ГРАФЫ
ПОИСК В ГЛУБИНУ (DFS)
ХРАНЕНИЕ ГРАФА В ПРОГРАММЕ

Школа::Кода

Олимпиадное программирование

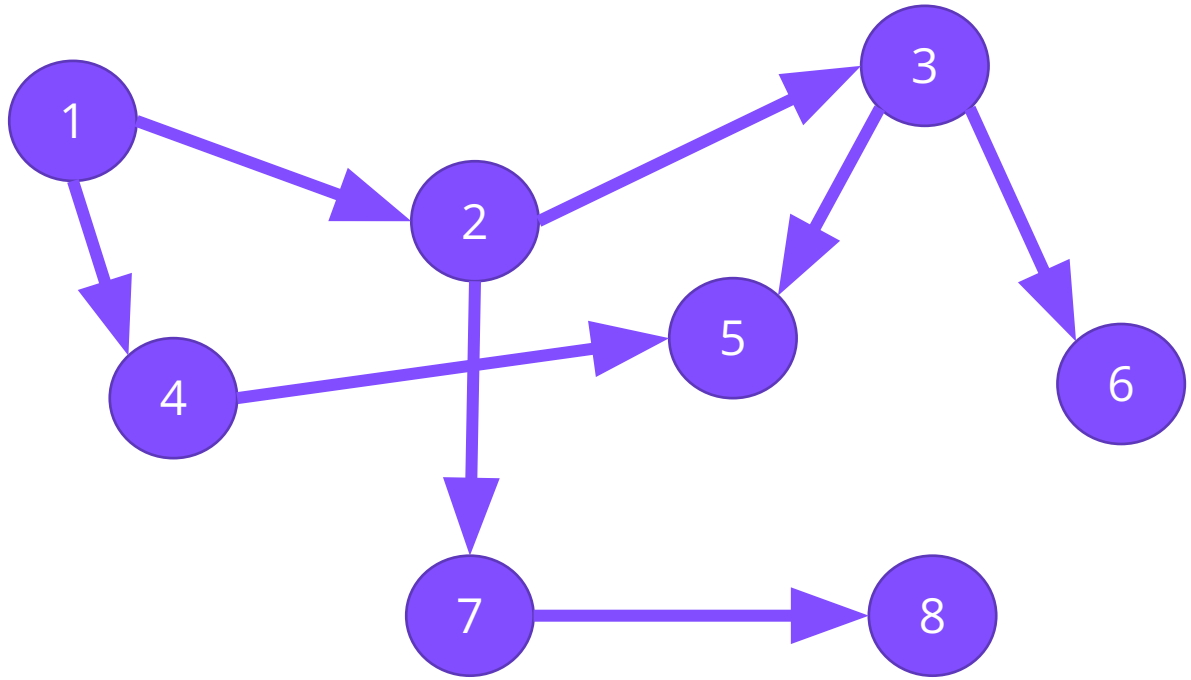
2020-2021 Таганрог

Определение

- Граф – это набор вершин, некоторые пары из которых соединены между собой рёбрами.
- Связный граф – это граф, из любой вершины которого по рёбрам можно попасть в любую другую.
- Направленный граф – это граф, по рёбрам которого можно передвигаться только в одном заданном направлении.
- Ациклический граф – это граф, не содержащий циклов.
- Дерево – это связный граф, имеющий N вершин и N-1 ребро.
- Взвешенный граф – это граф, в котором каждое ребро имеет свой вес (обычно длину).
- Компонента связности – максимальный по включению связный подграф.

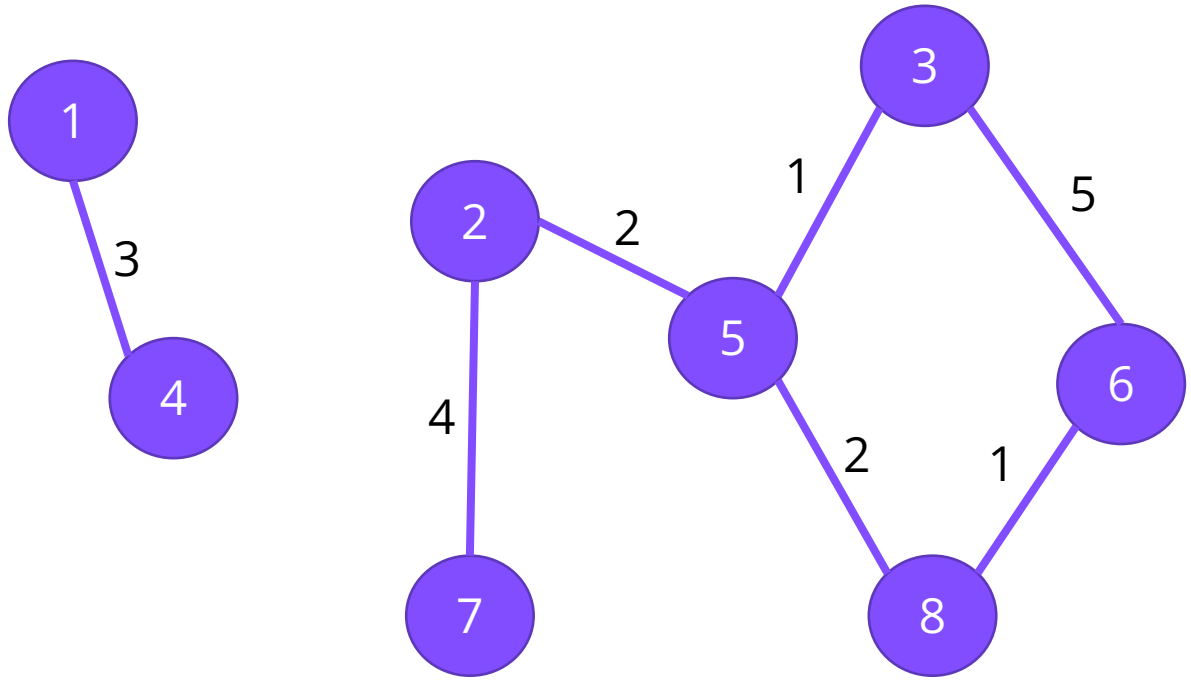
Задача:

- Охарактеризуйте граф:



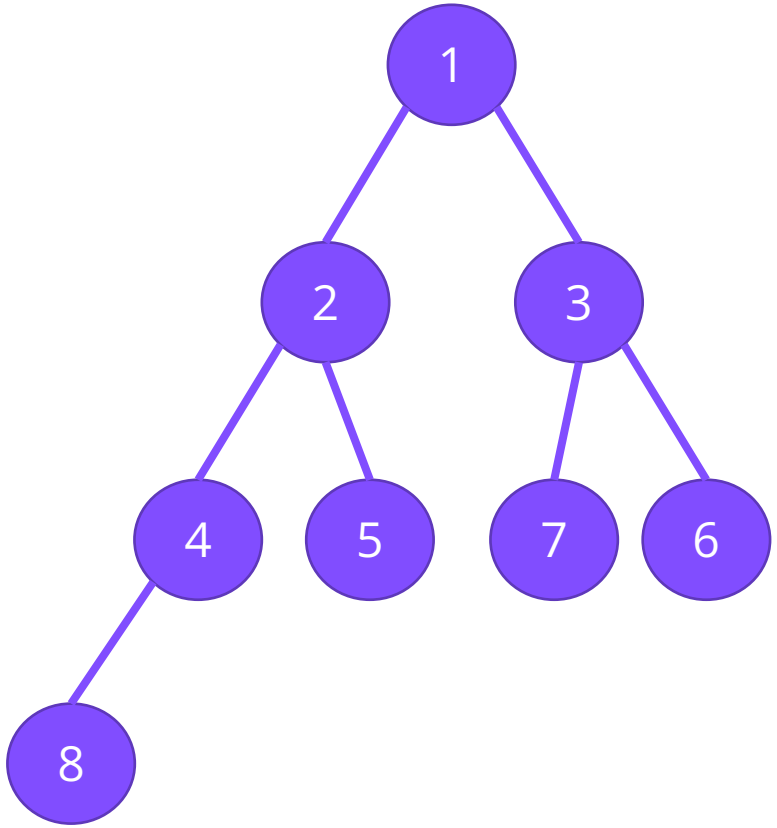
Задача:

- Охарактеризуйте граф:



Задача:

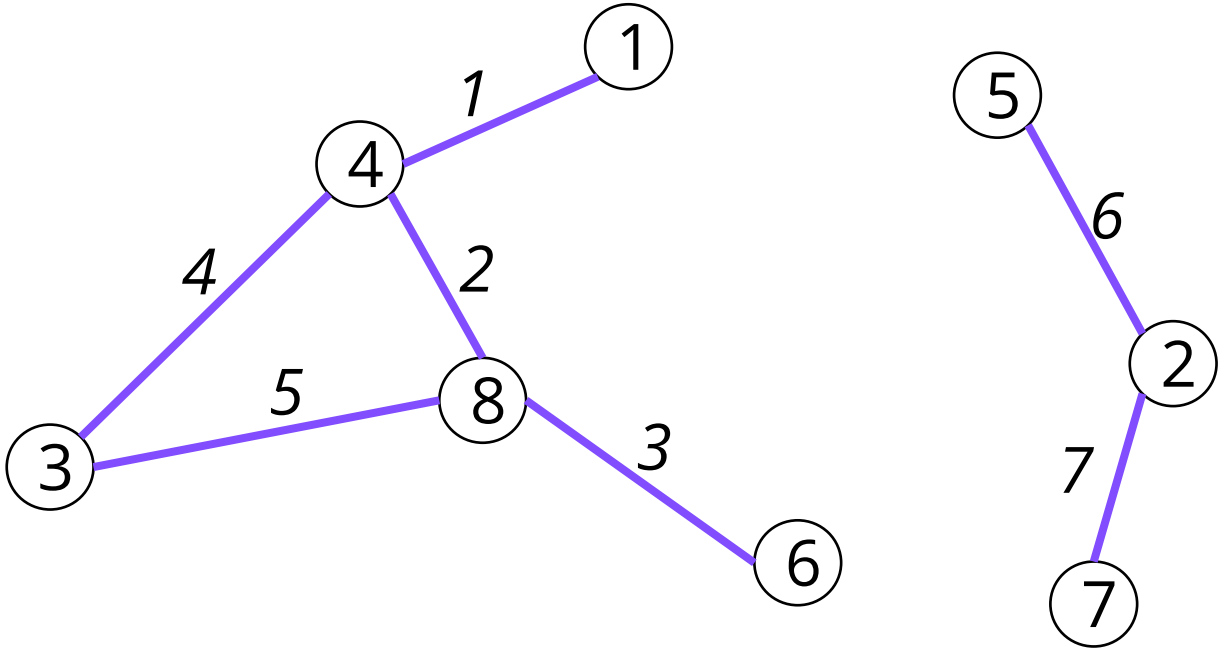
- Охарактеризуйте граф:



