пути)

Какой обход в основе?

- обход в ширину

Как хранить граф ?

- взвешенная матрица смежности

Какие нужны структуры вспомогательные?

- массивы: меток вершин, расстояний, предков

Точка оптимизации?

- поиск вершины с минимальным расстояние среди непомеченных

АЛГОРИТМ ДЕЙКСТРЫ

Суть:

1. Каждую итерацию выбираем вершину, до которой кратчайший путь и она не помечена
2. Помечаем выбранную вершину и просматриваем все ребра из нее (каждым пытаем ослабить кратчайшие пути)

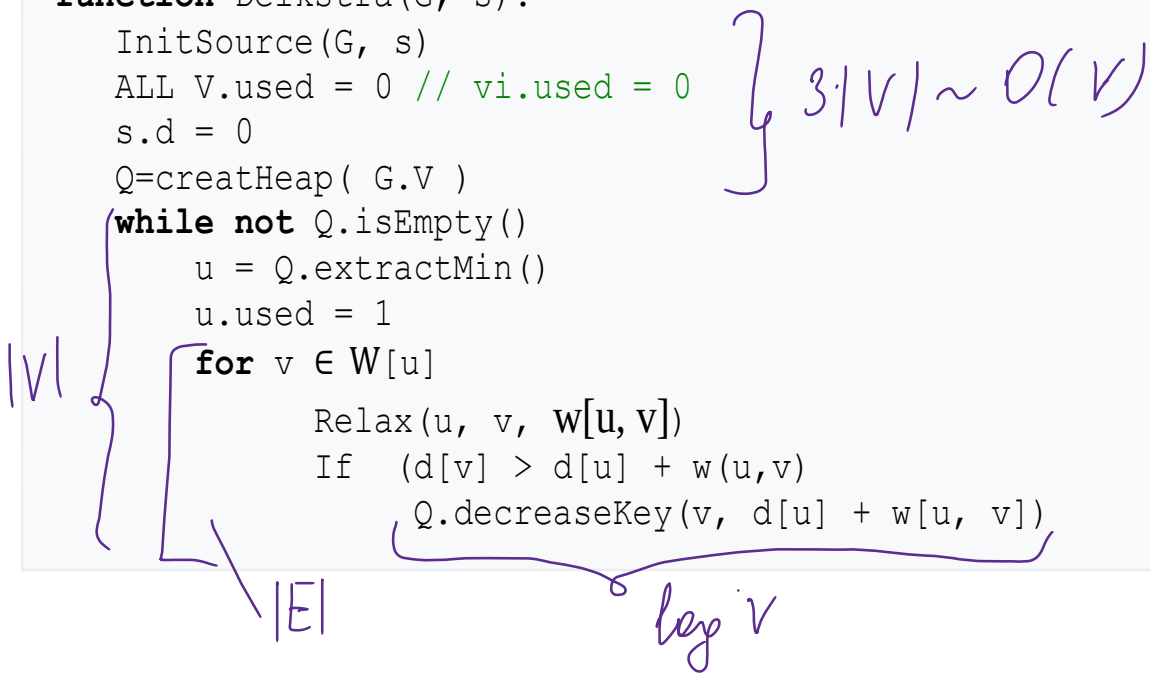
```
// G – исходный граф
// W – взвешенная матрица смежности
// Q - куча с минимумом (приоритетная очередь)
function Deikstra(G, s):
   InitSource(G, s)
   ALL V.used = 0 // vi.used = 0
   s.d = 0
   Q=creatHeap( G.V )
   while not Q.isEmpty()
       u = Q.extractMin()
       u.used = 1
       for v ∈ W[u]
             Relax(u, v, w[u, v])
             If  (d[v] > d[u] + w(u,v)
                  Q.decreaseKey(v, d[u] + w[u, v])
```

АЛГОРИТМ ДЕЙКСТРЫ

Суть:

1. Каждую итерацию выбираем вершину, до которой кратчайший путь и она не помечена
2. Помечаем выбранную вершину и просматриваем все ребра из нее (каждым пытаем ослабить кратчайшие пути)

```
// G – исходный граф
// W – взвешенная матрица смежности
// Q - куча с минимумом (приоритетная очередь)
function Deikstra(G, s):
    InitSource(G, s)
    ALL V.used = 0 // vi.used = 0
    s.d = 0
    Q=creatHeap( G.V )
    while not Q.isEmpty()
        u = Q.extractMin()
        u.used = 1
        for v ∈ W[u]
            Relax(u, v, w[u, v])
            If  (d[v] > d[u] + w(u,v)
                Q.decreaseKey(v, d[u] + w[u, v])
```



АЛГОРИТМ ДЕЙКСТРЫ

Суть:

1. Каждую итерацию выбираем вершину, до которой кратчайший путь и она не помечена
2. Помечаем выбранную вершину и просматриваем все ребра из нее (каждым пытаем ослабить кратчайшие пути)

```
// G – исходный граф
// W – взвешенная матрица смежности
// Q - куча с минимумом (приоритетная очередь)
function Deikstra(G, s):
    InitSource(G, s)
    ALL V.used = 0 // vi.used = 0
    s.d = 0
    Q=creatHeap( G.V )
    while not Q.isEmpty()
        u = Q.extractMin()
        u.used = 1
        for v ∈ W[u]
            Relax(u, v, w[u, v])
            If  (d[v] > d[u] + w(u,v)
                Q.decreaseKey(v, d[u] + w[u, v])
```

