Введение в дискретную математику и математическую логику

Лекция №6
Вершинная и реберная связность, сильная связность ориентированных графов

- Апанович Зинаида Владимировна

Часть 1 Вершинная и реберная связность

Насколько связен граф?

Предположим, что граф представляет собой компьютерную сеть.

Знание того, что этот граф связный, говорит нам, что любые два компьютера в сети могут общаться.

Однако мы также хотели бы понять, насколько надежна эта сеть.

Например, будет ли по-прежнему возможно обмениваться данными между всеми компьютерами после сбоя одного маршрутизатора или канала связи?

Разделяющие вершины и ребра

Иногда удаление из графа вершины и всех инцидентных ей ребер приводит к созданию подграфа с большим количеством компонент связности.

Такие вершины называются разделяющими вершинами (или точками сочленения).

Удаление разделяющей вершины из связного графа приводит к образованию подграфа, который не является связным.

Аналогично, ребро, удаление которого приводит к графу с большим количеством компонент связности, чем в исходном графе, называется разделяющим ребром или мостом.

Обратите внимание, что в графе, представляющем компьютерную сеть, разделяющая вершина и разделяющее ребро представляют собой важнейший маршрутизатор и важнейший канал связи, которые не должны выйти из строя, чтобы все компьютеры могли обмениваться данными.

Разделяющие вершины и ребра

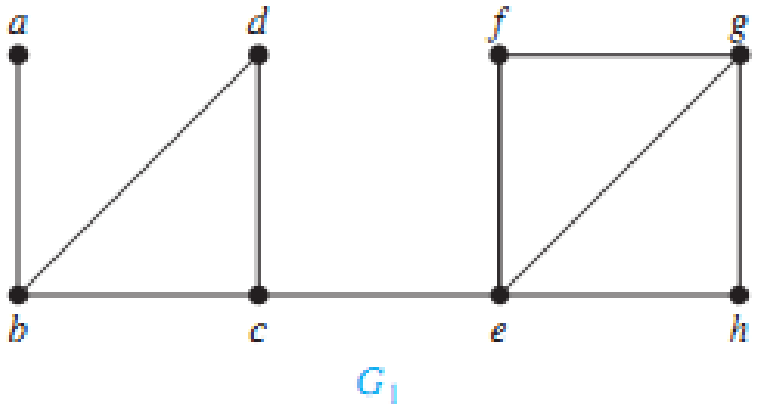
ПРИМЕР. Найдите разделяющие вершины и ребра в графе G_1 .

Решение: Разделяющие вершины графа G_1 — это *b*, *c* и *e*.

Удаление одной из этих вершин (и смежных с ней ребер) делают граф несвязным.

Разделяющие ребра — это { *a, b* } и { *c, e* }.

Удаление любого из этих ребер делает граф G_1 несвязным.



G_1

Вершинная связность

Не все графы имеют разделяющие вершины.

Например, полный граф K_n , где $n \geq 3$, не имеет разделяющих вершин.

При удалении вершины из K_n и всех ребер, инцидентных ей, результирующий подграф представляет собой полный граф K_{n-1}, то есть, связный граф.

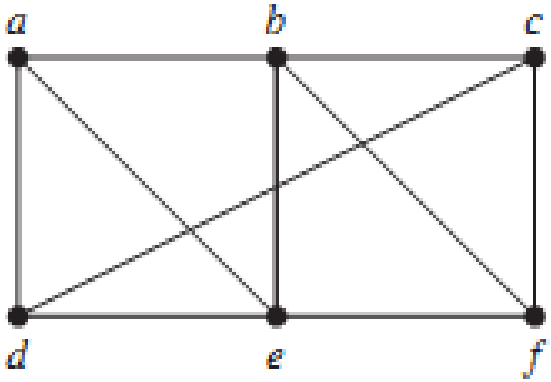
Графы без разделяющих вершин называются несепарабельными графами и могут рассматриваться как более связные, чем те, у которых есть разделяющая вершина.

Мы можем расширить это понятие, определив более детально меру связности графа, основанную на минимальном количестве вершин, которые надо удалить для превращения графа в несвязный граф.

Вершинное сечение (разделяющее множество вершин)

Подмножество V' множества вершин V графа $G = (V, E)$ называется вершинным сечением, или разделяющим множеством вершин, если $G - V'$ несвязен.

Например, в графе G множество $\{ b, c, e \}$ представляет собой вершинное сечение из трех вершин.



граф G

Число вершинной связности

Число вершинной связности графа G, не являющегося полным графом, обозначается $\kappa(G)$, и равно минимальному количеству вершин в вершинном сечении.

Число вершинной связности

Когда G — полный граф, он не имеет вершинных сечений, поскольку удаление любого подмножества его вершин и всех инцидентных ребер по-прежнему оставляют полный граф.

Следовательно, мы не можем определить $\kappa(G)$ как минимальное количество вершин в вершинном сечении, когда G является полным.

Вместо этого мы устанавливаем $\kappa(K_n) = n - 1$, количество вершин, которые необходимо удалить, чтобы получить граф с одной вершиной.

Число вершинной связности

Следовательно, для каждого графа G, $\kappa(G)$ — это минимальное количество вершин, которые надо удалить из G, чтобы либо сделать граф G несвязным, либо получить граф с одной вершиной.

Имеем $0 \le \kappa(G) \le n - 1$, если G имеет n вершин,

$\kappa(G) = 0 \Leftrightarrow G$ несвязен или $G = K_1$,

и $\kappa(G) = n - 1 \Leftrightarrow G$ полный .

Число вершинной связности

Чем больше $\kappa(G)$, тем более связным мы считаем G.

Несвязные графы и K_1 имеют $\kappa(G) = 0$,

связные графы с разделяющими вершинами и

K_2 имеют $\kappa(G) = 1$,

графы без разделяющих вершин, которые можно сделать несвязными, удалив 2 вершины и K_3 имеют $\kappa(G) = 2$,

и так далее.

Число вершинной связности

Мы говорим, что граф является k -связным (или k -вершинно-связным),

если $\kappa(G) \geq k$.

Граф G является 1-связным, если он связен и не является графом, содержащим единственную вершину;

граф G является 2-связным , или двусвязным,

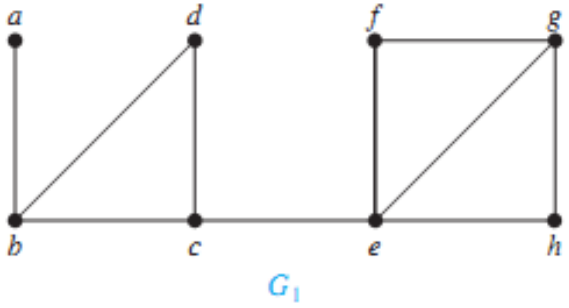
если он несепарабелен и имеет ≥ 3 вершин.

Обратите внимание, что если G является k-связным графом, то G является j-связным графом для всех

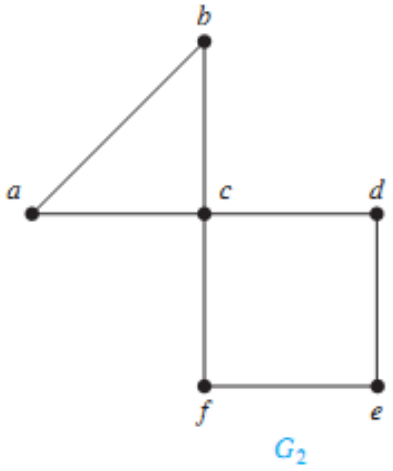
j: $0 \leq j \leq k$.

Вершинная связность

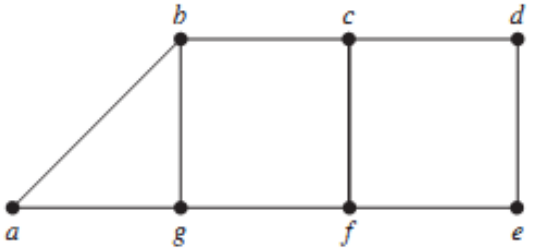
ПРИМЕР. Найдите вершинную связность для каждого из графов ниже.



