

# Двусвязность

Лекция 14

# СВЯЗНОСТЬ КОМПОНЕНТ

Граф  $G$  называется  *$k$ -связным* ( $k \geq 1$ ), если не существует

набора из  $k-1$  или меньшего числа узлов  $V' \subseteq V$ , такого, что

удаление всех узлов  $V'$  и инцидентных с ними ребер, сделают граф  $G$  несвязанным.

*Теорема Менгера:* граф  $G$  является  $k$ -связанным тогда и только тогда, когда любые два различные узла  $x$  и  $y$  графа  $G$

соединены по крайней мере  $k$  путями, не содержащими общих узлов (кроме  $x$  и  $y$ ).

Для  $\forall k$  существует алгоритм, отыскивающий  $k$ -связные компоненты за линейное время по числу вершин и

## Точки сочленения

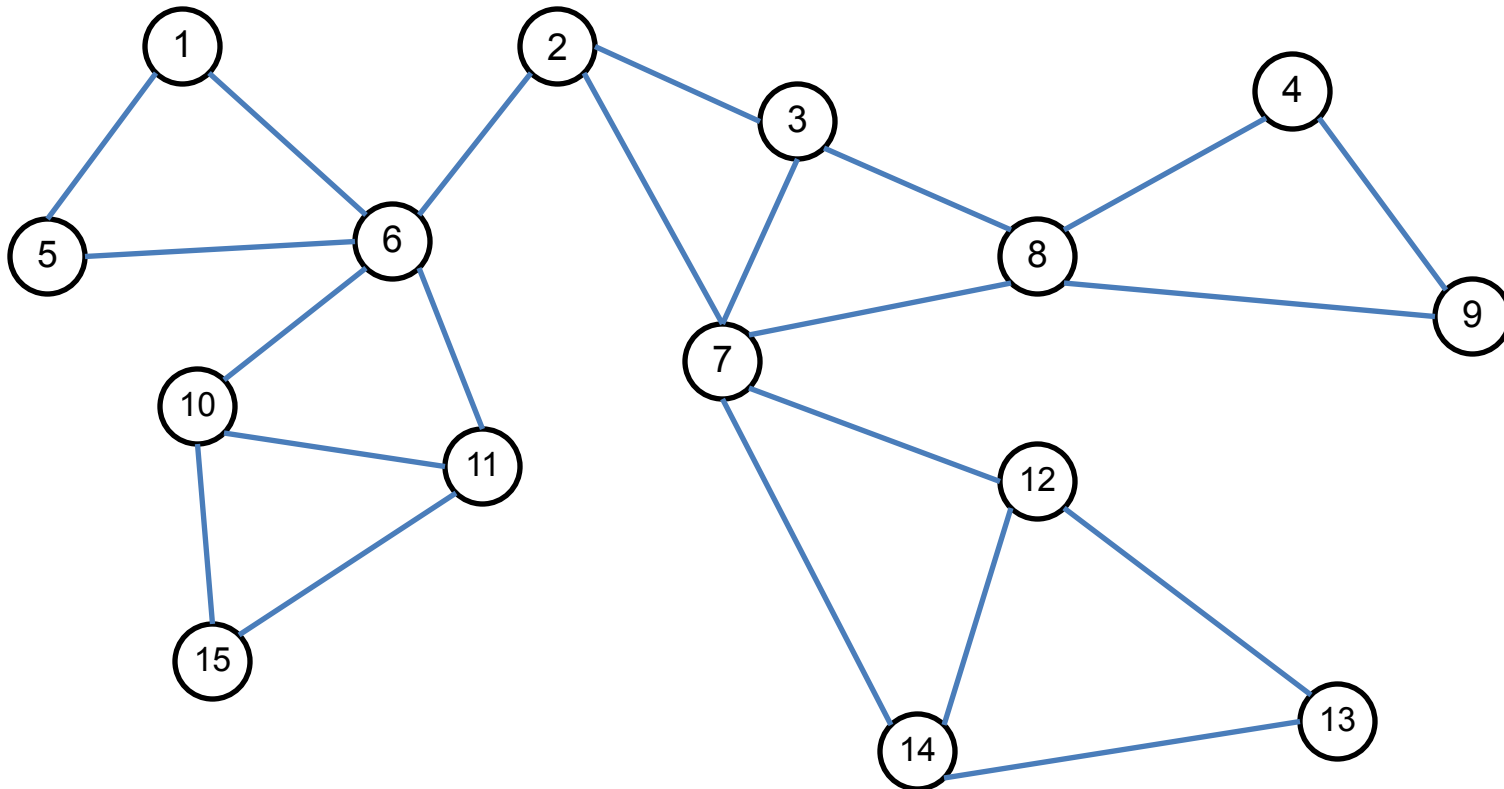
Пусть  $G = (V, E)$  — связный неориентированный граф. Узел  $a$  называют *точкой сочленения* графа  $G$ , если существуют такие узлы  $v$  и  $w$ , что  $v$ ,  $w$  и  $a$  различны и всякий путь между  $v$  и  $w$  содержит узел  $a$ .