$3|V| \sim O(V)$

$|V|$

$|E|$

$\log V$

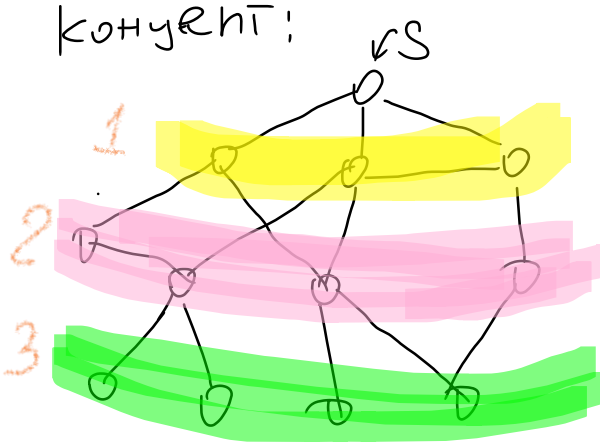
$O((V+E)\cdot \log V) = O(E \log V)$

ЛЕММА о BFS

BFS в невзвешенном графе находит длины кратчайших путей до всех достижимых вершин

ЛЕММА о BFS

BFS в невзвешенном графе находит длины кратчайших путей до всех достижимых вершин

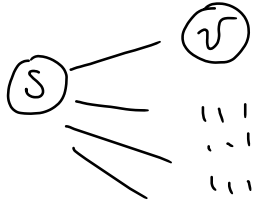


ЛЕММА о BFS

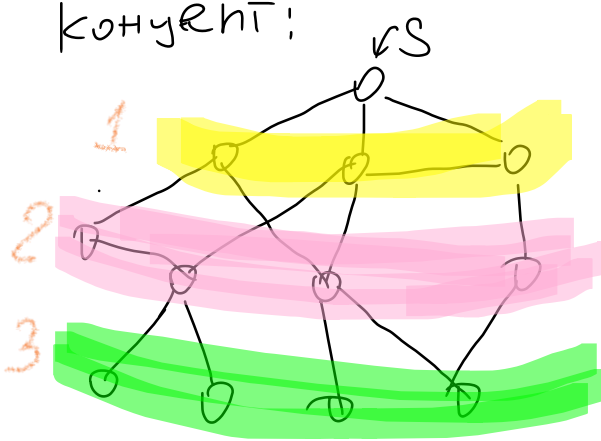
BFS в невзвешенном графе находит длины кратчайших путей до всех достижимых вершин

Prof! ← рассмотр. i-ый вызов от s

т.е. смешные (s)

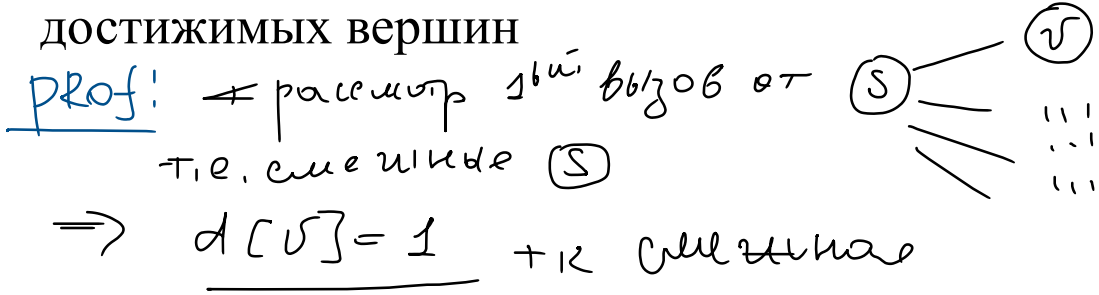


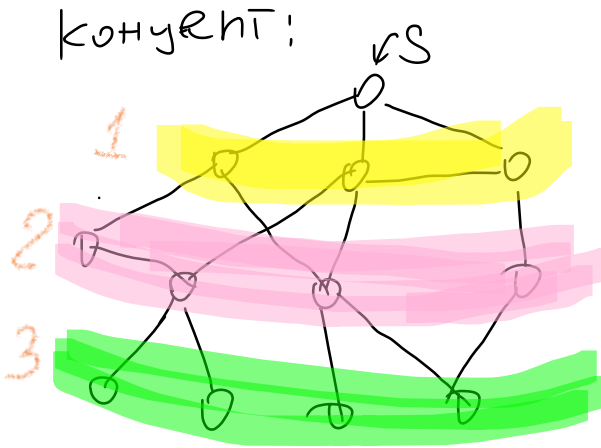
конкретно:



