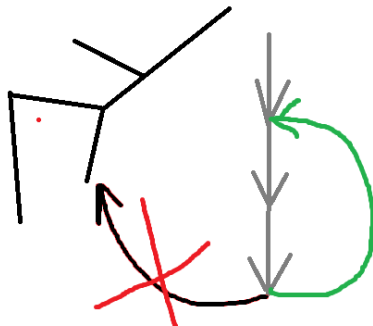


34. Алгоритм dfs на неориентированном графе. Дерево обхода dfs. Классификация рёбер на древесные и обратные. Проверка связности и ацикличности. Компоненты связности

В отличие от ориентированного графа, в неориентированном не будет рёбер в черные вершины, поскольку на графе нет ориентации. Когда мы обходим граф, у нас могут быть только рёбра в серые вершины.



Это - дерево dfs. Рёбра, идущие в порядке обхода dfs будем называть *древесные рёбра*, а те, что не были посещены dfs (они вели в серые вершины) - *обратные*

```
vector<vector<int>> g;
vector<bool> used;

void dfs(int v, int p = -1){
    used[v] = true;
    for(int to: g[v]){
        if(!used[to])
            dfs(to,v)
    }
}
```

Проверка на связность

Выберем какую-нибудь вершину в графе, запустим dfs из нее и проверим, что dfs обошел все вершины графа

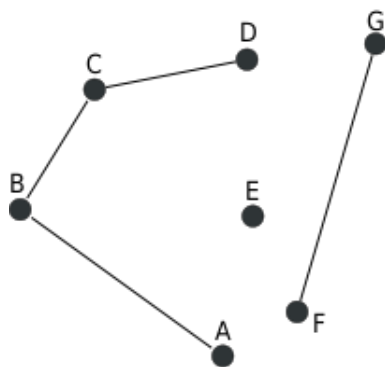
Проверка на ацикличность

Будем искать цикл почти так же, как и в ориентированном случае, но теперь нас интересует ребро не в серую вершину, а в любую использованную, не являющуюся непосредственным родителем текущей

Введем отношение \sim . $u \sim v$, если между вершинами u и v есть путь. Очевидно, это отношение является отношением эквивалентности

Тогда граф распадается на классы эквивалентности, называемые *компонентами связности*

Определение *Компонента связности графа* G (или просто компонента графа G) — максимальный (по включению) связный подграф графа G .



Здесь 3 компоненты связности