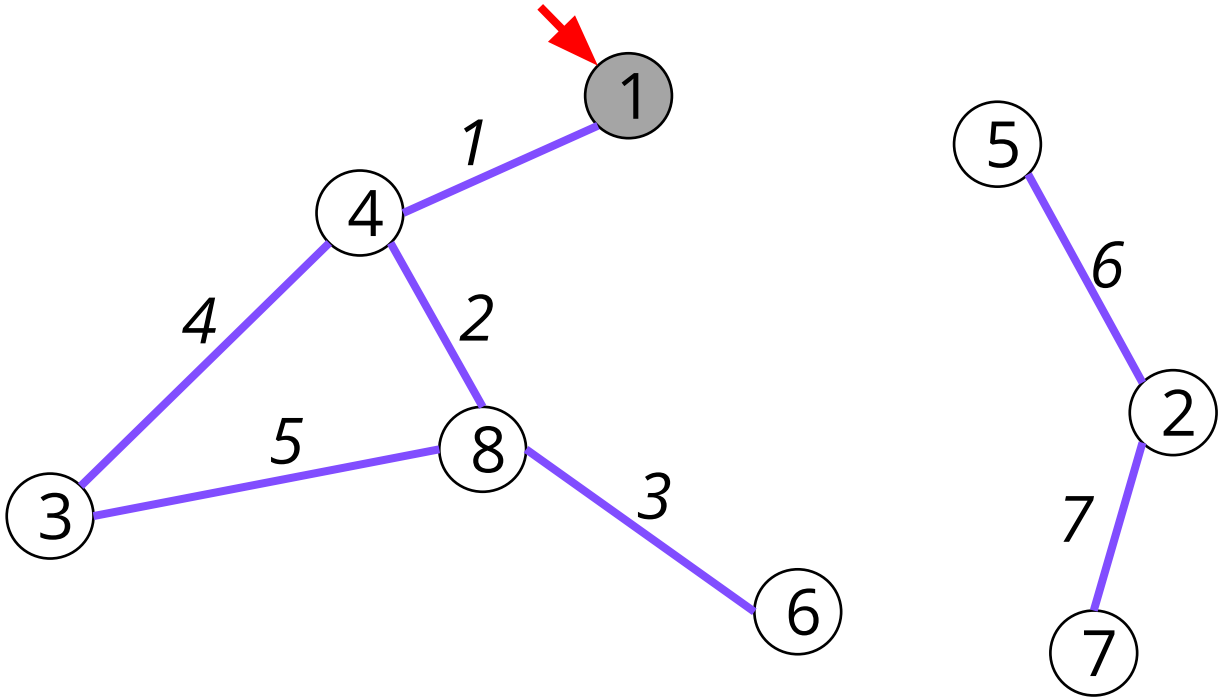
Поиск (обход) в глубину (DFS)

1. Для каждой вершины графа будем хранить её цвет. Пусть изначально вершины будут белыми, когда мы в них входим, они будут становиться серыми, а когда выходим – чёрными.
2. Будем перебирать вершины графа. Если встречаем белую вершину – запускаем из неё поиск в глубину.
3. Во время поиска в глубину перебираем все вершины, смежные той, в которой сейчас находится алгоритм, и по очереди запускаем из них поиск в глубину.
4. Если поиск запущен не из белой вершины, то дальше его продолжать не следует.

