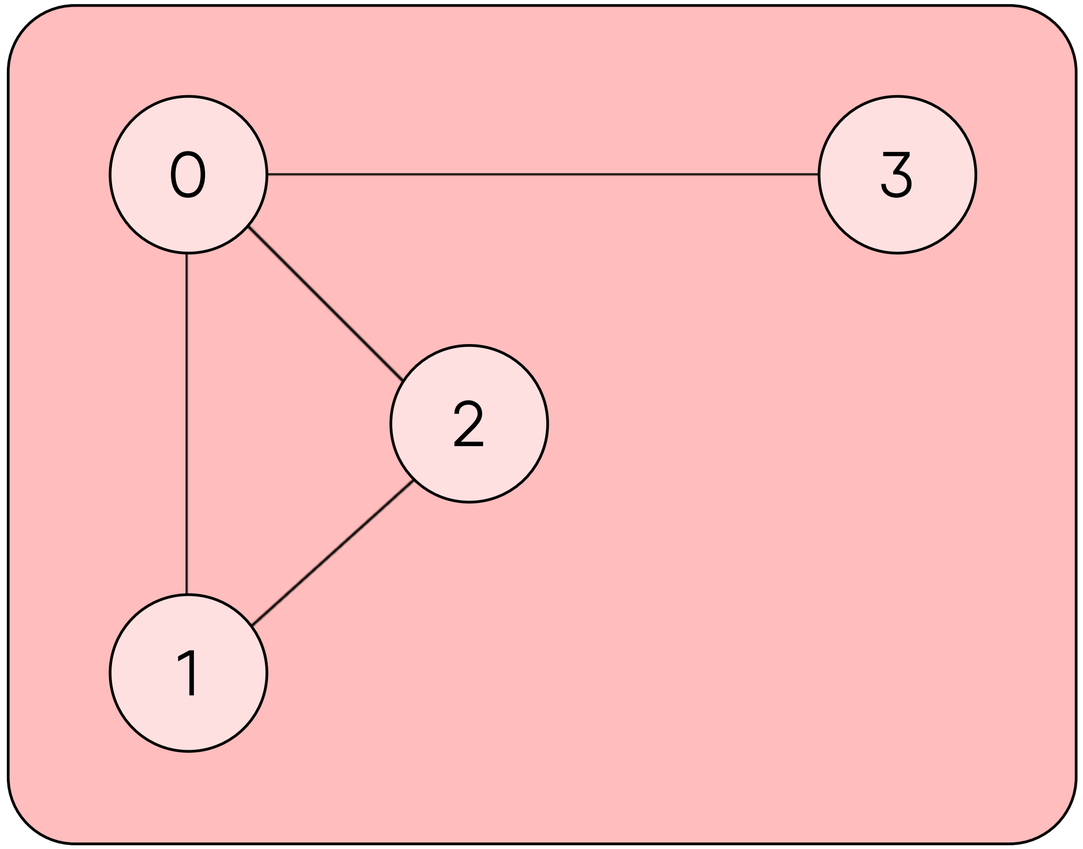
Его величество граф

Обход графа в глубину и ширину

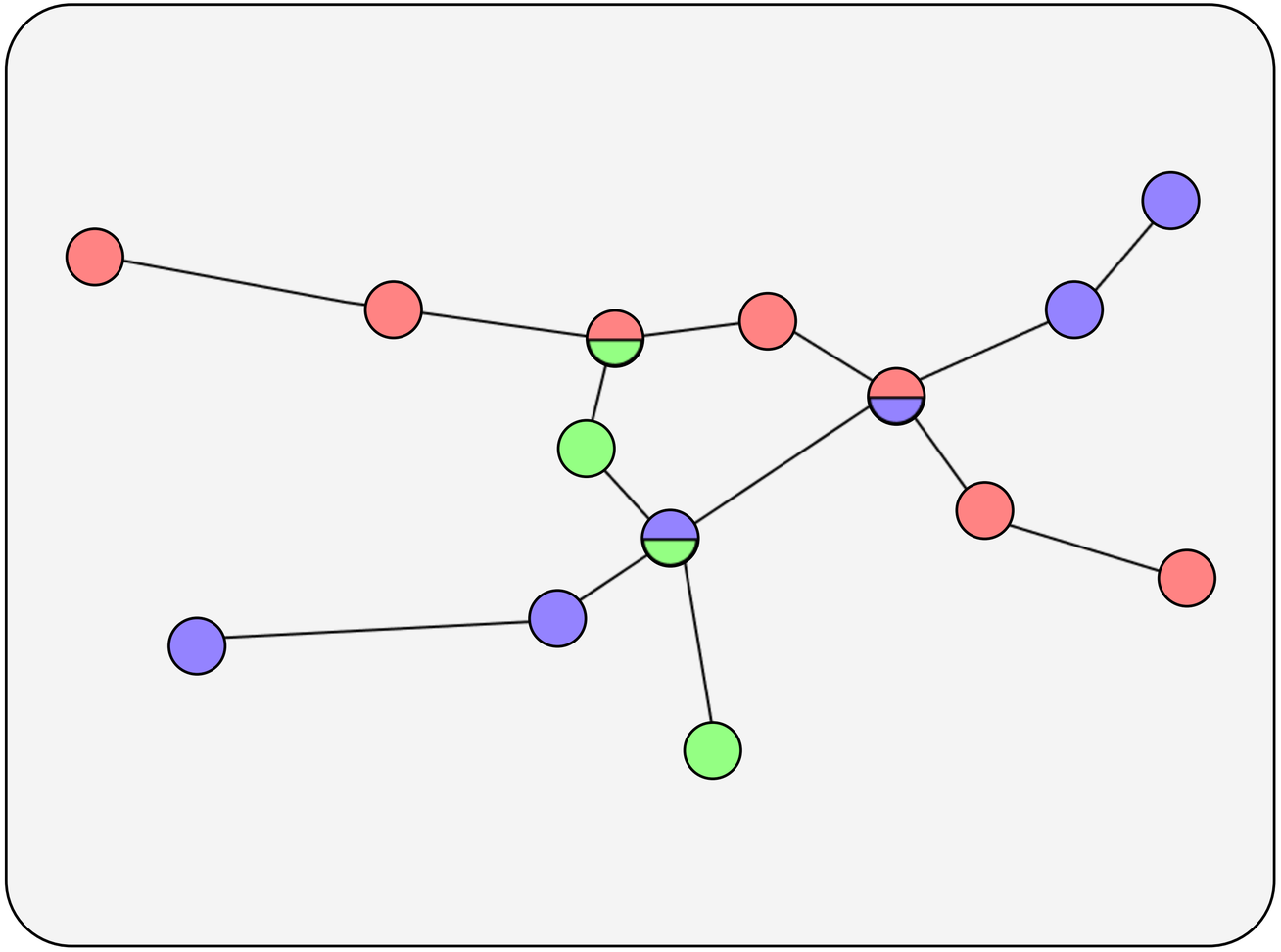
Связность в ориентированных и неориентированных графах



Демчишина Анна

Шиковец Егор

Его величество граф Орлов



Упрощенная схема минского метрополитена

Определения

DFS

Наглядная демонстрация

Принцип работы

Немного кода

BFS

Снова наглядная демонстрация

Снова принцип работы

Еще чуть-чуть кода

Связанность графов

Ориентированный граф

Неориентированный граф

Связанность в орграфе

Связанность в неорграфе

Орграф и неорграф простыми словами

Компонента связанности

Красивые картинки

Снова красивые картинки

Граф

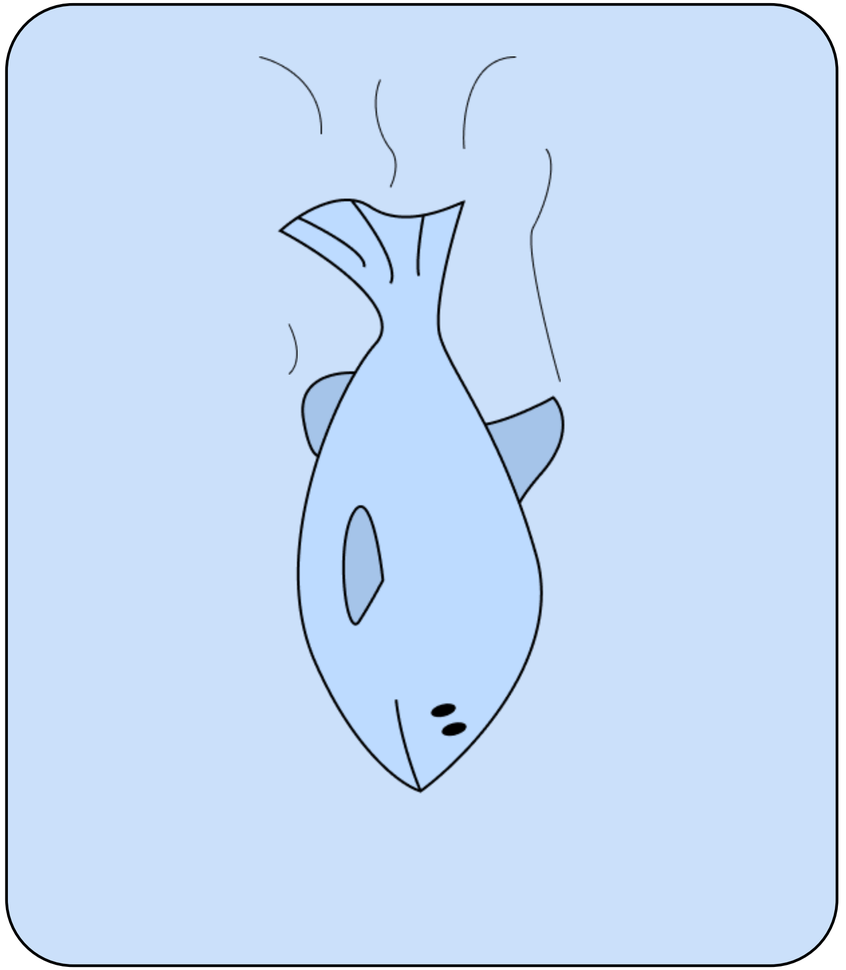
Абстрактный тип данных, который предназначен для реализации понятий неориентированного графа и ориентированного графа из области теории графов в математике

