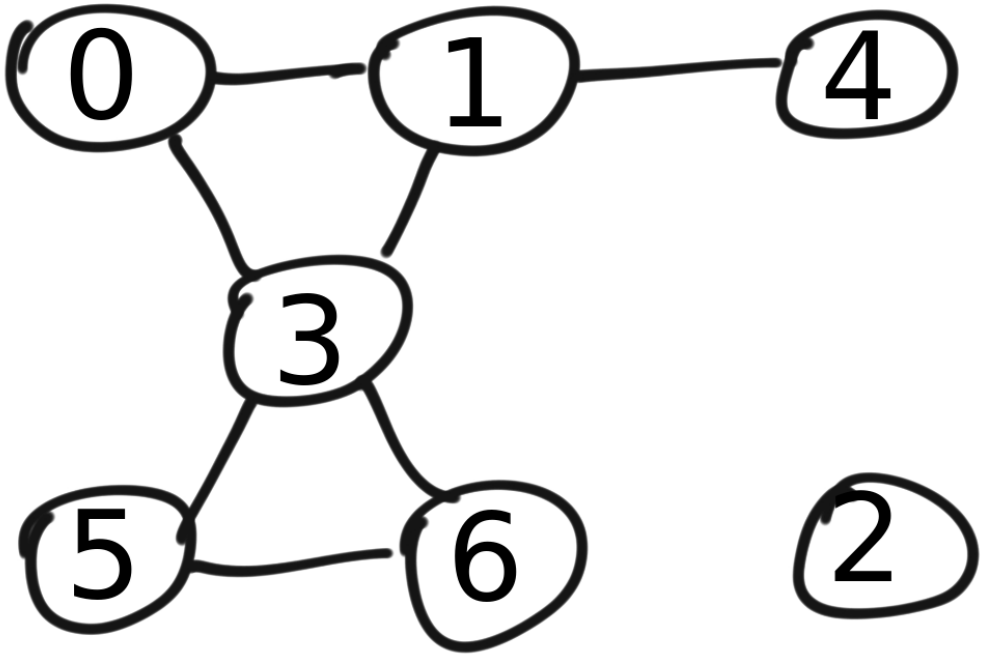
Точки сочленения

Articulation Points

Точки сочленения

Точкой сочленения в **неориентированном** графе G называется вершина, удаление которой (вместе со всеми инцидентными ей ребрами) приводит к **увеличению** числа компонент связности.



Точки сочленения и DFS

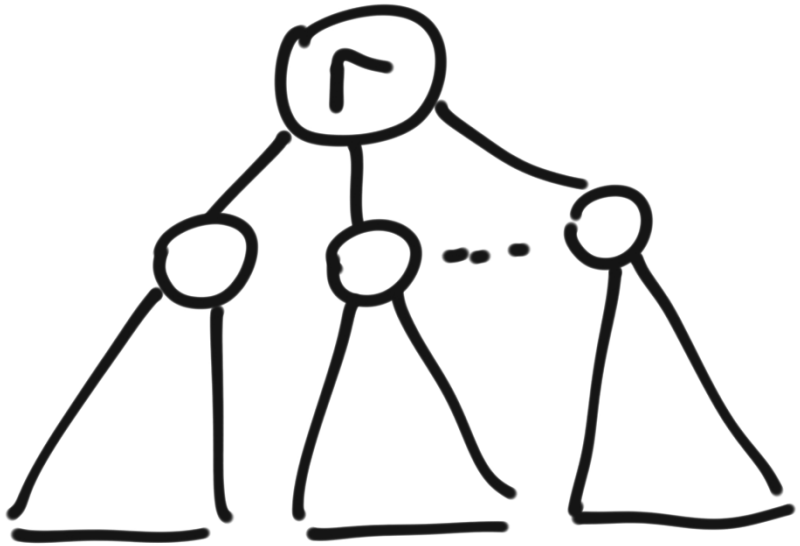
Теорема 1. *Пусть запустили алгоритм DFS со стартовой вершиной r. Тогда r - точка сочленения $\Leftrightarrow$ у r более одного сына в дереве обхода в глубину.*

Точки сочленения и DFS

Теорема 1. *Пусть запустили алгоритм DFS со стартовой вершиной r. Тогда r - точка сочленения $\Leftrightarrow$ у r более одного сына в дереве обхода в глубину.*

Доказательство.