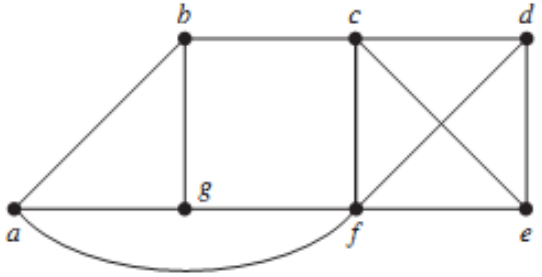
G_1



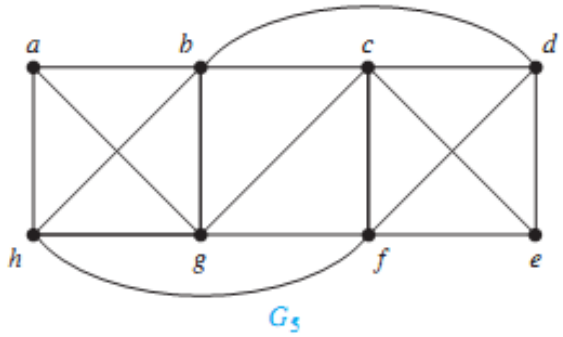
G_2



G_3



G_4



G_5

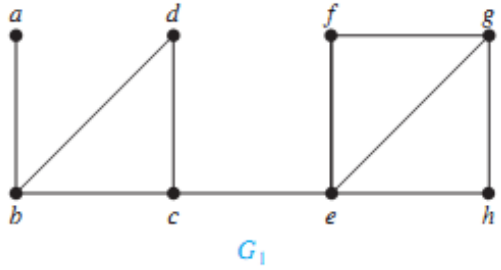
Вершинная связность

Решение: Каждый из 5 графов связен и имеет более 1 вершины, поэтому каждый из этих графов имеет положительную связность вершин.

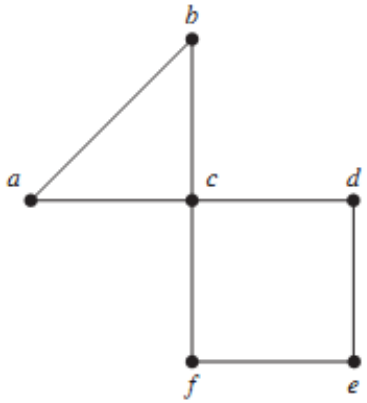
Поскольку G_1 является связным графом с разделяющей вершиной, мы знаем, что $\kappa(G_1) = 1$.

Аналогично, $\kappa(G_2) = 1$, поскольку c является точкой сочленения G_2 .

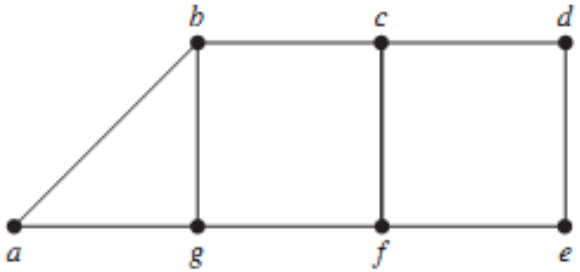
G_3 не имеет разделяющих вершин, но множество $\{ b, g \}$ является вершинным сечением. Следовательно, $\kappa(G_3) = 2$.



G_1



G_2



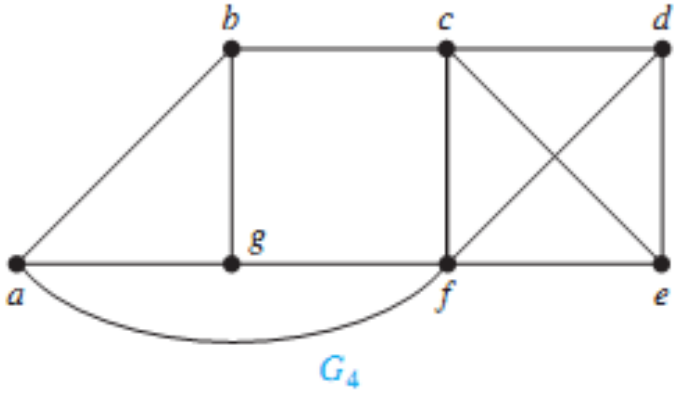
G_3

Вершинная связность

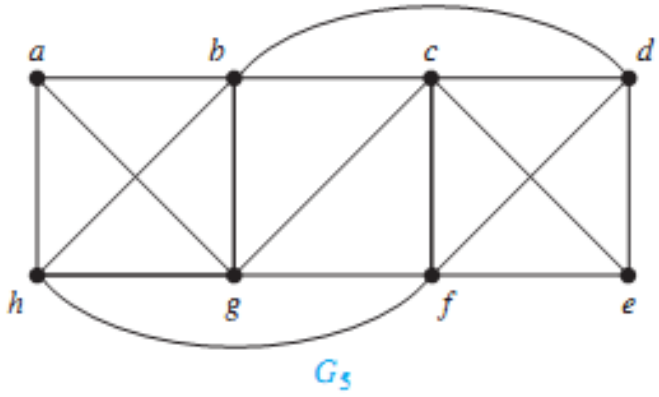
Аналогично, G_4 имеет вершинное сечение { *c, f* } размера 2, но не имеет разделяющих вершин.

Отсюда следует, что $\kappa(G_4) = 2$.

G_5 не имеет вершинного сечения размера два, но { *b, c, f* } является вершинным сечением G_5 . Следовательно, $\kappa(G_5)=3$.



G_4



G_5

Реберная связность

Мы также можем измерить связность графа $G = (V, E)$ с точки зрения минимального количества ребер , которые мы можем удалить, чтобы сделать его несвязным.

Если граф имеет разделяющее ребро, то нам достаточно удалить его, чтобы сделать граф G несвязным.

Если у G нет разделяющего ребра, мы ищем наименьшее множество ребер, которые можно удалить, чтобы сделать граф G несвязным.

Множество ребер E' называется реберным разрезом графа G, если подграф $G - E'$ является несвязным.

Реберная связность

Реберная связность графа G, обозначаемая $\lambda(G)$, — это минимальное количество ребер в реберном разрезе графа G.

Это определяет $\lambda(G)$ для всех связных графов,

имеющих > 1 вершин, поскольку всегда можно сделать такой граф несвязным, удалив все ребра, инцидентные одной из его вершин.

Обратите внимание, что $\lambda(G) = 0$, если G не является связным .

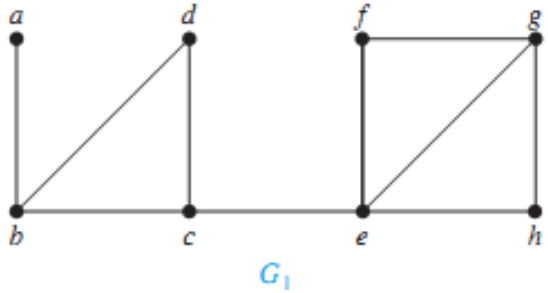
Мы также считаем, что $\lambda(G) = 0$, если G — граф, состоящий из одной вершины.

Отсюда следует, что если G — граф с n вершинами,

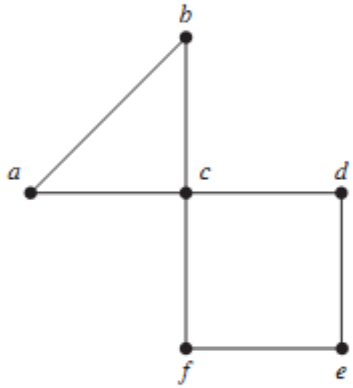
то $0 \leq \lambda(G) \leq n - 1$.

Реберная связность

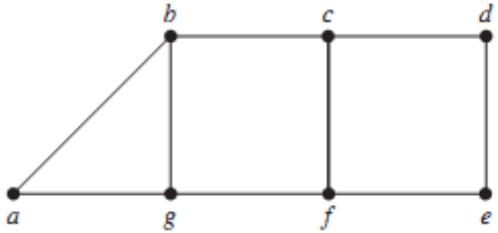
ПРИМЕР. Найдите реберную связность для каждого из графов ниже.



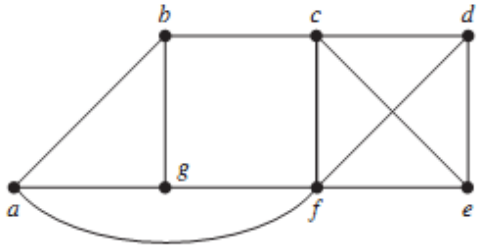
G_1



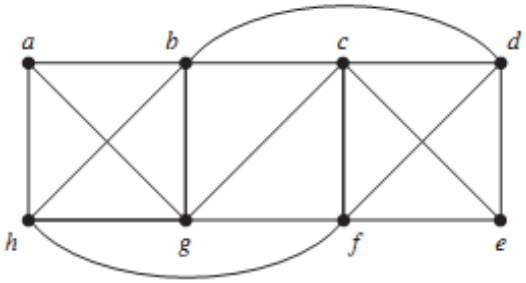
G_2



G_3



G_4



G_5

Реберная связность

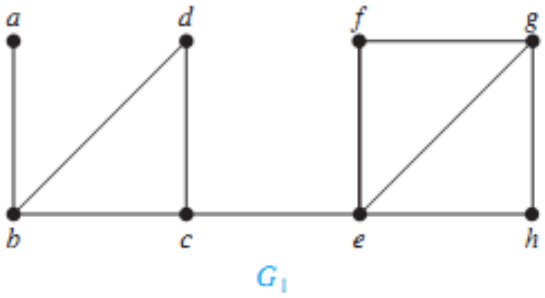
Решение: Каждый из 5 графов связен и имеет более 1 вершины, поэтому мы знаем, что все они имеют положительную реберную связность.