Обход графа

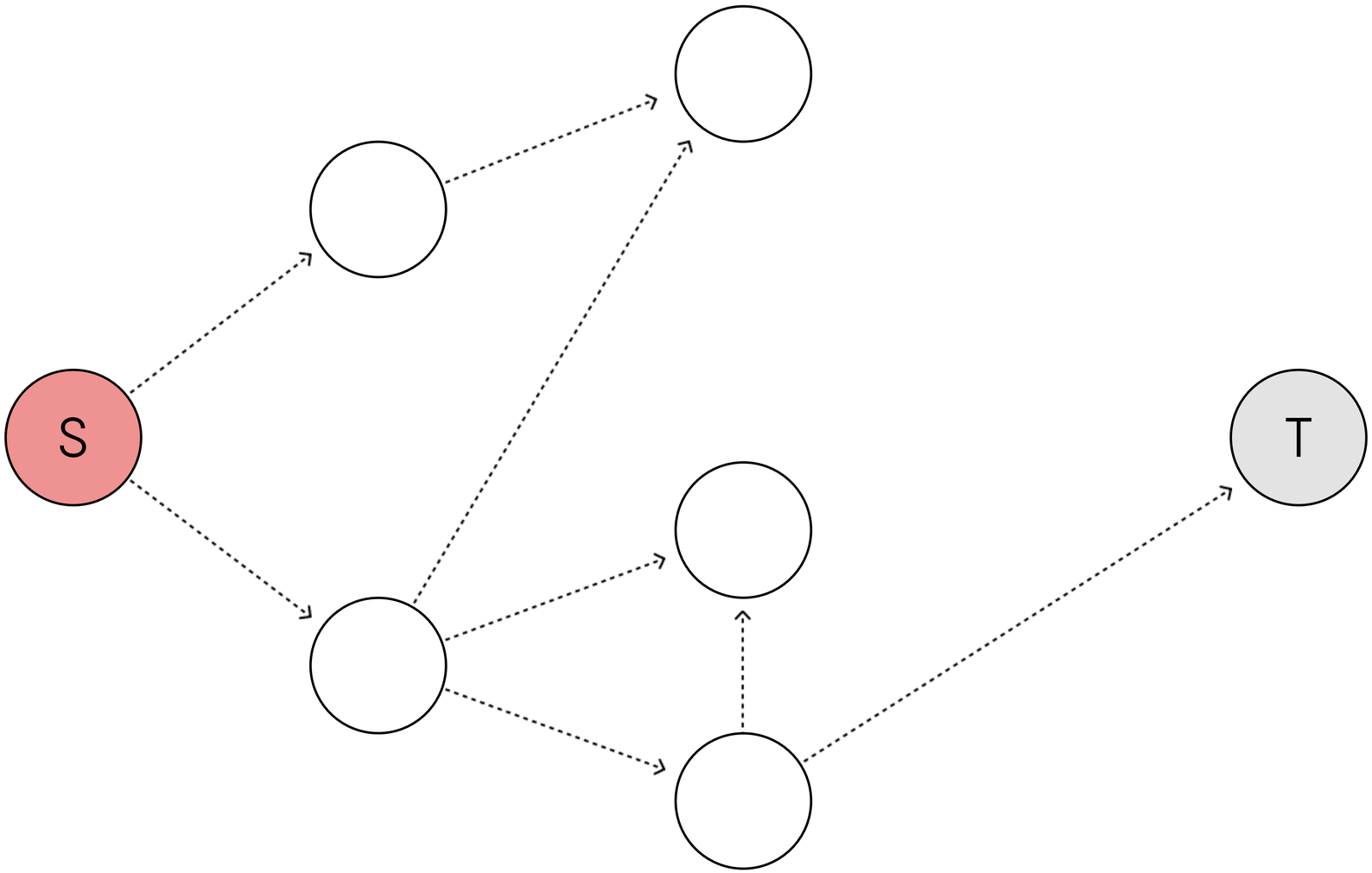
Процесс систематического просмотра всех ребер или вершин графа с целью отыскания ребер или вершин, удовлетворяющих некоторому условию

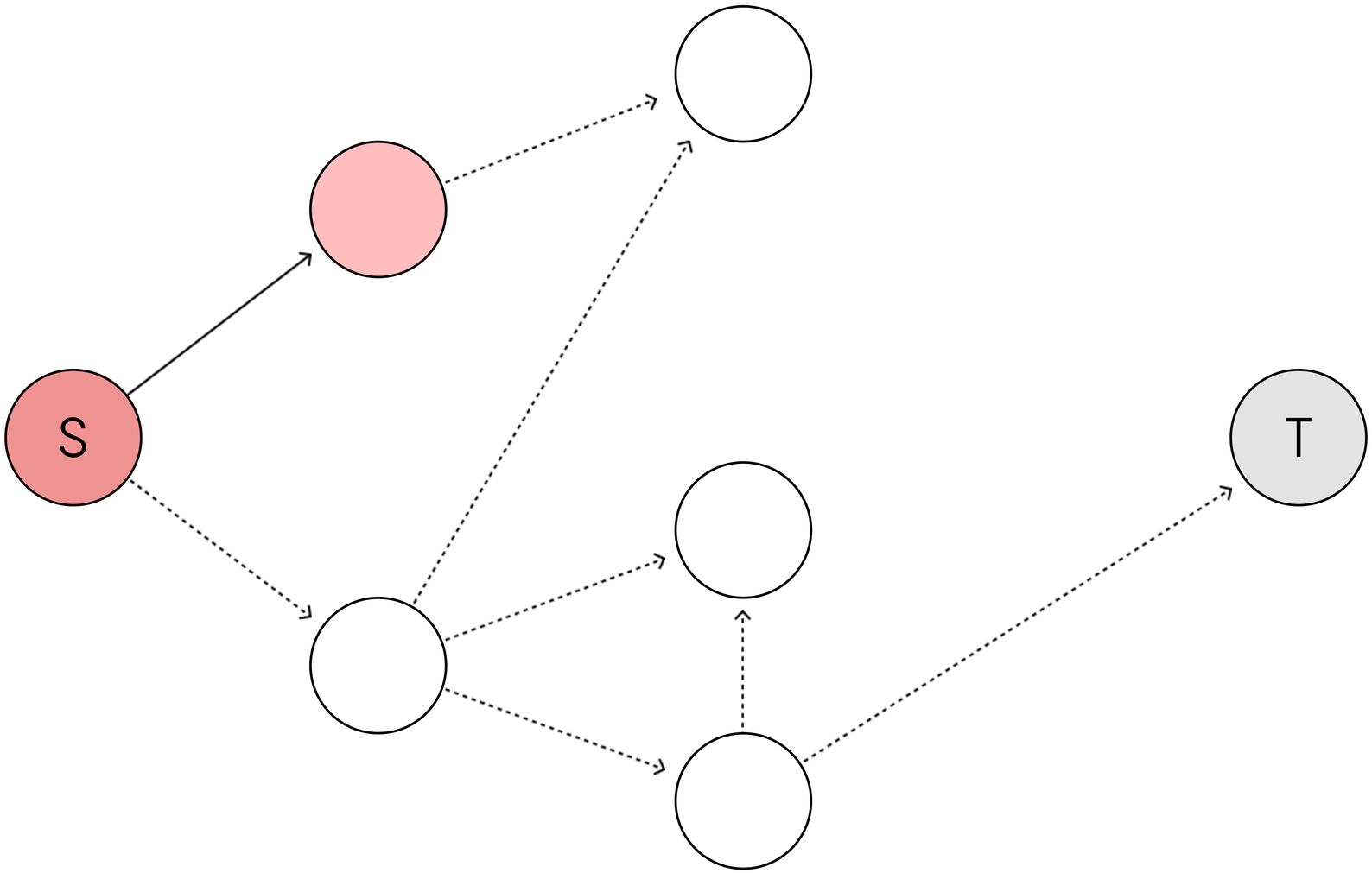
Depth-First Search (Поиск в глубину)