У r более одного сына $\Leftrightarrow$ из первого сына r не посетили других сыновей r $\Leftrightarrow$ по **лемме о белых путях** из первого сына r нет пути в других, который бы не проходил через r $\Leftrightarrow$ r - точка сочленения. ■



Точки сочленения и DFS

Теорема 2. *Пусть запустили алгоритм DFS со стартовой вершиной r. Тогда $v \neq r$ - точка сочленения $\Leftrightarrow \exists u - \text{сын } v$: из u нельзя попасть в предка v, двигаясь только по ребрам дерева и максимум 1 обратному ребру.*

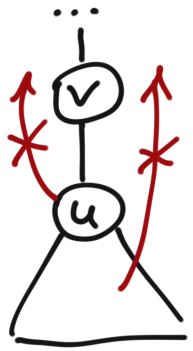
Точки сочленения и DFS

Теорема 2. *Пусть запустили алгоритм DFS со стартовой вершиной r. Тогда $v \neq r$ - точка сочленения $\Leftrightarrow \exists u - \text{сын } v$: из u нельзя попасть в предка v, двигаясь только по ребрам дерева и максимум 1 обратному ребру.*

Доказательство.

Из u нельзя попасть в предка v, двигаясь только по ребрам дерева и максимум 1 обратному ребру. $\Leftrightarrow$ из u совсем нельзя попасть в предков v, минуя v (так как ребер других типов нет - см. упражнение к лекции по DFS) $\Leftrightarrow$ удаление v приведет к потере связности между u и предками $v \Leftrightarrow v$ - точка сочленения.

■



Поиск точек сочленения: план

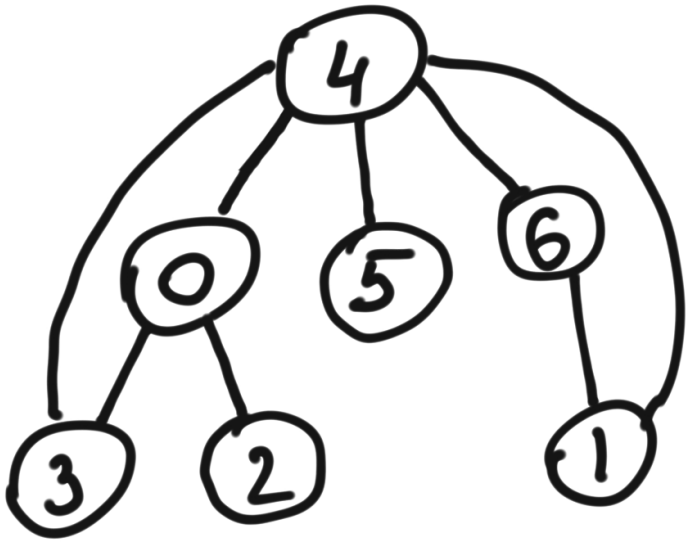
1. Запустим DFS из произвольной вершины r.
2. Если r смог найти хотя бы 2 белые соседние вершины, то r - ТС.
3. Если у какой-то из вершин $u \neq r$ нашелся ребенок, из которого нельзя попасть в предка u, двигаясь по ребрам дерева и максимум 1-му обратному, то u - ТС.
4. Если обошли не все вершины, то повторяем 1-3 для непосещенной компоненты.

Поиск точек сочленения : пункт 3

Как определить, что из ребенка u нельзя попасть в предка u?

Идея: для каждой вершины будем хранить "меру глубины" и "как высоко можно подняться". Тогда, если ребенок не может подняться выше родителя, то условие выполнено.

Хорошая новость: в качестве меры глубины можно использовать $time\_in$ (меньше - выше).