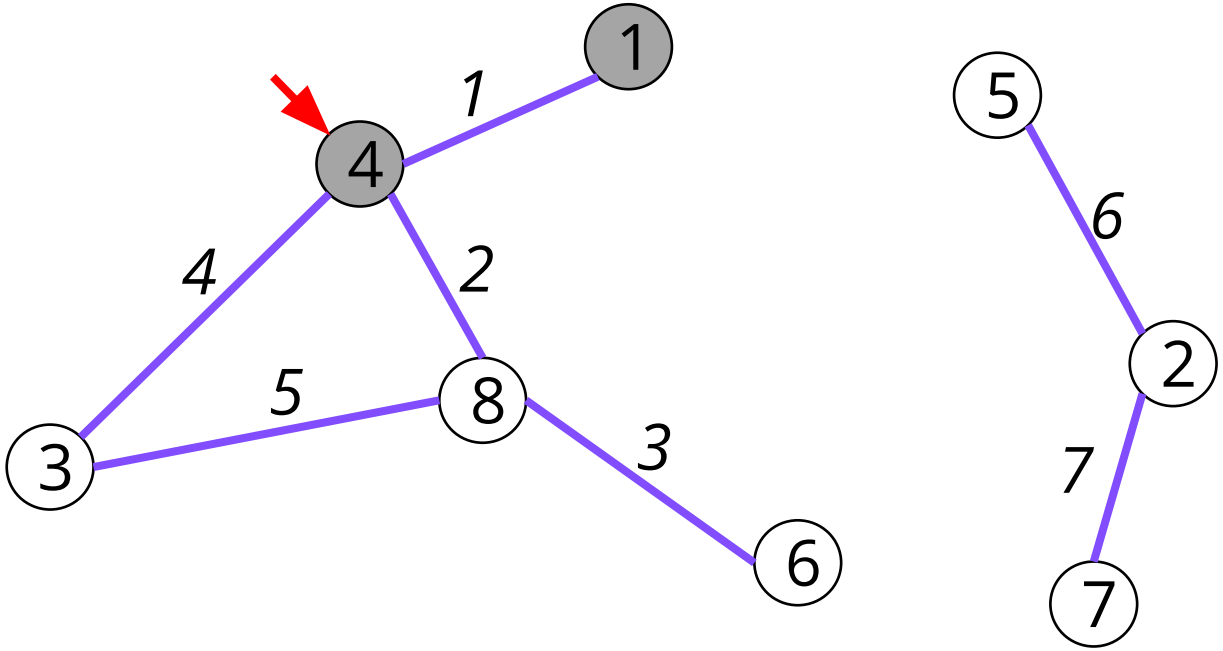
Поиск в глубину. Пример. Шаг 0



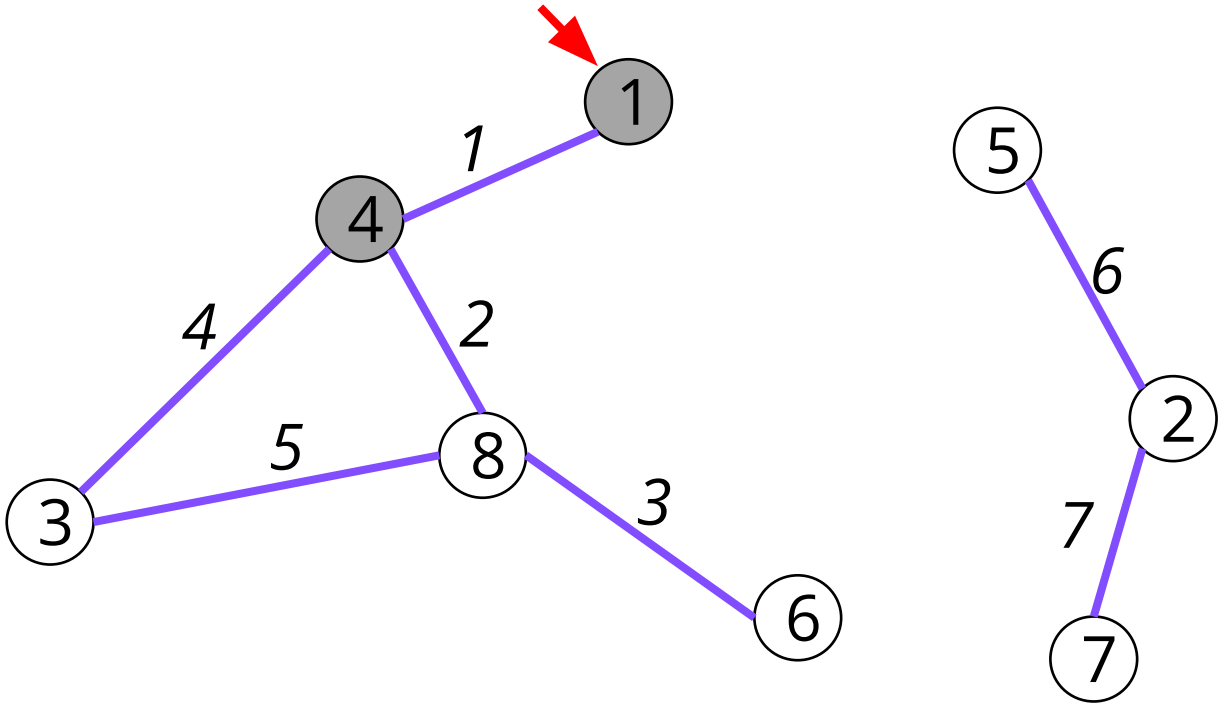
Поиск в глубину. Пример. Шаг 1



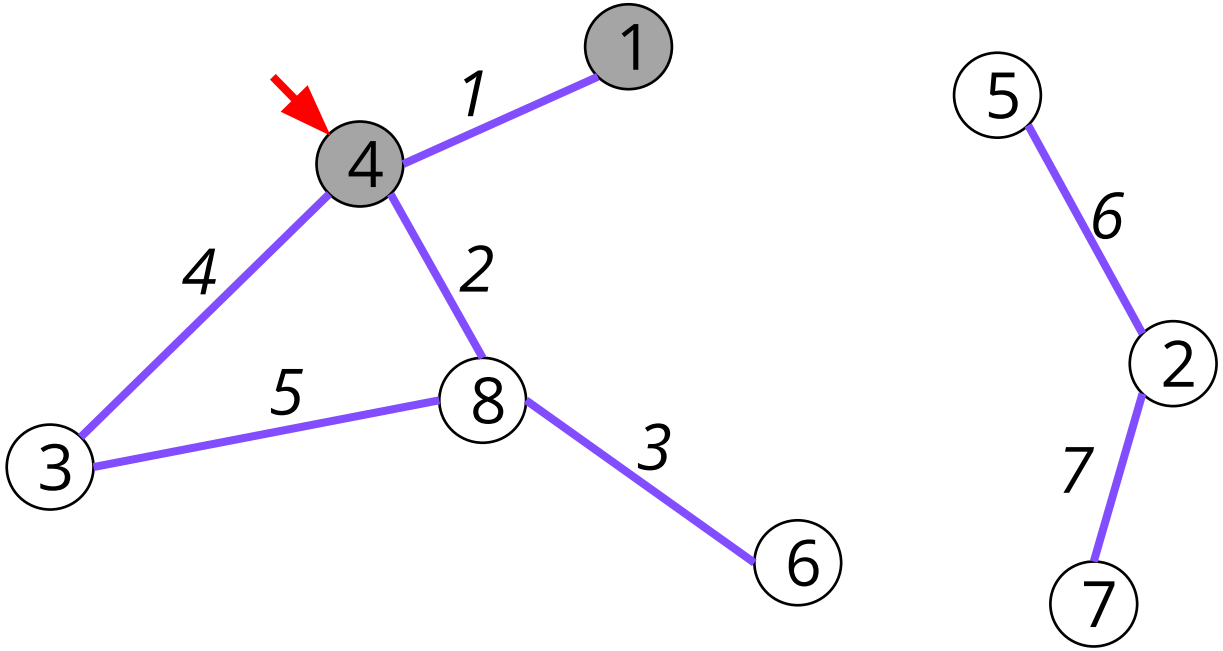
Поиск в глубину. Пример. Шаг 2



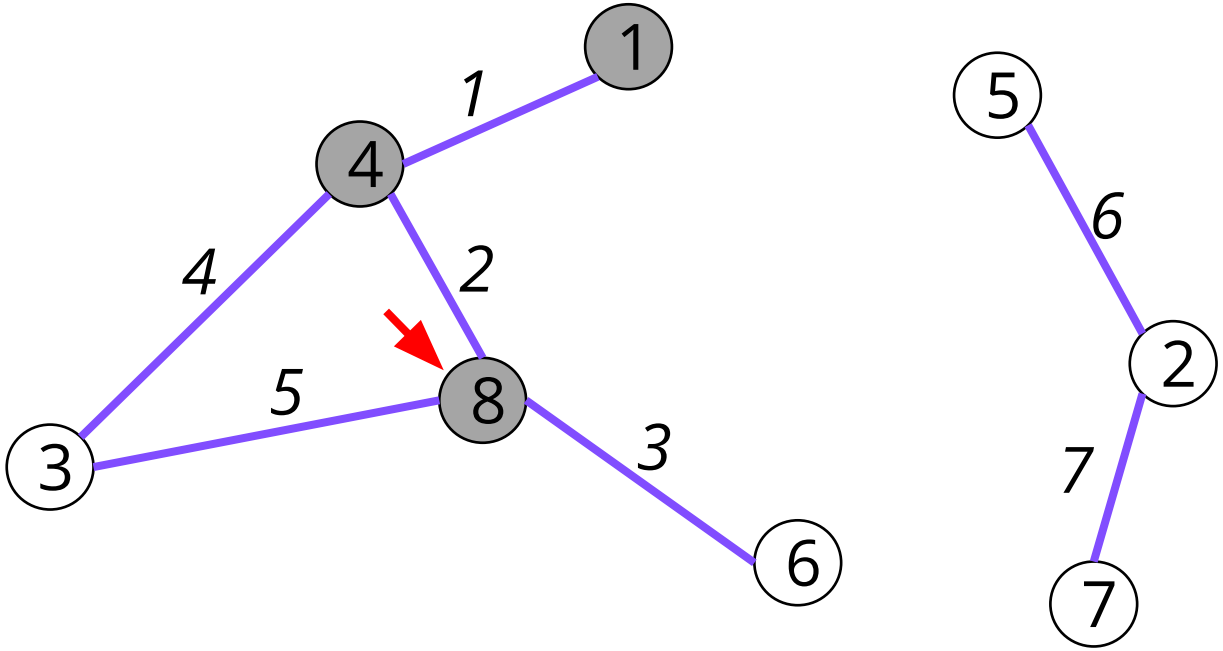
Поиск в глубину. Пример. Шаг 3



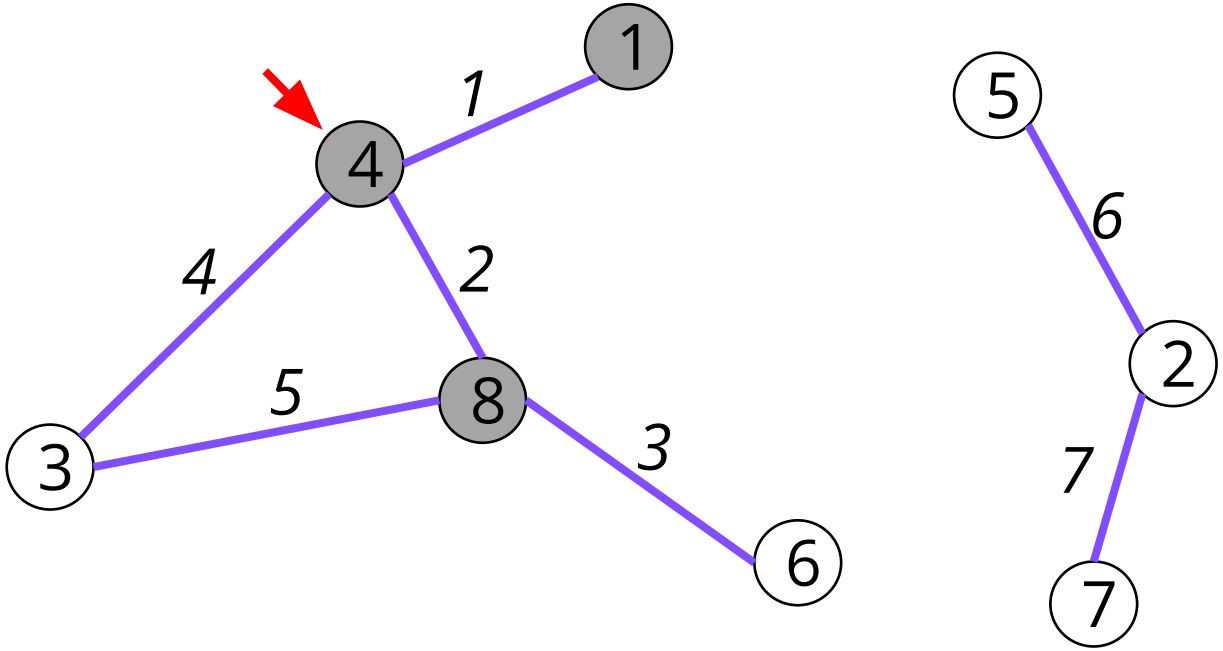
Поиск в глубину. Пример. Шаг 4



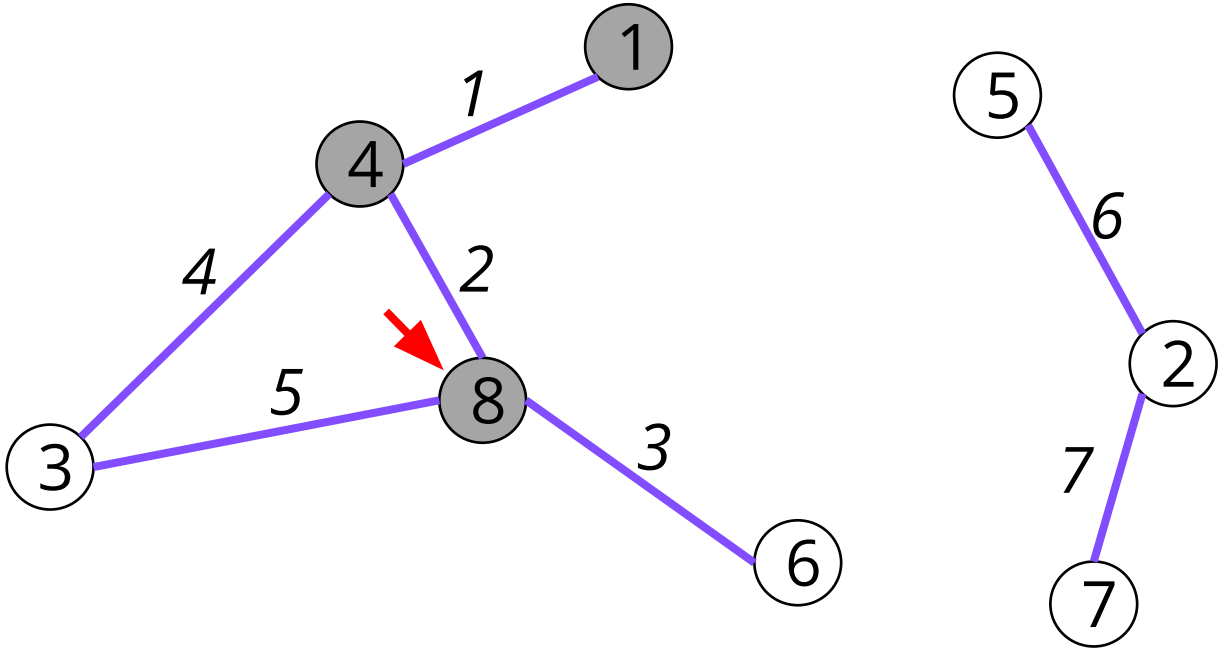
Поиск в глубину. Пример. Шаг 5



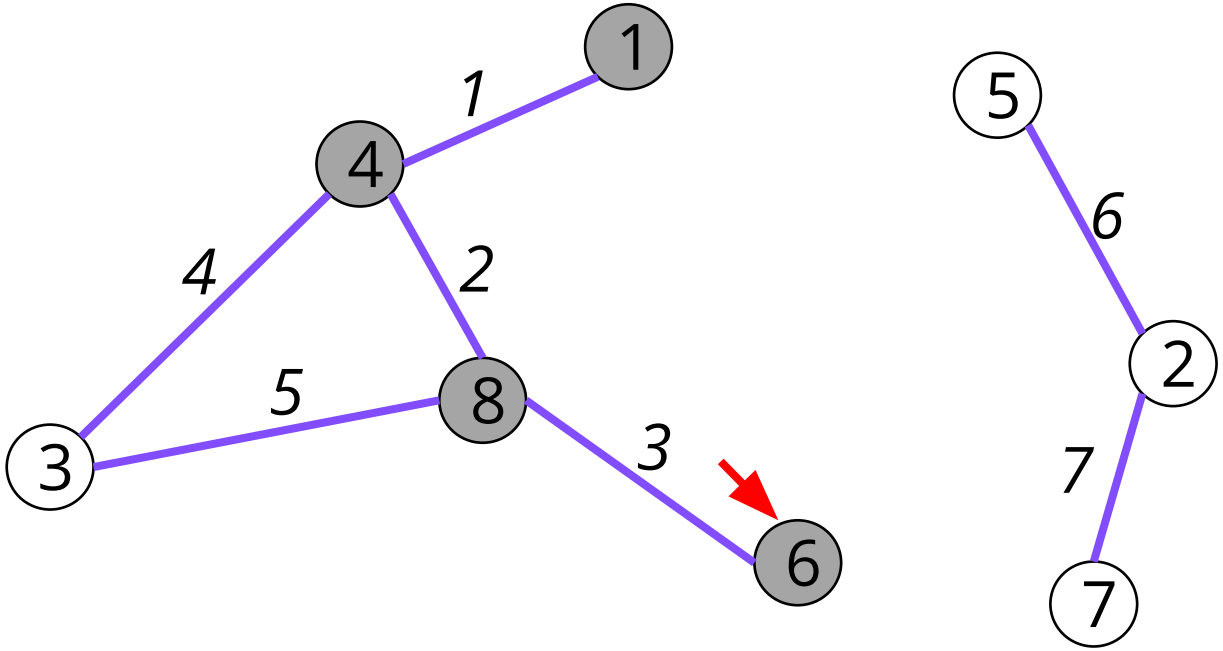
Поиск в глубину. Пример. Шаг 6



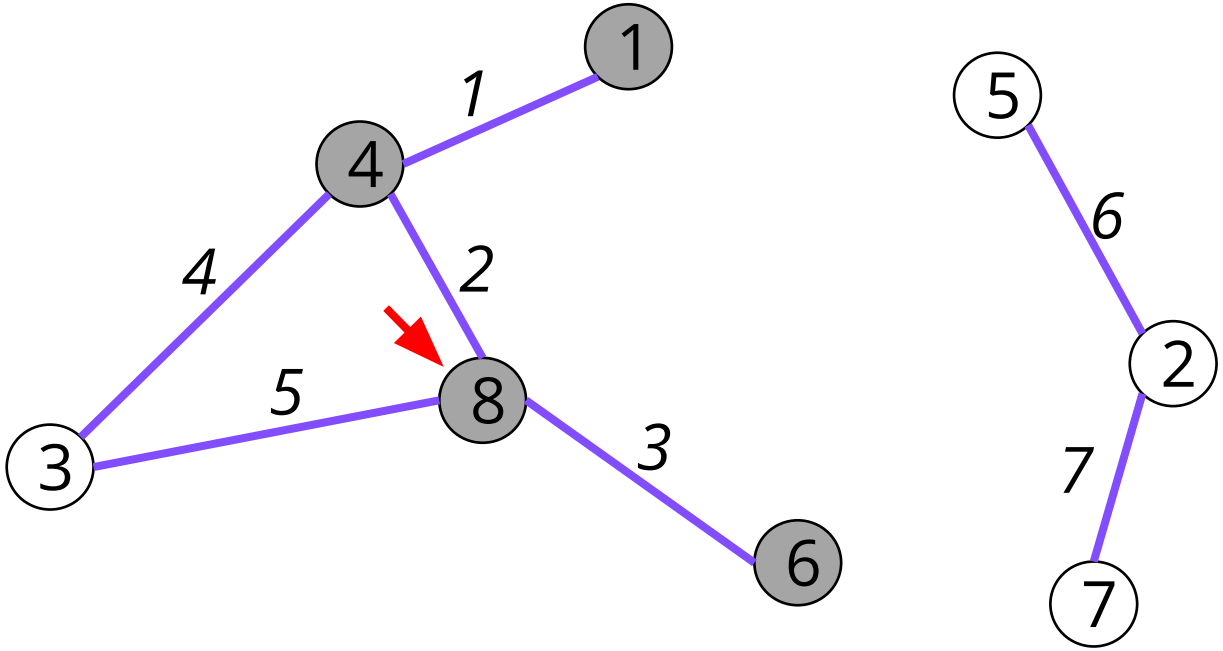
Поиск в глубину. Пример. Шаг 7



