Лекция 3
Графы

Способы задания графов. Обобщенный поиск. Поиск в ширину. Поиск в глубину. Топологическая сортировка, алгоритм Касарайю. Структура всемирной паутины.

Граф G(V, E) V-вершины (vertex), E – ребра (edge)

Способы задания:

1) Матрица смежности $A = (a_{ij})$

$$a_{ij} = \begin{cases} 1, \text{если есть ребро } (v_i, v_j) \\ 0, \text{в противном случае} \end{cases}$$

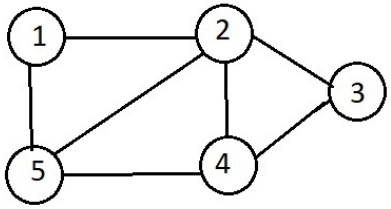
хранение (пространственная сложность) $O(|V|^2)$

2) Списки смежности: любой вершине v соответствует список смежных с ней вершин. $O(|V| + |E|)$

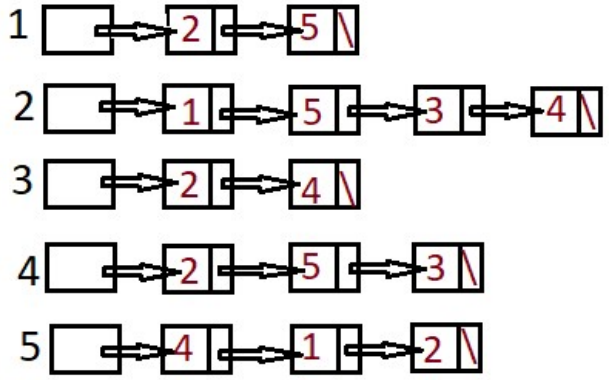
(1)

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

Пример:



(2)



Поиск в графе

Дан граф **G(V, E)** и стартовая вершина s.

Необходимо идентифицировать вершины, достижимые из s.

ОБОБЩЕННЫЙ ПОИСК (GENERICSEARCH)

Вход: граф $G = (V, E)$ и вершина $s \in V$.

Постусловие: вершина достижима из s тогда и только тогда, когда она помечена как «разведанная».

пометить вершину s как разведанную, все остальные вершины как неразведанные

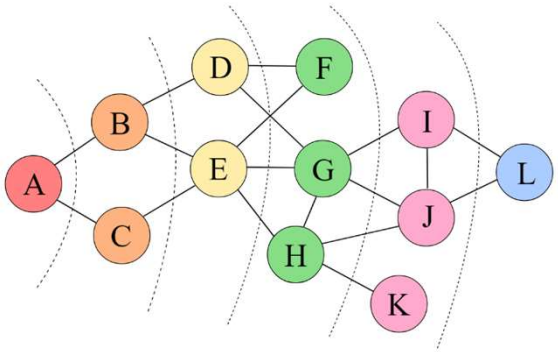
while существует ребро $(v, w) \in E$ с разведанной v и неразведанной w **do**

выбрать несколько таких ребер (v, w) // конкретизировано // неполно → Поиск в ширину BFS / Поиск в глубину DFS

пометить w как разведанную

Утв. (Правильность обобщенного поиска)
По завершении обобщенного поиска $v \epsilon V$ помечена как *разведанная* ⇔ ∃ путь из s в v

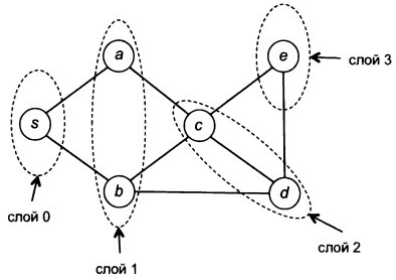
Поиск в ширину (BFS - breadth-first search)



Поиск в ширину обнаруживает вершины слоями.

Вершины i-го слоя являются соседями вершин *(i-1)*-го слоя, которые не появлялись ни в одном более раннем слое.

Время работы: $O(|V| + |E|)$



BFS

Вход: граф $G = (V, E)$, представленный в виде списков смежности, и вершина $s \in V$.

Постусловие: вершина достижима из s тогда и только тогда, когда она помечена как «разведанная».

---

1) пометить s как разведанную вершину, все остальные как не-разведанные
2) Q := очередь, инициализированная вершиной s
3) **while** Q не является пустой **do**
4) удалить вершину из начала Q, назвать ее v
5) **for** каждое ребро (v, w) в списке смежности v **do**
6) **if** w не разведана **then**
7) пометить w как разведанную
8) добавить w в конец Q

Определение кратчайшего пути *dist(s, v)* от s до любой достижимой вершины

BFS

Вход: граф $G = (V, E)$, представленный в виде списков смежности, и вершина $s \in V$.