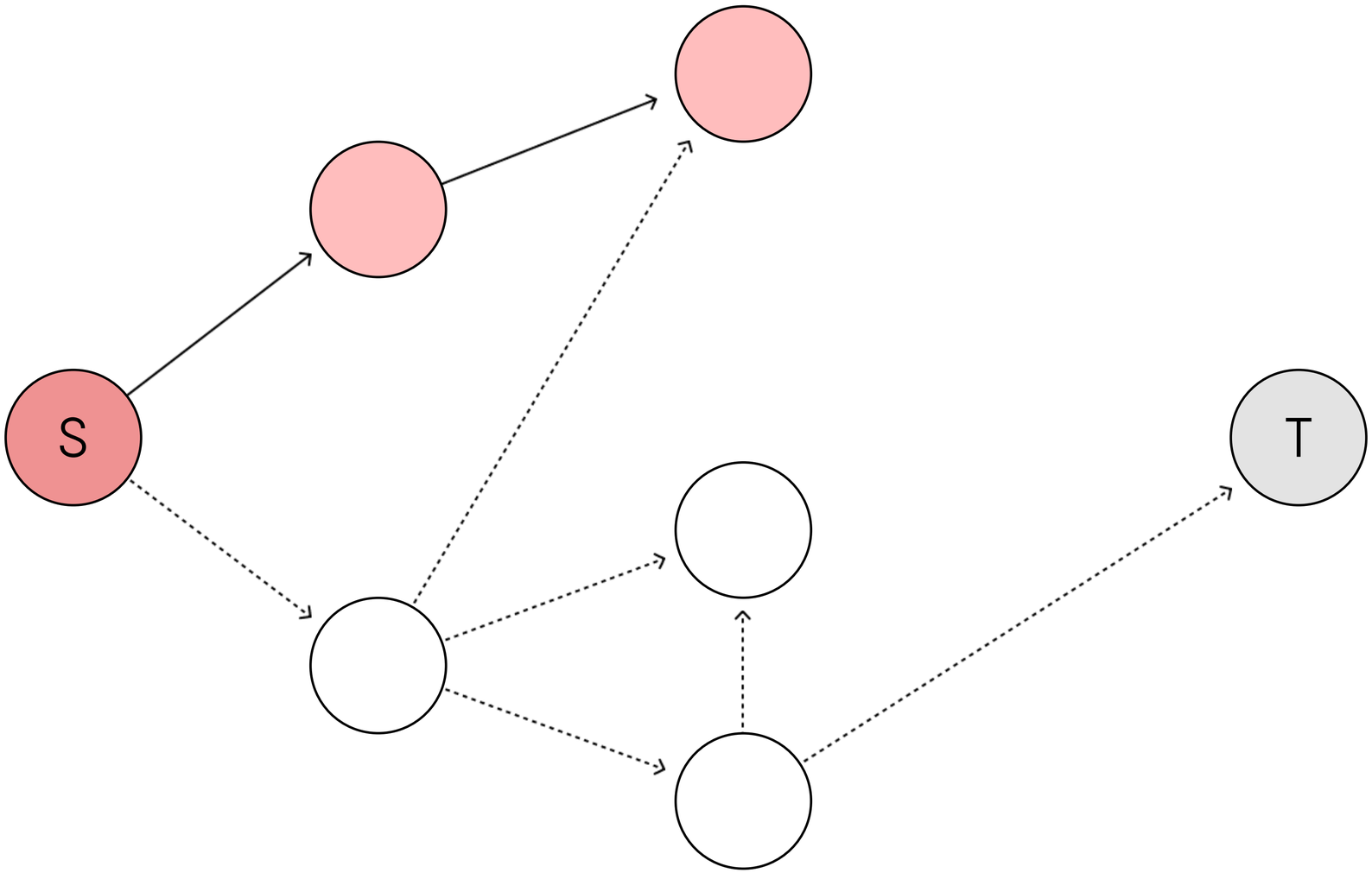
DFS следует концепции «погружайся глубже, головой вперед» («go deep, head first»). Мы двигаемся от начальной вершины в определенном направлении (по определенному пути) до тех пор, пока не достигнем конца пути или пункта искомой вершины. Если мы достигли конца пути, но он не является пунктом назначения, то мы возвращаемся назад и идем по другому маршруту.

