Графы. Базовые алгоритмы. 4.01.2015

Неориентированный граф

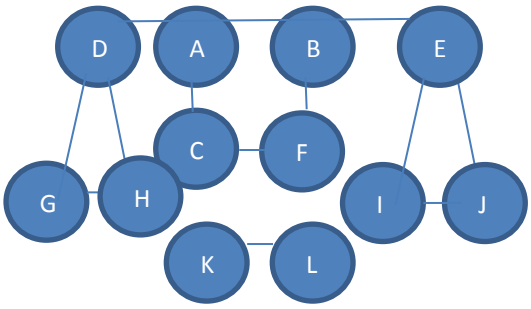


Рис. Исходный граф

void dfs (int v) Сложность O(|V|+|E|)

```
{
        used[v]=true; // вершина посещена
        cnum[v]=Ncomp; //№ компоненты связности для данной вершины
        pre[v]=clock; //время начала обработки вершины
        clock++;
        for (i=0; i<g[v].size(); i++)
        {
                if (!used[g[v][i]])
                dfs (g[v][i])
                post[v]=clock; // время окончания обработки вершины
                clock++;
        }
Ncomp=0;
        For(j=0; j<n; j++)
        If (!used[j]) { Ncomp++; dfs(j);}
```

Поиск в глубину

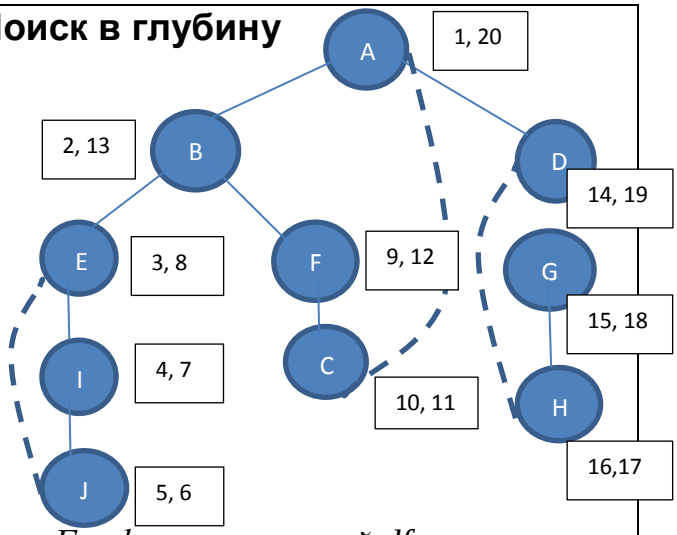


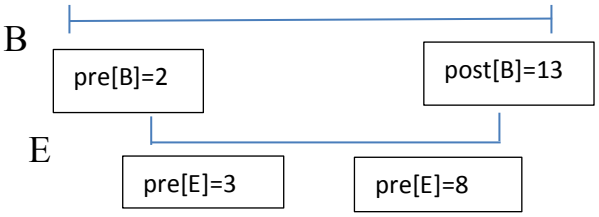
Рис. Граф, построенный dfs

Сплошные ребра ведут в ранее не встречающиеся вершины. Образуют дерево (связный граф без циклов) – *tree edges*

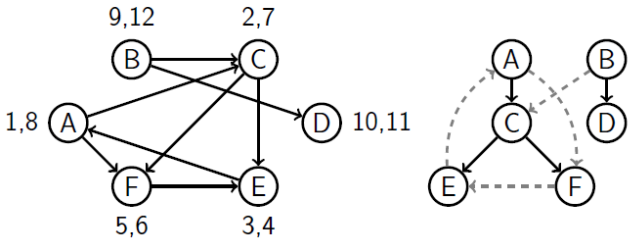
Пунктирные ребра ведут в вершины, которые уже встречались – обратные ребра *back edges*

Лемма

Для любых двух вершин u и v либо отрезки [pre(u),post(u)] и [pre(v),post(v)] не пересекаются, либо один содержатся в другом.

