Тема: Компоненты связности (найти количество компонент связности графа, вывести компоненты списком вершин согласно нумерации в матрице смежности).

Для начала рассмотрим алгоритм DFS поиска в глубину, на котором базируется алгоритм решения задачи. Существует множество вариантов алгоритма DFS, в зависимости от задачи: формирование леса поиска по исходному графу, расстановка меток времени и т.д. В данном случае нам нужно в процессе обхода графа, начав с определенной вершины, запомнить все вершины, до которых дойдет алгоритм, тем самым будет сформирована компонента связности этой вершины. DFS использует стек для хранения вершин в процессе обхода, в данном случае будем использовать стек вызовов для этой цели, то есть алгоритм DFS будет рекурсивным. Таким образом, функцию $DFS(v, visited, matrix)$ можно представить в виде следующих шагов:

Пусть G - неориентированный граф.
{**In:** вершина $v \in V(G)$; множество $visited$ посещенных вершин (при первом вызове пустое); матрица смежности $matrix$ графа G}
{**Out:** множество всех вершин, до которых дошел алгоритм в процессе обхода}

(1) Вершина v добавляется в множество $visited$;

(2) С помощью матрицы смежности обходятся все вершины, смежные с v;

(3) Для очередной вершины u, смежной с v и не лежащей в множестве $visited$, запускается: $visited = DFS(u, visited, matrix)$;

(4) Возвращается множество $visited$ всех посещенных в процессе обхода вершин.

Описание алгоритма поиска компонент связности $factorize(Q, matrix)$**:**
{**In:** множество Q всех вершин графа G, матрица смежности $matrix$ графа G}
{**Out:** множество всех компонент связности}
Пусть $result$ - множество компонент связности (инициализируется пустым множеством).

(1) Из Q извлекается вершина $v = Q[0]$. Запускается $component = DFS(v, \{\}, matrix)$. Функция DFS полностью определяет компоненту связности $component$, в которой находится вершина v;

(2) $component$ добавляется в $result$;

(3) Из Q удаляются все вершины, которые находятся в $component$;

(4) Если Q не пусто, то исполнение возвращается к шагу (1). В противном случае возвращается множество $result$.