ЛЕММА о BFS

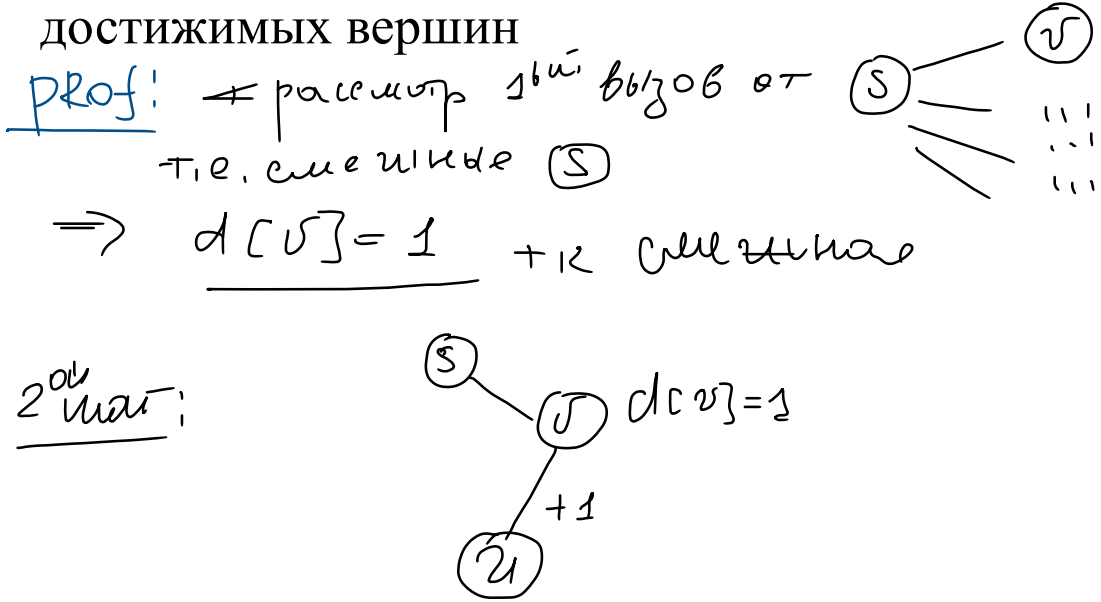
BFS в невзвешенном графе находит длины кратчайших путей до всех достижимых вершин

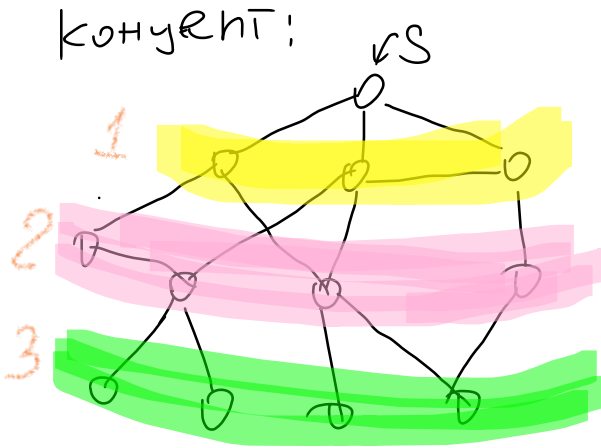




ЛЕММА о BFS

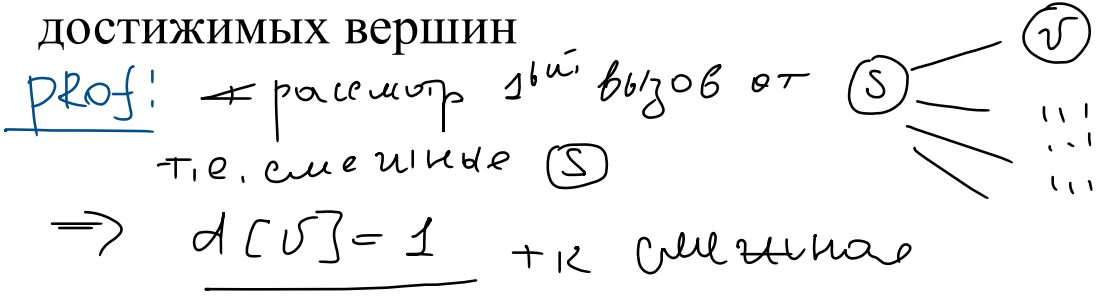
BFS в невзвешенном графе находит длины кратчайших путей до всех достижимых вершин

