Обход графа. Обход в глубину (DFS). Обход в ширину (BFS).

Обход графа

- **Обход графа** (graph traversal) – это процедура перебора (посещения) всех вершин графа начиная с заданной

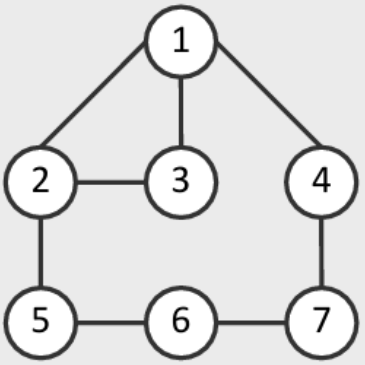
Обход в глубину (DFS)

- **Поиск в глубину** *(depth-first search – DFS)* – процедура посещения всех вершин графа начиная с заданного узла **v**
- Вершины обрабатываются от начальной до самой «глубокой»
- Например, при поиске выхода из лабиринта, всегда поворачивать направо, доходить до тупика, а потом возвращаться до ближайшей развилки.
- Обход в глубину графа, представленного **матрицей смежности**, имеет трудоёмкость **O(|V^2|)**
- Обход в глубину графа, представленного **списком смежности**, имеет трудоёмкость **O(|V| + |E|)**

```
void dfs(int v) {
    visited[v] = true; // Помечаем вершину как посещенную
    printf("Посещена вершина: %d\n", v); // Обработка вершины (можно
заменить на нужную логику)

    // Перебираем все смежные вершины
    for (int u = 0; u < vertex_count; u++) {
        // Если есть ребро и вершина не посещена
        if (adjacency[v][u] && !visited[u]) {
            dfs(u); // Рекурсивный вызов для смежной вершины
        }
    }
}
```

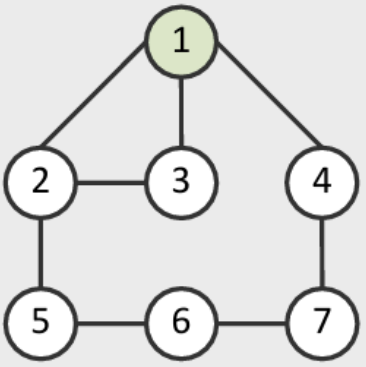
Пример работы:

```
function DFS(v)
    visited[v] = true
    for each u in Adj(v) do
        if visited[u] = false then
            DFS(u)
        end if
    end for
end function
```



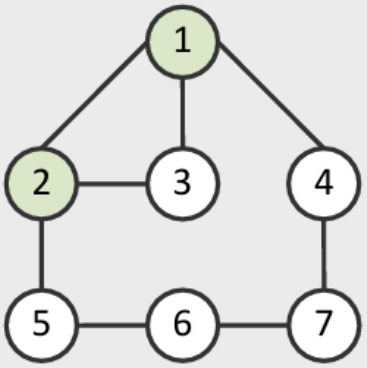
34

```
function DFS(1)
    visited[1] = true
    for each u in Adj(1) do
        if visited[2] = false then
            DFS(2)
        end if
    end for
end function
```

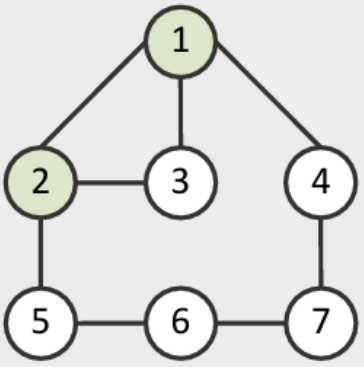


35

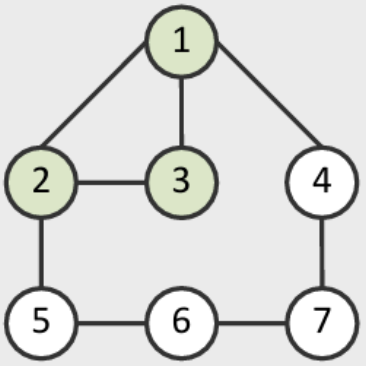
```
function DFS(2)
    visited[2] = true
    for each u in Adj(2) do
        if visited[1] = false then
            DFS(u)
        end if
    end for
end function
```



