March 6, 2025

Графы

1. Граф - совокупность двух множеств - вершин и ребер, их соединяющих (ребра не обязаны соединять все вершины графа между собой; также необязательно то, что из любой вершины графа можно попасть в любую).

2. Максимальное число ребер M простого графа зависит от ориентированности графа. Для ориентированного графа это число в два раза больше (так как любые две вершины могут соединять два ребра, а не одно, как в неориентированном). Пусть N - количество вершин графа. Для неориентированного графа

$$M_{\text{неориент}} = (N-1) + (N-2) + ... + 2 + 1 = N \cdot (N-1)$$

$$M_{\text{ориент}} = 2 \cdot N \cdot (N-1)$$

   Стоит отметить, что с точки зрения асимптотики эти величины равны.

3. Пусть A - матрица смежности.

   Граф неориентирован $\iff A^T = A$

4. Для хранения взвешенного графа в списке ребер, помимо вершин, определяющих ребро нужно будет хранить также и вес ребра. В списке смежности хранить вес ребра довольно неудобно. Наиболее рациональным кажется отказаться от хранения взвешенного графа таким образом. Однако если этот вариант в силу каких-либо причин не является приемлемым, может помочь создание словаря, где ключом будет пара вершин, определяющих ребро, а значением вес этого ребра.

5. Компонента связности графа - часть графа (наибольшего размера), из каждой вершины которой можно достичь любой другой вершины хотя бы одним способом. Очевидно, что число компонент связности не может быть больше N (N может достигаться в ориентированном графе) и не может быть меньше 1.

6. Наверное как-то можно, но мне это кажется довольно сложным в реализации, да и какая-то выгода по сравнению с DFS может быть только при поиске коротких (состоящих из малого числа вершин) циклов.

7. Если в графе содержится ребро с отрицательным весом, алгоритм Дейкстры неприменим независимо от того, есть отрицательный цикл (в этом случае задача некорректна (см. след. пункт)) или нет. Дело в том, что мы на каждом шаге выбираем лучший вариант, попадания в определенную вершину, отбрасывая все, которые хуже. Когда отрицательных ребер нет такой подход полностью оправдан. Однако при наличии отрицательных ребер более выгодным может оказаться перейти в новую вершину из вершины с большим весом, чем уже имеет она. Алгоритм Дейкстры не рассматривает такие случаи. Если же он будет рассматривать такие случаи, у нас упадет скорость работы и это уже не будет в полной мере алгоритмом Дейкстры. Так что я не вижу как можно решить эту проблему для Алгоритма Дейкстры не изменяя веса.

8. Алгоритм Форда-Беллмана нужно запускать в N-й раз, чтобы можно было определить отрицательный цикл. Если после N-й итерации расстояние между какими-то вершинами уменьшилось, это значит, что в графе есть отрицательный цикл, так как иначе кратчайший путь был бы уже (за N-1 итераций) найден. При наличии отрицательного цикла задача о поиске кратчайшего пути некорректна, так как в этом случае всегда можно сделать путь более коротким с помощью следующей итерации по отрицательному циклу.

9. Чтобы алгоритм Беллмана-Форда мог отработать быстрее Дейкстры, граф должен быть разреженным, так как в этом случае оба алгоритма имеют асимптотику $O(N^2)$. Я не совсем понимаю засчет чего даже в этом случае Форд-Беллман может отработать быстрее Дейкстры, если не рассматривать какие-то совсем уж специфические случаи, наподобие линейного графа. Однако Дейкстра с двоичной кучей в любом случае имеет лучшую асимптотику, чем Форд-Беллман.

10. Рекурсивная реализация DFS более интуитивно понятна и как будто бы проще в написании. Очевидно, она не может быть применена для очень больших графов, где неизбежно будет возникать переполнение стека рекурсии. Если этого не происходит я бы предпочел использовать рекурсивную реализацию, так как она более интуитивно понятна и проще в написании, однако я не вижу каких-либо принципиальных недостатков

в итеративной реализации. В итеративной версии применение списка в качестве стека вполне оправдано, так как удаление последнего элемента списка происходит за $O(1)$. Использовать список в качестве очереди, например в итеративной реализации BFS, однако, невыгодно, так как удаление первого элемента списка происходит за $O(N)$.