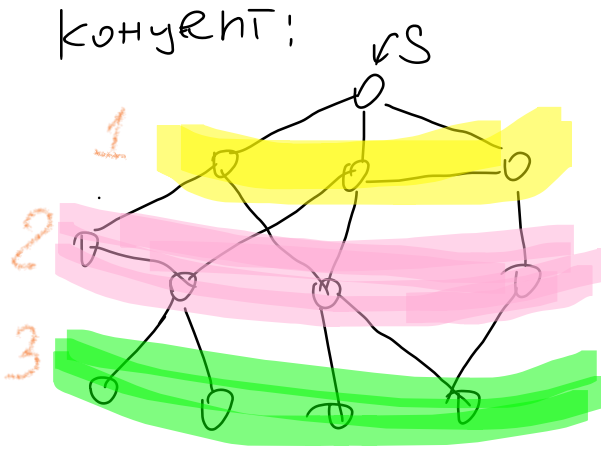


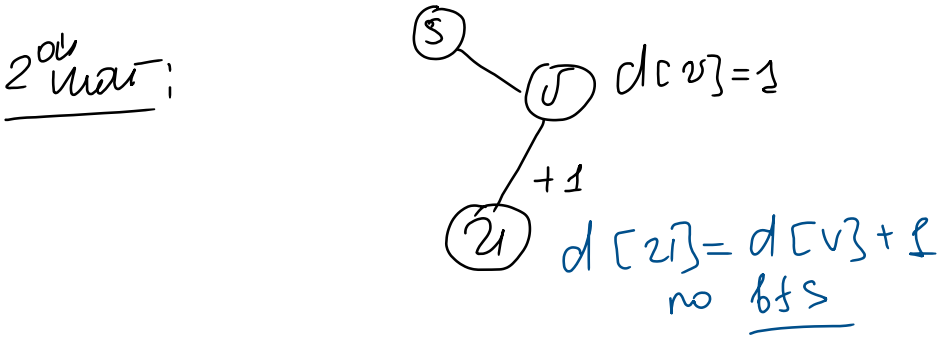


ЛЕММА о BFS

BFS в невзвешенном графе находит длины кратчайших путей до всех достижимых вершин

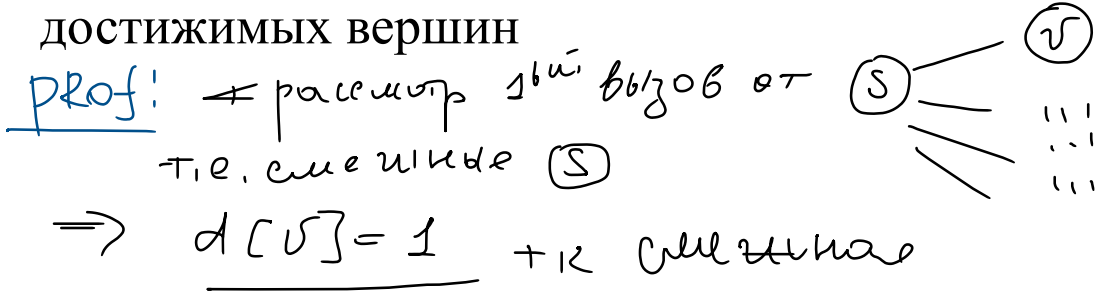


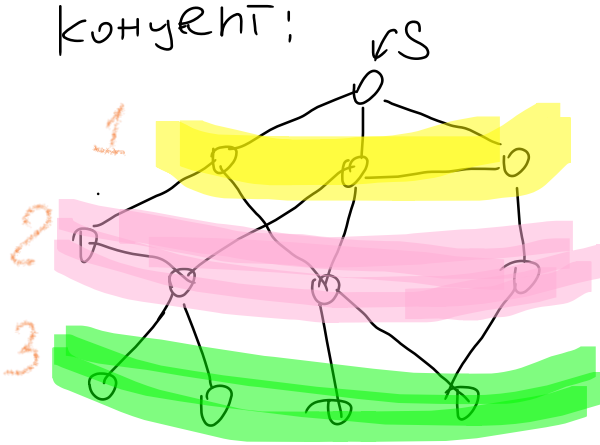




ЛЕММА о BFS

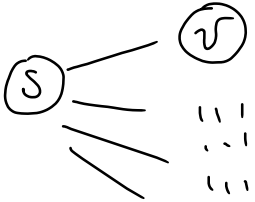
BFS в невзвешенном графе находит длины кратчайших путей до всех достижимых вершин





ЛЕММА о BFS

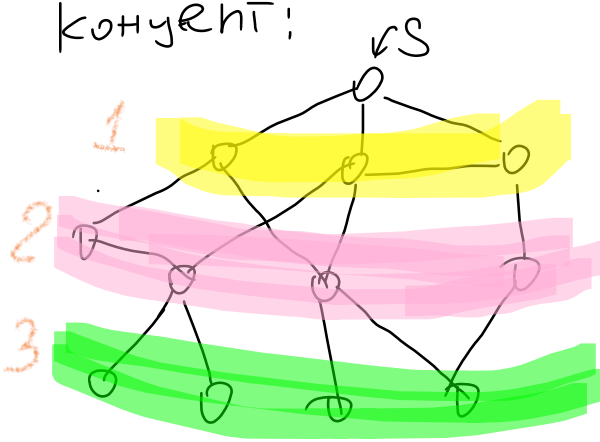
BFS в невзвешенном графе находит длины кратчайших путей до всех достижимых вершин



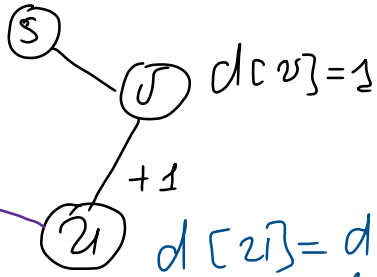
prof: рассмотр 1ый вызов от s

т.е. смежные s

$\Rightarrow d[v] = 1$ т.к. смежная



2ой мат:



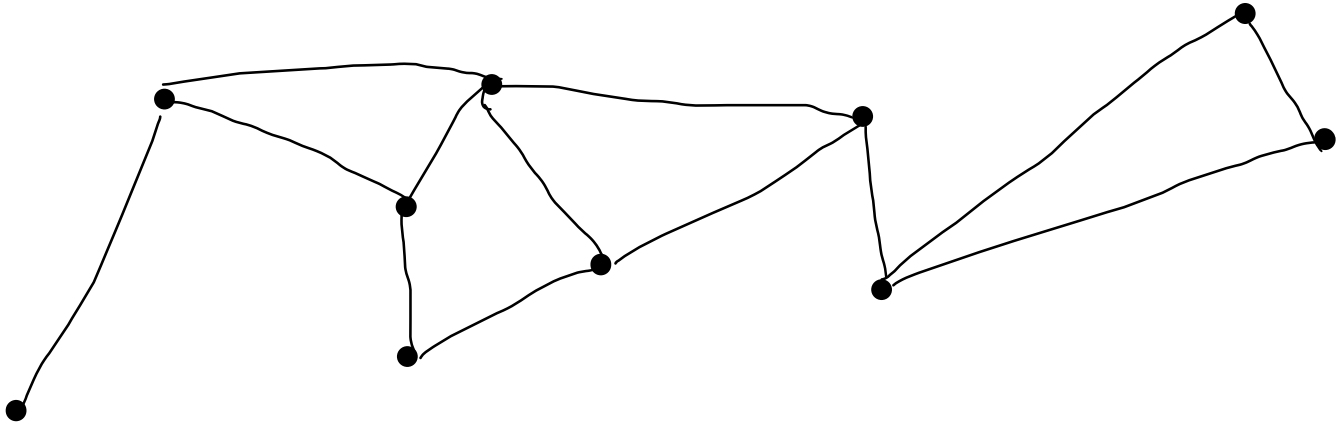
$\exists w: d[w] + 1$

$d[z] = d[v] + 1$ по bfs

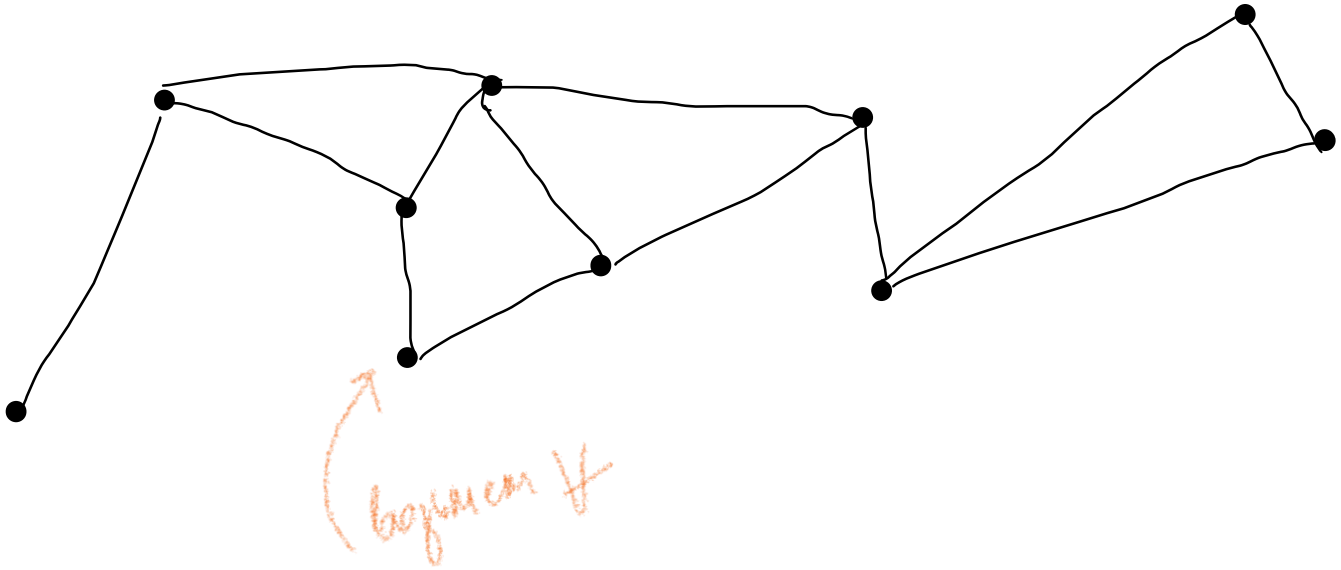
$d[w]+1 < d[v]+1$

$\Rightarrow d[w] < d[v] \Rightarrow$ w g.б до v, но это невозможно

