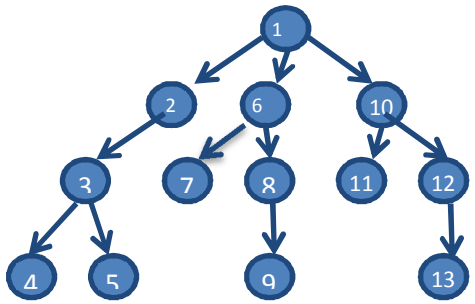
Прикладные задачи алгоритма DFS (DFS – поиск в глубину). O (n+m)

Поиск в глубину

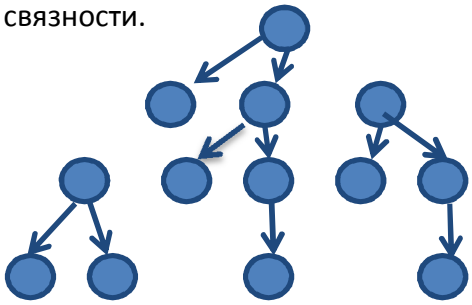
```
vector <vector<int>> g;        // граф
int n;                         // число вершин
vector <char> used;            // массив меток вершин
void dfs ( int v);
        {
                used[v]=true;
                for (i=0; i< g[v].size(); ++i)
                        if ( ! used[g[v][i]]) dfs (g[v,i])
        }
```



Подсчет количества компонент связности

```
void dfs(start)
{ used [start]=true;
        for (v=0; v< g[start].size(); ++v)
                if (! used [g[start][v]])
                        dfs( g[start][v])
}
Ncomp=0;
for (i=1; i<=n; ++i)
        if ( ! used [i])
                {       ++Ncomp;
                        dfs(i);
                }
```

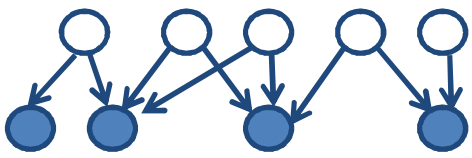
Обходим граф и проверяем, была ли вершина покрашена ранее. Если нет, то мы нашли новую компоненту связности. Граф G, изображенный на рисунке имеет три компоненты связности.



Проверка графа на двудольность

```
void dfs(start)
{       for (u=0; u< g[start].size(); ++u)
                if (color[g[start][u]]==0)
                        {color[g[start][u]]=3-color[start];
                        dfs( g[start][u]);
                        }
                else if (color[g[start][u]]==color[start])
                        Two=false;
}
Two=true;
for (i=1; i<=n; ++i)
        if ( color[i]==0)
                {       color[i]=1;
                        dfs(i);
                }
```

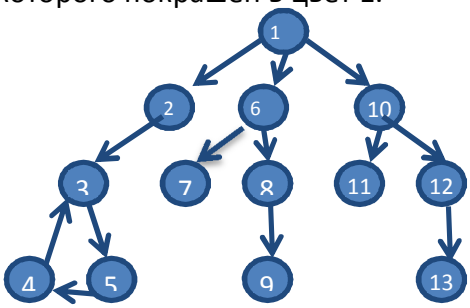
Двудольный граф – граф, вершины которого можно разбить на два множества так, чтобы концы каждого ребра принадлежали различным множествам.
Для проверки делаем раскраску графа в два цвета {1,2}. 0 – цвет непосещенной вершины.
Если встречаем ребро с концами одного цвета, то граф не является двудольным.



Поиск цикла в ориентированном графе

```
void dfs(start)
{       color[start]=1;
        for (u=0; u< g[start].size(); ++u)
                if (color[g[start][u]]==0)
                        dfs(g[start][u]);
                else if (color[g[start][u]]==1)
                        cycle=true;
        color[start]=2;
}
cycle=false;
for (i=1; i<=n; ++i)
        if ( color[i]==0) dfs(i);
```

Цикл – замкнутая цепь в графе. *Цепь* – маршрут, в котором все ребра различны. *Маршрут* – чередующая последовательность вершин и ребер графа.
При обходе используем три цвета вершин. 0 – не посещенная, 1 – в процессе обработки., 2 – обработанная вершина.
Цикл существует, если поиск в глубину обнаруживает ребро, конец которого покрашен в цвет 1.



Топологическая сортировка

```
void dfs(start)
{
        for (u=0; u< g[start].size(); ++u)
                if (color[g[start][u]]==0)
                        dfs(g[start][u]);
        color[start]=2;
        ans.push_back(start);
}

for (i=1; i<=n; ++i)
        if ( color[i]==0) dfs(i);

//Вывод вектора ans  в противоположном порядке
```

Дан ориентированный граф, не содержащий циклов (предварительно необходимо проверить отсутствие циклов). ТС упорядочивает вершины графа так, что все ребра идут от вершины с меньшим номером к вершине с большим номером.
В процедуре DFS при выходе из вершины ее номер добавляется в конец вектора с ответом.
Ответ выводится в противоположном порядке.