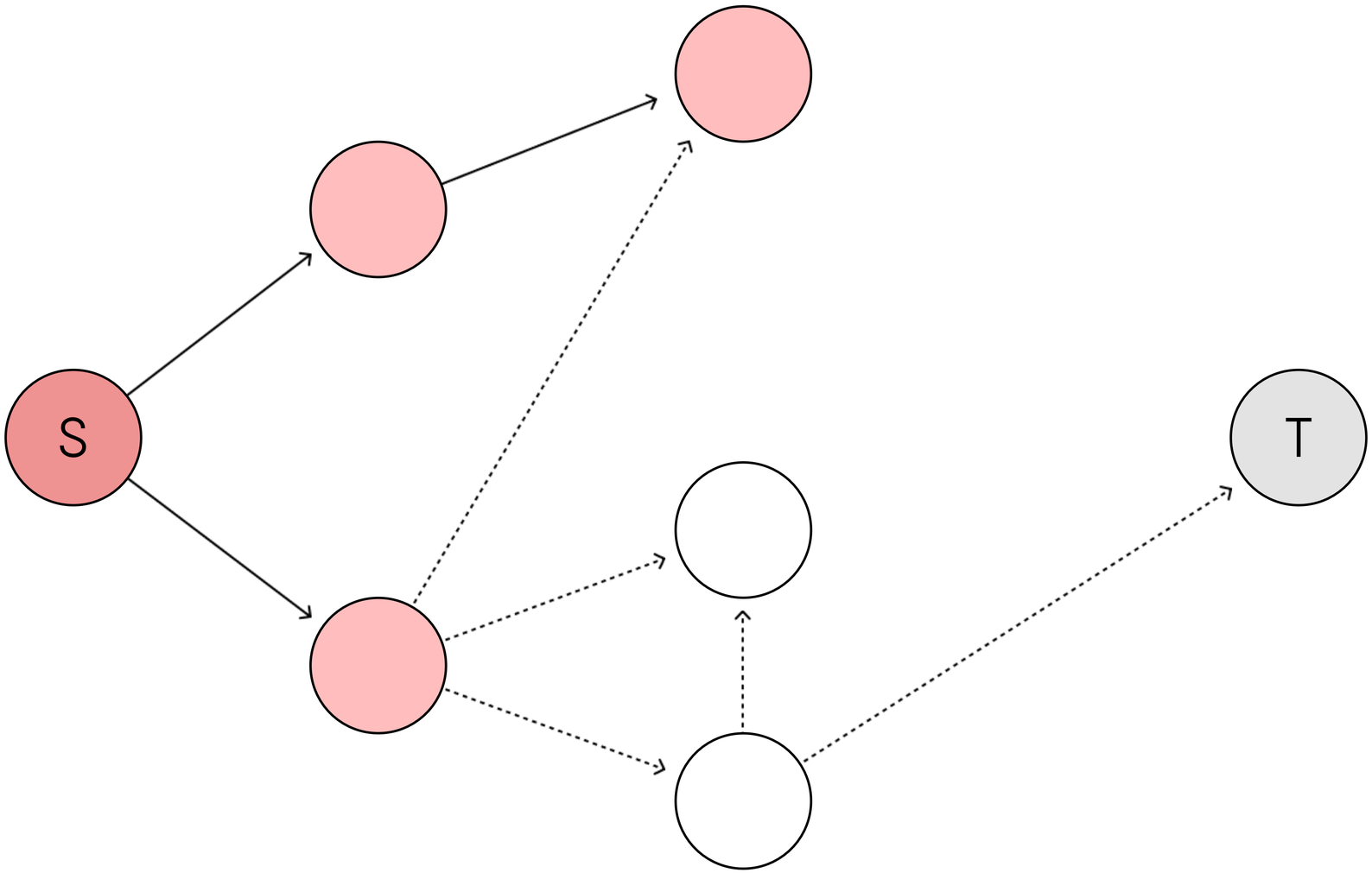


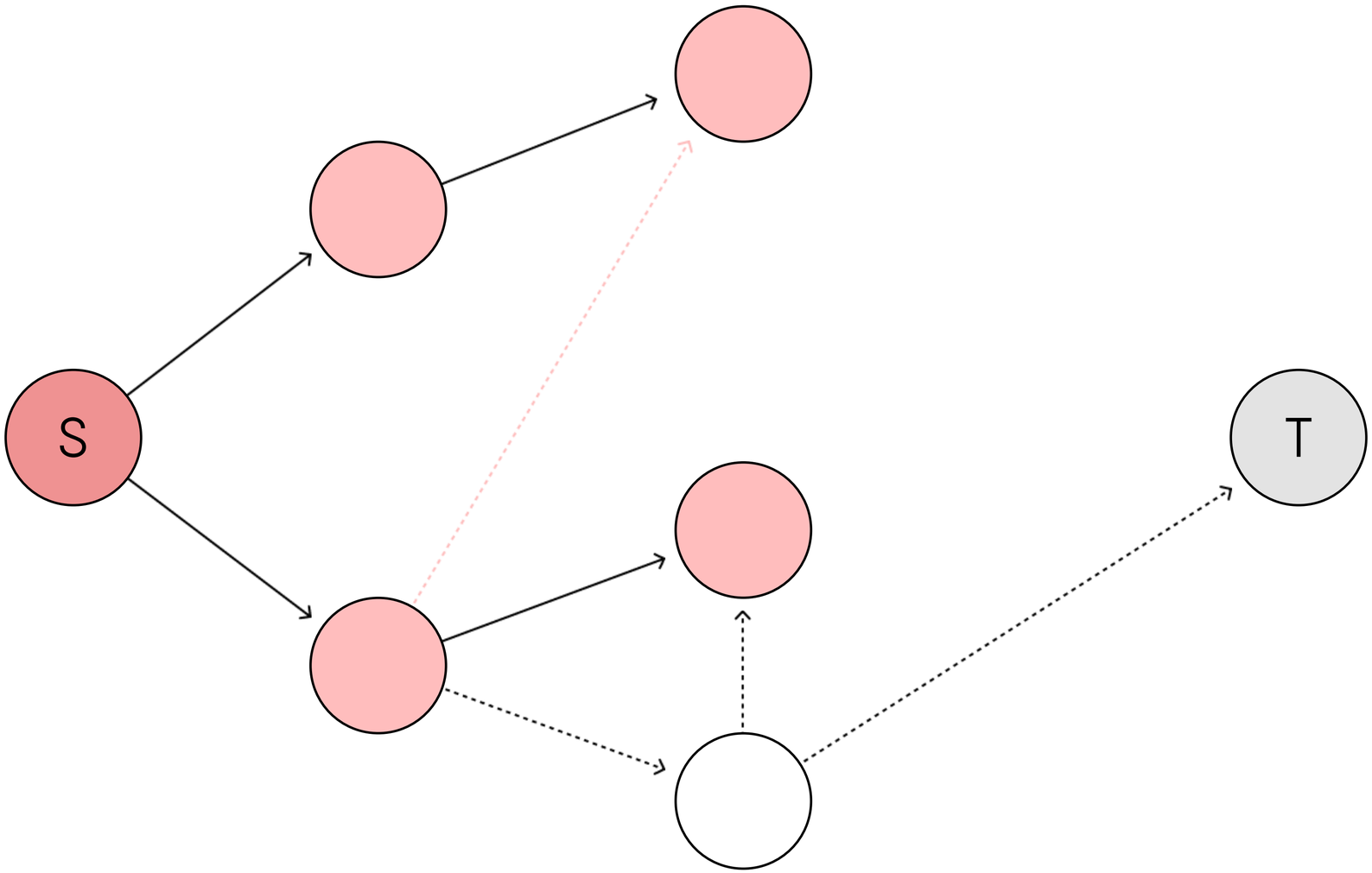
Go deep, head first

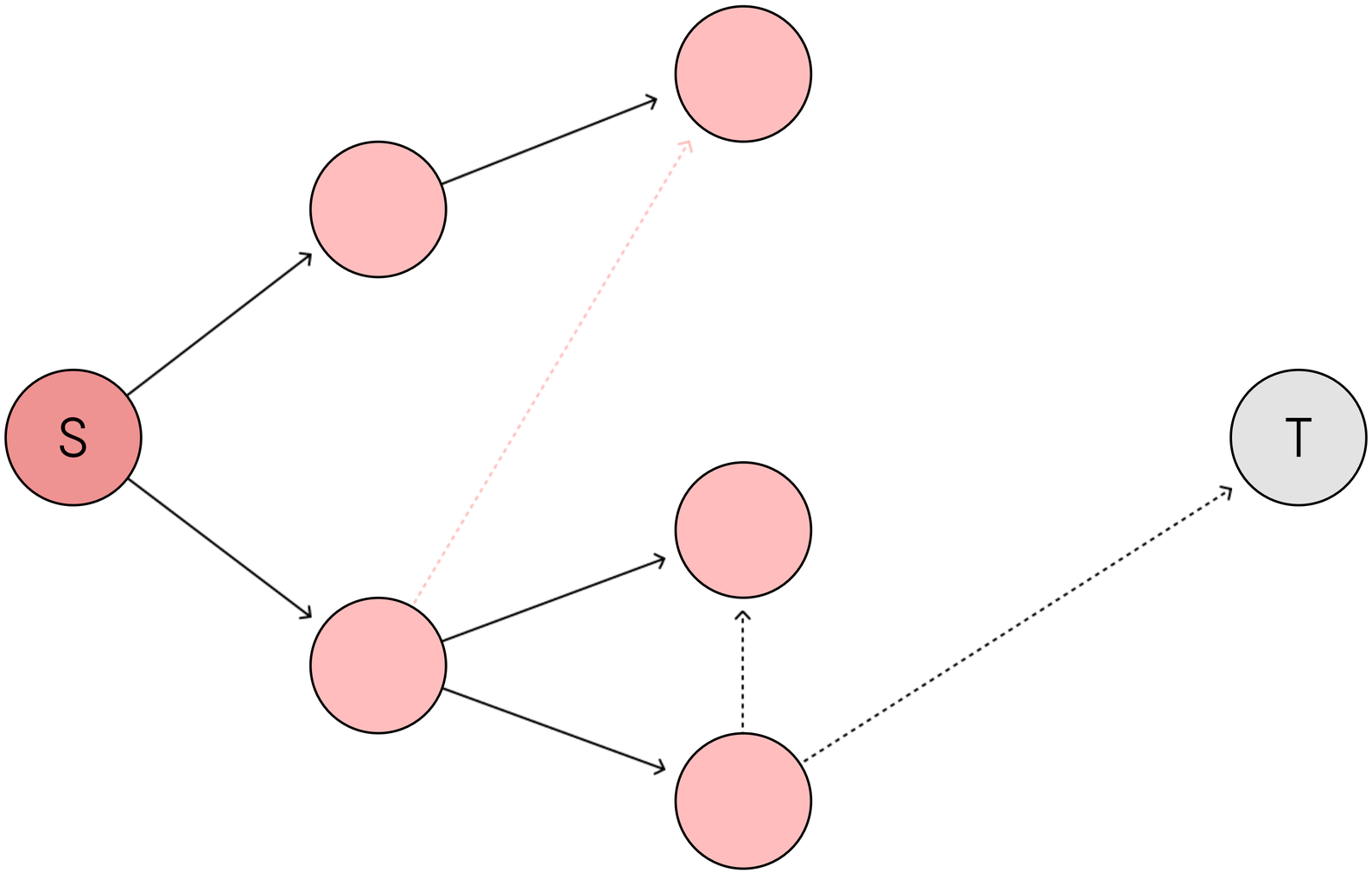

S
T

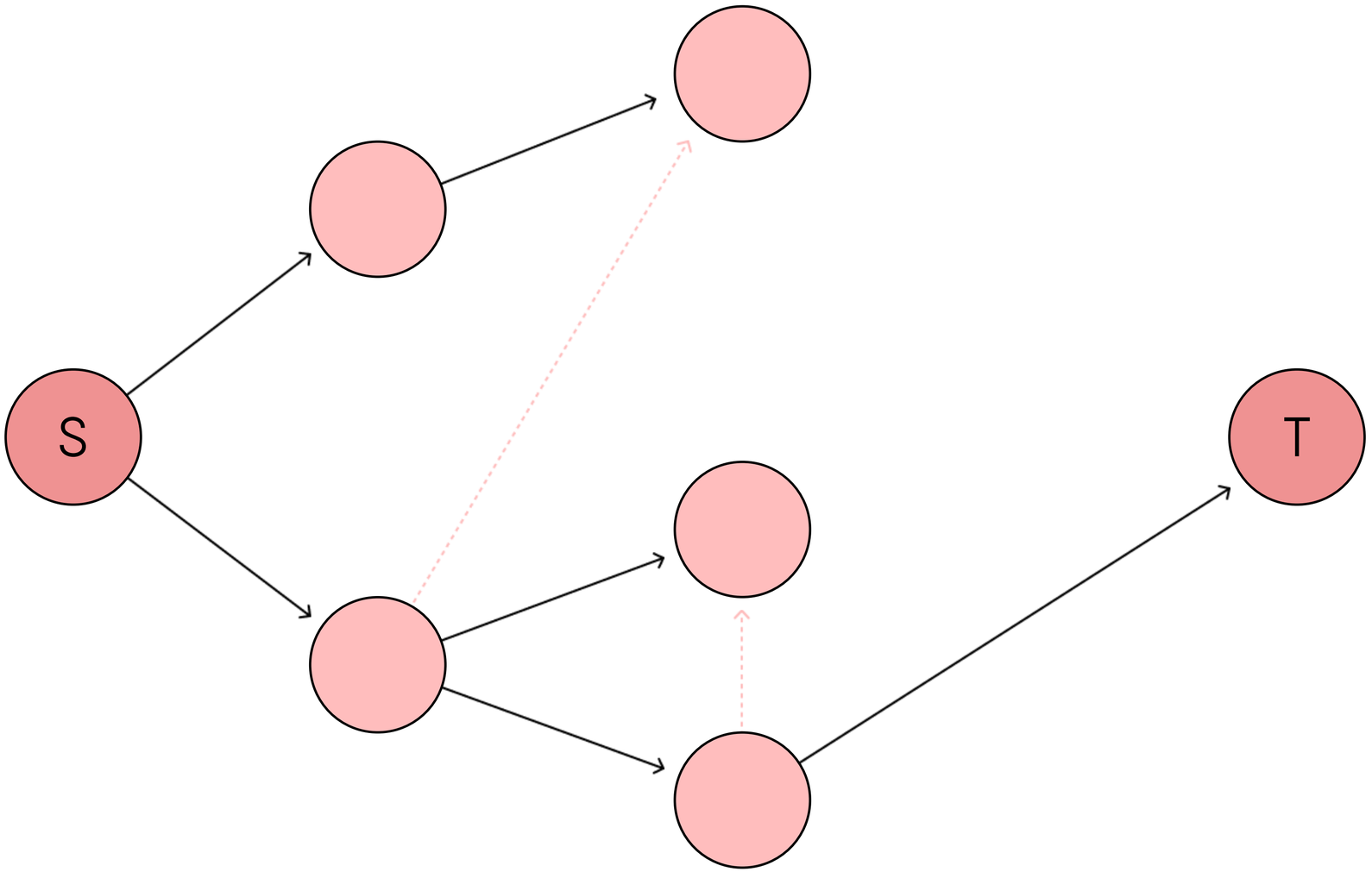

S
T


S
T


S
T


S
T


S
T


S
T

Как работает DFS?

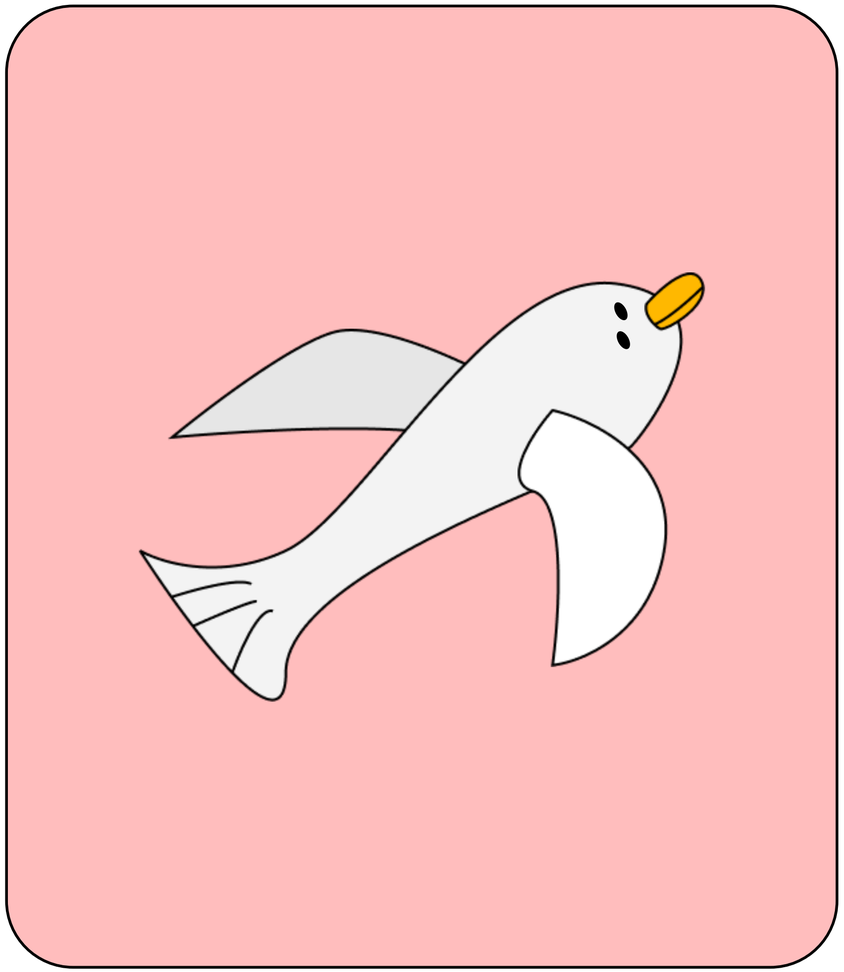
Двигаемся по определенному пути до конца. Если конец пути — это искомая вершина, мы закончили. Если нет, возвращаемся назад и двигаемся по другому пути до тех пор, пока не исследуем все варианты. Мы следуем этому алгоритму применительно к каждой посещенной вершине.