Постусловие: вершина достижима из s тогда и только тогда, когда она помечена как «разведанная».

---

1) пометить s как разведанную вершину, все остальные как неразведанные
2) Q := очередь, инициализированная вершиной s
3) **while** Q не является пустой **do**
4) удалить вершину из начала Q, назвать ее v
5) **for** каждое ребро (v, w) в списке смежности v **do**
6) **if** w не разведана **then**
7) пометить w как разведанную
8) добавить w в конец Q

ДОПОЛНЕННЫЙ ПОИСК В ШИРИНУ (AUGMENTED-BFS)

Вход: граф $G = (V, E)$, представленный в виде списков смежности, и вершина $s \in V$.

Постусловие: для каждой вершины $v \in V$ значение $l(v)$ равно истинному расстоянию кратчайшего пути $dist(s, v)$.

---

1) пометить s как разведанную вершину, все остальные как неразведанные
2) $l(s) := 0$, $l(v) := +\infty$ для каждой $v \neq s$
3) Q := очередь, инициализированная вершиной s
4) **while** Q не является пустой **do**
5) удалить вершину из начала Q, назвать ее v
6) **for** каждое ребро (v, w) в списке смежности вершины v **do**
7) **if** w не разведана **then**
8) пометить w как разведанную
9) $l(w) := l(v) + 1$
10) добавить w в конец Q

Теорема 1: $\forall G(V, E)$, представленного в виде списков смежности и $\forall\, s \in V$

- а) по завершении алгоритма AUGMENTED-BFS $\forall\, v \in V\ l(v) = dist(s, v)$ (кратчайшее расстояние);
- б) время работы $O(|V| + |E|)$

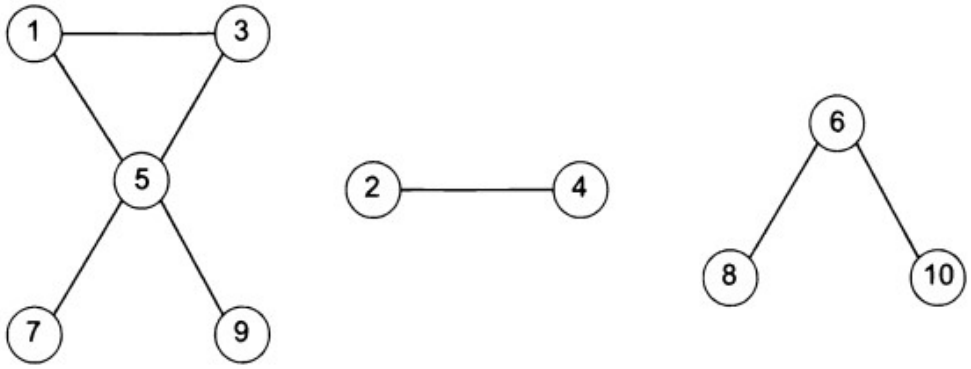
Доказательство:

а) Во-первых, вершины v, где $dist\ (s, v) = i$, являются как раз вершинами в i-м слое графа — вот почему мы определили слои именно так, а не иначе. Во-вторых, для каждой вершины w слоя i алгоритм Augmented-BFS в конечном счете устанавливает $l(w) = i$ (так как w обнаруживается посредством вершины v слоя $(i - 1)$, где $l(v) = i - 1$). Для вершин ни в одном из слоев, то есть не достижимых из s, как $dist(s, v)$, так и $l(v)$ равны $+\infty$[1].

б) очевидно.

Нахождение компонент связности в неориентированном графе (CC - Connectivity Component)

Компонента связности – это максимальное подмножество $S \subseteq V$ вершин, так что существует путь из любой вершины из S в любую другую вершину из S.



Граф из 10 вершин. 3 компоненты связности.

Вычисление компонент связности на основе BFS.
cc(v) – номер компоненты связности

BFS

Вход: граф $G = (V, E)$, представленный в виде списков смежности, и вершина $s \in V$.

Постусловие: вершина достижима из s тогда и только тогда, когда она помечена как «разведанная».