Иначе говоря,  $a$  — точка сочленения графа  $G$ , если удаление узла  $a$  расщепляет  $G$  не менее чем на две части.

Граф  $G$  называется *двусвязным*, если для любой тройки различных узлов  $v$ ,  $w$ ,  $a$  существует путь между  $v$  и  $w$ , не содержащий  $a$ .

Таким образом, неориентированный связный граф двусвязен тогда и только тогда, когда в нем нет точек сочленения.

# Точки сочленения. Пример



Двусвязная компонента состоит из трех или более его вершин, в котором любые две вершины соединены, по крайней мере, двумя путями, не имеющими общих ребер.

Двусвязная компонента может представлять собой просто две вершины, соединенные одним ребром.

Двусвязная компонента - устойчивая часть графа: если в ней удалить вершину и все примыкающие к ней ребра, то любые две из оставшихся вершин по-прежнему оказываются соединенными между собой.

Если граф, образованный компьютерами сети, двусвязен, то сеть сохранит способность функционировать даже при выходе из строя одного из компьютеров.

При планировании авиаперевозок двусвязные компоненты графа гарантирует возможность переправить пассажиров по другому маршруту при закрытии аэропорта по погодным условиям.

Точки сочленения можно обнаружить с помощью грубой силы, удаляя из графа поочередно каждую вершину и пользуясь одним из наших алгоритмов обхода графа для проверки связности оставшейся части. Если оставшийся граф

связен, то удаленная вершина не является точкой сочленения.

При таком подходе мы должны выполнить  $n$  обходов графа с

$n$  вершинами, что потребует  $O(n^2)$  операций.

Однако сохраняя незначительное количество дополнительной информации при обходе графов, можно обнаружить все точки сочленения и компоненты двусвязности при единственном обходе.