G_1 имеет разделяющее ребро, поэтому $\lambda(G_1) = 1$.

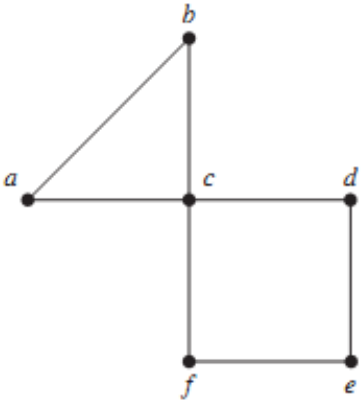
Граф G_2 не имеет разделяющих ребер, но удаление двух ребер $\{a, b\}$ и $\{a, c\}$ делает его несвязным. Следовательно, $\lambda(G_2) = 2$.

Аналогично, $\lambda(G_3) = 2$, поскольку G_3 не имеет разделяющих ребер, но удаление двух ребер

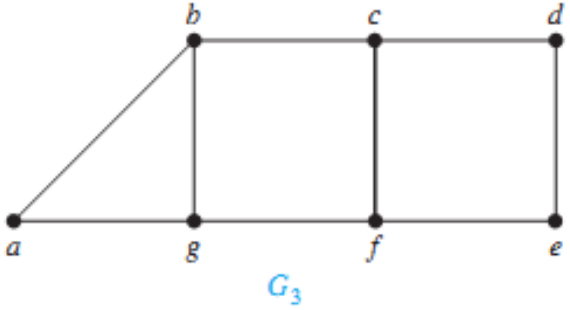
$\{b, c\}$ и $\{f, g\}$ делает его несвязным.



G_1



G_2

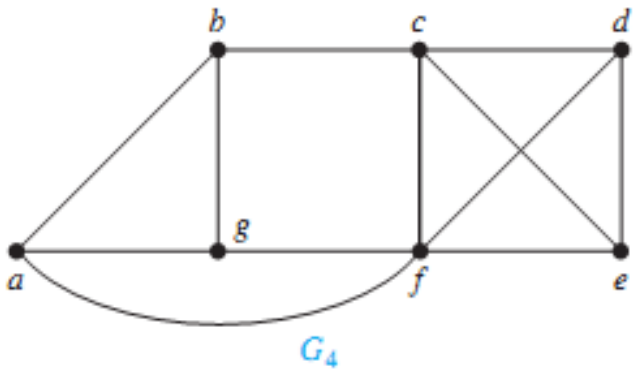


G_3

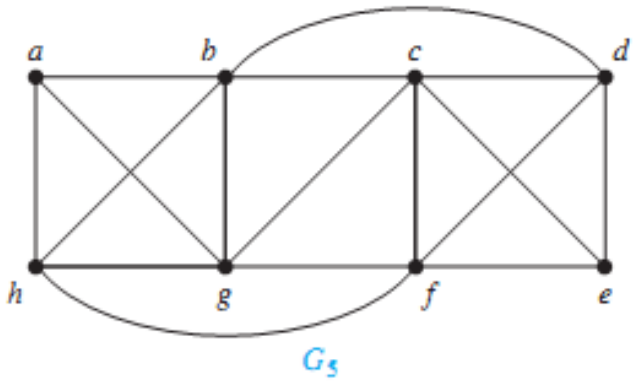
Реберная связность

Удаление двух ребер не делает несвязным G_4 , но удаление трех ребер $\{b, c\}$, $\{ a, f \}$, и $\{ f, g \}$ делает его несвязным. Следовательно, $\lambda(G_4) = 3$.

Наконец, $\lambda(G_5) = 3$, поскольку удаление любых двух его ребер не делает его несвязным, но удаление $\{ a, b\}$, $\{ a, g \}$, и $\{ a, h \}$ делает.



G_4



G_5

НЕРАВЕНСТВО ДЛЯ ВЕРШИННОЙ И РЕБЕРНОЙ СВЯЗНОСТИ $\kappa(G) \leq \lambda(G) \leq \delta(G)$

Если $G = (V, E)$ — не полный связный граф, имеющий не менее чем 3 вершины, минимальная степень вершин G является верхней границей как для вершинной связности G, так и для реберной связности G.

То есть, $\kappa(G) \leq \min_{v \in V} \deg(v)$ и

$$\lambda(G) \leq \min_{v \in V} \deg(v).$$

Чтобы увидеть это, заметим, что удаление всех соседей вершины минимальной степени делает G несвязным, и

Удаление всех ребер, инцидентных вершине минимальной степени, делает G несвязным.

НЕРАВЕНСТВО ДЛЯ ВЕРШИННОЙ И РЕБЕРНОЙ СВЯЗНОСТИ $\kappa(G) \le \lambda(G) \le \delta(G)$

Доказательство. Для доказательства неравенства $\kappa(G) \le \lambda(G)$ рассматриваются различные случаи.

Если G несвязен или тривиален, то $\kappa = \lambda = 0$.

Если G связен и имеет мост x, то $\lambda = 1$.

В этом случае $\kappa = 1$, поскольку либо G имеет точку сочленения, инцидентную x, либо G является графом K_2 .

Наконец, предположим, что G имеет $\lambda > 2$ ребер, удаление которых делает его несвязным.

Очевидно, что удаление $\lambda - 1$ из этих ребер дает граф с мостом

$x = \{ u , v \}$.

Для каждого из этих $\lambda - 1$ ребер выберем инцидентную вершину, отличную от u или v .

Удаление этих вершин также удаляет $\lambda - 1$ ребро и, вполне возможно, больше.

Если полученный граф несвязен , то $\kappa < \lambda$;

если нет, *то x является мостом* , и, следовательно, удаление u или v приведет либо к несвязному, либо к тривиальному графу, поэтому $\kappa \le \lambda$ в каждом случае.

НЕРАВЕНСТВО ДЛЯ ВЕРШИННОЙ И РЕБЕРНОЙ СВЯЗНОСТИ $\kappa(G) \leq \lambda(G) \leq \delta(G)$

Отметим также, что $\kappa(K_n) = \lambda(K_n) = \min_{v \in V} \deg(v) = n - 1$, когда n положительно целое число и

что $\kappa(G) = \lambda(G) = 0$, когда G — несвязный граф.

Объединив эти факты, мы можем сформулировать

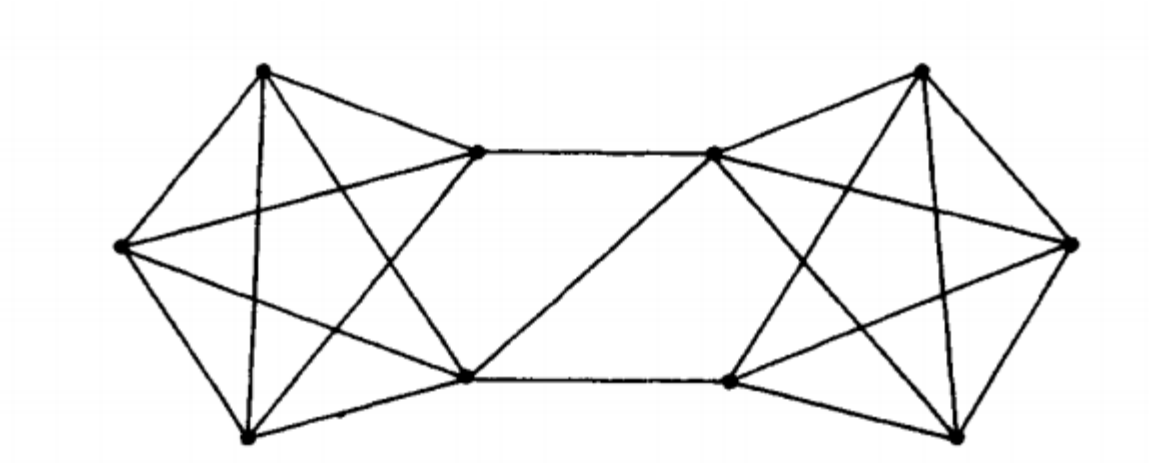
Теорема 1:

Для всех графов G ,

$\kappa(G) \leq \lambda(G) \leq \min_{v \in V} \deg(v)$.