---

1) пометить s как разведанную вершину, все остальные как не-разведанные
2) Q := очередь, инициализированная вершиной s
3) **while** Q не является пустой **do**
4) удалить вершину из начала Q, назвать ее v
5) **for** каждое ребро (v, w) в списке смежности v **do**
6) **if** w не разведана **then**
7) пометить w как разведанную
8) добавить w в конец Q

UCC

Вход: неориентированный граф $G = (V, E)$, представленный в виде списков смежности, где $V = \{1,2,3, ..., n\}$.

Постусловие: для каждой $u, v \in V$, $cc(u) = cc(v)$ тогда и только тогда, когда u, v находятся в одной и той же связной компоненте.

пометить все вершины как неразведанные

numCC := 0

for *i* := от 1 до *n* **do** // перебрать все вершины

if i не разведана **then** // избежать избыточности

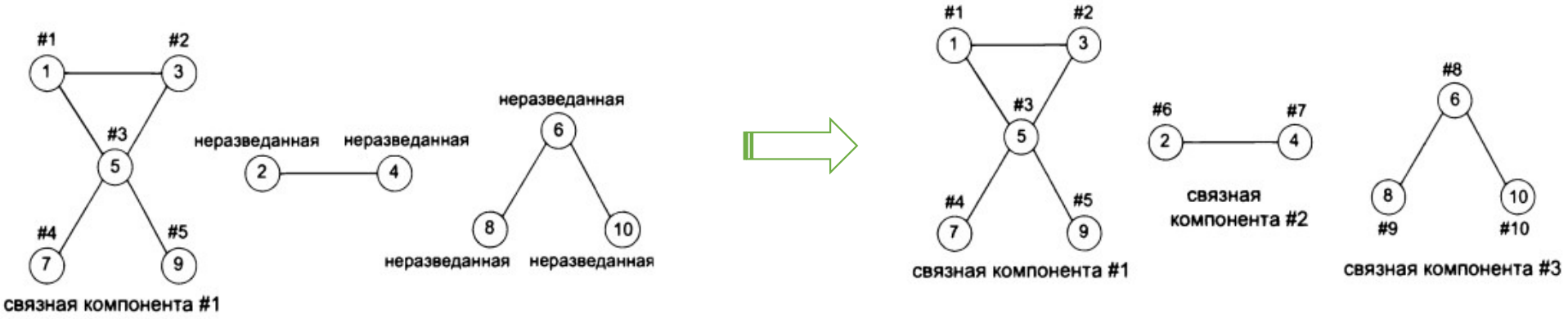
Пометить *i* как разведанную

numCC := *numCC* + 1 // новая компонента

```
// вызвать алгоритм BFS, начиная с i (строки 2-8)
Q := очередь, инициализированная значением i
while Q не является пустой do
  удалить вершину из начала Q, назвать ее v
  cc(v) := numCC
  for каждая (v, w) в списке смежности вершины v do
    if w не разведана then
      пометить w как разведанную,
      добавить w в конец Q
```

Используем внешний цикл для выполнения одного обхода вершин. BFS используем в качестве подпрограммы.

Пример нахождения компонент связности



Теорема **(Свойства алгоритма UCC.)** *Для каждого неориентированного графа* $G = (V, E)$, *представленного в виде списков смежности:*

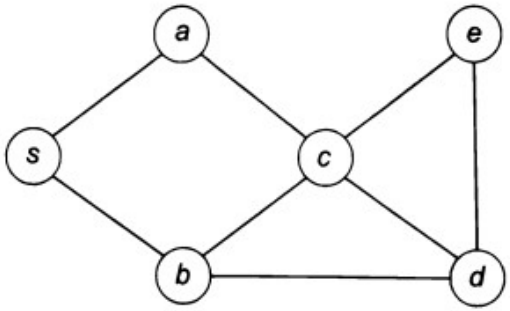
а) по завершении алгоритма UCC *для каждой пары* u, v *вершин* $cc(u) = cc(v)$ *тогда и только тогда, когда* u *и* v *принадлежат одной и той же связной компоненте графа* G;

б) время работы алгоритма UCC *равно* $O(|V| + |E|)$.

Поиск в глубину (DFS - Depth-first search)

Идти «вглубь» графа, насколько это возможно. Алгоритм поиска описывается рекурсивно: перебираем все исходящие из рассматриваемой вершины рёбра. Если ребро ведёт в вершину, которая не была рассмотрена ранее, то запускаем алгоритм от этой нерассмотренной вершины, а после возвращаемся и продолжаем перебирать рёбра. Возврат происходит в том случае, если в рассматриваемой вершине не осталось рёбер, которые ведут в нерассмотренную вершину. Если после завершения алгоритма не все вершины были рассмотрены, то необходимо запустить алгоритм от одной из нерассмотренных вершин.