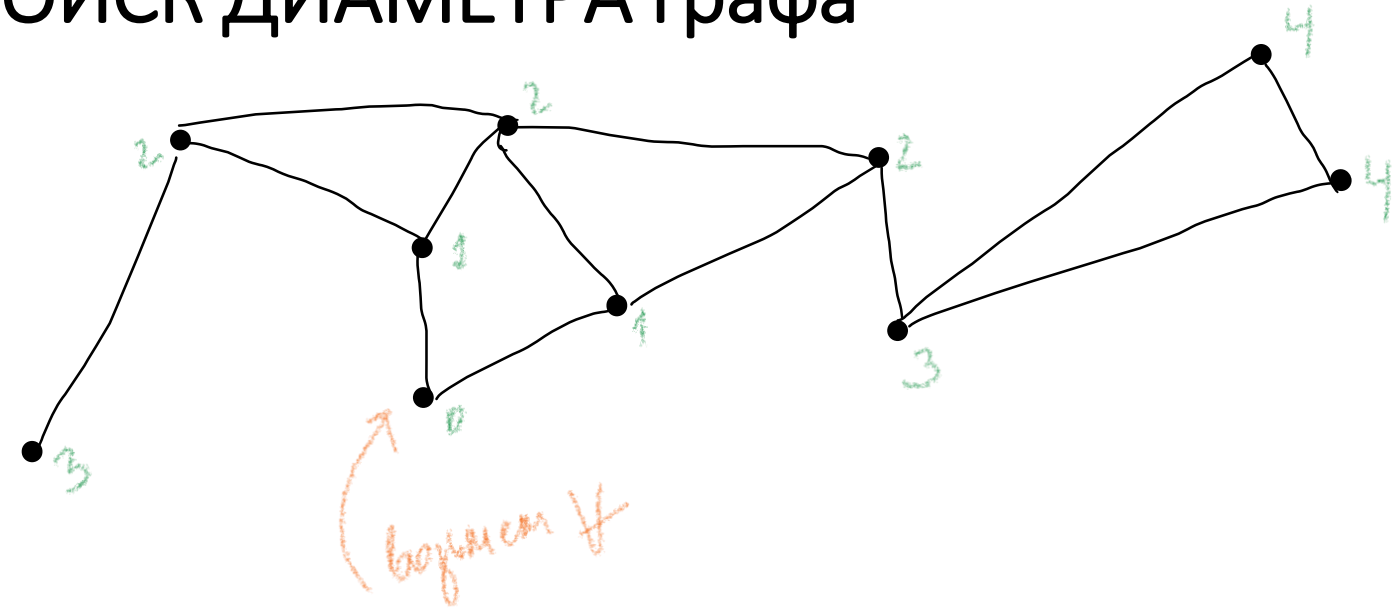
ПОИСК ДИАМЕТРА графа



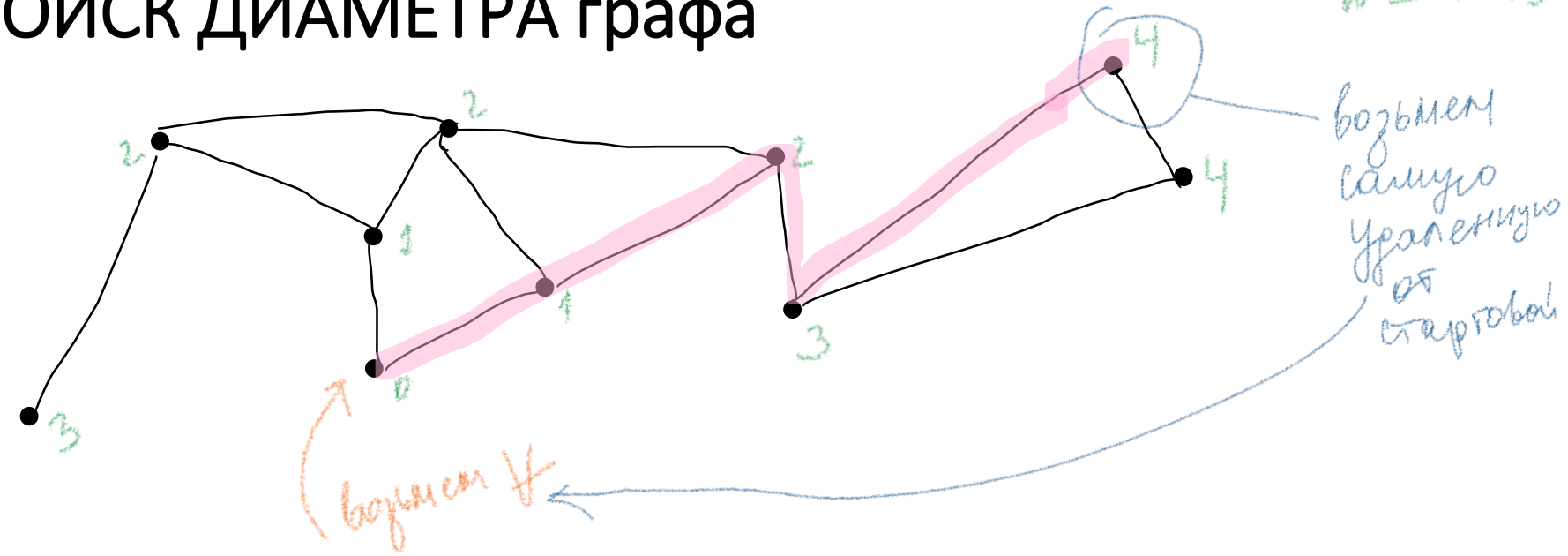
ПОИСК ДИАМЕТРА графа



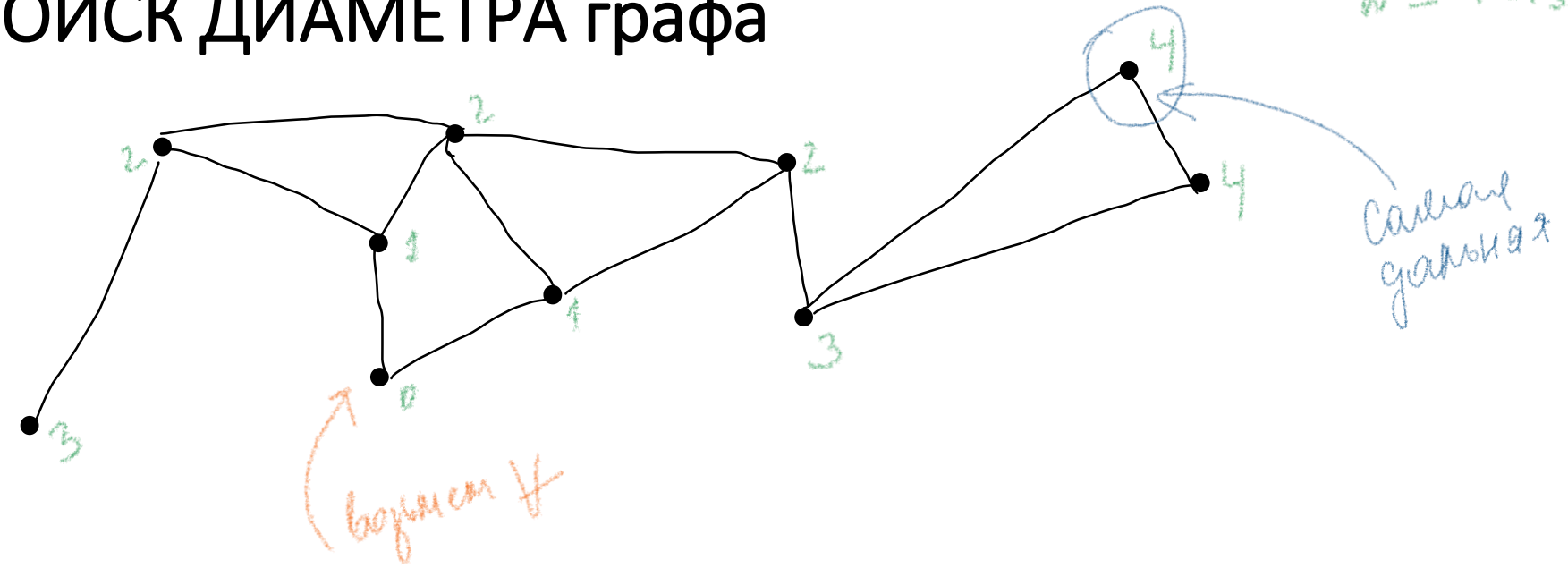
ПОИСК ДИАМЕТРА графа



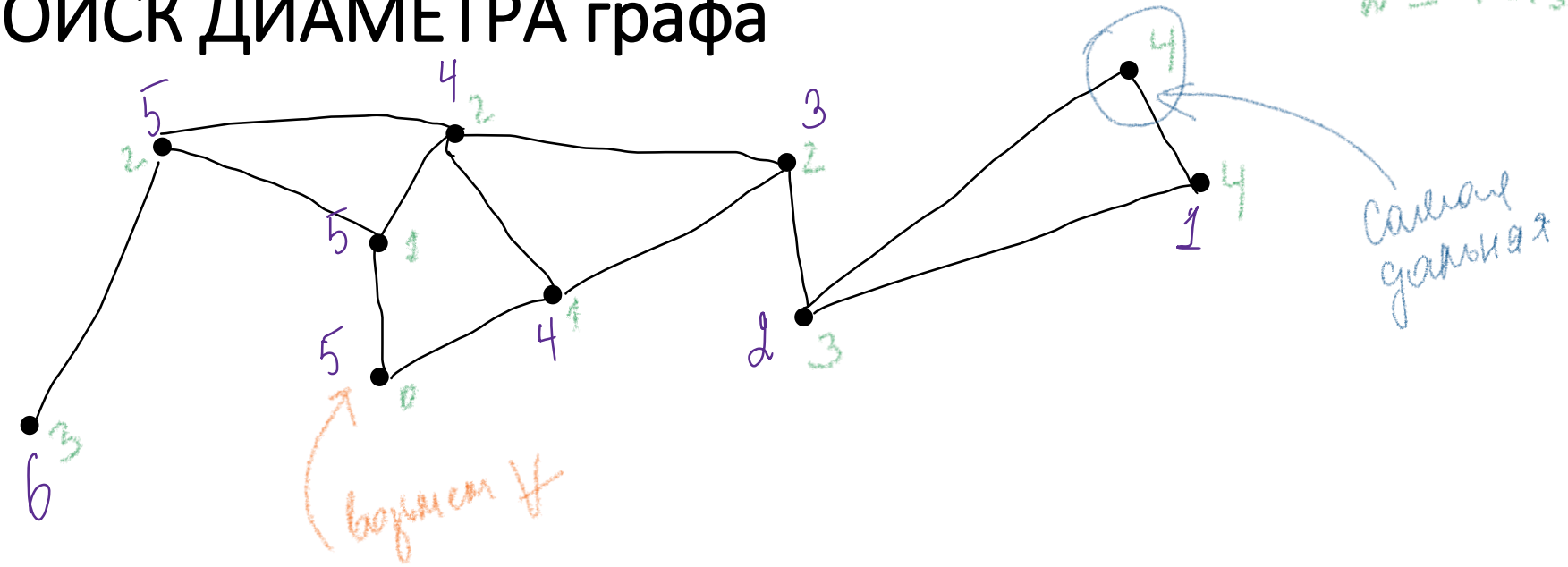
ПОИСК ДИАМЕТРА графа