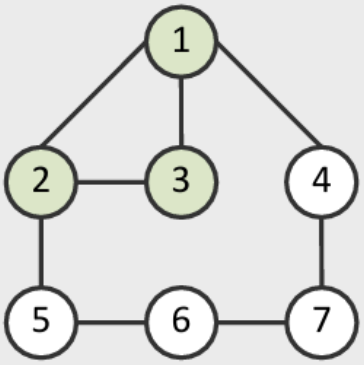

```
function DFS(2)
    visited[2] = true
    for each u in Adj(2) do
        if visited[3] = false then
            DFS(3)
        end if
    end for
end function
```



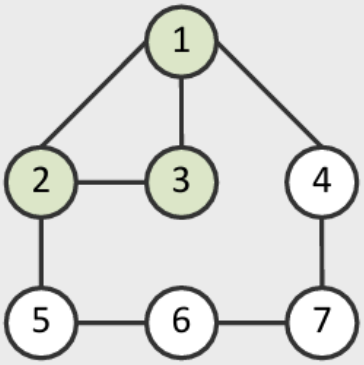
```
function DFS(3)
    visited[3] = true
    for each u in Adj(3) do
        if visited[1] = false then
            DFS(u)
        end if
    end for
end function
```



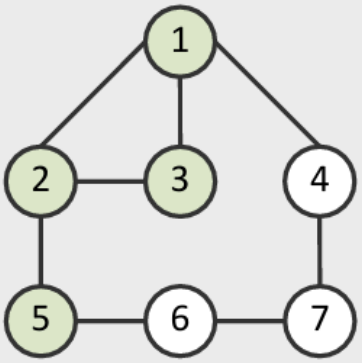

```
function DFS(3)
    visited[3] = true
    for each u in Adj(3) do
        /* Все узлы, смежные с 3,
           посещены. Возвращаемся к 2. */
        if visited[2] = false then
            DFS(u)
        end if
    end for
end function
```



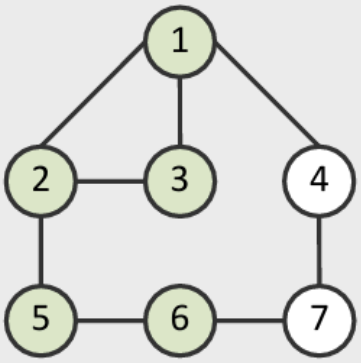
```
function DFS(2)
    visited[2] = true
    for each u in Adj(2) do
        if visited[5] = false then
            DFS(5)
        end if
    end for
end function
```



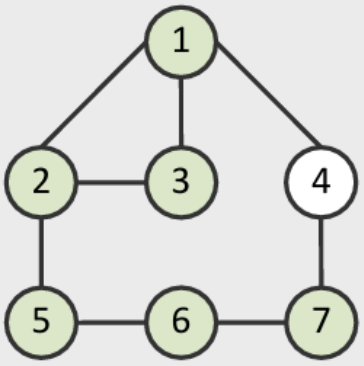

```
function DFS(5)
    visited[5] = true
    for each u in Adj(5) do
        if visited[6] = false then
            DFS(6)
        end if
    end for
end function
```



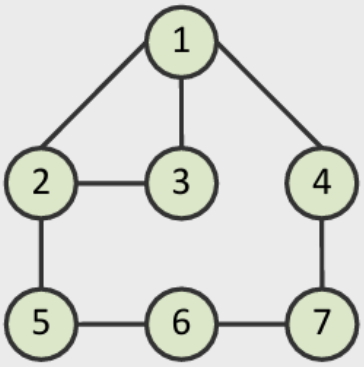
```
function DFS(6)
    visited[6] = true
    for each u in Adj(6) do
        if visited[7] = false then
            DFS(7)
        end if
    end for
end function
```




```
function DFS(7)
    visited[7] = true
    for each u in Adj(7) do
        if visited[4] = false then
            DFS(4)
        end if
    end for
end function
```