Пример (тот же, что для BFS):



1) s – разведанная, остальные - неразведанные

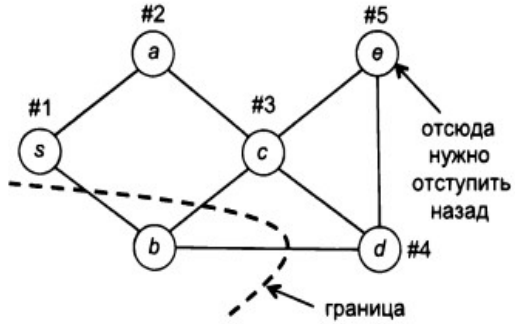
2) $s \rightarrow a, a$ - разведанная

3) $a \rightarrow c, c$ - разведанная

4) $c \rightarrow d, d$ - разведанная

5) $d \rightarrow e, e$ - разведанная

6) из e – нет неразведанн



идем назад в $d, d \rightarrow b, b$ - разведанная

Алгоритм DFS (псевдокод)

DFS (ИТЕРАТИВНАЯ ВЕРСИЯ)

Вход: граф $G = (V, E)$, представленный в виде списков смежности, и вершина $s \in V$.

Постусловие: вершина достижима из s тогда и только тогда, когда она помечена как «разведанная».

пометить все вершины как неразведанные

S := стек, инициализированный вершиной s

while S не является пустым **do**

  удалить (вытолкнуть) вершину v из головы стека S

  if v не разведана **then**

    пометить v как разведанную

    for каждое ребро (v, w) в списке смежности вершины v **do**

      добавить (втолкнуть) w в голову стека S

DFS (РЕКУРСИВНАЯ ВЕРСИЯ)

Вход: граф $G = (V, E)$, представленный в виде списков смежности, и вершина $s \in V$.

Постусловие: вершина достижима из s тогда и только тогда, когда она помечена как «разведанная».

```
// перед внешним вызовом все вершины не разведаны
```

пометить s как разведанную

for каждое ребро (s, v) в списке смежности вершины s **do**

  if v не разведана **then**

    DFS (G, v)

Теорема 2. Для каждого ориентированного или неориентированного графа $G(V, E)$, представленного в виде списков смежности и стартовой вершины $s \epsilon V$

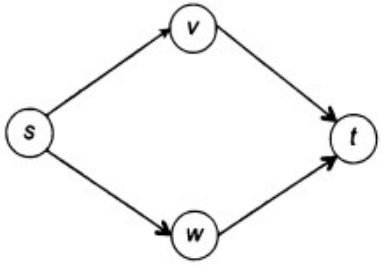
а) по завершении алгоритма DFS *вершина* $v \in V$ *помечается как разведанная тогда и только тогда, когда в* G *существует путь из* s *в* v;

б) время работы алгоритма DFS *равно* $O(m + n)$, *где* $m = |E|$ *и* $n = |V|$.

Доказательство:
(а) см. обобщенный поиск;
(б) инициализация – $O(n)$, *каждое ребро не более 2 раза*

Топологическая сортировка (в ориентированном графе)

$\forall v \in V$ определим $f(v)$: $\forall$ребра $(v, w) \in E$ $f(v) < f(w)$



Примеры упорядочений

s, w, v, t
f_1 : 1, 2, 3, 4

s, v, w, t
f_2 : 1, 2, 3, 4

Топологическую сортировку можно проводить только в ориентированных ациклических графах

Для любого такого графа существует минимально
одно топологическое упорядочение, т.к. каждый орграф
имеет минимально 1 исток.
(пойдем обратно из любой вершины). Находим и отсекаем исток и все го ребра. В оставшемся пять ищем исток и т.д. Получаем упорядочение.

DFS и топологическая сортировка

DFS (РЕКУРСИВНАЯ ВЕРСИЯ)

Вход: граф $G = (V, E)$, представленный в виде списков смежности, и вершина $s \in V$.

Постусловие: вершина достижима из s тогда и только тогда, когда она помечена как «разведанная».

```
// перед внешним вызовом все вершины не разведаны
```

пометить s как разведанную

for каждое ребро (s, v) в списке смежности вершины s **do**

  if v не разведана **then**

    DFS (G, v)

TOPOSORT

Вход: ориентированный ациклический граф $G = (V, E)$, представленный в виде списков смежности.