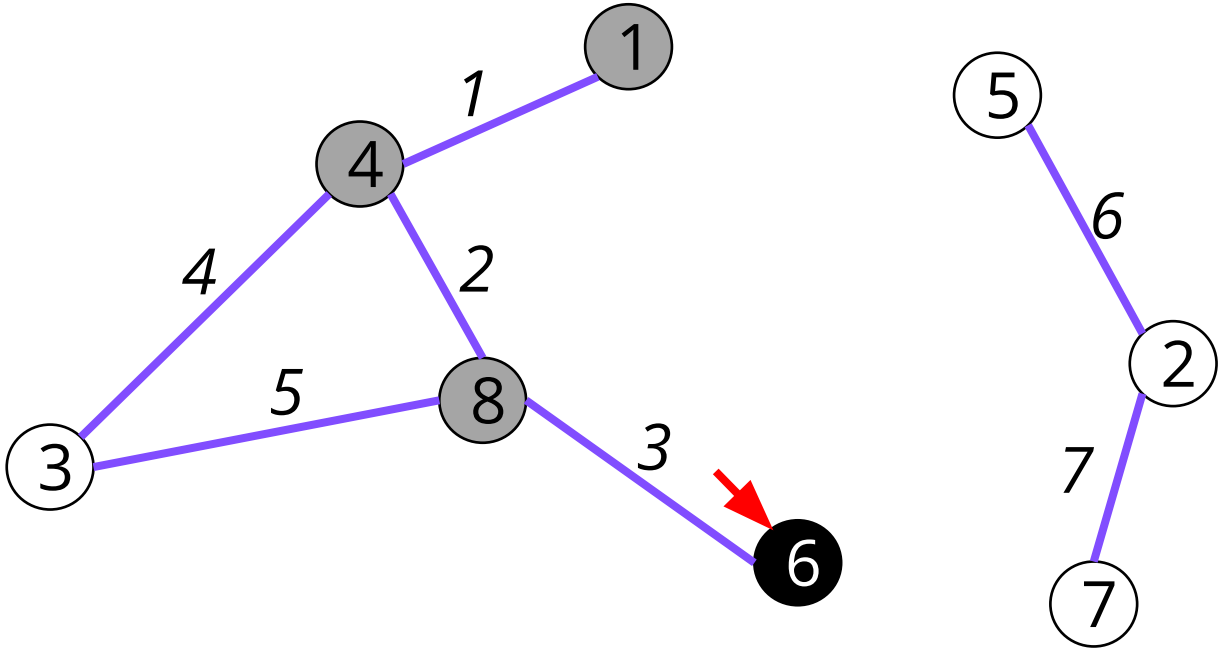
Поиск в глубину. Пример. Шаг 8



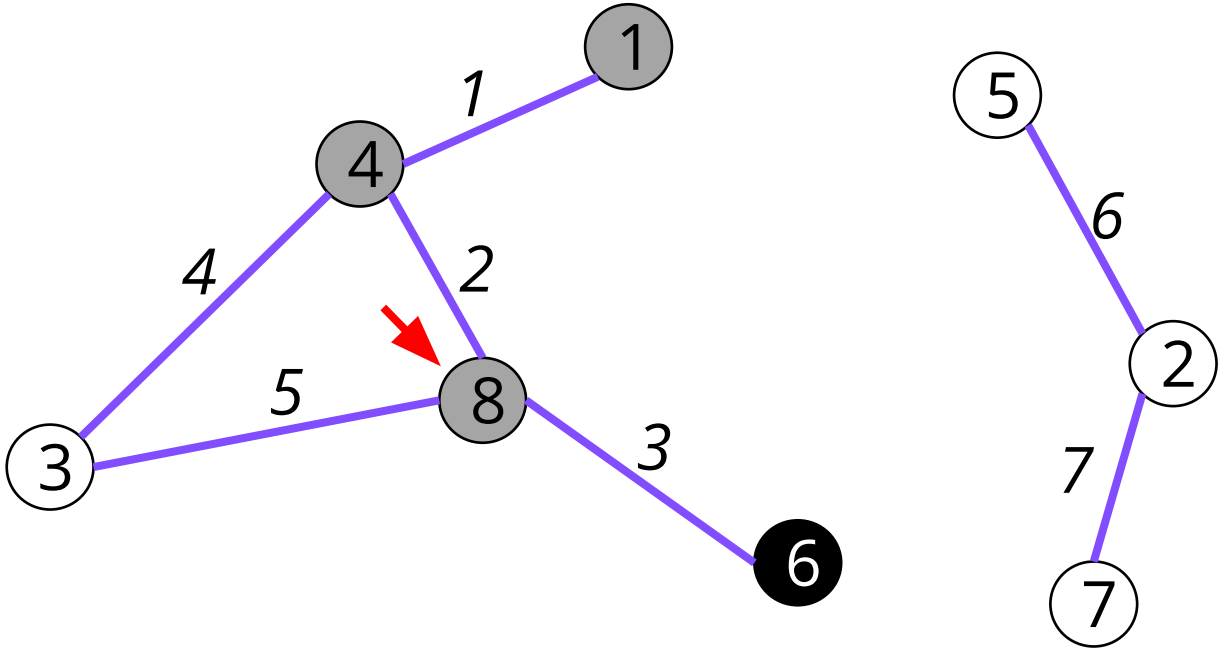
Поиск в глубину. Пример. Шаг 9



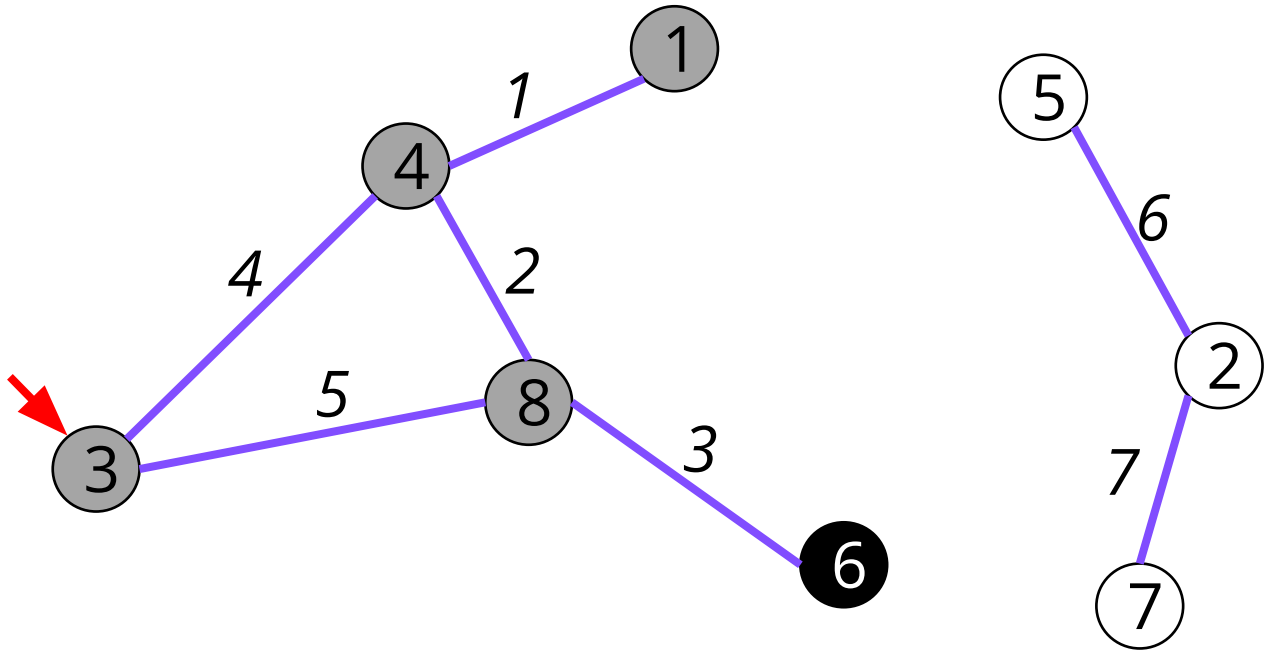
Поиск в глубину. Пример. Шаг 10



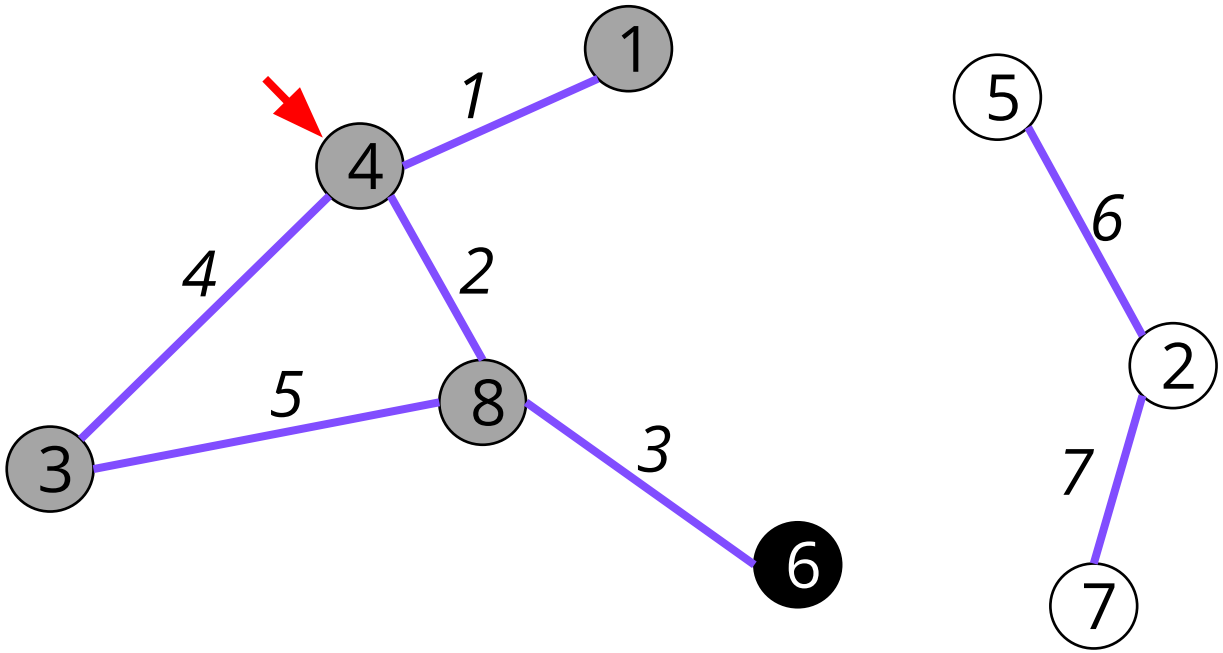
Поиск в глубину. Пример. Шаг 11



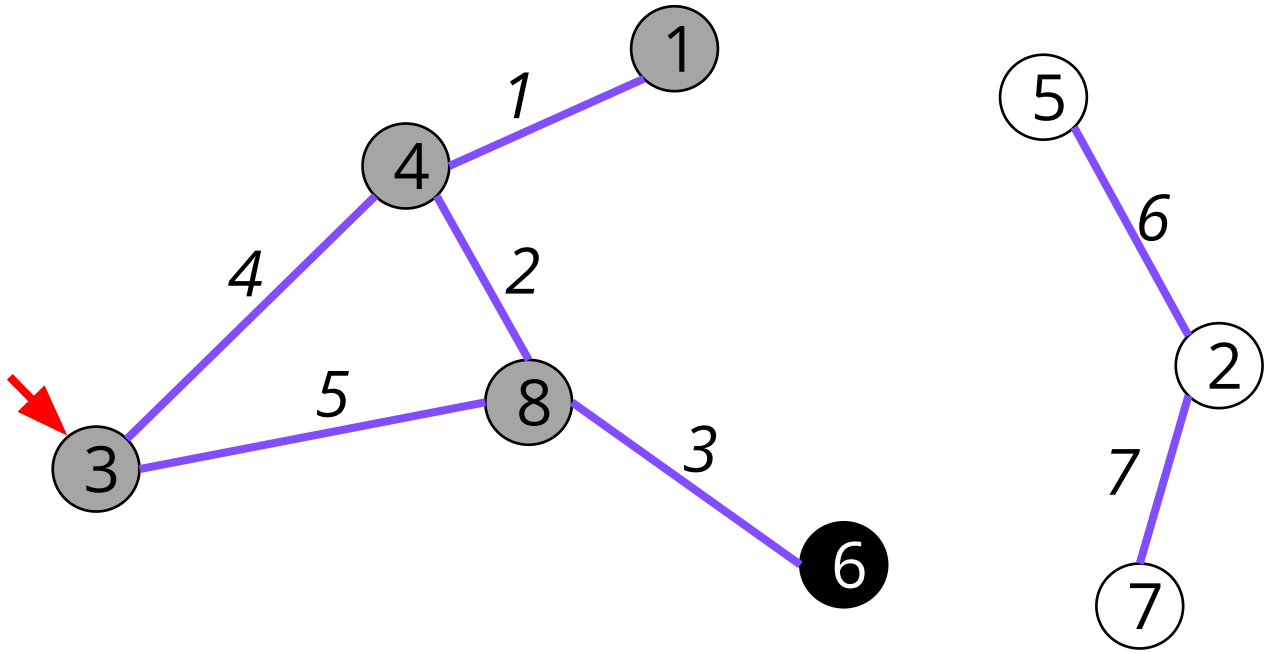
Поиск в глубину. Пример. Шаг 12



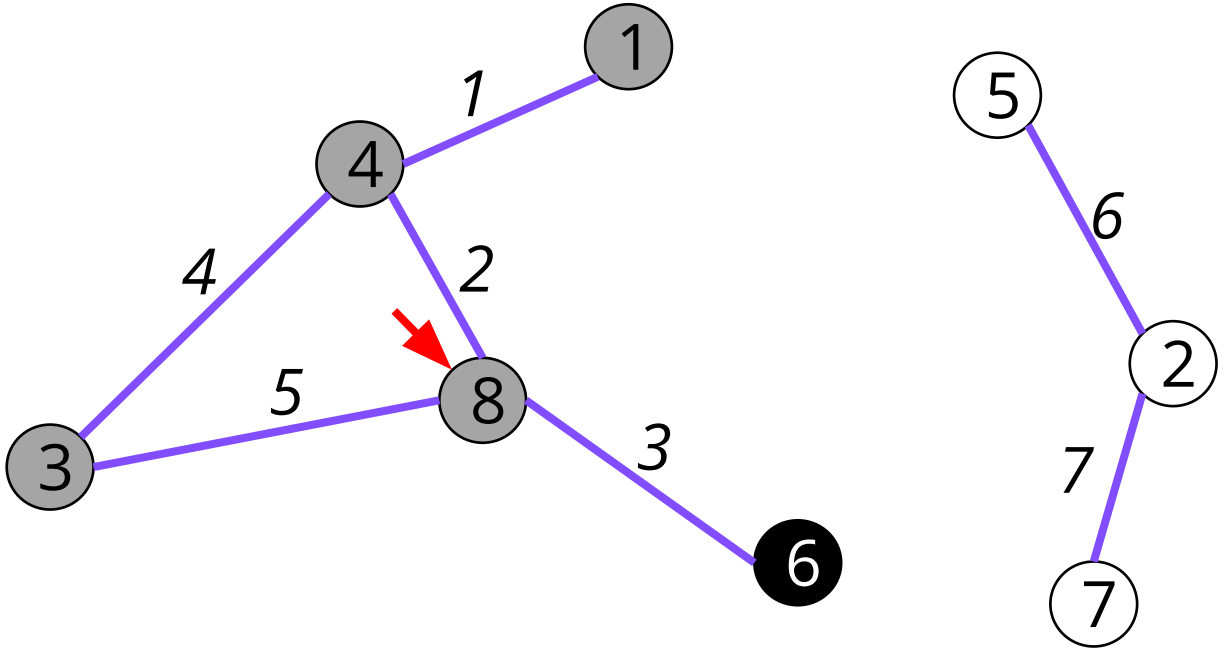
Поиск в глубину. Пример. Шаг 13



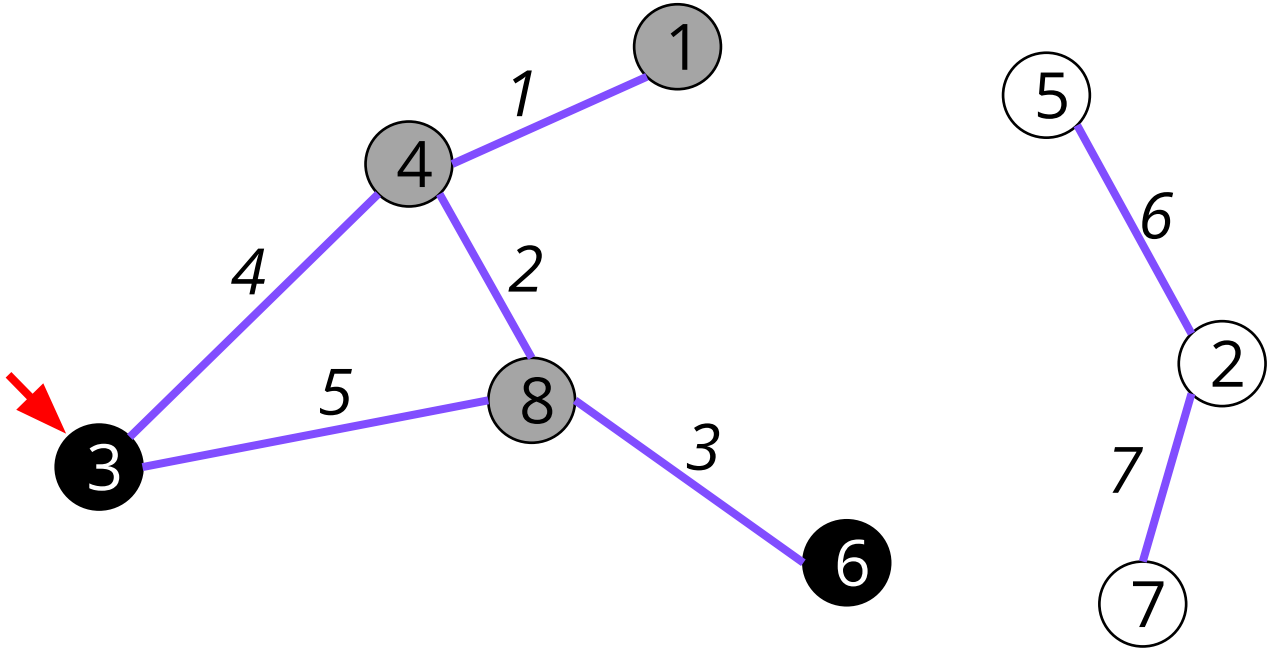
Поиск в глубину. Пример. Шаг 14



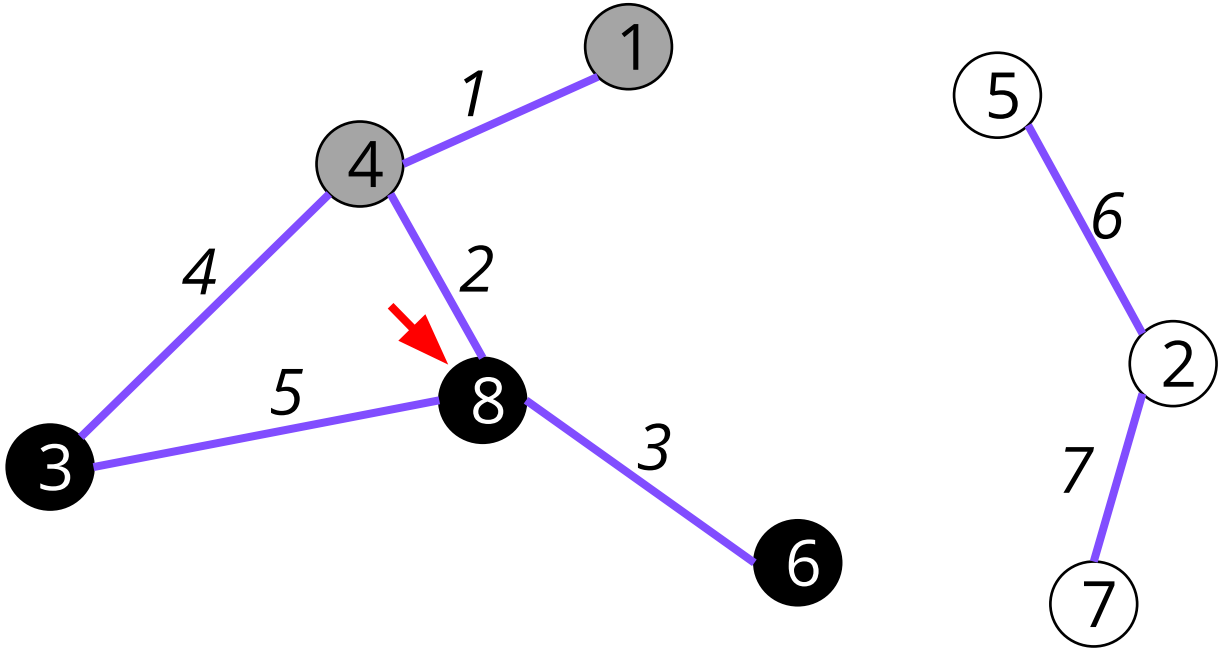