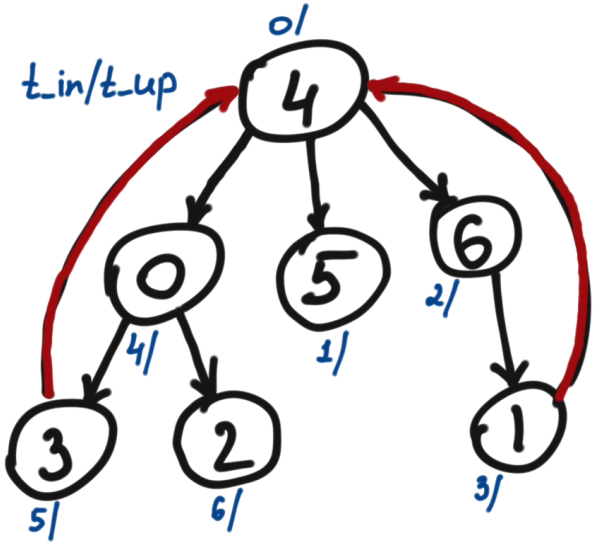


Поиск точек сочленения : как высоко подняться

Параллельно будем подсчитывать высоту, на которую можно подняться, двигаясь только по ребрам дерева и максимум 1-му обратному, - $time\_up$.

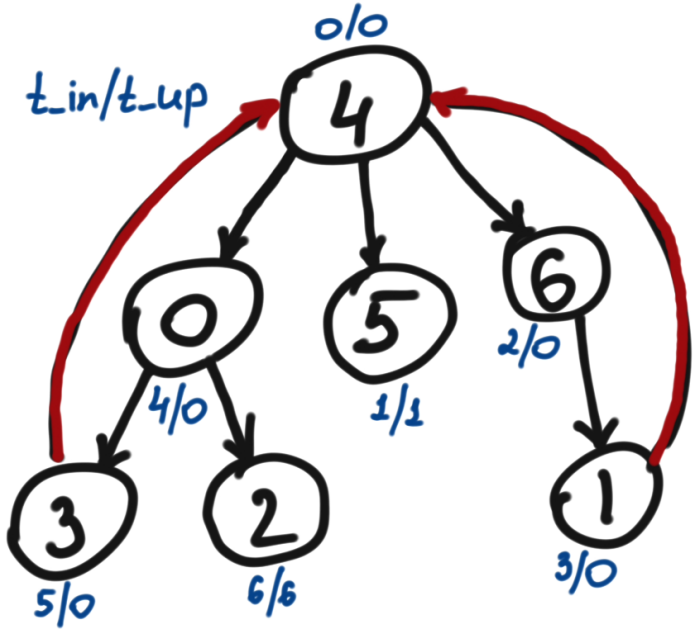
Легко понять, что $time\_up[v]$ - это минимум из

- $time\_in[v]$ (можно остаться на месте)
- $time\_in[back]$ (подняться по обратному ребру в $back$)
- $time\_up[tree]$ (спуститься по древесному ребру в $tree$ и подняться оттуда)



Поиск точек сочленения: критерий

1. Для корня обхода считаем количество детей (встретившихся белых вершин). Если больше 1, то значит - точка сочленения.
2. Условие недостижимости предков вершины v из ее ребенка u можно записать так: $time\_in[v] \leq time\_up[u]$



Точки сочленения: алгоритм

```
def DfsVisit(G, v, is_root):
  colors[v] = GRAY
  time_in[v] = time_up[v] = ++time
  n_children = 0
  for u in G.neighbors(v):
    if colors[u] == GRAY:  # back edge
      time_up[v] = min(time_up[v], time_in[u])
    if colors[u] == WHITE:
      ++n_children
      DfsVisit(G, u, false)
      time_up[v] = min(time_up[v], time_up[u])
      if !is_root and time_in[v] <= time_up[u]:
        articulation_points.add(v)
  if is_root and n_children > 1:
    articulation_points.add(v)
  colors[v] = BLACK
```