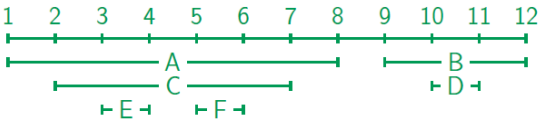


Ориентированный граф



Поиск в глубину. Используется DFS как для неориентированного графа. Ребра ориентированного графа:

- ребра дерева tree edges (AC, CE, CF, BD)
- обратные ребра back edges (EA)
- прямые ребра forward edges ведут от вершины к потомку, не являющемуся ребенком. (AF)
- перекрёстные ребра cross edges от вершины к другой вершине, не являющейся ни предком, ни потомком (BC, FE)

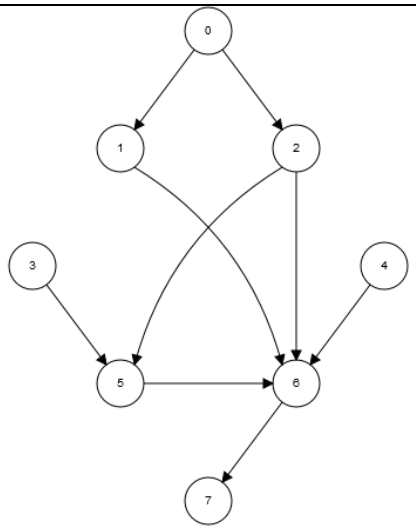


Лемма. Ориентированный граф содержит циклы тогда и только тогда, когда при поиске в глубину обнаруживается обратное ребро.

Топологическая сортировка – упорядочение вершин графа в таком порядке, при котором каждое ребро соединяет вершину с меньшим номером в вершину с большим номером.
0 4 3 2 5 1 6 7 - порядок вершин при топологической сортировке.

Ориентированный граф может быть топологически упорядочен тогда и только тогда, когда в нем нет циклов.

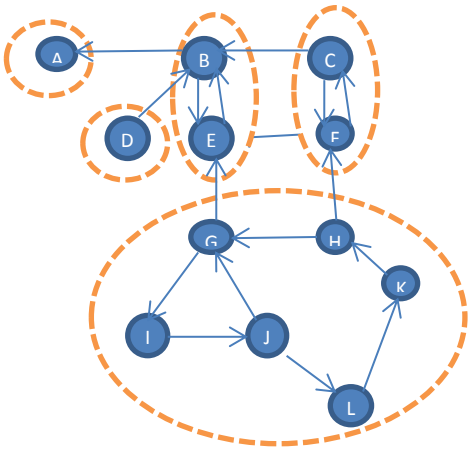
Поиск в глубину расставляет post значения. Топологически отсортированные вершины располагаются в порядке убывания post значений.



0 – исток(source). В него не ведет ни одно ребро
7 – сток (sink). Из него не выходит ни одного ребра

У каждого ациклического ориентированного графа есть хотя бы один исток и хотя бы один сток.

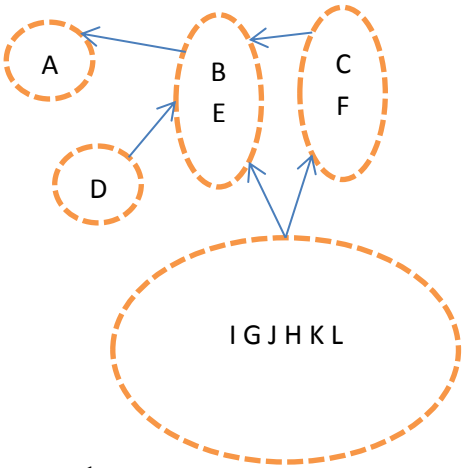
Исходный ориентированный граф
Вершины u и v связаны если из u достижима v и из v достижима u



Компоненты сильной связности состоят из вершин, каждая из которых достижима из любой другой в этой компоненте.

Если запустить dfs из вершины, находящейся в стоке метаграфа (A), то он обойдет только вершины стока (A). Далее удаляем эту вершину-сток метаграфа и снова запускаем dfs от вершины-стока (B,E) и т.д.

Метаграф (Конденсация графа, Граф компонент сильной связности)



Метаграф является ациклическим.
Свойство. Пусть C и C1 – компоненты сильной связности графа и в графе есть ориентированное ребро из C в C1. Тогда максимальное post-значение вершин в C больше, чем максимальное значение вершин в C1.

Алгоритм поиска компонент сильной связности

1. Построить транспонированный граф G^R (это позволит найти вершины в компоненте-стоке метаграфа). Для транспонирования строится второй граф с измененным направлением ребер.
2. Расставить post значения в вершинах транспонированного графа.
3. Применить поиск dfs **в исходном графе** из вершин в порядке убывания их post значений, найденных на втором шаге.