Поиск в глубину. Пример. Шаг 15



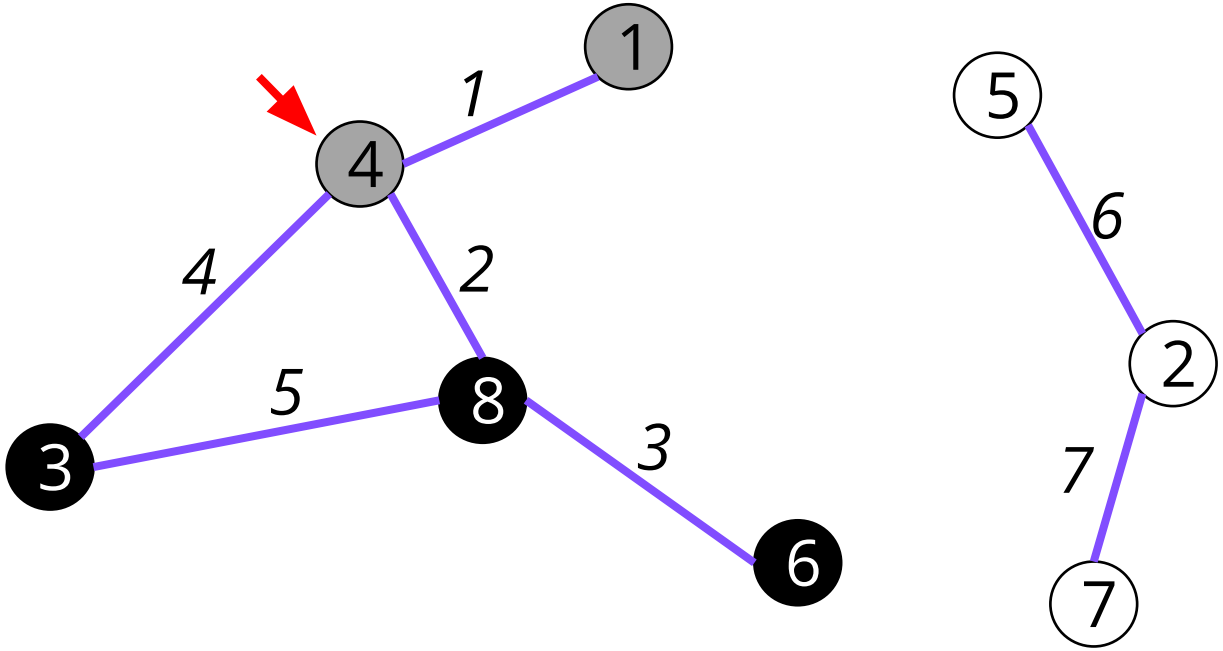
Поиск в глубину. Пример. Шаг 16



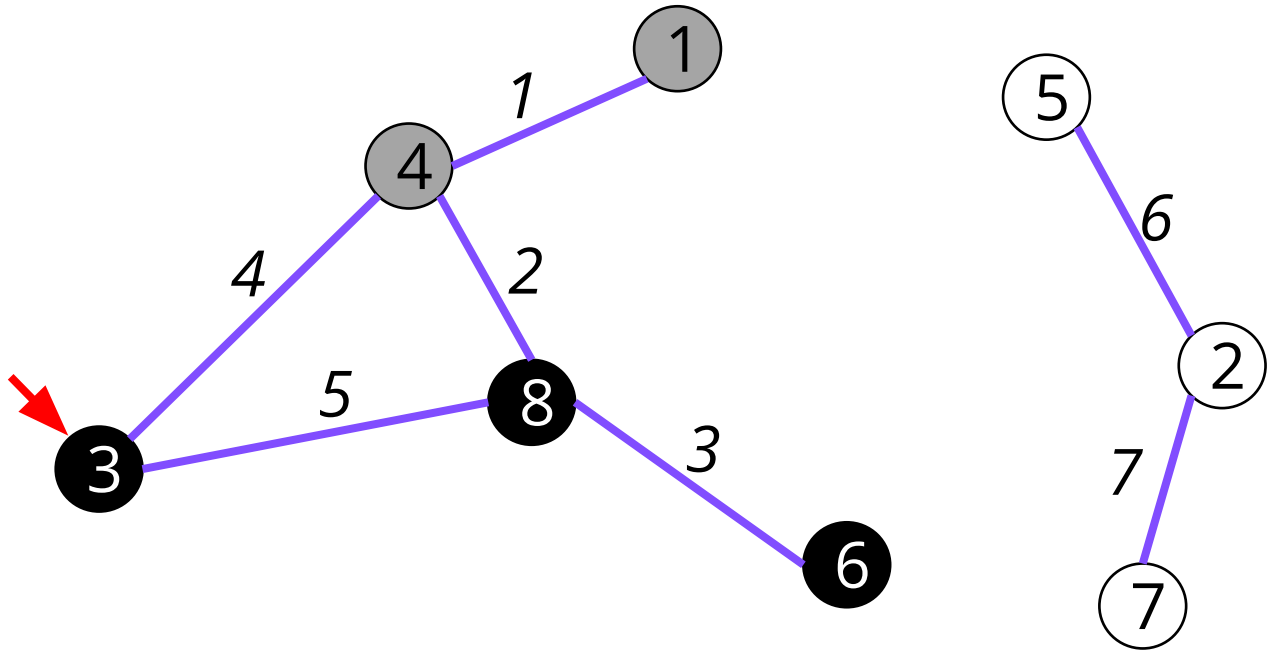
Поиск в глубину. Пример. Шаг 17



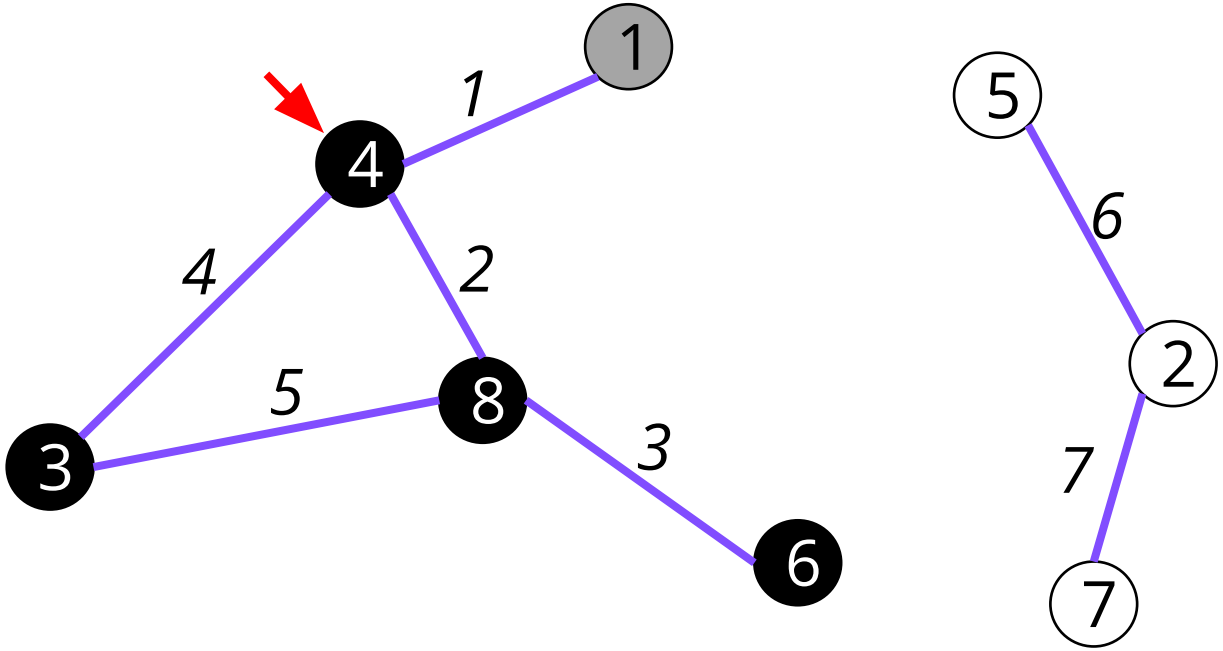
Поиск в глубину. Пример. Шаг 18



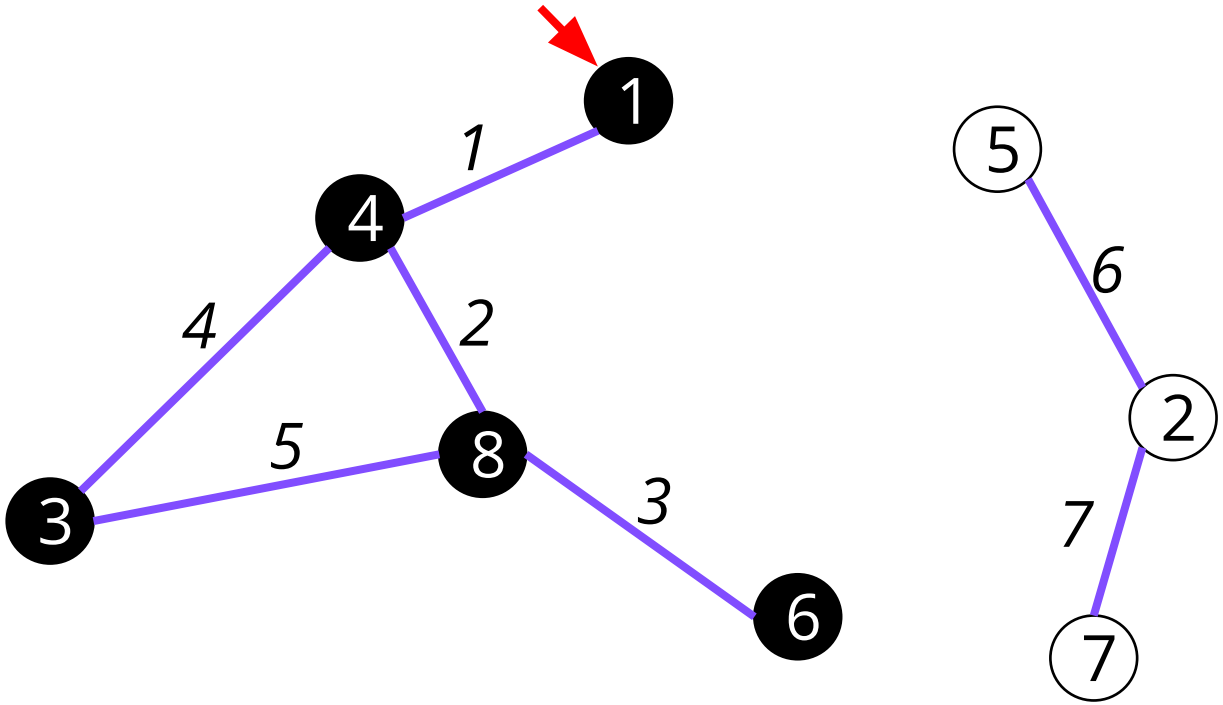
Поиск в глубину. Пример. Шаг 19



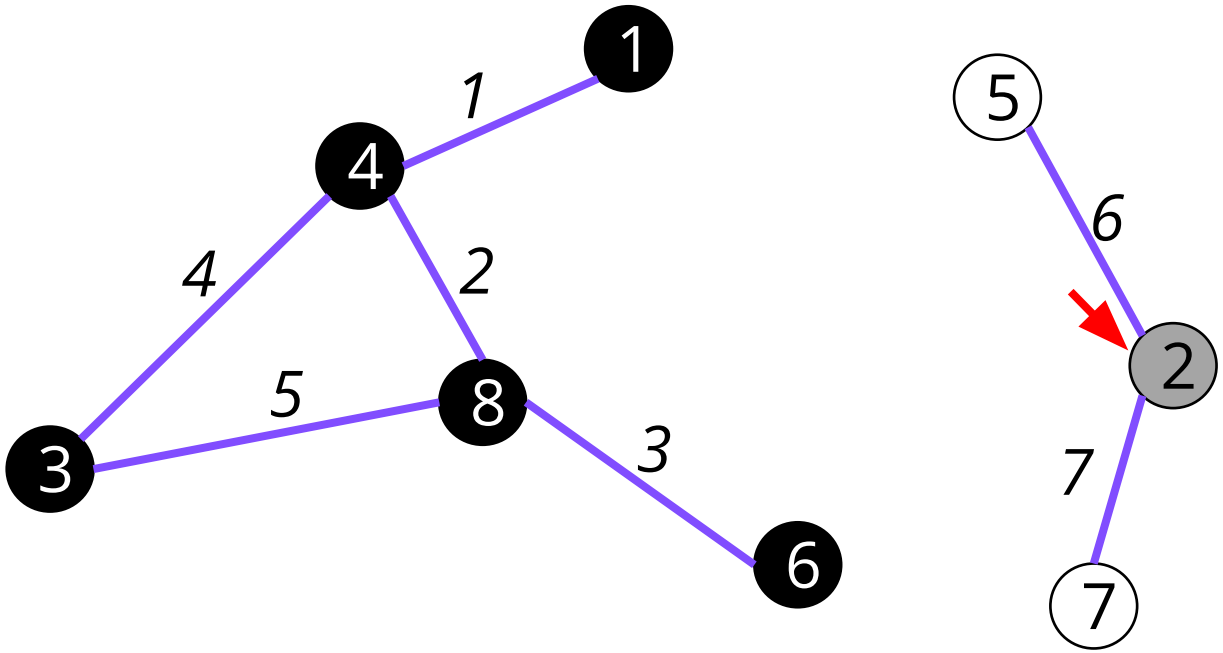
Поиск в глубину. Пример. Шаг 20



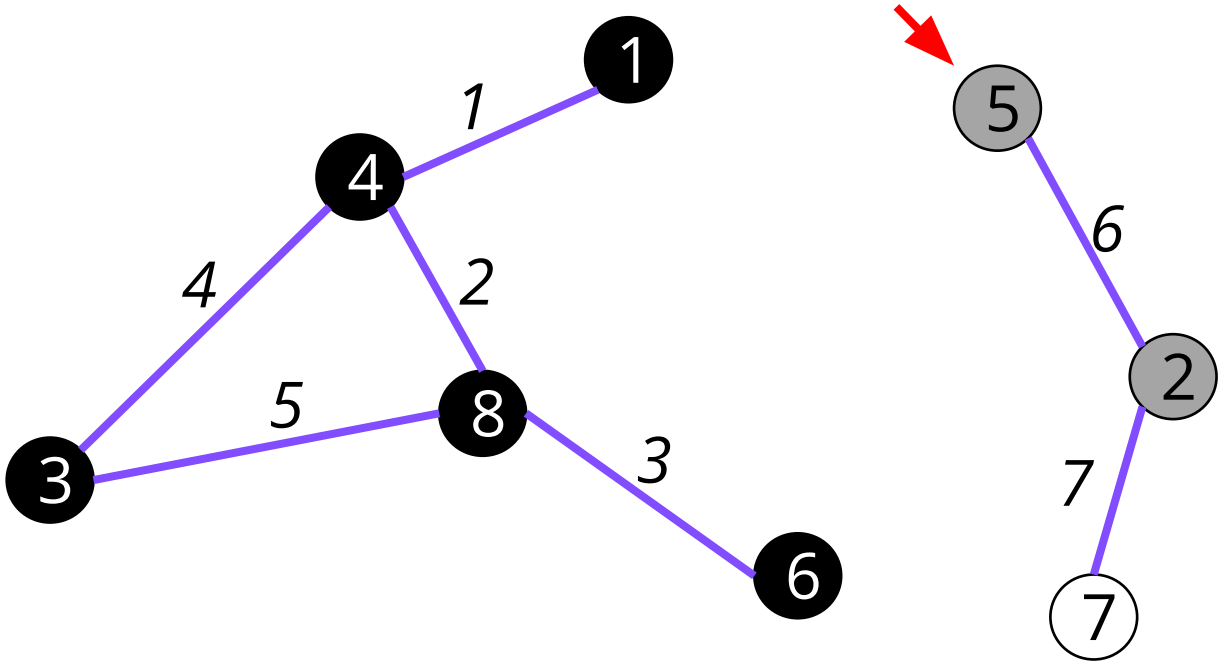
Поиск в глубину. Пример. Шаг 21



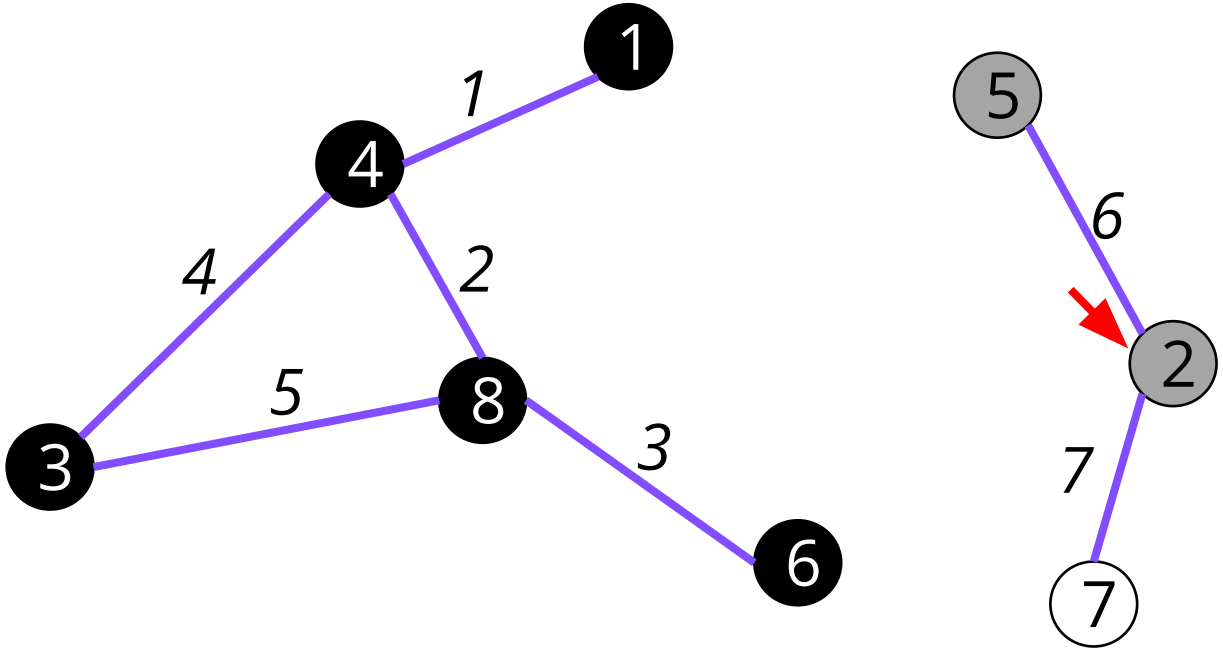
Поиск в глубину. Пример. Шаг 22



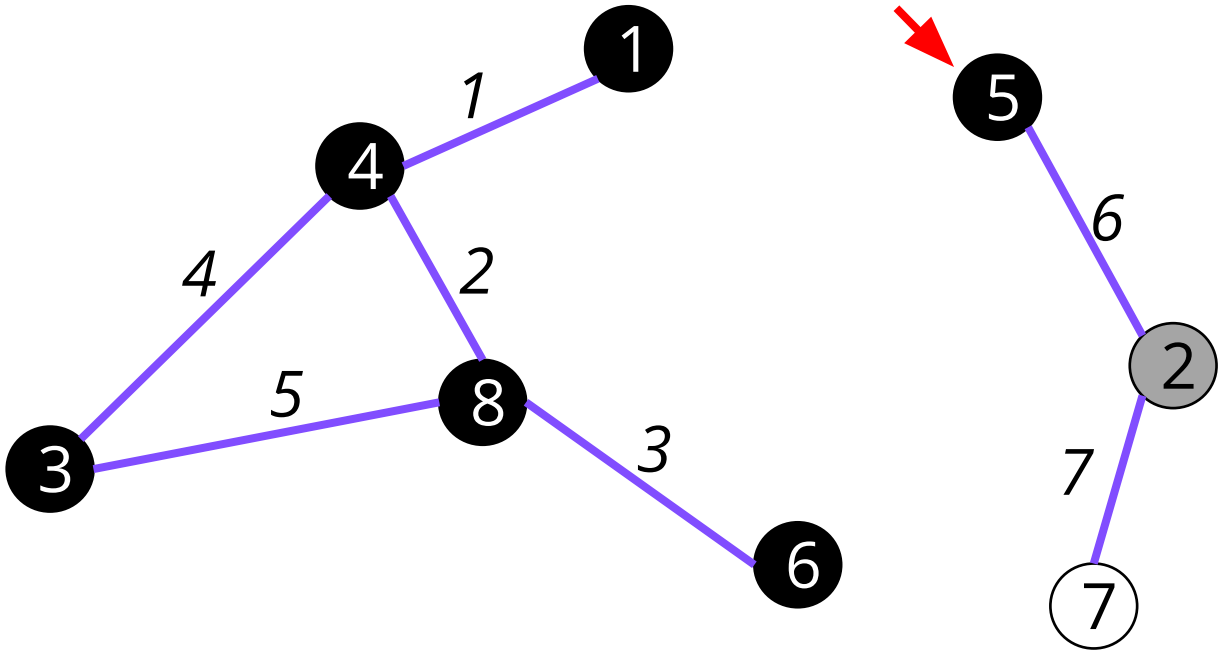