Постусловие: значения f вершин образуют топологическую упорядоченность графа G.

пометить все вершины как неразведанные

$curLabel := |V|$ `// отслеживает упорядочивание`

for каждая $v \in V$ **do**

  if v не разведана **then** `// в предыдущем DFS`

    `DFS-Topo` (G, v)

DFS-TOPO

Вход: граф $G = (V, E)$, представленный в виде списков смежности, и вершина $s \in V$.

Постусловие: каждая вершина, достижимая из s, помечается как «разведанная» и имеет присвоенное ей значение f.

пометить s как разведанную

for каждое ребро (s, v) в исходящем списке смежности s **do**

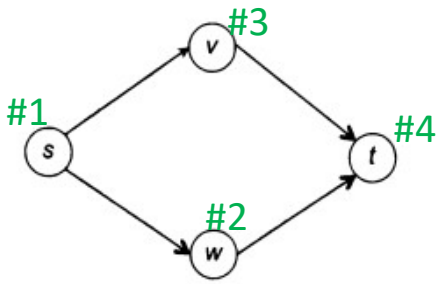
  if v не разведана **then**

    `DFS-Topo` (G, v)

$f(s) := curLabel$ `// позиция s в упорядочении`

$curLabel := curLabel - 1$ `// двигаться справа налево`

Пример: пусть вершины (для TOPOSORT) перечислены как v, t, s, w



Теорема. Для графа G(V, E), представленного в виде списков смежности

а) по завершении алгоритма `TopoSort` *каждой вершине v было присвоено значение f, и эти значения f образуют топологическое упорядочение графа G;*

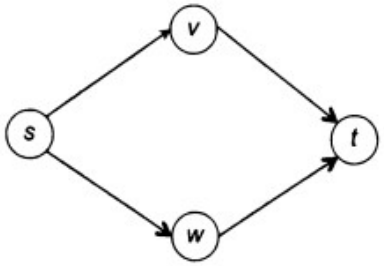
б) время работы алгоритма `TopoSort` *равно* $O(m + n)$, *где* $m = |E|$ *и* $n = |V|$.

Без доказательства.

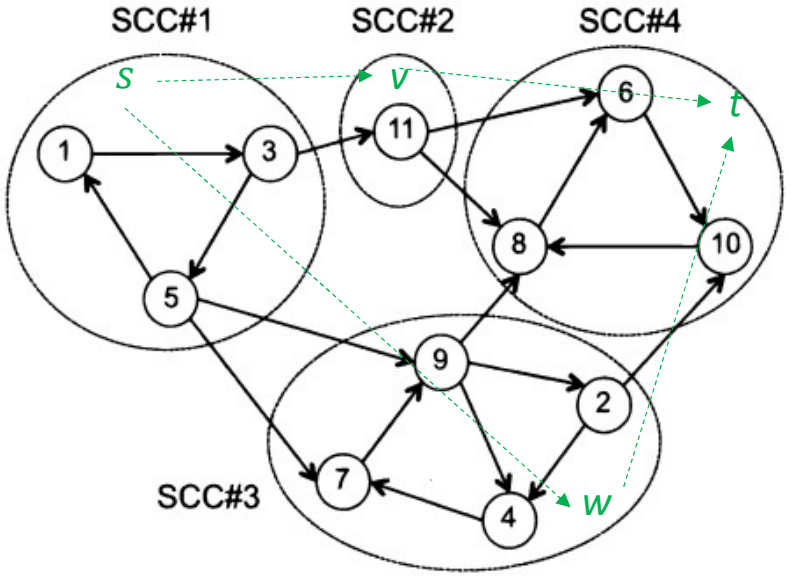
Нахождение сильно-связнных компонент

Сильно-связнная компонента (Strongly Connected Component – SCC) ориентированного графа – это максимальное подмножество $S \subseteq V$ такое, что существует путь из любой $v \in S$ в любую другую $v' \in S$.

Пример:



4 сильно-связных компоненты



метаграф

Сам граф не является ациклическим
Можно сортировать метавершины.

