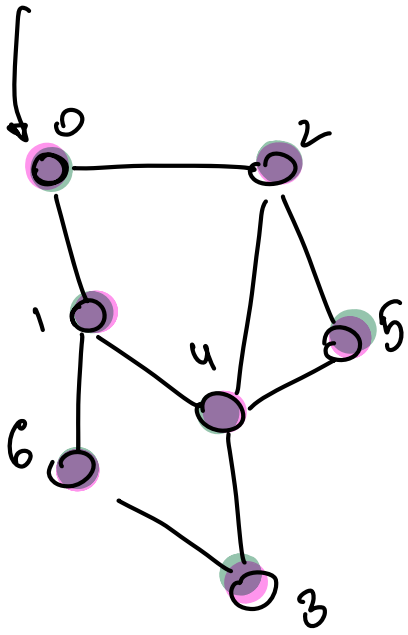
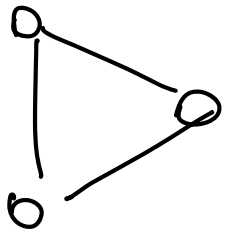


Обход графа



Алгоритм DFS

depth – first search

эйлеров обход

Обход в глубину

Опр Белая вершина – непосещенная вершина.

Опр Серая вершина – вершина, в кот-го DFS зашел, но не вышел

Опр Черная вершина – вершина из которой

DFS Волков.

Опр tin_v - время входа DFS в вершину V.

Опр $tout_v$ - время выхода DFS из вершины V.

Алгоритм

G ← граф [G.get-children() ...]

visited = [False, ... False] размера N

DFS изначально не посетил ни одной вершины

tin = [-1 .... -1]

tout = [-1 ... -1]

t = 0

функция DFS(G, start, visited)

{

visited[start] = true // рассмотрели вершину.

tin[start] = t ← время

t++;

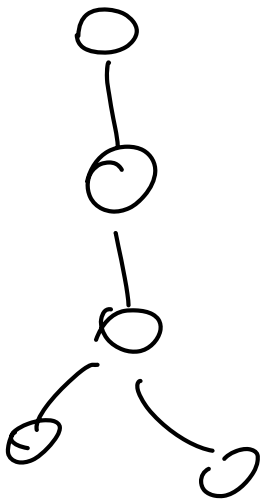
children = G.getChildren(start)

```
for  i = 0 .... children.size():
}
    если  visited[children[i]] → false
        ↳ DFS(G, children[i], visited)
}
tout[start] = t ↙ тогда время выхода
t++;
}
```

граф.

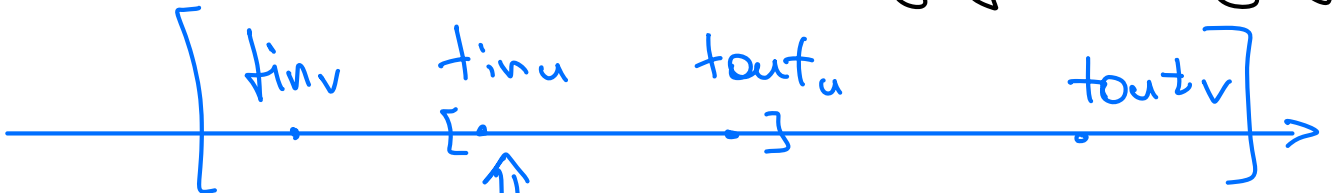
Асимптотика $T(G(V,E)) = O(|V| + |E|)$

Свойства:

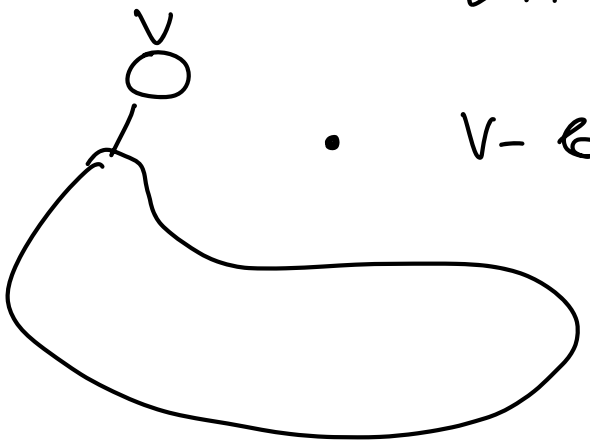


- u - предок v, если $tin_v \in [tin_u, tout_u]$
- $[tin_v; tout_v] \cup [tin_u; tout_u]$