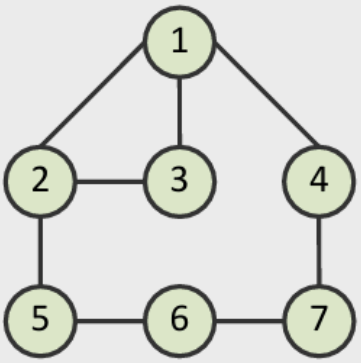


```
function DFS(4)
    visited[4] = true
    for each u in Adj(4) do
        if visited[1] = false then
            DFS(u)
        end if
    end for
end function
```



44

```
function DFS(4)
    visited[4] = true
    for each u in Adj(4) do
        if visited[7] = false then
            DFS(u)
        end if
    end for
end function
```



45

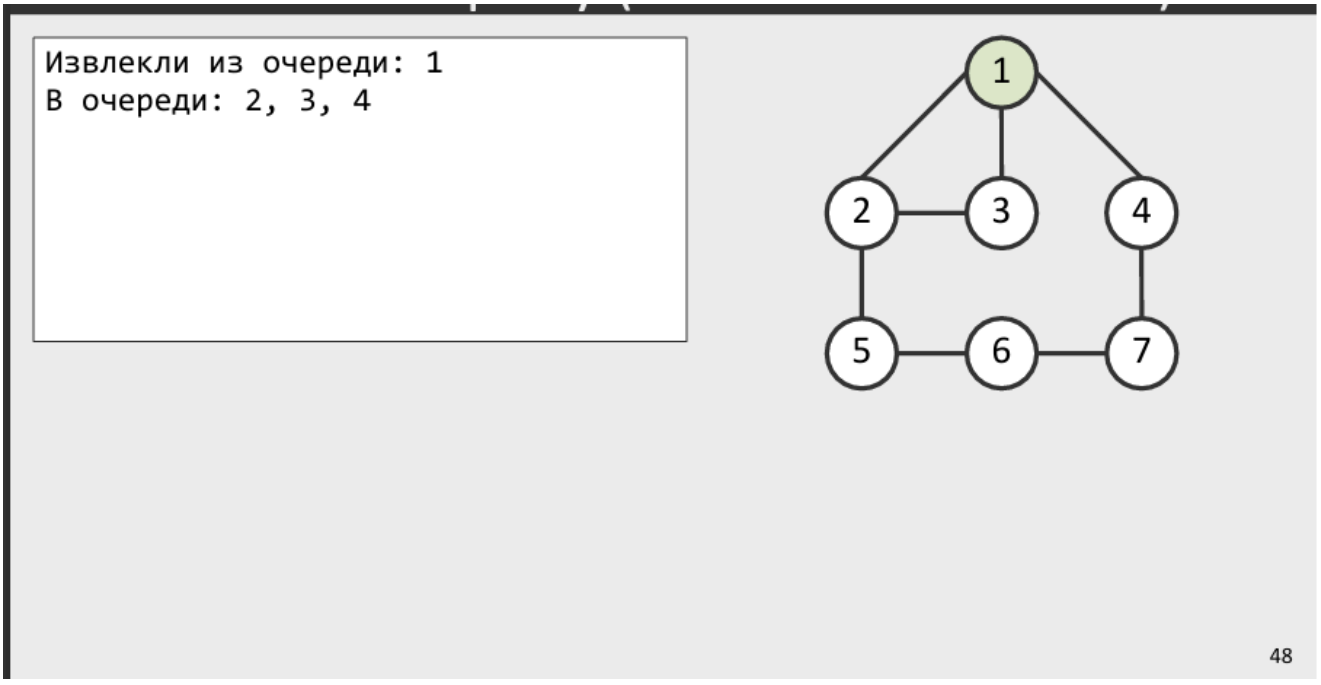
Обход в ширину (BFS)