НЕРАВЕНСТВО ДЛЯ ВЕРШИННОЙ И РЕБЕРНОЙ СВЯЗНОСТИ $\kappa(G) \leq \lambda(G)$

Пример строгого неравенства $\kappa < \lambda < \delta$.



Граф, для которого $\kappa = 2$, $\lambda = 3$, $\delta = 4$

Свойства точек сочленения

Теорема 2 Пусть a — вершина связного графа

Следующие утверждения эквивалентны:

(1) a является точкой сочленения графа G.

(2) Существуют вершины u и w, отличные от a, такие, что a принадлежит каждому пути из u в w.

(3) Существует разбиение множества вершин $V - \{ a \}$ на подмножества U и W такое, что для любых вершин $u \in U$ и $w \in W$, вершина a находится на каждом пути из u в w .

Доказательство.

(1) => (3) Так как a является точкой сочленения G, то $G - a$ несвязен и имеет как минимум 2 компоненты.

Образуем разбиение $V — \{ a \}$, положив U состоящим из вершин одной из этих компонент, а W — из вершин другой.

Тогда любые 2 вершины $u \in U$ и $w \in W$ лежат в разных компонентах $G — a$.

Следовательно, каждый путь из u в w G содержит a .

Свойства точек сочленения

(3) => (2) Очевидно, поскольку (2) является частным случаем (3).

(2) => (1) Если a находится на каждом пути в G, соединяющем u и w, то не может быть пути, соединяющего эти вершины в $G - a$.

Таким образом $G - a$ несвязен, поэтому a является точкой сочленения графа G.

ПРИЛОЖЕНИЯ ВЕРШИННОЙ И РЕБЕРНОЙ СВЯЗНОСТИ

Связность графов играет важную роль во многих проблемах, связанных с надежностью сетей.

Например, мы можем смоделировать сеть передачи данных, используя вершины для представления маршрутизаторов и ребра для представления связей между ними .

Вершинная связность полученного графа равна минимальному количеству маршрутизаторов, которые делают сеть несвязной, когда они выходят из строя.

Если число неисправных маршрутизаторов меньше, передача данных между каждой парой маршрутизаторов по-прежнему возможна.

Реберная связность представляет собой минимальное количество волоконно -оптических соединений, отключение которых может привести к отключению сети.

Если число неисправных каналов уменьшится, передача данных между каждой парой маршрутизаторов все равно будет возможна.

ПРИЛОЖЕНИЯ ВЕРШИННОЙ И РЕБЕРНОЙ СВЯЗНОСТИ

Мы можем смоделировать сеть автомагистралей ,

- используя вершины для представления перекрестков автомагистралей и
- ребра для представления участков дорог, проходящих между перекрестками.

Вершинная связность полученного графа представляет собой минимальное количество перекрестков, которые могут быть закрыты в определенный момент времени, что сделает невозможным перемещение между каждыми двумя перекрестками.

Если будет закрыто меньше перекрестков, проезд между каждой парой перекрестков все равно будет возможен.

Реберная связность представляет собой минимальное количество дорог, которые можно закрыть, чтобы сделать сеть автомагистралей несвязной.

Если будет закрыто меньше автомагистралей, проехать между любыми двумя перекрестками все равно можно будет.

Очевидно, что департаменту автодорог было бы полезно учитывать эту информацию при планировании ремонта дорог.

Часть 2. Связность в ориентированных графах

Связность в ориентированных графах

В ориентированных графах существует два понятия связности, в зависимости от того, учитываются ли направления ребер.

Сильно связный ориентированный граф

ОПРЕДЕЛЕНИЕ. Ориентированный граф сильно связен, если существует путь из *a* в *b* и из *b* в *a* для каждой пары вершин графа.

Для того чтобы ориентированный граф был сильно связным, должна существовать последовательность ориентированных ребер из любой вершины графа в любую другую вершину.

Ориентированный граф может не быть сильно связанным, но при этом оставаться «целым».

Слабо связный ориентированный граф

ОПРЕДЕЛЕНИЕ Ориентированный граф является слабо связным, если между каждыми двумя вершинами его основания существует путь .

То есть, ориентированный граф слабо связен

⇔ существует путь между любыми двумя вершинами, если не учитывать направления ребер.

Очевидно, что любой сильно связный ориентированный граф является также слабо связным .

ПРИМЕР

Являются ли ориентированные графы *G* и *H*, показанные ниже, сильно связными? Являются ли они слабо связными?

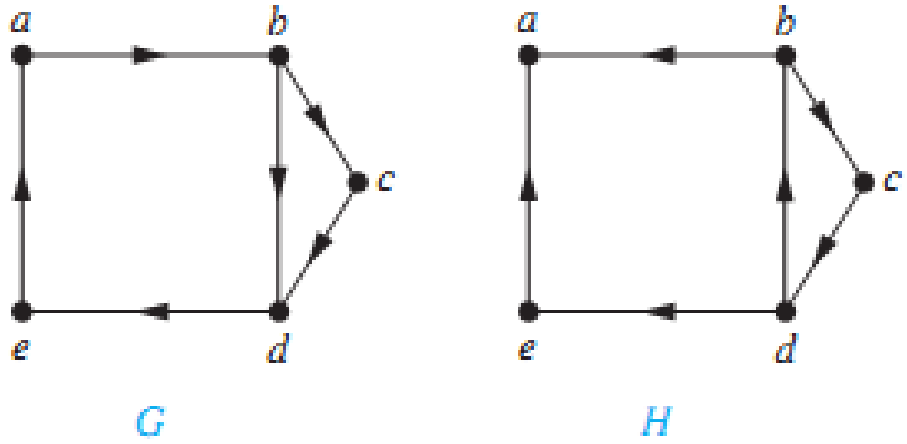
Решение: Граф *G* сильно связен, поскольку между любыми двумя вершинами этого ориентированного графа существует путь.

Следовательно, *G* также слабо связен .

Граф *H не* является сильно связным .

В *H* нет ориентированного пути из *a* в *b* .

Однако, *H* слабо связен, поскольку между любыми двумя вершинами в основании графа *H* существует путь .



СИЛЬНЫЕ КОМПОНЕНТЫ ОРИЕНТИРОВАННОГО ГРАФА

Подграфы ориентированного графа *G*, которые сильно связны, но не содержатся в более крупных сильно связанных подграфах, то есть,

максимальные сильно связные подграфы называются сильно связными компонентами или

сильными компонентами графа *G*.

Обратите внимание, что если *a* и *b* — две вершины в ориентированном графе, то их сильные компоненты либо совпадают, либо не пересекаются.

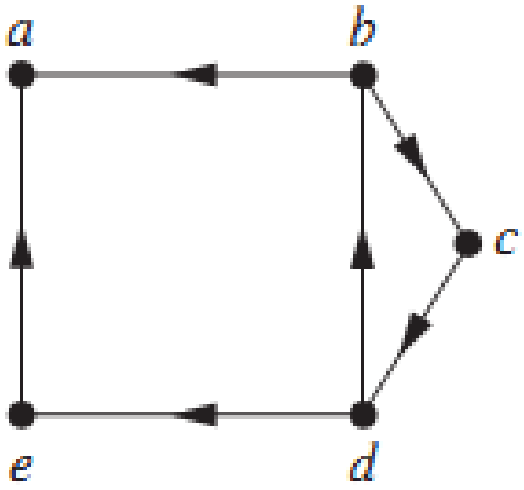
Сильно связные компоненты

ПРИМЕР. Граф *H* имеет 3 сильно связные компоненты, состоящие из

вершины *a* ;

вершины *e* ;

и подграфа, состоящего из вершин *b, c* и *d* и ребер (*b, c*), (*c, d*) и (*d, b*).