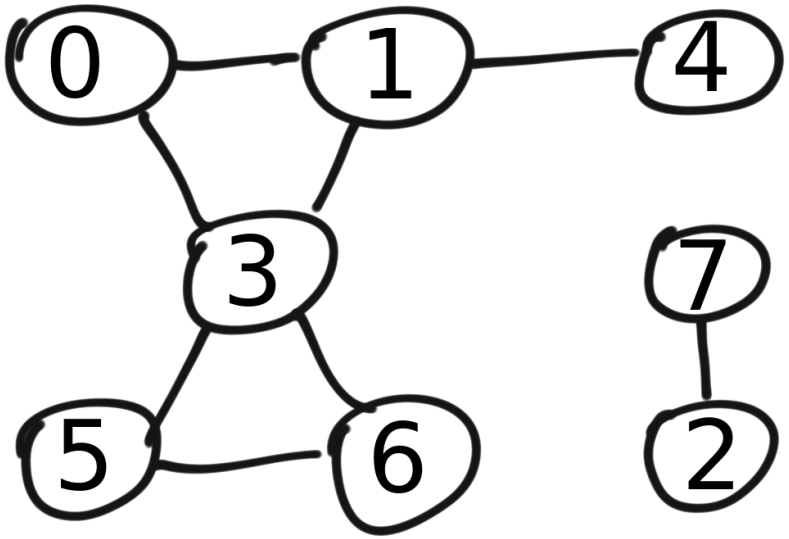
Время работы = Время работы DFS

Мосты

Bridges

Мост

Мостом в неориентированном графе называется ребро, удаление которого приводит к увеличению числа компонент связности.



Критерий моста

Теорема (критерий моста).

Пусть запустили обход в глубину. Ребро vu - мост $\Leftrightarrow$ v - родитель u в дереве обхода, и из u нельзя подняться выше себя, двигаясь только по ребрам дерева и обратным ребрам.

Критерий моста

Теорема (критерий моста).
Пусть запустили обход в глубину. Ребро vu - мост $\Leftrightarrow$ v - родитель u в дереве обхода, и из u нельзя подняться выше себя, двигаясь только по ребрам дерева и обратным ребрам.

Доказательство.
Из u нельзя подняться выше себя, двигаясь только по ребрам дерева и обратным ребрам $\Leftrightarrow$ из поддерева u нет пути в v и ее предков, кроме ребра vu (так как перекрестных ребер тоже нет) $\Leftrightarrow$ vu - единственный путь, соединяющий v и u $\Leftrightarrow$ vu - мост. ■

Поиск мостов: алгоритм

Таким образом, поиск мостов отличается от поиска точек сочленинея:

1. Отсутствием случая корня
2. Заменой условия на $time\_in[v] < time\_up[u]$

Поиск мостов: алгоритм

```
def DfsVisit(G, v):
  colors[v] = GRAY
  time_in[v] = time_up[v] = ++time
  for u in G.neighbors(v):
    if colors[u] == GRAY:  # back edge
      time_up[v] = min(time_up[v], time_in[u])
    if colors[u] == WHITE:
      DfsVisit(G, u)
      time_up[v] = min(time_up[v], time_up[u])
      if time_in[v] < time_up[u]:
        bridges.add(vu)
  colors[v] = BLACK
```

Время работы = Время работы DFS

Поиск мостов: подводные камни

- Просматривать ребро дерева в обратном направлении запрещено. То есть, если в вершину u пришли по ребру $e = (v, u)$, то серая вершина v не должна учитываться при подсчете $time\_up[u]$.
- Однако в случае мультиграфа первый пункт будет давать сбой. В таких ситуациях достаточно проверять является ли ребро кратным и удалять такие ребра из списка мостов.

Теорема Роббинса

Теорема (G.Robbins, 1939).

Неориентированный связный граф можно сильно ориентировать (задать ориентацию ребер так, что граф будет сильно связным) $\Leftrightarrow$ *В связном графе нет мостов.*

Теорема Роббинса

Теорема (G.Robbins, 1939).
Неориентированный связный граф можно сильно ориентировать (задать ориентацию ребер так, что граф будет сильно связным) $\Leftrightarrow$ В связном графе нет мостов.

Доказательство.
$\Rightarrow$ Допустим, есть мост vu. Сильно ориентируем граф. Без ограничения общности предположим, что vu ориентировано в направлении от v к u. В исходном графе не было пути из u в v, не проходящего по ребру vu $\Rightarrow$ теперь в графе совсем нет пути из u в v. Противоречие.

$\Leftarrow$ Ориентируем ребра согласно обходу DFS (ребра дерева в направлении предок-потомок, обратные - потомок-предок). Из любой вершины можно подняться выше нее самой $\Rightarrow$ из любой вершины можно подняться в корень $\Rightarrow$ из любой вершины можно добраться до любой другой. ■