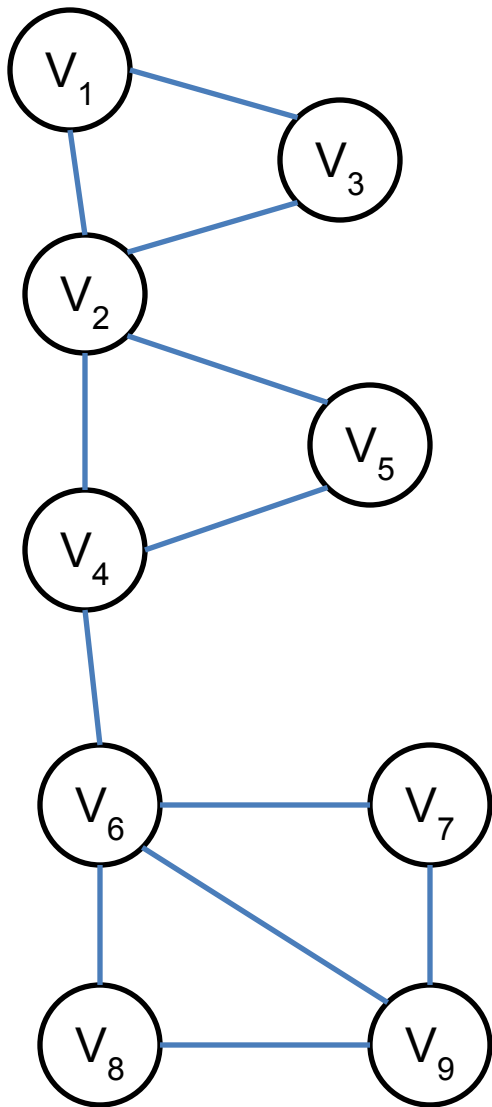
## Двусвязные компоненты

На множестве ребер графа  $G$  можно задать естественное отношение, полагая, что для ребер  $e_1$  и  $e_2$  выполняется это отношение, если  $e_1 = e_2$  или они лежат на некотором цикле.

Легко показать, что это отношение является **отношением эквивалентности**, разбивающим множество ребер графа  $G$  на такие классы эквивалентности  $E_1, E_2, \dots, E_k$ , что два различных ребра принадлежат одному и тому же классу тогда и только тогда, когда они лежат на общем цикле.

Для  $1 \leq i \leq k$  обозначим через  $V_i$  множество узлов, лежащих на ребрах из  $E_i$ . Каждый граф  $G_i = (V_i, E_i)$  называется **двусвязной компонентой** графа  $G$ .

# Пример

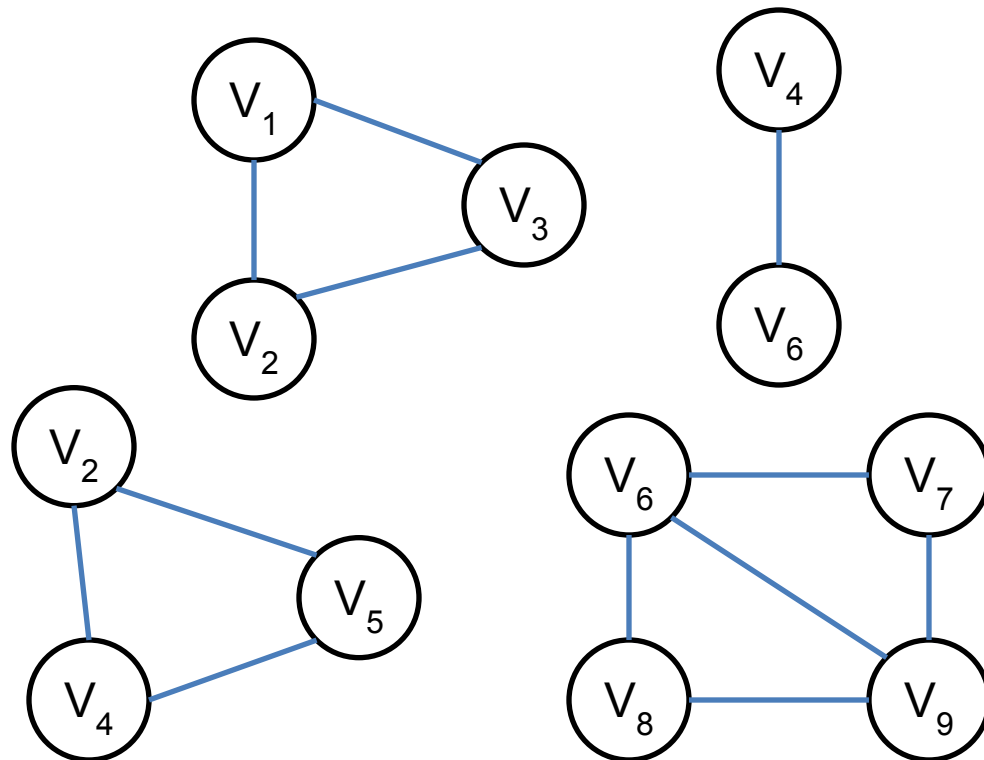


$$E_1 = \{ (v_1, v_2), (v_1, v_3), (v_2, v_3) \},$$

$$E_2 = \{ (v_2, v_4), (v_2, v_5), (v_4, v_5) \},$$

$$E_3 = \{ (v_4, v_6) \},$$

$$E_4 = \{ (v_6, v_7), (v_6, v_8), (v_6, v_9), (v_7, v_9), (v_8, v_9) \}$$





# Лемма 1.

Пусть  $G_i = (V_i, E_i)$  для  $1 \leq i \leq k$  — двусвязные компоненты связного неориентированного графа  $G = (V, E)$ .