Реализация C++

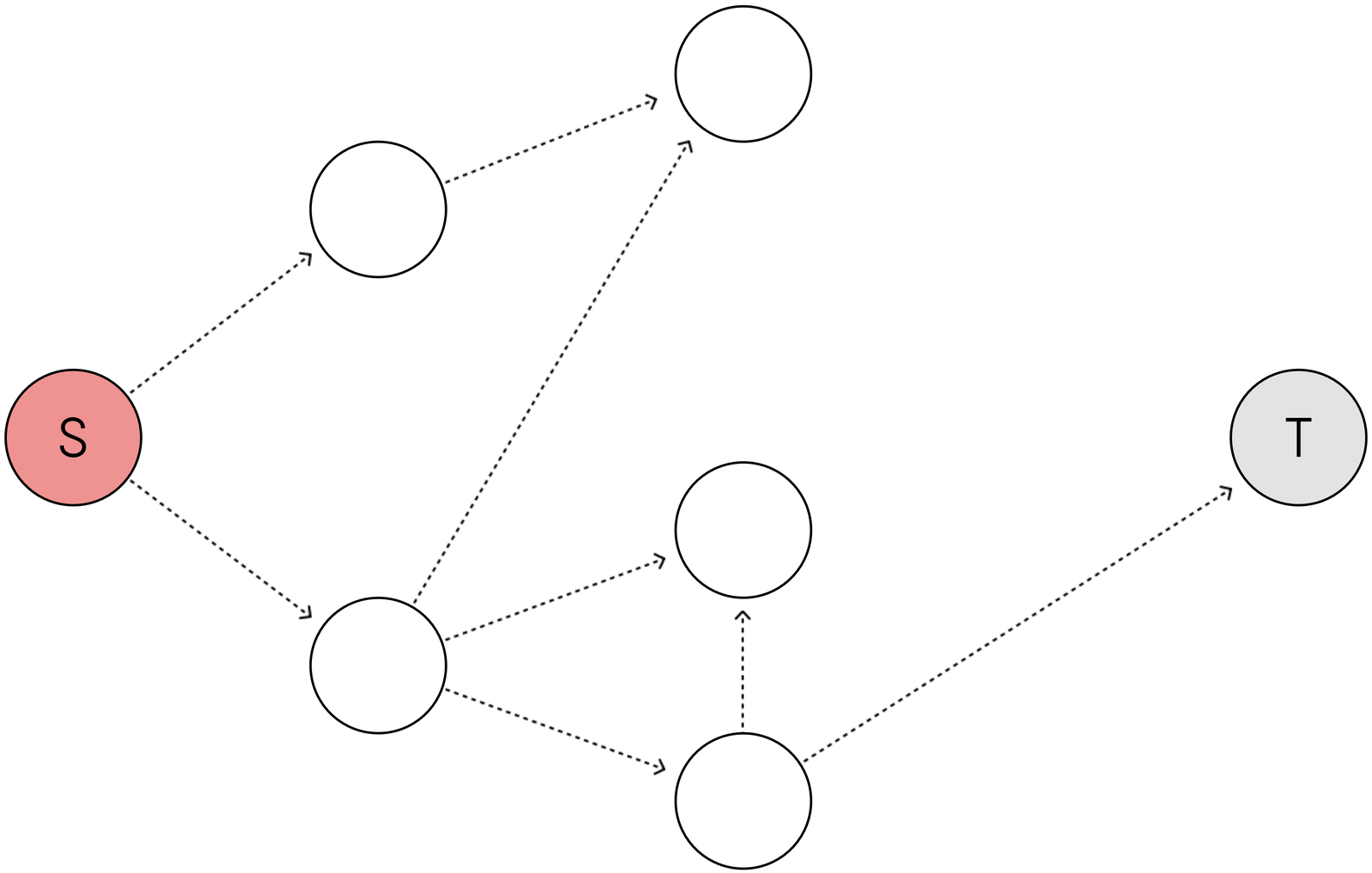
```
void dfs (int v) {
    if (flag == 1) return;
    else {
        used[v] = 1;
        path.push_back(v);
        for (int i = 0; i < graph[v].size(); i++) {
            int to = graph[v][i];
            if (used[to] == 1) {
                path.push_back(to);
                flag = 1;
                return;
            }
            else { dfs(to); }
            if (flag == 1) return;
        }
        used[v] = 2; path.pop_back();
    }
}
```