либо не пересекаются, либо вложены друг в друга



v - предок u

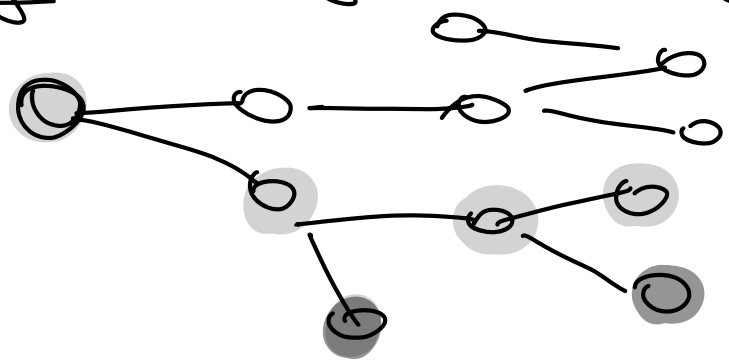


- tin - массив с номерами $[0 \dots N-1]$
- v - вершина, поддерево ее $(tout_v - tin_v)$ ← не увеличивается второй раз ↖

Для того чтобы за $O(1)$ находить предков и потомков

Лемма (о белых путях)
Возьмем вершину v графа $G(V, E)$, которая покрашена в серый цвет. Тогда все вершины, достижимые из v по белым путям покрасятся в черный цвет до выхода DFS из v.

Визуализация:



D-во:
по индукции.

Решаемые задачи

① Проверка на связность:

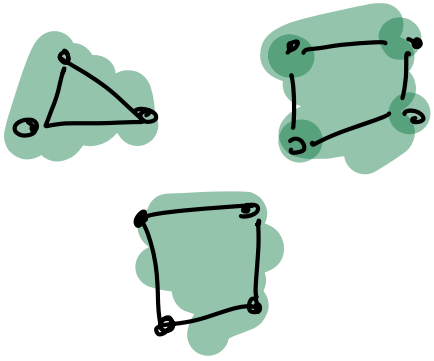
Напоминание: связность → ∃ путь из $v \to u$ $\forall v, u$

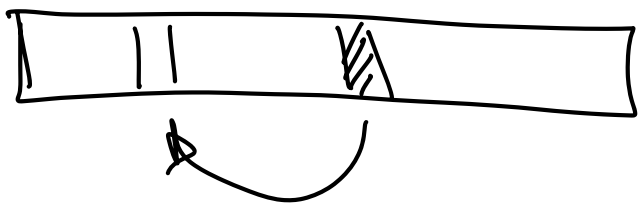
Решение:
→ DFS (0)
→ если в visited есть false
⇒ он несвязный

② Поиск кол-ва компонент связности

компонента связности: связный подграф

```
for i=0... N-1:
  if ( visited[i] → false ) {
        DFS( G, i, visited);
        count_comp ++;
  }
}
```

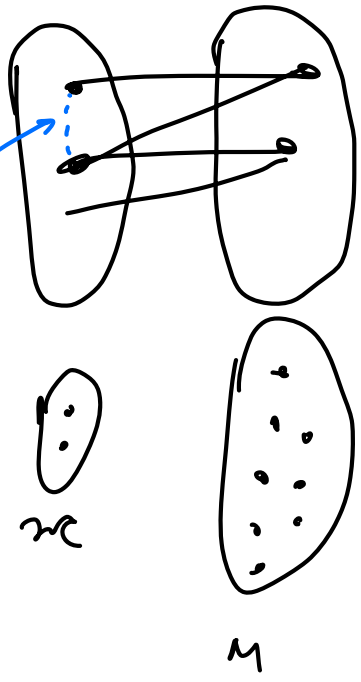


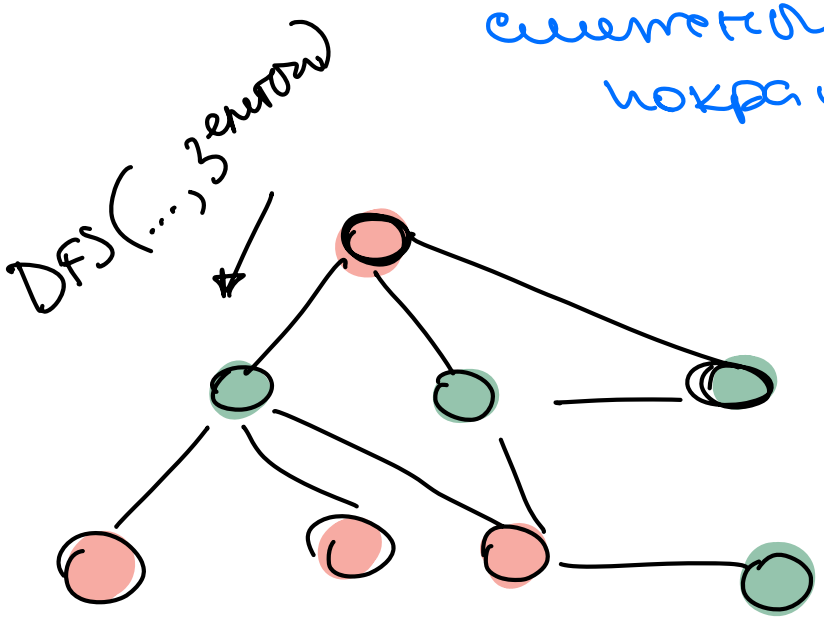


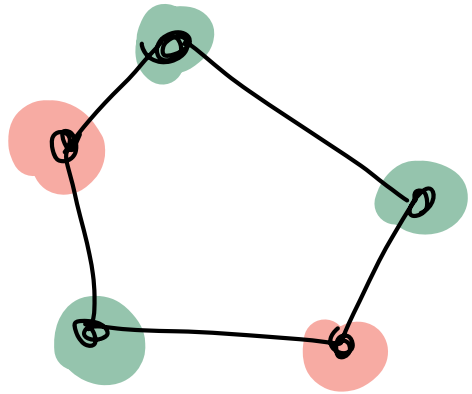
③ Проверка графа на двудольность.