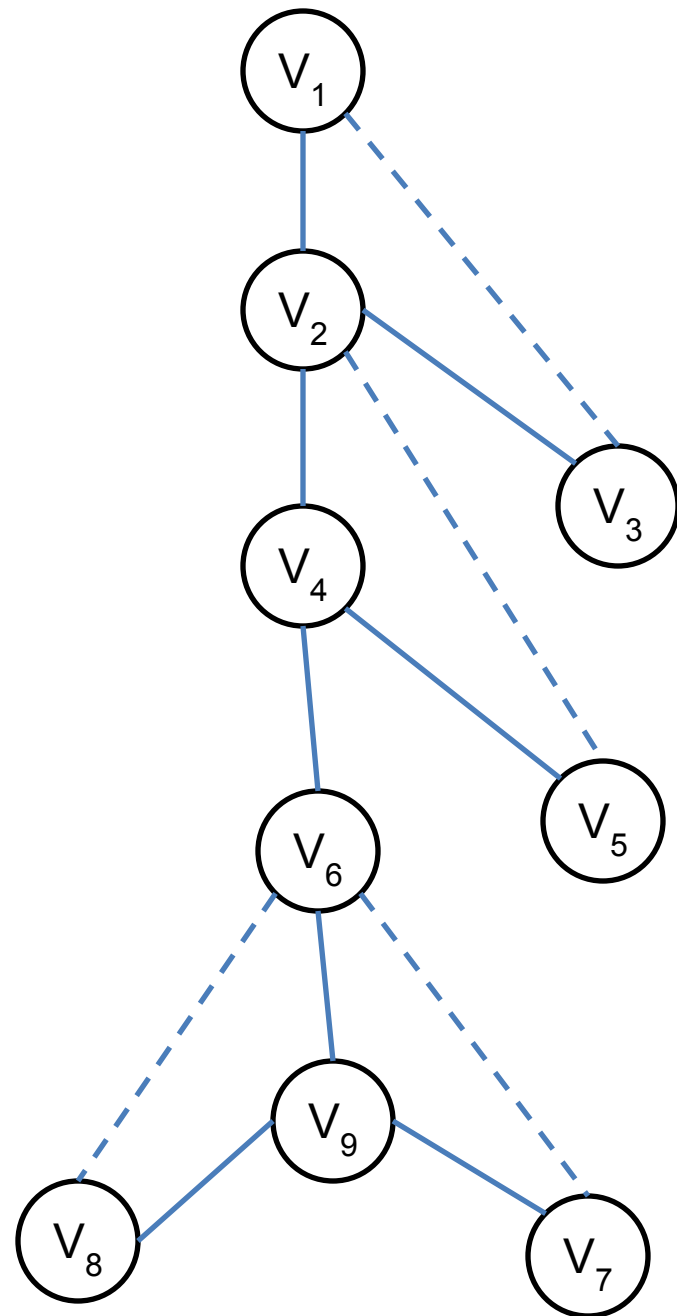
Тогда

- 1) граф  $G_i$  двусвязен для каждого  $i$ ,  $1 \leq i \leq k$ ;
- 2) для всех  $i \neq j$  пересечение  $V_i \cap V_j$  содержит не более одного узла;
- 3)  $a$  — точка сочленения графа  $G$  тогда и только тогда, когда  $a \in V_i \cap V_j$  для некоторых  $i \neq j$ .