H

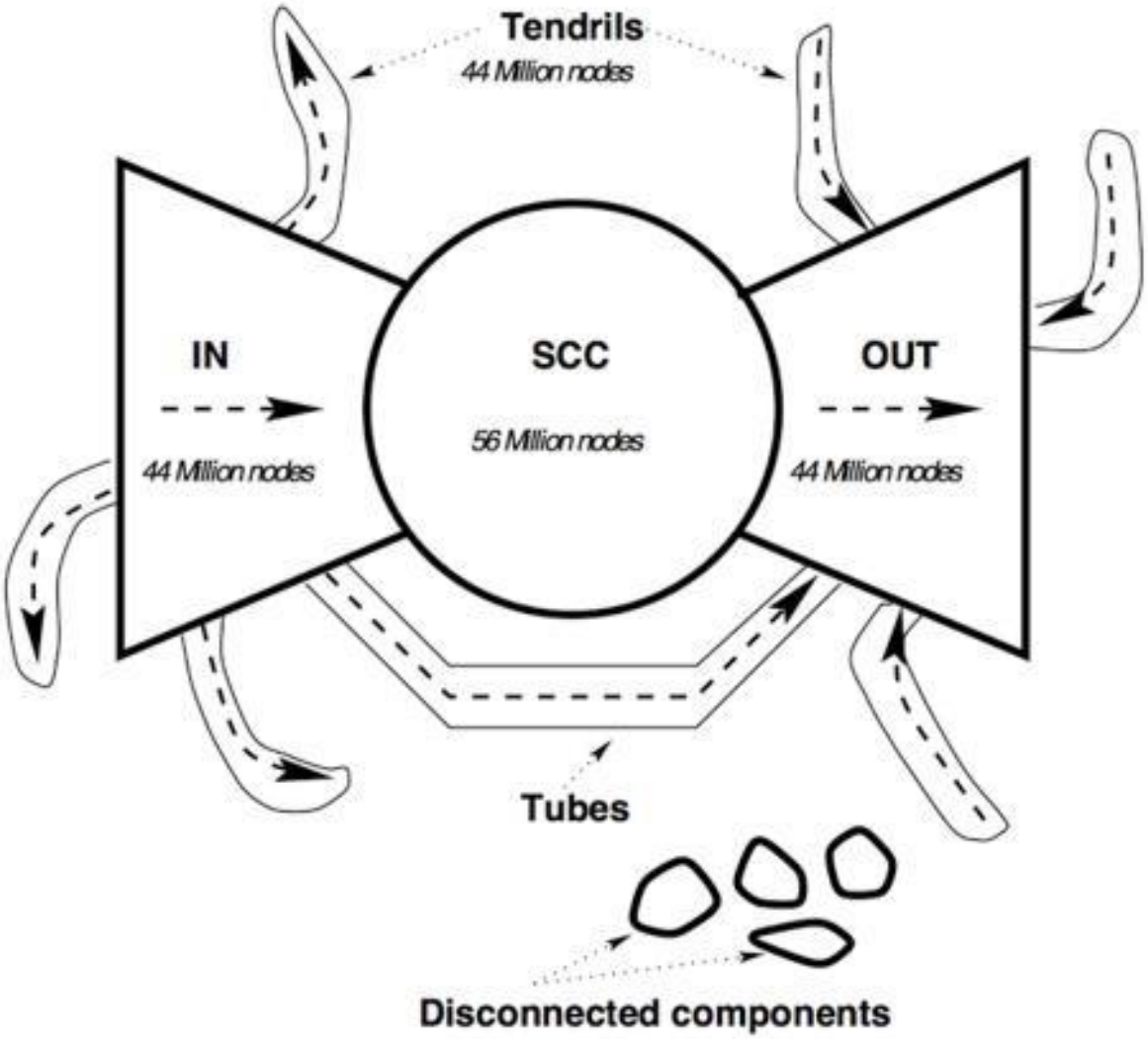
Премия The Seoul Test of Time Award (2017),

- Статья, получившая премию 2017 года, « Структура графа в Интернете » Андрея Бродера, Рави Кумара, Фарзина Магхул , Прабхакар Рагхаван, Шридхар Раджагопалан, Рэйми Стата, Эндрю Томкинс и Джанет Винер, была первоначально представлена в 2000 году на 9-й конференции WWW в Амстердаме.

Премия The Seoul Test of Time Award (2017)

- Статья внесла два важных вклада в понимание структуры Интернета.
- 1) ее масштабные эксперименты показали, что узлы сети распределены по степенному закону.
- То есть, вероятность того, что узел сети имеет *i* входящих ссылок, примерно пропорциональна $1/i^{2.1}$.
- 2) в отличие от предыдущих исследований, предполагавших, что WWW почти полностью связен, в этом исследовании описывается гораздо более сложная структура WWW, которая с тех пор изображается в форме «бабочки»:

Премия The Seoul Test of Time Award (2017)



- В статье описано несколько характерных классов веб-страниц:
- *сильно связная компонента-ядро*, где каждая страница доступна с любой другой страницы,
- так называемые кластеры *IN* и *OUT*, которые имеют только однонаправленные пути в ядро или из него,
- *«усики»* , свисающие с двух кластеров, и «трубки», соединяющие кластеры, в обход ядра, и, наконец,
- *несвязанные компоненты*, которые изолированы от остальной части графа.

Часть 3.
Двусвязность

ДВУСВЯЗНОСТЬ

Пусть $G = (V, E)$ — связный неориентированный граф.

Вершина a называется точкой сочленения графа G, если существуют вершины v и w такие, что v, w и a различны, и каждый путь между v и w содержит вершину a .

Другими словами, a является точкой сочленения G, если удаление a разделяет G на ≥ 2 части.

Граф G двусвязен, если для каждой тройки вершин v, w, a существует путь между v и w, не содержащий a .

Таким образом, неориентированный связный граф является двусвязным $\Leftrightarrow$ он не имеет точек сочленения.

Двусвязная компонента (блок)

Мы можем определить естественное отношение на множестве ребер графа G, сказав, что 2 ребра e_1 и e_2 находятся в отношении R, если

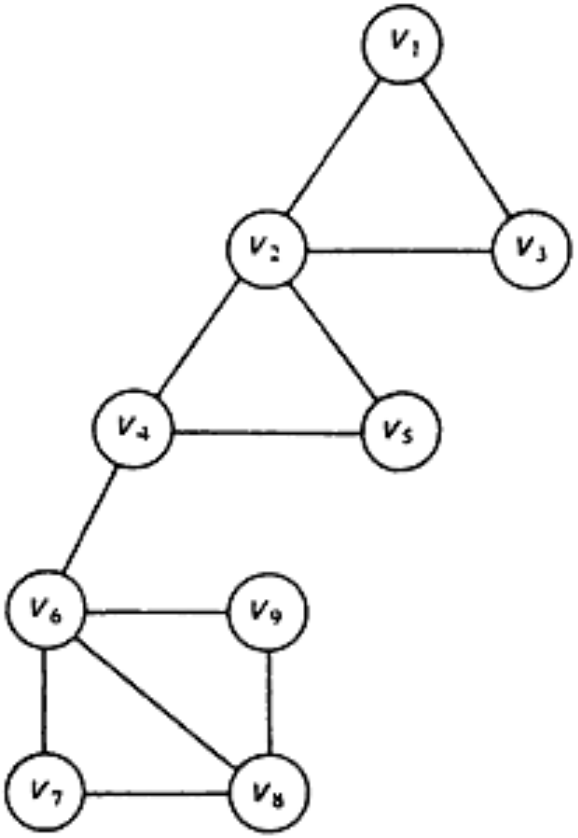
$e_1 = e_2$ или существует цикл, содержащий e_1 и e_2 .

Легко показать, что это отношение является отношением эквивалентности, которое разбивает ребра графа G на классы эквивалентности $E_1, E_2, \ldots, E_k$ таким образом, что 2 различных ребра находятся в одном классе $\Leftrightarrow$ они лежат на общем цикле.

Для $1 \le i \le k$, пусть V_i будет множеством инцидентных вершин для ребер во множестве E_i .

Каждый граф $G_i = (V_i , E_i)$ называется двусвязной компонентой (блоком) графа G .

Пример. Рассмотрим неориентированный граф ниже. Сколько двусвязных компонент показано?



Решение Имеется 4 двусвязных компоненты.

Вершина v_4 , например, является точкой сочленения , поскольку каждый путь между v_1 и v_7 проходит через v_4

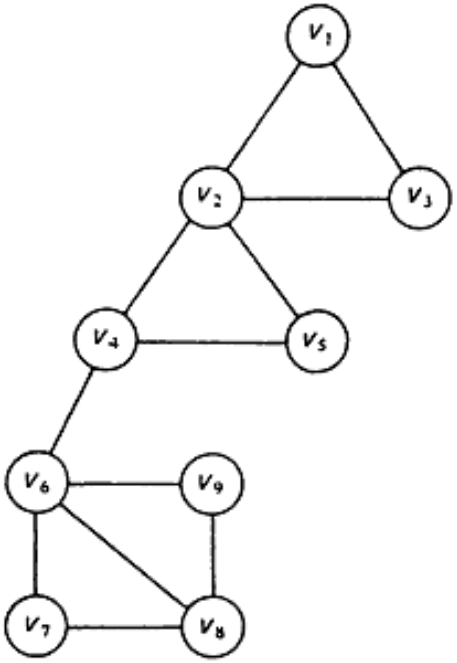
Классы эквивалентности ребер, лежащих на общих циклах, равны