- Поиск в ширину (breadth-first search, BFS) — процедура посещения всех вершин графа, начиная с заданного узла v
- Обрабатываются все смежные вершины начальной, потом обрабатываются все вершины первой из смежных с начальной, потом второй…
- При поиске выхода из лабиринта на каждой развилке проверять все варианты поворотов, потом все варианты одного из поворотов…
- Обход в ширину графа, представленного матрицей смежности, имеет трудоёмкость **O(|V^2|)**

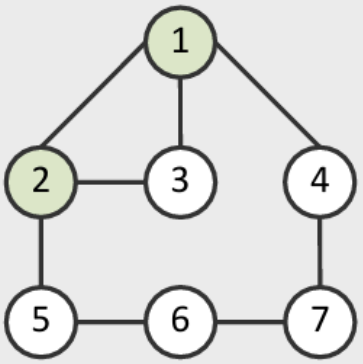
- Обход в ширину графа, представленного списком смежности, имеет трудоёмкость **O(|V| + |E|)**

```c
void bfs(int start) {
    Queue q;
    initQueue(&q);

    visited[start] = true;
    enqueue(&q, start);

    printf("Обход графа в ширину (BFS):\n");

    while (!isEmpty(&q)) {
        int u = dequeue(&q);
        printf("Посещена вершина: %d\n", u);

        for (int x = 0; x < vertex_count; x++) {
            if (adjacency[u][x] && !visited[x]) {
                visited[x] = true;
                enqueue(&q, x);
            }
        }
    }
}
```

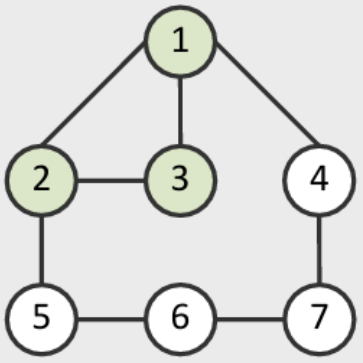
Пример работы :



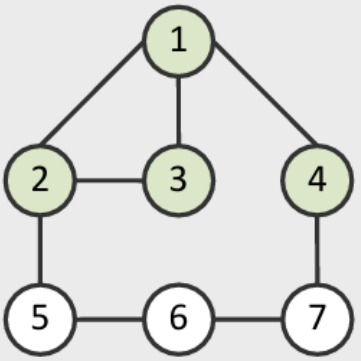
```
Извлекли из очереди: 2
В очереди: 3, 4
Добавили: 5
```



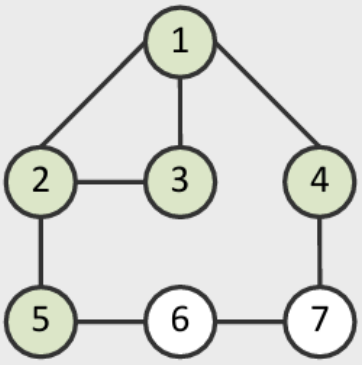

```
Извлекли из очереди: 3
В очереди: 4, 5
Добавили:
```



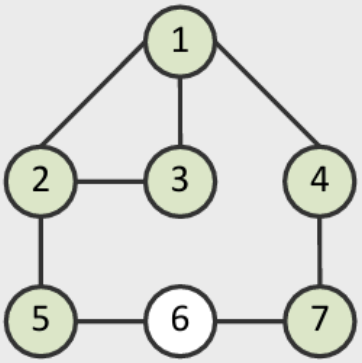
```
Извлекли из очереди: 4
В очереди: 5
Добавили: 7
```




```
Извлекли из очереди: 5
В очереди: 7
Добавили: 6
```



```
Извлекли из очереди: 7
В очереди: 6
Добавили:
```




```
Извлекли из очереди: 6
В очереди:
Добавили:
```