Контрольные вопросы по информатике №1

Паршина Эмилия

Февраль-март 2025

1 Что такое граф?

Граф – объект в математике из двух множеств: вершин (узлов) и рёбер (их парных связей). Т.е. у нас вершины это какие-то объекты, а рёбра показывают связи между этими объектами. Много что можно представить в виде графа (транспортные пути (отсюда логистика идёт), соцсети, семейное древо и тд.) Графы бывают ориентированными и неориентированными, связаными и не связаными и тд.

2 В случае простого графа (отсутствуют петли и кратные ребра), какое максимальное возможное количество ребер M? Зависит ли это число от ориентированности графа?

В простом графе с N вершинами максимальное количество рёбер M равно N(N-1)/2. То есть это когда каждая вершина соединена со всеми остальными и нет петель или кратных рёбер. Для ориентированного графа, где важен порядок соединения (из 1 во 2 и из 2 в 1 разные рёбра идут), максимальное количество рёбер будет N(N-1) (то есть N(N-1)/2 в одну сторону и столько же в обратную). Так что от ориентированности графа число максимальных рёбер зависит, да.

3 Как по заданной матрице смежности быстро проверить граф на орентированность?

Для этого можно просто проверить матрицу на симметричность: если матрица симметрична относительно главной диагонали (то есть, A[i][j] == A[j][i] для всех i и j), то граф неориентированный (ребро из любой вершины i в любую вершину j будет также являться ребром из j в i). Если же хоть одна пара вершин нарушает симметричность, то граф будет ориентированным.

4 Как изменятся структуры списка ребер и списка смежности в случае взвешенного графа?

Для взвешенного графа будет добавляться значение веса каждого ребра. То есть, каждый элемент списка рёбер будет не просто парой вершин (A, B), а тройкой (A, B, вес). В списке смежности тоже будет храниться вес: там будет указываться не только следующая (т.е. соседняя) вершина, а будет указываться пара (соседняя вершина, вес ребра), то есть будут такие кортежи храниться.

5 Что такое компонента связности графа? Каким может быть максимальное и минимальное кол-во компонент в одном графе?

Компонента связности – это набор из максимального числа таких вершин графа, в котором из любой вершины можно дойти до любой другой, двигаясь по рёбрам. То есть максимальное количество компонент связности - это когда каждая вершина изолирована, то есть в графе с N вершинами будет N компонент связности. Соответственно, минимальное количество компонент в графе - один, когда весь граф является одной компонентой связности.

6 Можно ли с помощью BFS искать циклы в графе? Выгоднее ли это чем поиск через DFS?

Да, с помощью BFS можно искать циклы в неориентированном графе. То есть если при обходе в BFS мы наткнулись на вершину, которая уже посещена и не является "родителем" текущей вершины, то это будет циклом. Но обычно BFS больше подходит для поиска кратчайшего пути, а DFS – для обхода всего графа или поиска каких-то конкретных путей, особенно с ориентированными графами удобен.

7 Зачем вводится требование на неотрицательность веса ребра в алгоритме Дейкстры? Можно ли заставить алгоритм корректно работать в случае отрицательных весов (без их изменения).

Алгоритм Дейкстры всегда выбирает ближайшую непосещенную вершину. Если же есть отрицательные веса, то пройдя по отрицательному ребру, можно “улучшить” (то есть уменьшить) уже найденное расстояние до вершины, что порушит всю логику алгоритма. Заставить Дейкстру корректно работать с отрицательными весами без их изменения нельзя, поэтому в случае отрицательных весов пользуются алгоритмом Форда-Беллмана.

8 Зачем запускать алгоритм Форда-Беллмана в N-й раз? Почему в определенных случаях задача поиска кратчайщего пути в принципе некорректна на нашем графе?

Алгоритм Форда-Беллмана запускают N-й раз, чтобы проверить наличие циклов отрицательного веса, то есть если после N-1 запуска на N-ом запуске расстояние до какой-то вершины уменьшилось, то, значит, в графе точно есть цикл с отрицательным весом, и тогда задача поиска кратчайшего пути становится некорректной, потому что можно бесконечно “улучшать” (т.е. уменьшать) путь, проходя по этому циклу. Поэтому алгоритм запускается N-й раз для обнаружения циклов таких как раз.

9 Подумайте, в каком случае Форд-Беллман может работать быстрее чем Дейкстра. Каким образом можно оптимизировать эту ситуацию?

Форд-Беллман может работать быстрее Дейкстры, если нам нужно найти кратчайшие пути из одной вершины до всех остальных, и в графе мало ребер. Это бывает в сильно разреженных графах (такой граф, в котором число рёбер сильно меньше максимально возможного). Тогда перебор ребёр по одному в алгоритме Форда-Беллмана будет осуществляться быстрее, чем в алгоритме Дейкстры поиск расстояний для всех вершин (то есть Дейкстра выгоднее на больших графах, а на разреженных, а ещё лучше - с фиксированным количеством вершин и малым количеством ребёр, Форд-Беллман будет работать быстрее).

Как можно попробовать оптимизировать: если после какой-то итерации в Форде-Беллмане ни одно расстояние не изменилось, то можно остановить программу, так как дальше ничего меняться тоже не будет. Также можно попробовать с обоими алгоритмами использовать списки смежности, в Форде-Беллмане - сортировать граф по возрастанию весов, в Дейкстре - останавливать алгоритм после рассчёта и извлечения данных для нужной нам вершины.

10 В каких ситуациях вы предпочтете рекурсивную реализацию DFS итеративной и наоборот? Эффективно ли применение списка в качестве стэка в итеративной версии? Что насчет очереди в случае BFS?

Рекурсивный DFS проще написать и читать, если глубина обхода небольшая. Но при большой глубине стек вызовов переполнится скорее всего. Итеративный DFS в случае большой глубины лучше использовать, но там стеком занимаемся мы сами.Со списком можно использовать append и pop, что удобно будет. Для BFS очередь удобно использовать, потому что она как раз использует порядок где первый пришедший первым и уходит, что подходит как раз для алгоритма BFS.