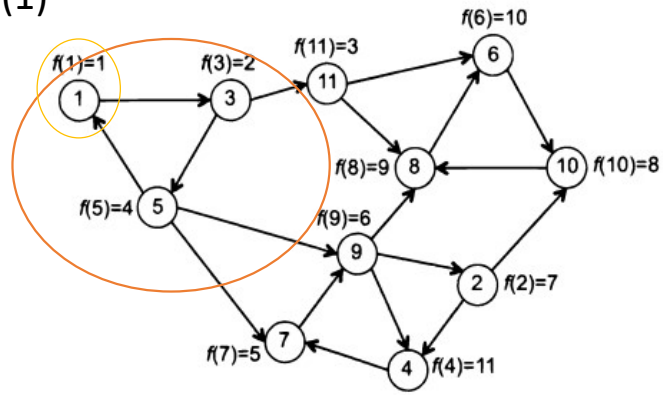
Как определить метавершины?

Используем топологическую сортировку на исходном графе.

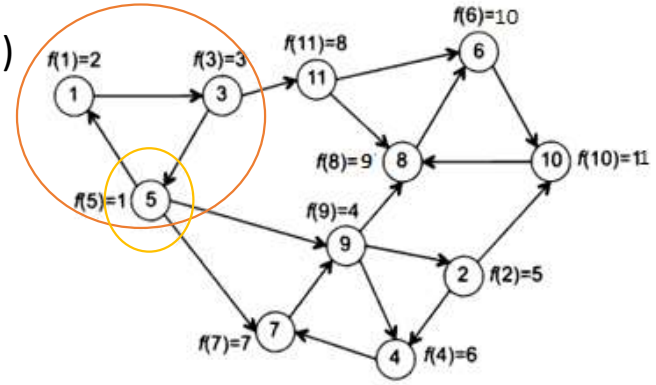
Сортируем вершины так, чтобы найти метавершину-сток. Найдем и удалим ее из графа. Повторим на оставшемся графе и т.д., пока не переберем все вершины.

Как найти сток? - Рассмотрим на G две различных сортировки:

(1)



(2)



В обоих случаях (порядка обработки вершин) вершина в первой позиции принадлежит истоковой компоненте SCC#1

Пометим SCC наименьшей позицией одной из ее вершин. Тогда метки будут образовывать топологическое упорядочение метавершин.

Теорема. Топологическое упорядочение сильно-связных компонент.

Пусть G – орграф, вершины которого произвольно упорядочены. $\forall v$ определена позиция $f(v)$ с помощью TOPOSORT. Пусть S_1 и S_2 -компоненты сильной связности, (v, w) - ребро, причем $v \in S_1$, $w \in S_2$.

Тогда $\min_{x \in S_1} f(x) < \min_{y \in S_2} f(y)$

Доказательство: возможны 2 случая:

а) TOPOSORT обнаруживает и инициирует поиск в глубину из $s \in S_1$ перед любой вершиной из S_2. Т.к. существует ребро из S_1 в S_2 и позиции назначаются в убывающем порядке, то номер вершины v из S_1 будет меньше любого номера из S_2 .

б) Если TOPOSORT сначала обнаруживает и разведывает вершины из S_2 , путь из S_2 в S_1 отсутствует, то номера вершин из S_2 будут больше, чем номера из S_1. ч. т. д.

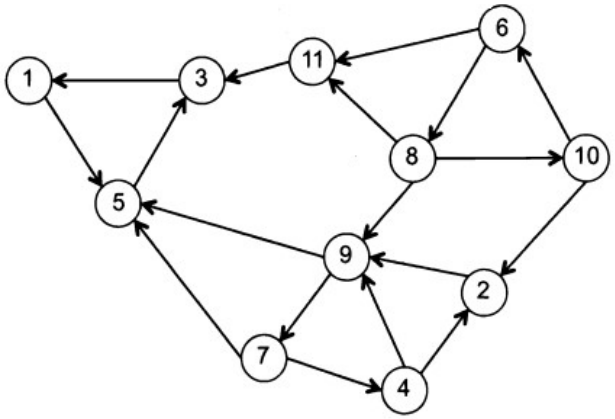
Следствие: вершина в первой позиции всегда лежит в истоковой компоненте.

РАЗВОРОТ ГРАФА

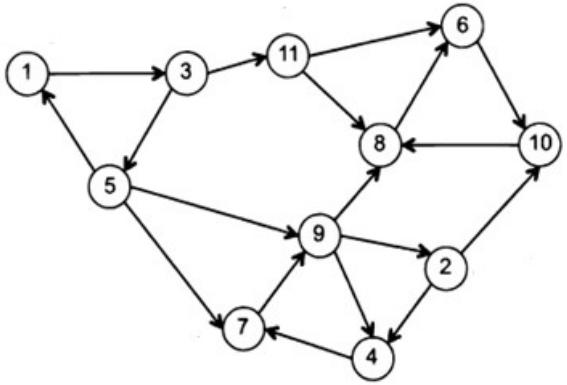
Развернем граф G в обратную сторону и запустим на G^R TOPOSORT, получив некоторое упорядочение. Найдем в G^R истоковую SCC. Она будет стоковой для исходного графа G.