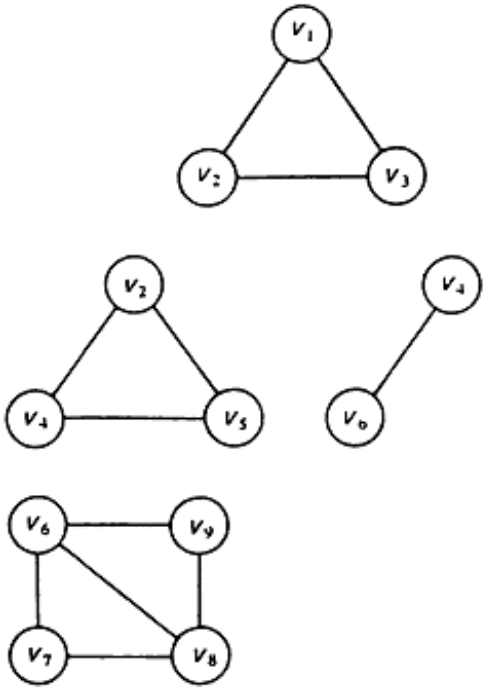
$\{\{ v_1 , v_2 \}, \{ v_1 , v_3 \}, \{ v_2 , v_3 \}\}$,

$\{\{ v_2 , v_4 \}, \{ v_2 , v_5 \}, \{ v_4 , v_5 \}\}$,

$\{\{v_4 , v_6 \}\}$,

$\{\{v_6 , v_7 \}, \{ v_6 , v_8 \}, \{ v_6 , v_9 \}, \{ v_7 , v_8 \}, \{ v_8 , v_9 \}\}$.





Свойства двусвязных компонент

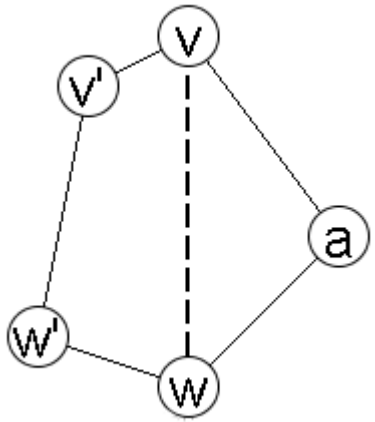
Лемма 1. Для $1 \le i \le k$, пусть $G_i = (V_i , E_i)$ будут двусвязными компонентами связного неориентированного графа $G = (V, E)$.

Тогда

1. G_i двусвязен для каждого i , $1 \le i \le k$.
2. Для всех $i \ne j$, $V_i \cap V_j$ содержит не более 1 вершины.
3. a — точка сочленения $G \Leftrightarrow$ $a \in G_i \cap B_j$ для некоторых $i \ne j$.

1) G_i двусвязен для каждого i , $1 \le i \le k$.

Доказательство



в V есть 3 различные вершины v , w и a , такие, что все пути в G_i между v и w проходят через a .

Тогда, конечно, (v, w) не является ребром в E_i . Таким образом, существуют различные ребра (v , v') и (w, w') в E_i и существует цикл в G_i, содержащий эти ребра.

По определению двусвязной компоненты все ребра и вершины этого цикла находятся в E_i и V_i соответственно.

в G_i есть 2 пути между v и w , только 1 из которых может содержать a,

противоречие .

2. Для всех $i \neq j$, $V_i \cap V_j$ содержит не более 1 вершины.

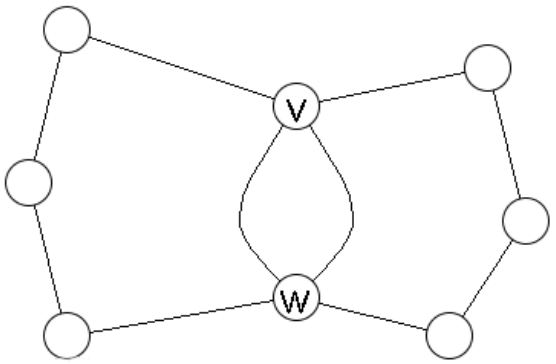
Предположим, что имеются 2 различные вершины v и w, принадлежащие $V_i \cap V_j$.

Тогда существует цикл C_1 в G_i , содержащий v и w, и цикл C_2 в G_j , также содержащий v и w .

Так как E_i и E_j не пересекаются, то множества ребер в C_1 и C_2 не пересекаются.

Однако мы можем построить цикл, содержащий v и w , который использует ребра как из C_1, так и *из* C_2, и, значит, по крайней мере 1 ребро в E_i эквивалентно ребру в E_j .

Таким образом, E_i и E_j не являются классами эквивалентности, как предполагалось.



3. a — точка сочленения $G \Leftrightarrow a \in V_i \cap V_j$ для некоторых $i \neq j$.

=> Предположим , что вершина a является точкой сочленения G.

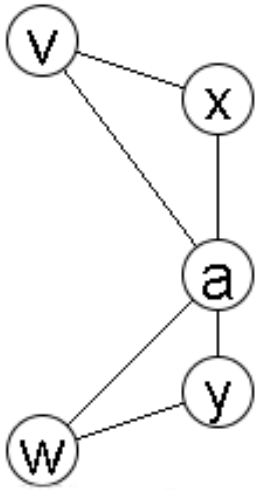
Тогда существуют 2 вершины v и w такие, что v, w и a различны, и каждый путь между v и w содержит a .

Поскольку G связен, существует по крайней мере 1 такой путь.

Пусть $\{ x , a \}$ и $\{ y , a \}$ — два ребра на пути между v и w, инцидентные a .

Если существует цикл, содержащий эти 2 ребра, то существует путь между v и w , не содержащий a .

Таким образом, $\{ x , a \}$ и $\{ y , a \}$ находятся в различных двусвязных компонентах , а a находится в пересечении множеств их вершин.

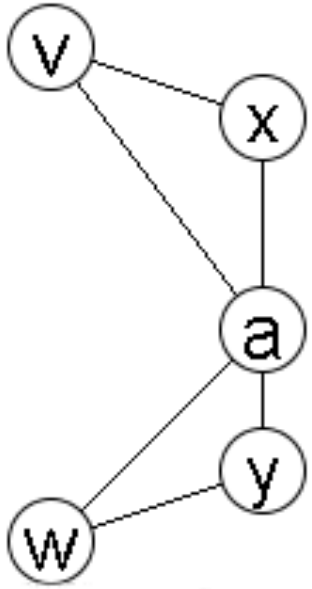


< = Если $a \in V_i \cap V_j$,т огда есть ребра { *x*, *a* } и { *y*, *a* } в

E_i и E_j соответственно.

Поскольку оба эти ребра не встречаются ни в одном цикле, то каждый путь от *x* до *y* содержит *a* .

Таким образом, *a* является точкой сочленения .



Поиск в глубину особенно полезен в нахождение двусвязных компонентов неориентированного графа .

Одной из причин этого является отсутствие «поперечных ребер».

То есть, если вершина *v* не является ни предком, ни потомком вершины *w* в остовном лесу, то не может быть ребра из *v* в *w* .

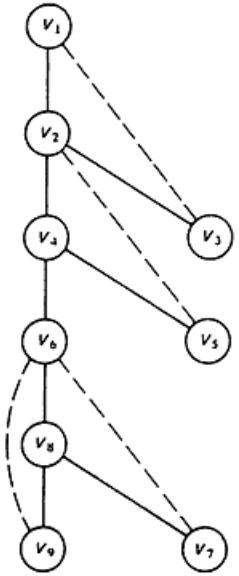
Если вершина *a* является точкой сочленения, то удаление *a* и всех ребер, инцидентных a, *разбивает* граф на 2 или более частей.

Один состоит из сына *a* и *всех* его потомков в дереве поиска в глубину.

Таким образом, в дереве поиска в глубину *a* должна иметь сына *s* , такого, что не существует обратного ребра между потомком *s* и собственным предком *a* .

Наоборот, за исключением корня остовного дерева, отсутствие поперечных ребер подразумевает, что вершина *a* является точкой сочленения, если нет обратного ребра от любого потомка некоторого сына *a* к *правильному предку a*.

Корень остовного дерева поиска в глубину является точкой сочленения ⇔ у него есть ≥ 2 сына.

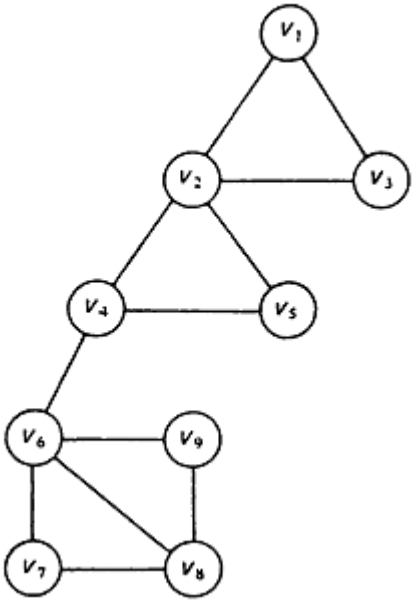


Пример. Глубинное остовное дерево для графа (a) показано на (b).