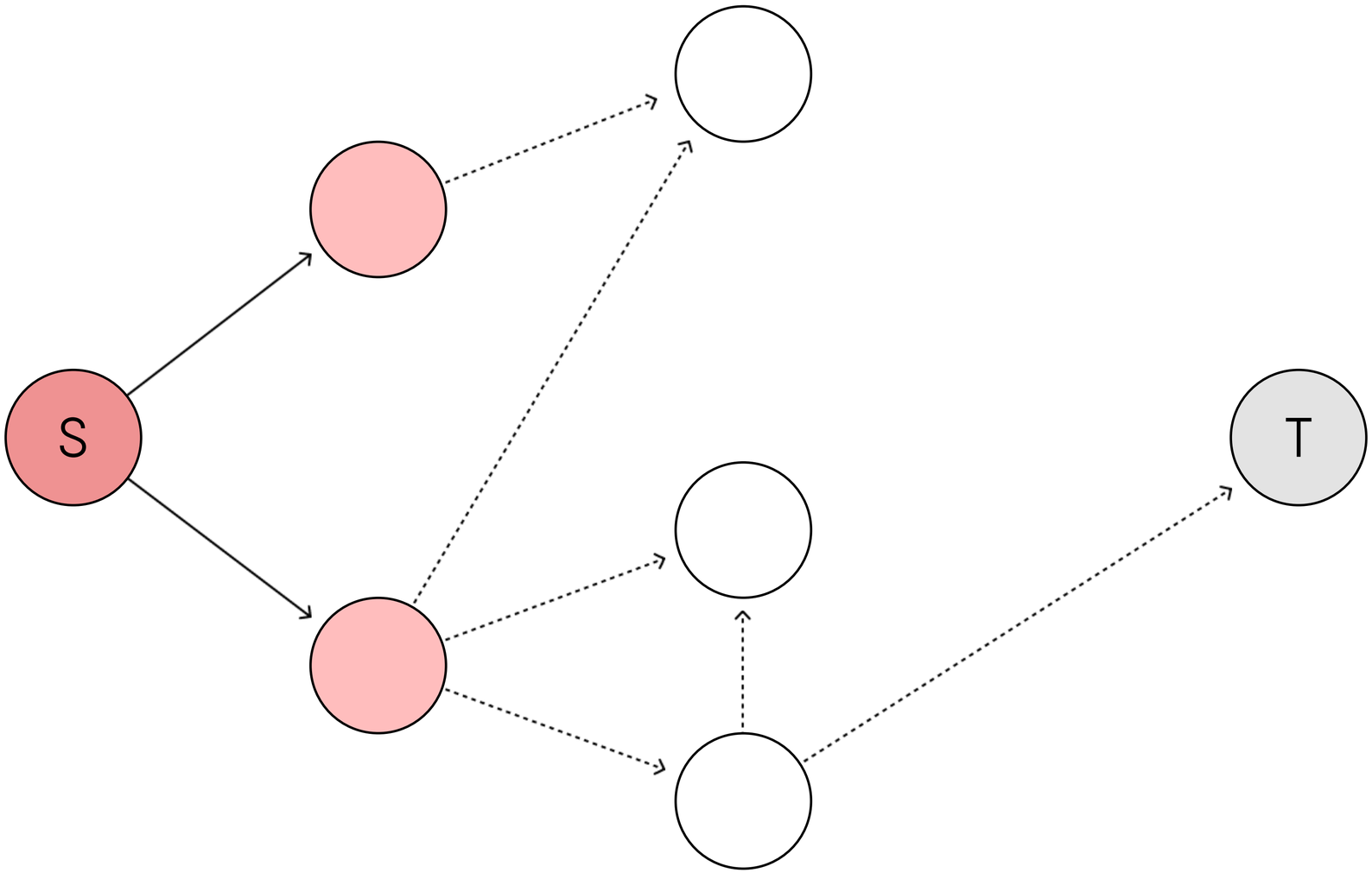
Breadth-First Search (Поиск в ширину)

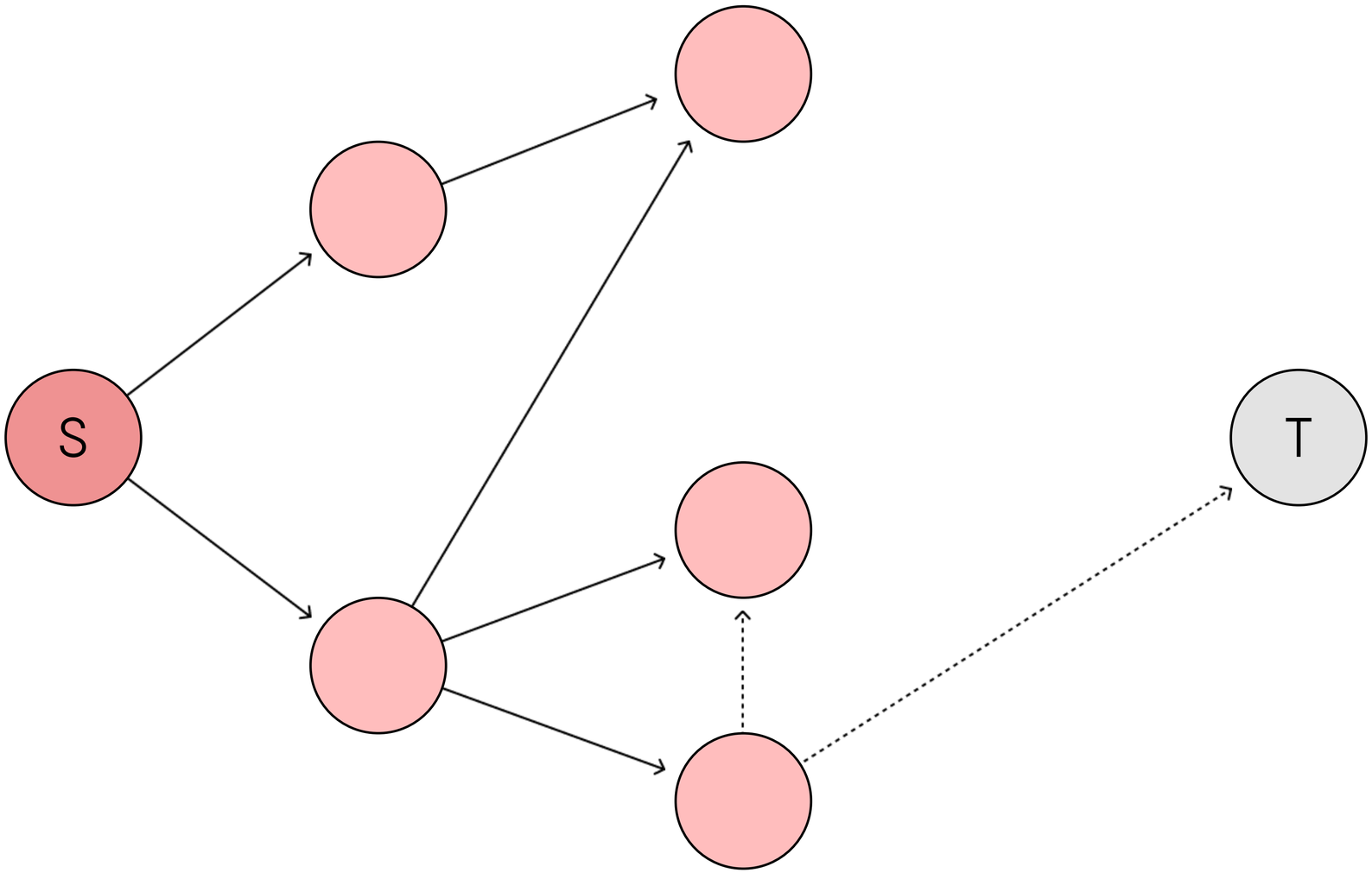
BFS следует концепции «расширяйся, поднимаясь на высоту птичьего полета» («go wide, bird’s eye-view»). Вместо того, чтобы двигаться по определенному пути до конца, BFS предполагает движение вперед по одному соседу за раз.