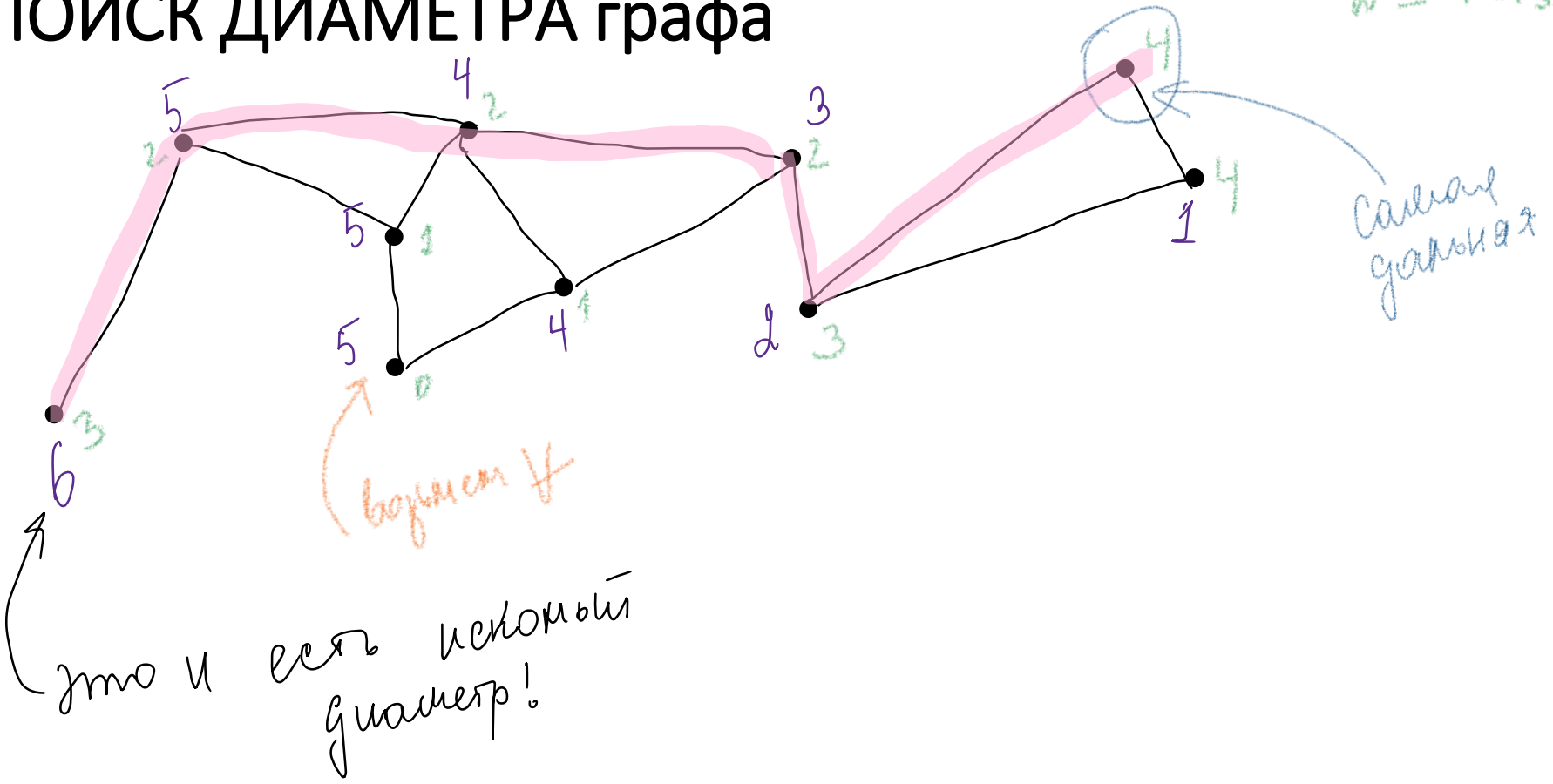
ПОИСК ДИАМЕТРА графа



ПОИСК ДИАМЕТРА графа



ПОИСК ДИАМЕТРА графа



ПОИСК ДИАМЕТРА графа

Алгоритм:

1. Посчитать кратчайшие пути из любой вершины (невзвешенные) bfs 1
2. Найти самую удалённую вершину от выбранной на шаге 1 max
3. От неё найти все кратчайшие пути (невзвешенные) bfs 2
4. Выбрать максимальны: самый длинный из них - **искомый диаметр** max

$O(|V| + |E|) \sim$ bfs

Спасибо за внимание!

www.ifmo.ru