Поиск в глубину. Пример. Шаг 23



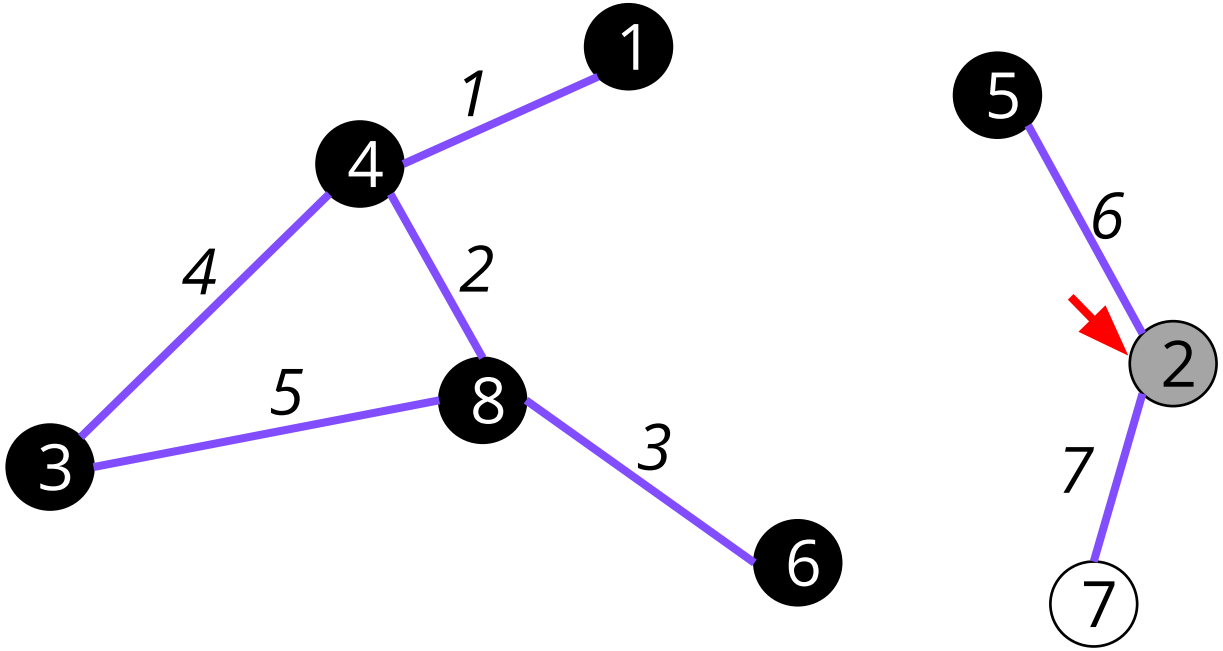
Поиск в глубину. Пример. Шаг 24



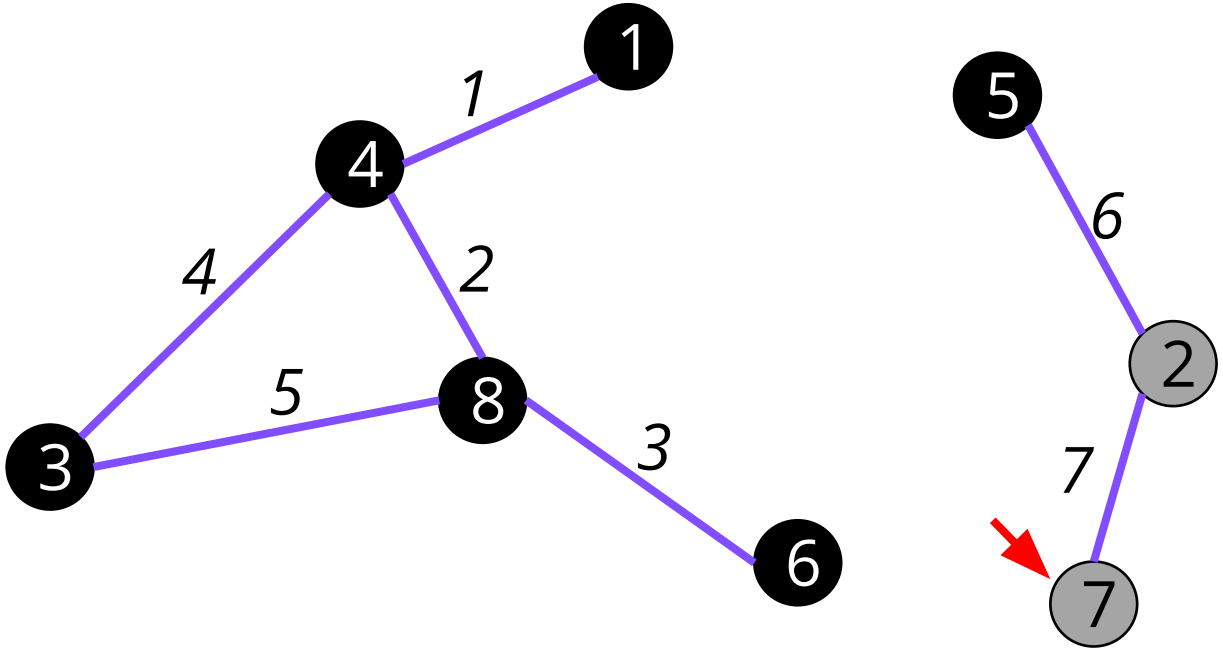
Поиск в глубину. Пример. Шаг 25



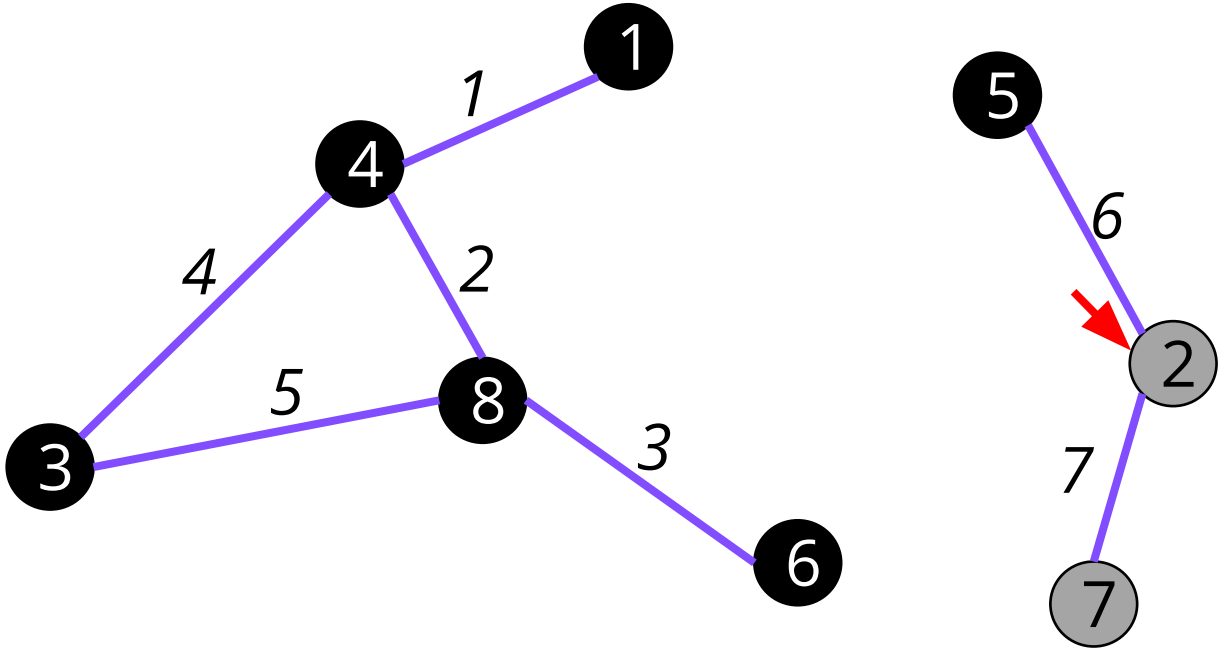
Поиск в глубину. Пример. Шаг 26



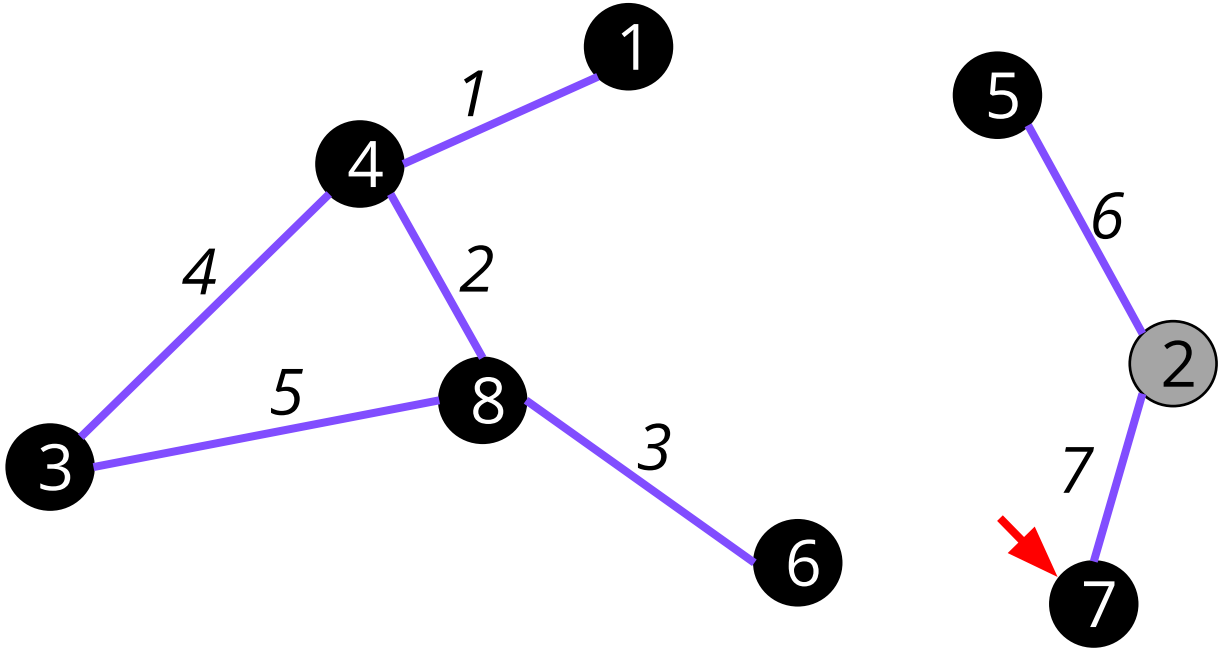
Поиск в глубину. Пример. Шаг 27



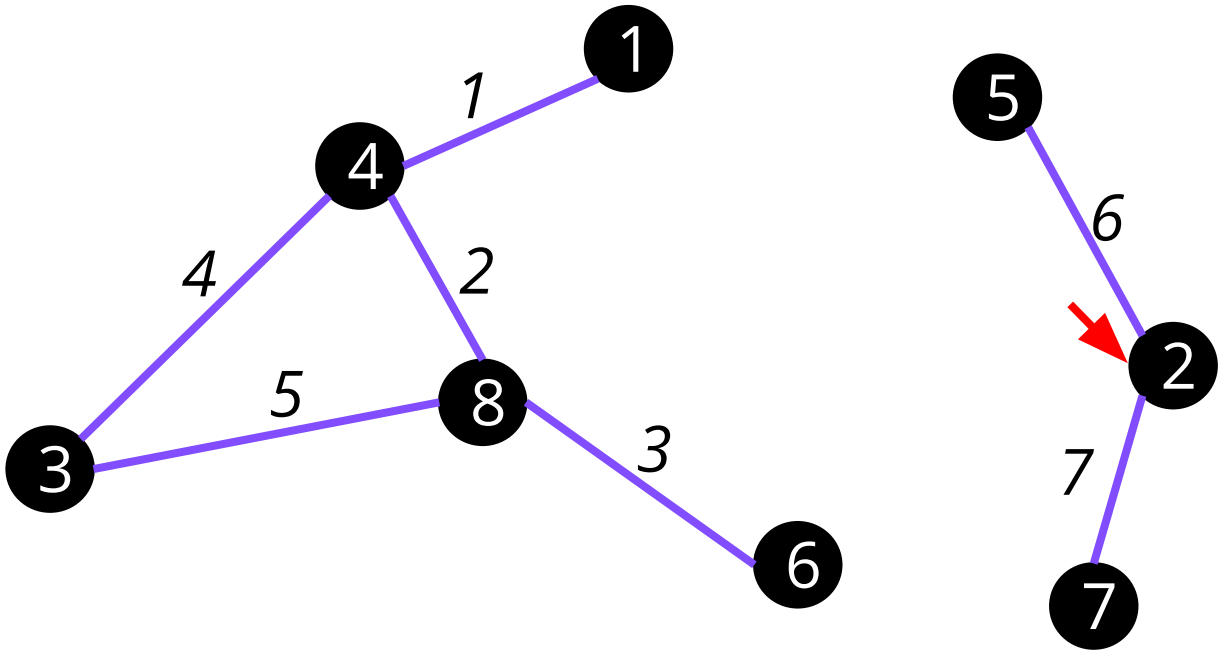
Поиск в глубину. Пример. Шаг 28



Поиск в глубину. Пример. Шаг 29



Поиск в глубину. Пример. Шаг 30



Хранение графа в программе

- Создадим собственную структуру Node для хранения информации о вершине графа.