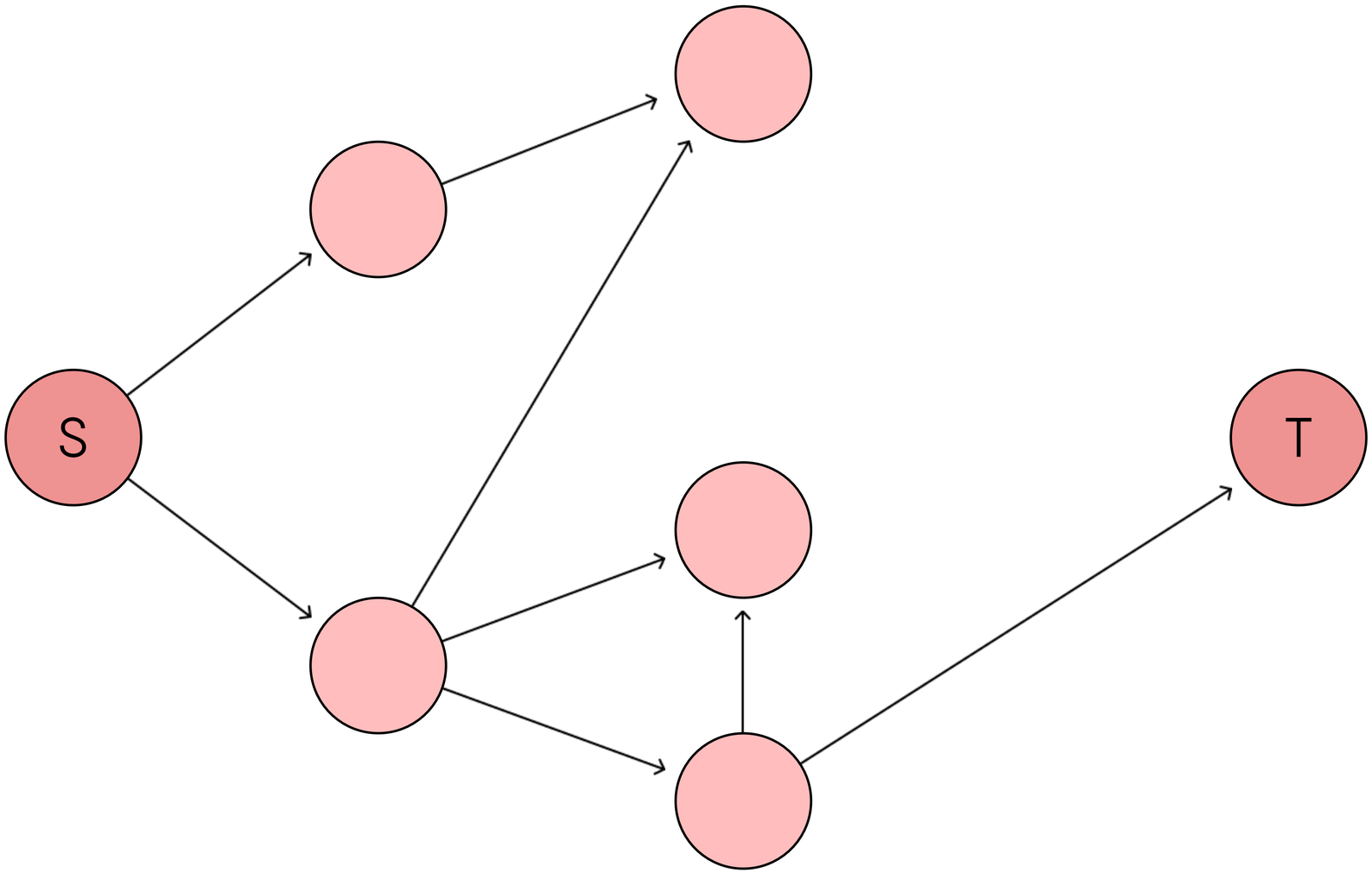


Go wide, bird’s eye-view


S
T


S
T


S
T


S
T

Как узнать каких соседей посетить первыми?

Для этого мы можем воспользоваться концепцией «первым вошел, первым вышел» (first-in-first-out, FIFO) из очереди (queue).

Мы помещаем в очередь сначала ближайшую к нам вершину, затем ее непосещенных соседей, и продолжаем этот процесс, пока очередь не опустеет или пока мы не найдем искомую вершину.

Реализация C++

```
vector<int> graph[100000];
bool used[100000];

int main()    {
    queue<int> q;
    q.push(0);
    used[0] = true;
    while (!q.empty())    {
        int cur = q.front();
        q.pop();
        cout << "BFS at vertex " << cur + 1 << endl;
        for (int neighbor: graph[cur])    {
            if (!used[neighbor])    {
                q.push(neighbor);
                used[neighbor] = true;
            }
        }
    }
}
```

DFS

Время выполнения O(V + E).

V — общее количество вершин.
E — общее количество граней (ребер).

BFS

Время выполнения O(V + E).

V — общее количество вершин.
E — общее количество граней (ребер).

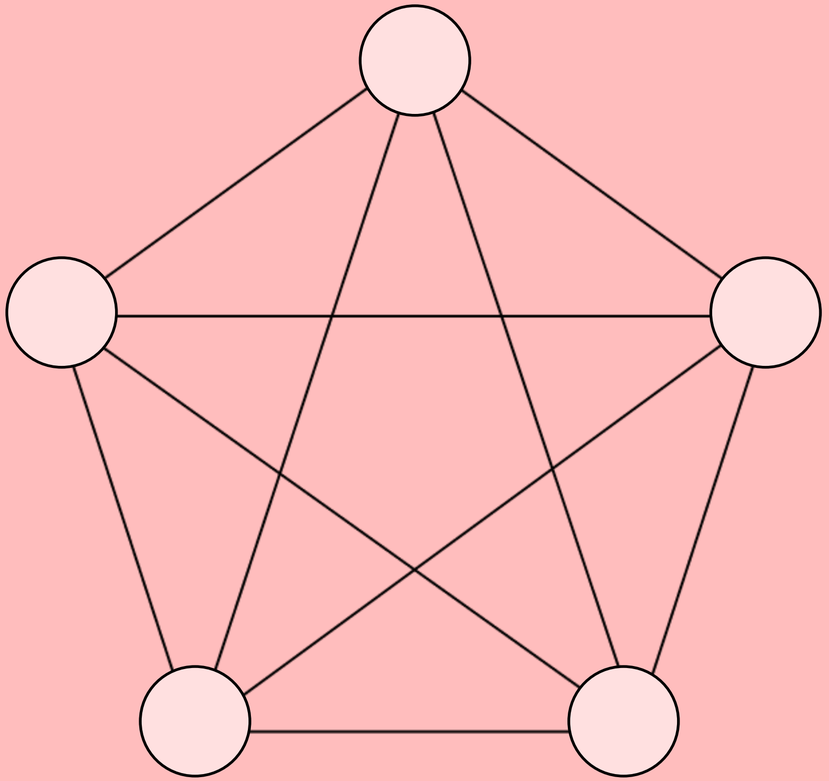
Чем DFS отличается от BFS?

Несмотря на то, что BFS может казаться медленнее, на самом деле он быстрее, поскольку при работе с большими графами обнаруживается, что DFS тратит много времени на следование по путям, которые в конечном счете оказываются ложными. BFS часто используется для нахождения кратчайшего пути между двумя вершинами.

Связность в ориентированных и неориентированных графах

Связный граф

Это граф, содержащий ровно одну компоненту связности. Это означает, что между любой парой вершин этого графа существует как минимум один путь.



Связный граф

Ориентированый граф

Это пара (V, E), где V – конечное множество вершин (узлов, точек) графа, а E – некоторое множество пар вершин, т.е. подмножество множества V x V или бинарное отношение на V. Элементы E называют ребрами (дугами, стрелками, связями). Для ребра e= (u,v) ϵ E вершина u называется началом e, а вершина v – концом e, говорят, что ребро e ведет из u в v.