Запустим на G TOPOSORT, перебирая вершины в найденном порядке. Будем находить и отсекать стоковые компоненты. Т.е. будем регистрировать метавершины в обратном топологическом порядке.

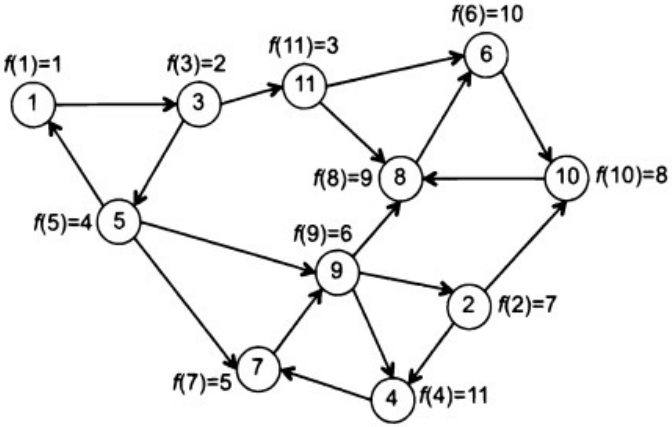
Пример: G



G^R

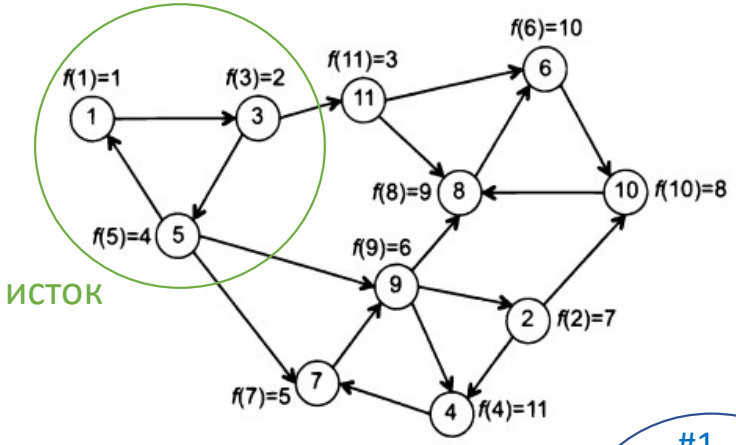


G^R после TOPOSORT

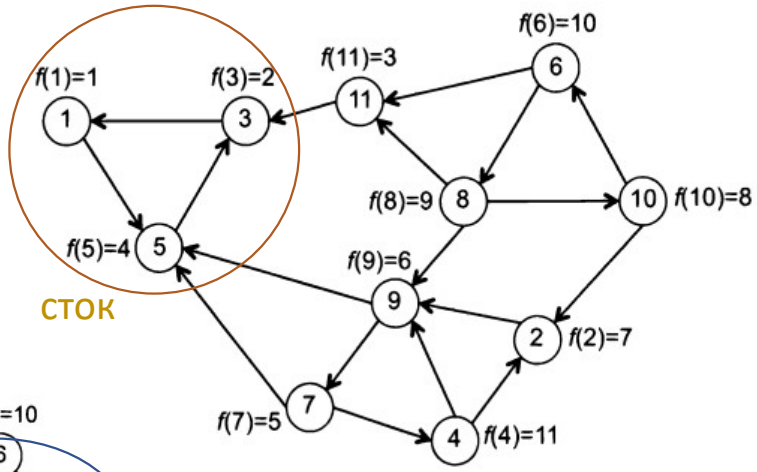


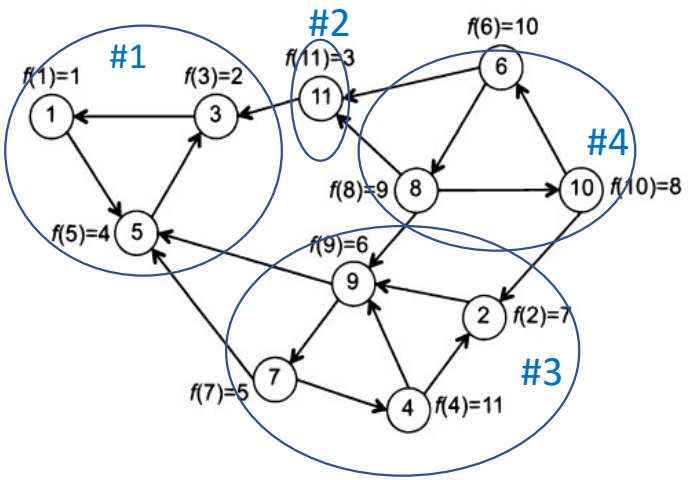
Продолжение примера

G^R после TOPOSORT



G, вершины упорядочены как в G^R





Алгоритм Косарайю

KOSARAJU

Вход: ориентированный граф $G = (V, E)$, представленный в виде списков смежности, с $V = \{1, 2, 3, ..., n\}$.

Постусловие: для каждой $v, w \in V$, $scc(v) = scc(w)$ тогда и только тогда, когда v, w находятся в одной и той же сильной связной компоненте графа G.

G^{rev} := G, в котором все ребра развернуты в обратную сторону

пометить все вершины G^{rev} как неразведанные

```
// первый проход поиска в глубину
// (вычисляет позиции f(v), волшебную упорядоченность)
```

TopoSort (G^{rev})

```
// второй проход поиска в глубину
// (находит сильно связные компоненты
// в обратном топологическом порядке)
```

пометить все вершины G как неразведанные

$numSCC := 0$ `// глобальная переменная`

for каждая $v \in V$, в порядке возрастания $f(v)$ **do**

  if v не разведана **then**

    $numSCC := numSCC + 1$