- В Node будем хранить цвет самой вершины и перечень смежных вершин.
- Граф будем хранить в виде массива (вектора) вершин.

```
struct Node
{
    vector<int> to;
    int color = 0;
};

vector<Node> g;
```

Реализация поиска в глубину

```
struct Node
{
    vector<int> to;
    int color = 0;
};

vector<Node> g;

void dfs(int v)
{
    if (g[v].color != 0)
        return;
    g[v].color = 1;
    for (int i = 0; i < g[v].to.size(); ++i)
        dfs(g[v].to[i]);
    g[v].color = 2;
}
```

Поиск в глубину реализуется в виде рекурсивной функции, критерием остановки которой является повторное посещение вершины.

Ввод графа и запуск поиска в глубину

- Пусть дан граф из N вершин и M рёбер. Во входных данных сначала даются числа N и M, а затем M пар чисел U и V – номера вершин графа, между которыми есть ребро. ($1 \le U, V \le N$)

```
int main()
{
    int n, m;
    cin >> n >> m;
    g.resize(n);
    for (int i = 0; i < m; ++i)
    {
        int u, v;
        cin >> u >> v;
        --u; --v;
        g[u].to.push_back(v);
        g[v].to.push_back(u);
    }
    for (int i = 0; i < n; ++i)
        if (g[i].color == 0)
            dfs(i);

    return 0;
}
```