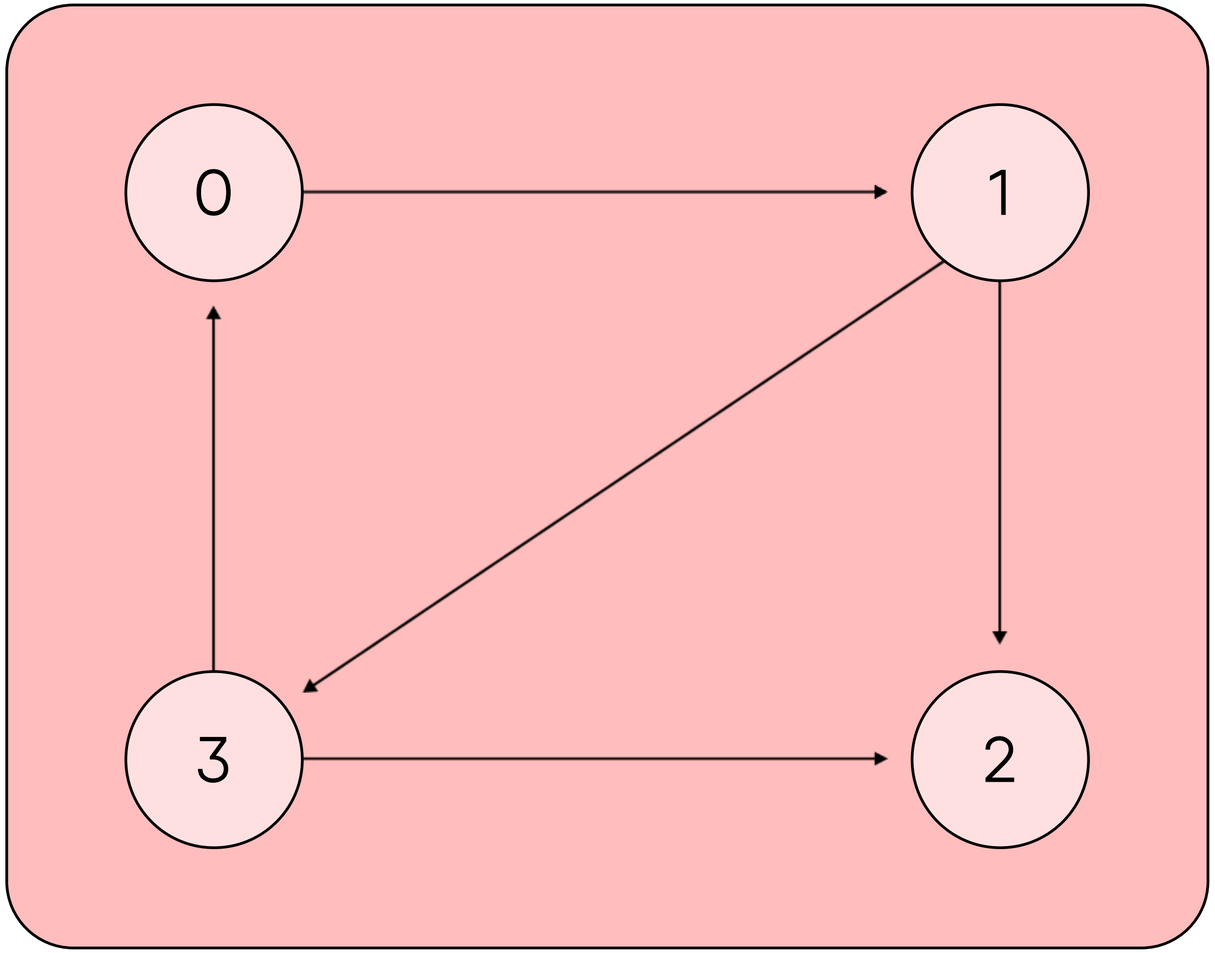
Неориентарованный граф

Это ориентированный граф, у которого для каждого ребра (u,v) ϵ E имеется противоположное ребро (v,u) ϵ E, т.е. отношение E симметрично. Такая пара (u,v), (v,u) называется неориентированным ребром. Для его задания можно использовать обозначение для множества концов: {u, v}, но чаще используется указание одной из пар в круглых скобках. Если e= (u,v) ϵ E, то вершины u и v называются смежными в G, а ребро e и эти вершины называются инцидентными.

Простыми словами

Ориентированный граф (орграф) — (мульти) граф, рёбрам которого присвоено направление. Направленные рёбра именуются также дугами.

Граф, ни одному ребру которого не присвоено направление, называется неориентированным графом или неорграфом.



Ориентированный граф

Связность

В неорграфе

Неориентированный граф называется связным, если все его вершины достижимы из некоторой вершины (эквивалентно, из любой его вершины).

В орграфе

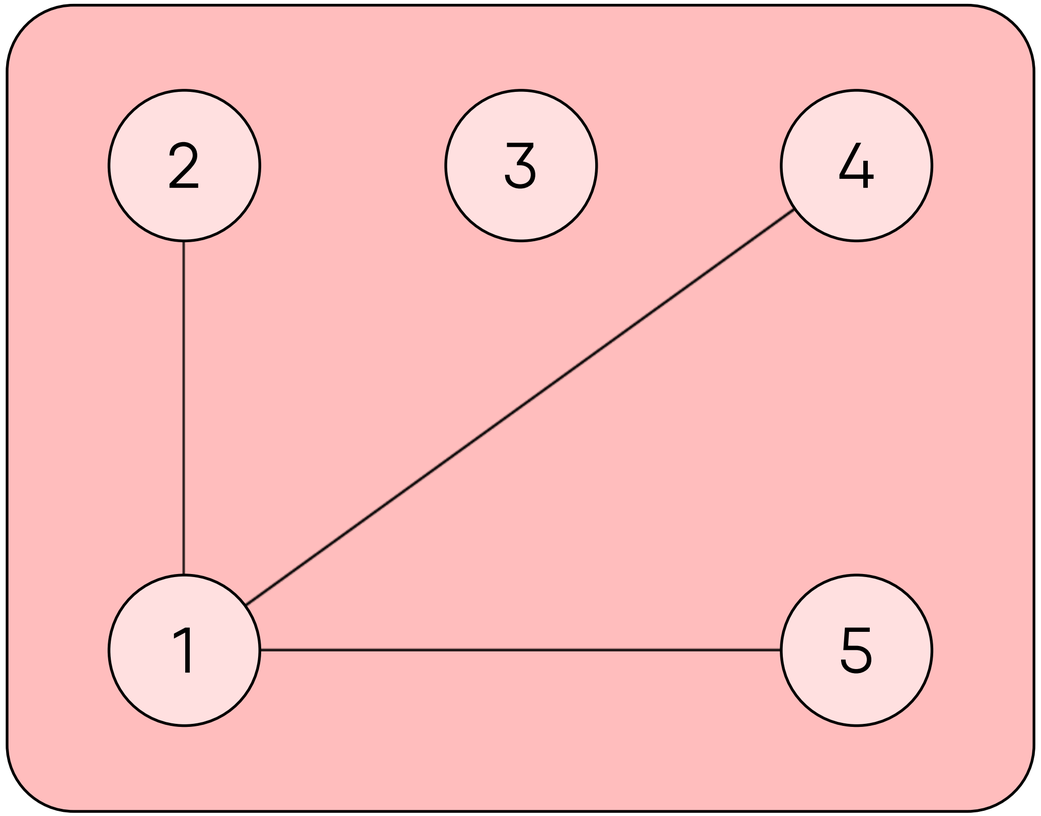
Ориентированный граф называется

- слабо связным, если соответствующий неориентированный граф является связным
- сильно связным, если всякая вершина достижима из любой другой вершины.

Компонента связности

Компонентой связности неориентированного графа называется его связный подграф, не являющийся собственным подграфом никакого другого связного подграфа данного графа (максимально связный подграф).

У неориентированного графа, изображенного на этом рисунке две компоненты связности. Первая компонента связности включает вершины 1, 2, 4, 5, а вторая состоит из одной вершины (3)



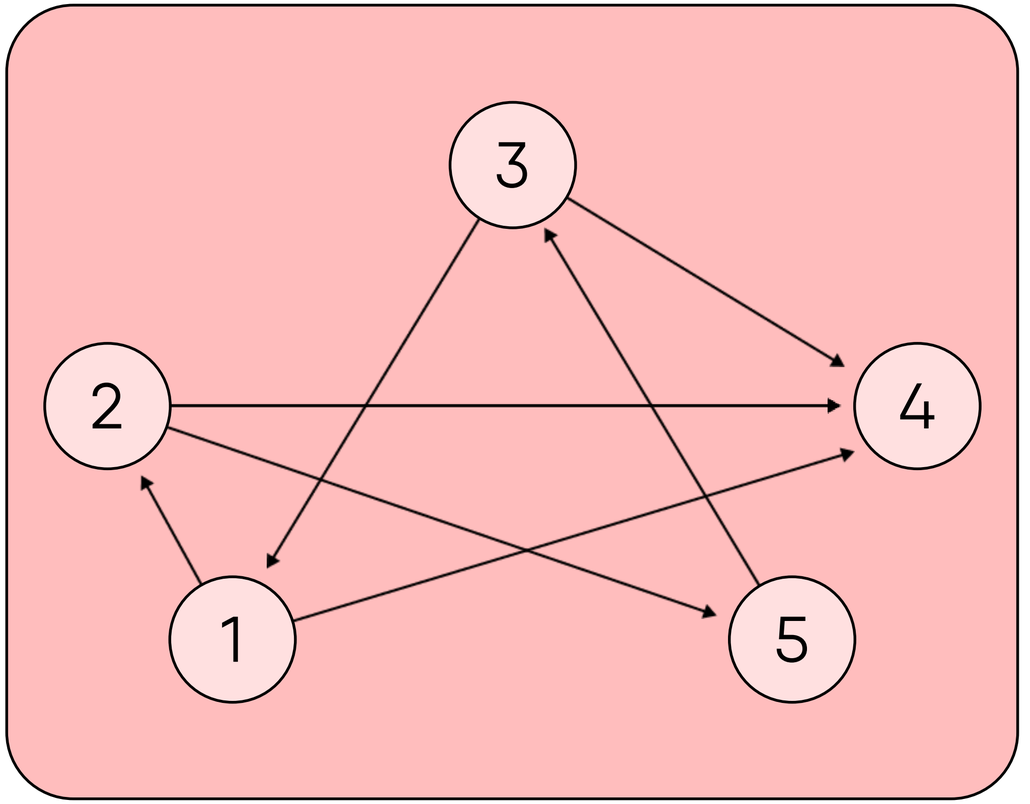
Неориентированный граф

У ориентированного графа, изображенного на этом рисунке две компоненты сильной связности.

Первая компонента связности включает вершины 1, 2, 3, 5, а вторая состоит из одной вершины (4).

Для любой пары вершин из первого компонента существует хотя бы один путь, соединяющий эти вершины.

Из вершины 4 нет пути ни в одну вершину графа.



Ориентированный граф