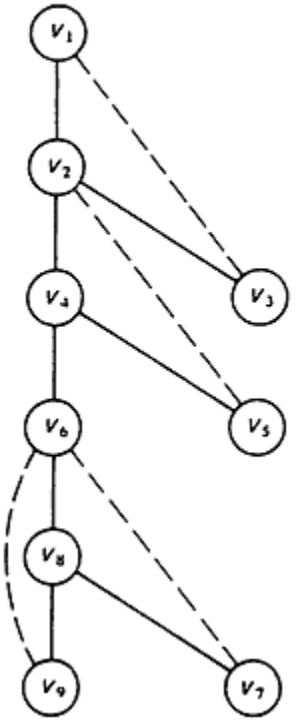
Точки сочленения — v_2, v_4 и v_6.

У вершины v_2 есть сын v_4, и ни один потомок v_4 не имеет обратных ребер к собственному предку v_2.

Аналогично, *у вершины* v_4 есть сын v_6, и у вершины v_6 есть сын v_8 с аналогичным свойством.



(а)



(б)

Предшествующие идеи воплощены в следующей лемме.

Лемма 2. Пусть $G = (V, E)$ — связный неориентированный граф, а $S = (V , T)$ — дерево поиска в глубину для G .

Вершина a является точкой сочленения G $\Leftrightarrow$ либо

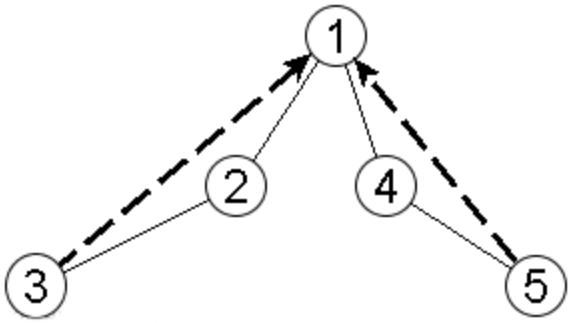
(1) a является корнем и a имеет > 1 сына,

или

(2) *a не* является корнем и для некоторого сына s вершины a *нет* обратного ребра между любыми *потомками* из (включая сам s) и правильным предком a .

Легко показать, что корень является точкой сочленения ⇔ у него > 1 сына.

(Упражнение).



=> Предположим, что *условие* 2 истинно (вершина *a не* является корнем и для некоторого сына вершины *a* нет обратного ребра между каким- либо *потомком s* и *собственным* предком *a*).

Пусть *p* будет родителем *a*. Каждое обратное ребро идет от вершины к ее предку.

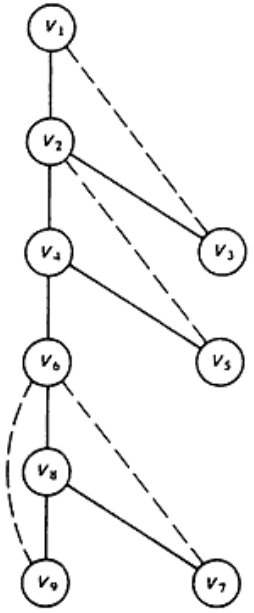
Таким образом, любое обратное ребро *от* потомка *s* вершины *v* из переходит к предку *s*.

По условию леммы обратное ребро не может идти к собственному предку *a.*

Следовательно, оно идет либо *к a* или

потомку *a*.

Таким образом, каждый путь от *s* до *p* содержит *a*, и , значит, *a* является точкой сочленения .



<= Чтобы доказать обратное, предположим, что *a* является точкой сочлененя, но не корнем .

Пусть *x* и *y* — различные вершины, отличные от *a,* такие, что каждый путь в *G* между *x* и *y* содержит *a* .

По *крайней мере одна из x и y, скажем x , является правильным потомком a* в *S*, в противном случае существует путь в *G* между *x* и *y*, использующий ребра в *T* и избегающий *a* .

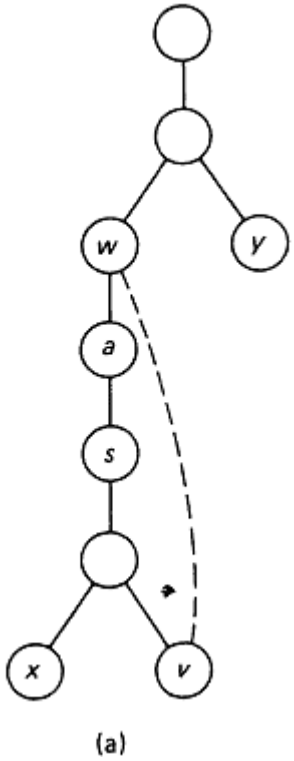
Пусть *s будет* таким сыном, что *x* является потомком *s* (возможно, *x* = *s*).

Возможны 2 варианта.

Либо нет обратного ребра между потомком *v* вершины *s* и собственным предком *w* вершины *a*, в этом случае Лемма 2 сразу же становится истинной,

или такое ребро { *v*, *w* } существует.

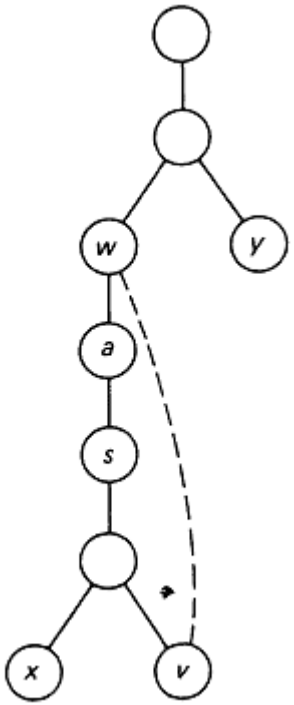
В последнем случае следует рассмотреть два случая .



(a)

СЛУЧАЙ 1. Предположим , *что y* не является потомком *a.*

Тогда существует путь от *x в v* и в *y* , который не проходит через *a*, противоречие.



(a)

СЛУЧАЙ 2. Предположим, что *y* является потомком *a* .

Конечно, *y* не является потомком *s*, иначе существует путь от *x* к *y* , который обходит *a* .

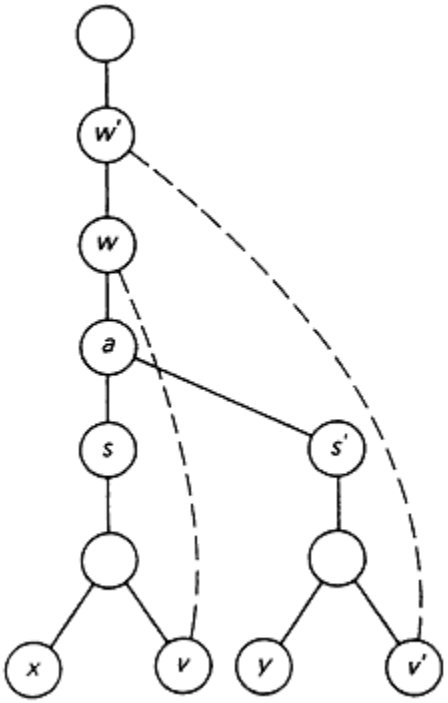
Пусть *s*' будет сыном вершины *a*, таким, что *y* является потомком *s*.

Опять возможны 2 варианта. Либо нет обратного ребра между потомком *v* вершины *s*' и собственным предком *w* вершины *a* , в этом случае лемма 2 сразу же становится истинной,

или такое ребро (*v*', *w*') существует.

В последнем случае имеется путь из *x* в *v, w, w', v , y* , что позволяет не проходить через *a*. *Противоречие* .

Приходим к выводу, что Лемма 2 верна.



Пусть T и B — множества деревесных и обратных ребер полученные путем поиска в глубину на связном неориентированном графе $G = (V, E)$.

Мы предполагаем, что вершины в V проименованы по их номерам поиска в глубину ($v.d$).

Для каждой v в V мы определяем

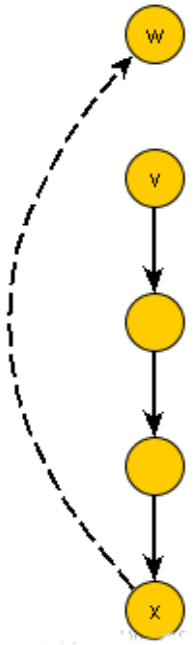
$v.LOW$ = MIN({v} U {w | существует обратное ребро { x, w } $\in B$

- такое, что x является потомком v,
- а w предком v в глубинном остовном лесу (V , T)}) (1)

Нумерация $v.d$ подразумевает, что если x является потомком из v и { x , w } — обратное ребро такое, что $w < v$, то w является собственным предком v.