    `// назначить scc-значения (подробности ниже)`

    DFS-SCC (G, v)

DFS-SCC

Вход: ориентированный граф $G = (V, E)$, представленный в виде списков смежности, и вершина $s \in V$.

Постусловие: каждая вершина, достижимая из s, помечается как «разведанная» и имеет присвоенное ей значение scc.

пометить s как разведанную

$scc(s) := numSCC$ `// приведенная выше глобальная переменная`

for каждое ребро (s, v) в исходящем списке смежности s **do**

  if v не разведана **then**

    DFS-SCC (G, v)

Теорема. (Свойства алгоритма Касараю)

Для каждого ориентированного графа, $G = (V, E)$, представленного в виде списков смежности:

а) по завершении алгоритма Косарайю для каждой пары v, w вершин $scc(v) = scc(w)$ тогда и только тогда, когда v и w принадлежат одной и той же сильно связной компоненте графа G;

б) время работы алгоритма Косарайю равно $O(m + n)$, где $m = |E|$ и $n = |V|$.

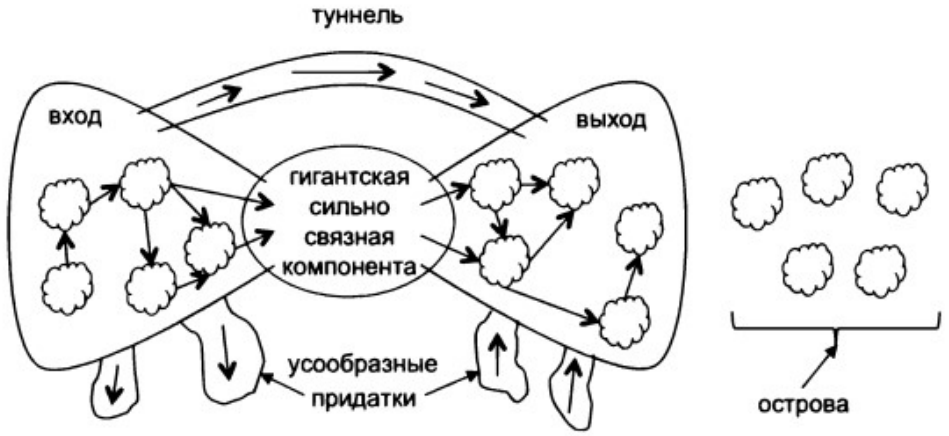
(2 поиска в глубину с небольшим постоянным множителем)

Структура всемирной паутины (орграф)

Веб-граф. Вершины – веб-страницы.

Ребра – гиперссылки.

Галстук-бабочка



. Визуализация веб-графа в виде «галстука-бабочки». Примерно одинаковое число веб-страниц принадлежит гигантской сильно связной компоненте, входу, выходу и остальной части графа

Гигантская сильно компонента – 28%,
остальные компоненты $\sim 24 - 28\%$