Министерство образования Республики Беларусь

Учреждение образования «Полоцкий государственный университет»

Факультет информационных технологий

Кафедра технологий программирования

**Реферат «Обход графа в глубину и ширину. Связность в ориентированных и неориентированных графах.»**

Выполнили: Демчишина А.Ю., гр. 21-ИТ-3

Шиковец Е.А., гр. 21-ИТ-1

Проверила: преподаватель

Деканова М.В.

Полоцк 2022

Введение

Граф — это абстрактный тип данных, который предназначен для реализации понятий неориентированного графа и ориентированного графа из области теории графов в математике. Программирование графами улучшает характеристики по наглядности, простоте, компактности, скорости ввода и занимаемой памяти компьютера. Для поиска определенного элемента в графе используется обход графа.

Простыми словами, обход графа — это переход от одной его вершины к другой в поисках свойств связей этих вершин. Связи (линии, соединяющие вершины) называются направлениями, путями, гранями или ребрами графа. Вершины графа также именуются узлами.

Двумя основными алгоритмами обхода графа являются поиск в глубину (Depth-First Search, DFS) и поиск в ширину (Breadth-First Search, BFS). Несмотря на то, что оба алгоритма используются для обхода графа, они имеют некоторые отличия. Рассмотрим каждый из них подробнее и начнем с DFS.

**Обход графа в глубину**

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

Так 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. Необходимость многократного повторения процедуры указывает на необходимость использования рекурсии для реализации алгоритма.

Давайте проанализируем этот алгоритм. Поскольку мы обходим каждого «соседа» каждого узла, игнорируя тех, которых посещали ранее, мы имеем время выполнения, равное O(V + E), где V — общее количество вершин. E — общее количество граней (ребер).

Может показаться, что правильнее использовать V\*E, однако стоит разобрать, что означает V\*E. V\*E означает, что применительно к каждой вершине, мы должны исследовать все грани графа безотносительно принадлежности этих граней конкретной вершине.

С другой стороны, V+E означает, что для каждой вершины мы оцениваем лишь примыкающие к ней грани. Возвращаясь к примеру, каждая вершина имеет определенное количество граней и, в худшем случае, мы обойдем все вершины (O(V)) и исследуем все грани (O(E)). Мы имеем V вершин и E граней, поэтому получаем V+E.

Далее, поскольку мы используем рекурсию для обхода каждой вершины, это означает, что используется стек (бесконечная рекурсия приводит к ошибке переполнения стека). Поэтому пространственная сложность составляет O(V).

**Обход графа 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.

Чем DFS отличается от BFS? Простым языком, DFS идет напролом, а BFS не торопится, а изучает все в пределах одного шага.

Далее возникает вопрос: как узнать, каких соседей следует 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. Очередь предполагает обработку каждой вершины перед достижением пункта назначения. Это означает, что, в худшем случае, BFS исследует все вершины и грани.

Может показаться, что BFS работает медленнее, но 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.

Таким образом, время выполнения BFS также составляет O(V + E), а поскольку мы используем очередь, вмещающую все вершины, его пространственная сложность составляет O(V).

**Связность в ориентированных и неориентированных графах**

Ориентированный 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.

Неориентированный граф G = (V, E) - 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.

Проще говоря, 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.

Вершины произвольного графа можно разбить на классы, такие, что для любых двух вершин одного класса v1 и v2 существует путь из v1 в v2. Эти классы называются компонентами связности. Если у графа одна компонента связности, то граф называется связным.

Неориентированный граф называется связным, если все его вершины достижимы из некоторой вершины (эквивалентно, из любой его вершины). Компонентой связности неориентированного графа называется его связный подграф, не являющийся собственным подграфом никакого другого связного подграфа данного графа (максимально связный подграф). Ориентированный граф называется слабо связным, если соответствующий неориентированный граф является связным, и сильно связным, если всякая вершина достижима из любой другой вершины.

Algorithms on Graphs: Let’s talk Depth-First Search (DFS) and Breadth-First Search (BFS) - https://trykv.medium.com/algorithms-on-graphs-lets-talk-depth-first-search-dfs-and-breadth-first-search-bfs-5250c31d831a