Таким образом, по лемме 2, если вершина v *не* является корнем T, то v является *точкой сочленения* ⇔ у v есть сын s такой что

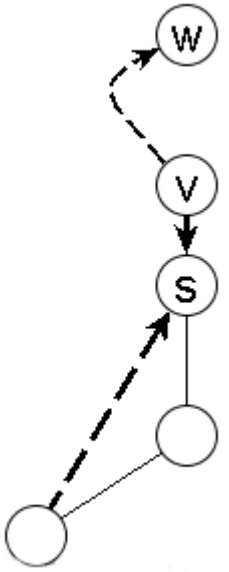
$s.Low \geq v$.



Мы можем внедрить в процедуру DFS_visit вычисление для определения значения *Low* каждой вершины, если перепишем (1) так, чтобы выразить *v.Low* через вершины, смежные с *v* , через обратные ребра и значения *Low* в сыновьях *v* .

В частности, *v.Low* можно вычислить, определив минимальное значение таких вершин *w*, что либо

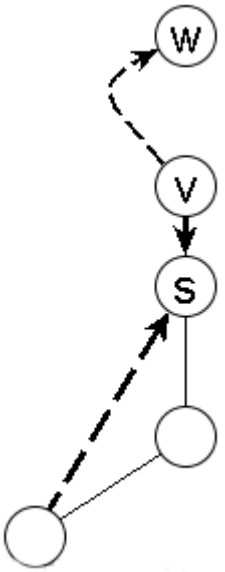
1. *w* = *v* , или
2. *w* = *s.Low* и *s* является сыном *v* , или
3. (*v*, *w*) —обратное ребро в *B* .



Минимальное значение *w* можно определить, как только список вершин, смежных с *v*, Adj[*v*], будет исчерпан.

Таким образом, (1) эквивалентно

v.Low = MIN({ *v* } U { *s* . *Low* | *s* является сыном *v* } U { *w* |{ *v* , *w* } ∈ *B* }). (2)



DFS_visit с вычислением LOW

Процедура *DFS _ Visit _Biconn(v);*

v.color = GRAY ;

v.d = time;

time = time +1;

v.Low = v.d ;

DFS с вычислением LOW (продолжение)

```
for each vertex w ∈ Adj[v] do {
    if w.color == "WHITE" then{ /* (v, w) – древесное ребро*/
        w.π = v /* add (v, w) to T/
        DFS_Visit_Biconn(w);
        if w.Low ≥ v.d  then /* найдена двусвязная компонента*/
            v.Low = MIN(v.Low, w.Low);
      } /* конец обработки древесного ребра*/
   else if v.π != w then /*(v, w) – обратное ребро*/
            v.Low =MIN(v.Low, w.d);
 }
}}
}
```

Мы включили как переименование вершин по первому посещению, так и вычисление *Low* в пересмотренную версию DFS_visit.

Сначала мы инициализируем *v.Low* до максимально возможного значения.

Если вершина *v* имеет сына *w* в глубинном остовном лесу, то мы корректируем *v.Low* , если *w.Low* < *v.Low* .

Если вершина *v* соединена обратным ребром с вершиной *w*, то мы делаем *v.Low равным w.d , если номер при поиске в* глубину вершины *w* меньше текущего значения *v.Low* .

Тест проверяет случай, когда (*v, w*) на самом деле не является обратным ребром, поскольку *w* является отцом *v в* глубинном остовном дереве.

Найдя *v*. *Low* для каждой вершины *v,* мы можем легко определить точки сочленения.

Алгоритм 3. Нахождение двусвязных компонент.

Вход. Связный неориентированный граф $G = (V, E)$.

Выход . Список ребер каждой двусвязной компоненты графа G.

1. Изначально установить $T = \varnothing$, a time = 0.

Также пометить каждую вершину в V как «БЕЛУЮ».

Затем выбрать произвольную вершину v_0 в V и вызвать DFS _ Visit _Biconn(v_0) для построения остовного дерева поиска в глубину $S = (V , T)$ и вычисления $v.LOW$ для каждой v в V .

Когда вершина w встречается в DFS _ Visit_Biconn поместить ребро (v, w) в STACK, если его там еще нет.

После обнаружения пары (v, w) такой, что w является сыном v и $w.Low \geq v$, извлечь из STACK все ребра до (v, w) включительно.

Эти ребра образуют двусвязную компоненту графа

(Обратите внимание, что если (v, w) является ребром, то v находится на Adj [w], а w находится на Adj [v]. Таким образом, (v, w) встречается дважды: один раз при посещении вершины v u один раз при посещении вершины w .

Мы можем проверить, находится ли (v, w) уже в STACK, проверив, что $v < w$ и w *является* "старой" или если $v > w$ и $w = v.\pi$.

Пример. Дерево поиска в глубину на графе *G* показано с вершинами, переименованными в соответствии с *v*.*d* и также указаны значенbz.*Low* .

Например, *DFS _ Visit _Biconn* (6) определяет, что 6. *Low* = 4, поскольку существует обратное ребро (6, 4).

Затем *DFS _ Visit _Biconn* (5), вызвавший *DFS _ Visit _Biconn* (6), устанавливает 5. *Low* = 4, так как 4 меньше начального значения

5.*Low*, которое равно 5.

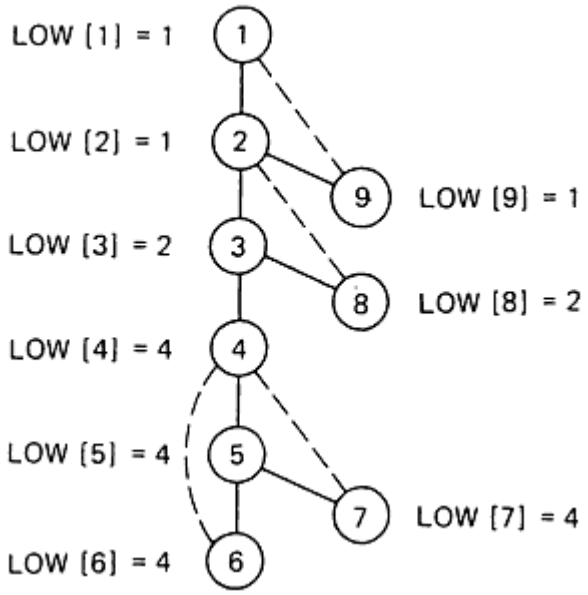
По завершении *DFS _ Visit _Biconn* (5) мы обнаруживаем, что 5.*Low* = 4.

Таким образом, 4 — это точка сочленения.

На этом этапе стек содержит ребра (снизу вверх)

(1, 2) (2, 3), (3, 4), (4, 5), (5, 6), (6, 4). (5, 7), (7, 4).

Таким образом, мы сдвигаем ребра вниз до (4, 5) включительно.

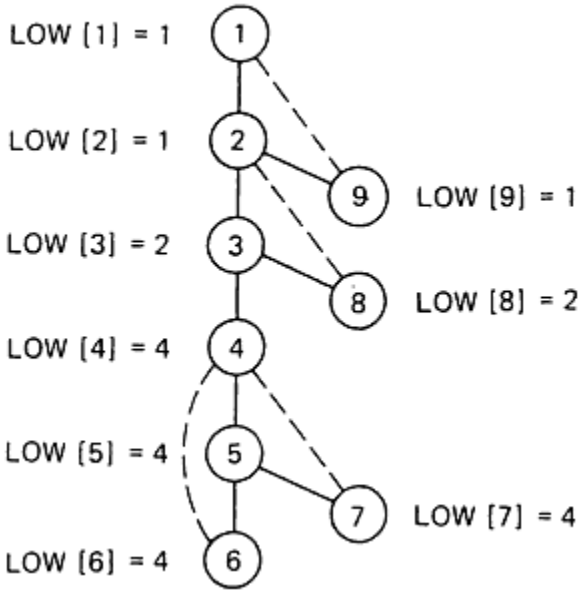


То есть мы выводим ребра (7, 4), (5, 7), (6, 4), (5, 6) и (4, 5), которые являются ребрами первой найденной двусвязной компоненты.

Обратите внимание, что по завершении *DFS_visit_Biconn* (2) мы обнаруживаем, что

2.Low = 1 и очистиv стек ребер, даже если 1 не является точкой сочленения.

Это гарантирует, что двусвязная компонента, содержащая корень, будет обнаружена.



- Ваши вопросы?
- Контакты лектора:
  apanovich_09@mail.ru