## Лемма 2.

Пусть  $G = (V, E)$  — связный неориентированный граф, а  $S = (V, T)$  — глубинное остовное дерево для него. Узел  $a$  является точкой сочленения графа  $G$  тогда и только тогда, когда выполнено одно из условий:

- 1)  $a$  — корень и  $a$  имеет более одного сына;
- 2)  $a$  — не корень и для некоторого его сына  $s$  нет обратных ребер между потомками узла  $s$  (в том числе самим  $s$ ) и подлинными предками узла  $a$



# Нахождение двусвязных компонент и точек сочленения методом поиска в глубину

1. Для всех вершин  $v$  вычисляются числа  $dfnumber[v]$ . Они фиксируют последовательность обхода вершин глубинного остовного дерева в прямом порядке.
2. Для каждой вершины  $v$  вычисляется число  $dfnumber[v]$ ;  
 $low[v] = \min dfnumber[z], (v, z) \text{ – обратное ребро};$   
 $low[x], x \text{ – потомок } v.$
3. Точки сочленения определяются следующим образом:
  - корень остовного дерева будет точкой сочленения тогда и только тогда, когда он имеет двух и более сыновей;
  - вершина  $v$ , отличная от корня, будет точкой сочленения тогда и только тогда, когда имеет такого сына  $w$ , что  $low[w] \leq dfnumber[v]$ .

## Алгоритм нахождения двусвязных компонент и точек сочленения

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

*Выход.* Список ребер каждой двусвязной компоненты графа  $G$ .

*Метод.*

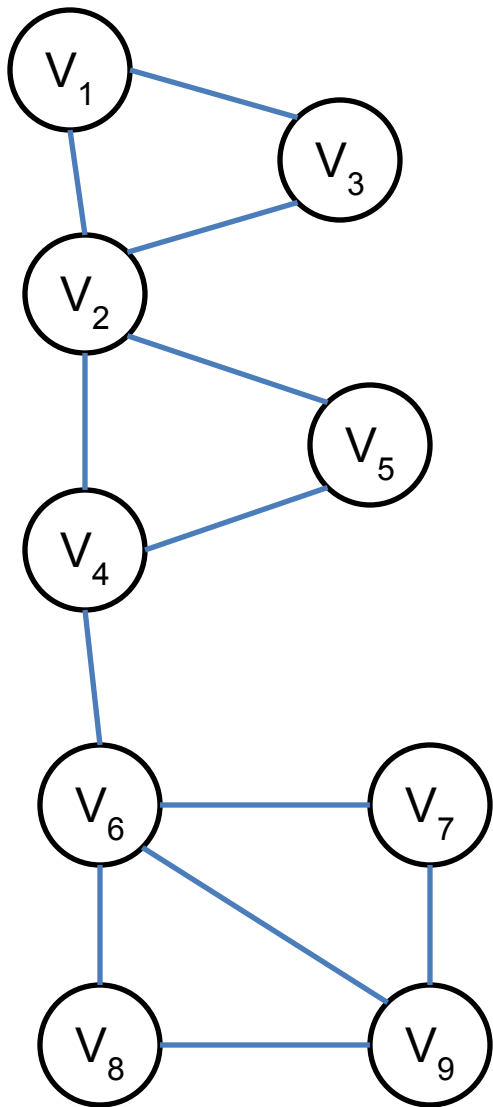
Вначале полагаем  $T = \emptyset$  и СЧЕТ=1. Помечаем каждый узел в  $V$  как "белый", выбираем произвольный узел  $v_0$  в  $V$ ,  $\Pi[v_0] = 0$  и вызываем Поиск\_дк( $v_0$ ), чтобы построить глубинное остовное дерево  $S = (V, T)$  и вычислить  $low[v]$  для каждого  $v$  из  $V$ .