Граф двудольный ⇔ Можно "правильно" раскрасить в 2 цвета.

(правильная раскраска – смежные вершины покрашены в разные цвета)







+1 → красный

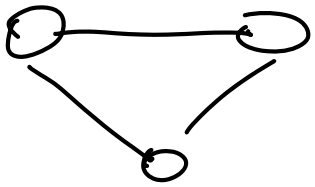
-1 → зеленый

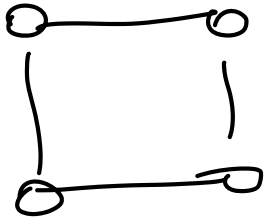
colors = [0, ... 0]

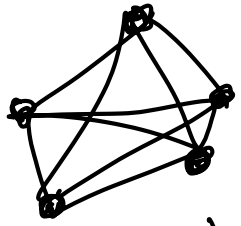
```
DFS(G, start, colors, color)
    colors[start] = color
    children = G.getChildren(start)
    for i -0... children.size)
    }
        если  colors [children[i]] == 0
            ↳ DFS(G, children[i], colors, -color)
        иначе:
            если  color[children[i]] == color:
                ↳ граф не двудольный,
                    return
                    exit()
    }
```

④ Хроматическое число графа

минимальная правильная раскраска графа

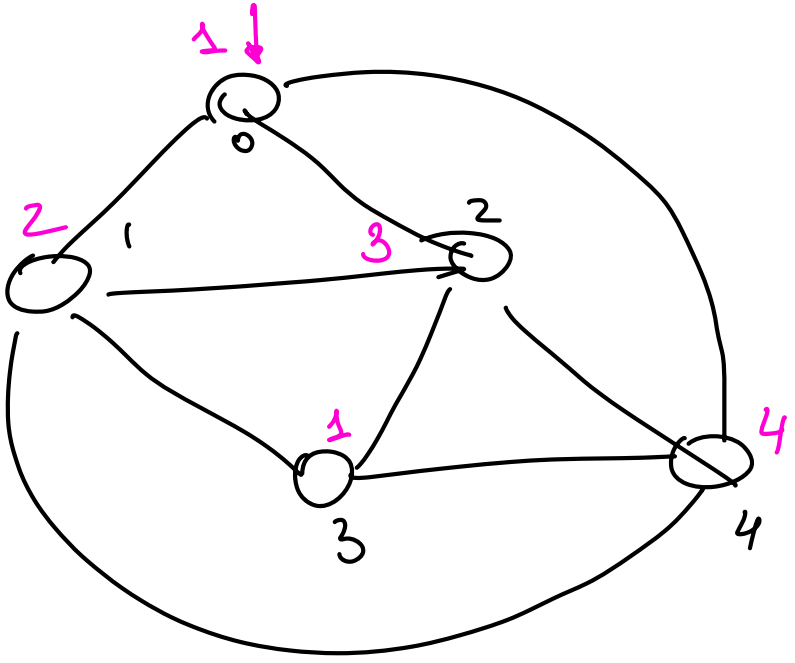






$\chi(G) = 5$

Жадный алгоритм раскраски: красим в минимально возможный цвет.



Проблема: найти минимально возможный цвет.

→ ch = G.get_children()

ch_color ← цвета детей

ch_color.sort()

→ выбираем первый

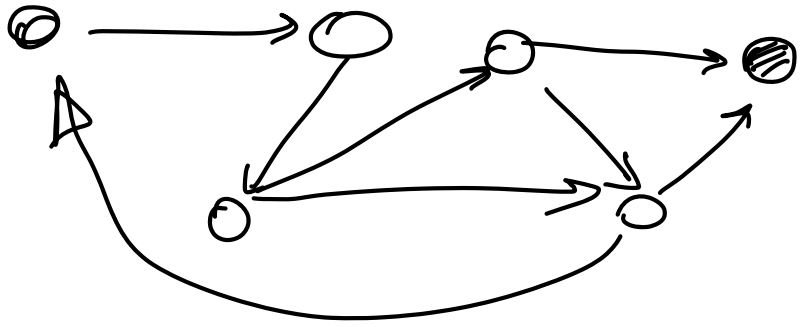
цвет, которого нет.

⑤ Поиск цикла:

ориентированный граф

если мы попали в вершину, кот-я является серой. [tin(v) tout]

↳ цикл есть.



неорграф

—//— visited, но хранится родитель вершины

DFS(..., parent)