# Процедура Поиск\_дк( $v$ )

Поиск\_дк( $v$ )

```
{  цвет [ $v$ ] ← серый;   $dfnumber[v] \leftarrow$  СЧЕТ++;  
   $low[v] \leftarrow dfnumber[v]$  ;  
  для  $\forall w \in$  смежные( $v$ ) выполнить  
  {  если (цвет[ $w$ ] == белый) то  
    {  поместить ( $v, w$ ) в СТЕК;  добавить ( $v, w$ ) к  $T$ ;   $p[w] \leftarrow v$ ;  
      Поиск_дк ( $w$ );  
      если  $low[w] \geq dfnumber[v]$  то  
      {  обнаружена двусвязная компонента:  
        вытолкнуть из СТЕКа все ребра вплоть до ребра ( $v, w$ ) ;  
      }  
       $low[v] \leftarrow \min ( low[v], low[w])$ ;  
    }  
  иначе  
    если ( $w \neq p[v]$ ) то  
    { если ( $dfnumber[w] < dfnumber[v]$  ) то поместить ( $v, w$ ) в СТЕК;  
       $low[v] \leftarrow \min ( low[v], dfnumber[w]$  )  
    }  
  }  
  цвет[ $v$ ] ← чёрный;  
}
```

# Пример



$low[1]= 1$

$low[2]= 2$

$low[3]= 2$

$low[4]= 4$

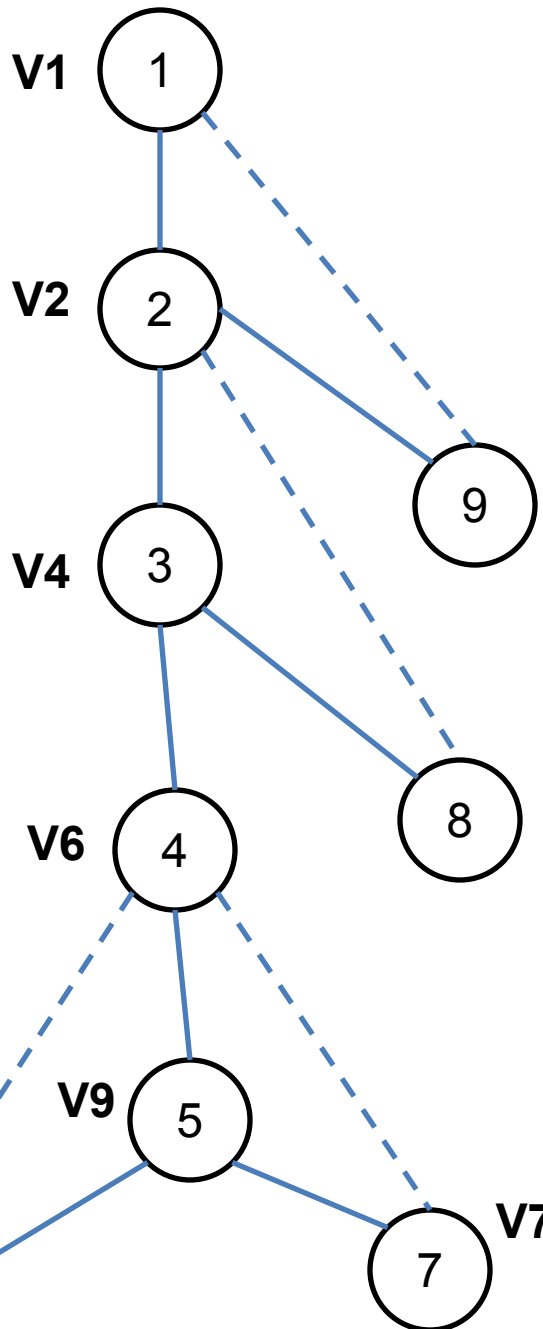
$low[5]= 5$

$low[6]= 6$

$low[7]= 7$

$low[8]= 8$

$low[9]= 9$



Сте

K

$(V_7, V_6)$

$(V_9, V_7)$

$(V_8, V_6)$

$(V_9, V_8)$

$(V_5, V_2)$

$(V_3, V_6)$

$(V_2, V_3)$

$(V_1, V_2)$

# Теорема

Алгоритм правильно находит двусвязные компоненты графа  $G$  с  $e$  ребрами и тратит на это время  $O(e)$ .