1) Что такое граф?

Граф это структура данных, элементы которой называются вершинами и некоторые(или все) пары вершин соединены ребрами.

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

В случае неориентированного графа максимальное число ребер = N*(N-1)/2, N - количество вершин, так как каждая вершина N может быть соединена максимум с N-1 вершиной (граф простой), при этом каждую связь-ребро мы посчитали дважды, поэтому делим на два. В ориентированном графе меняется последний пункт: каждая вершина может быть сединена как бы двумя ребрами разной ориентации, поэтому макс. число M=N*(N-1).

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

Транспонируем матрицу и смотрим осталась ли она такой же. Если нет, то граф ориентированный. Если да, то точно сказать нельзя.

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

В списке смежности будет пара объектов вместо одного, указывающего лишь на наличие ребра. В списке ребер будет тройка объектов: 2 вершины и вес.

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

Компонентой связности можно назвать максимальное множество вершин между которыми существует путь(нет изолированных точек), то есть до каждой вершины мы можем дойти по ребрам, считая граф неориентированным. Так как изолированная точка - это компонента связности, то максимальное число таких компонент=число вершин, а минимальное соотвественно 1(если считаем что пустых графов нет).

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

Один из пунктов алгоритма заключается в том что мы выбираем наименьшую вершину на каждой итерации и считаем ее значение перманентным с этих пор так как никак лучше ее обновить не получится. Отрицательные ребра ломают эту логику. Можно сначала вытащить наименьшую по величине вершину, а далее релаксировать ребра от нее и засовывать ее соседей в очередь. Далее делаем то же самое со следующей наименьшей вершиной пока очередь не станет пустой. Но алгоритм сломается если будет отрицательный цикл, оптимизировать под него наверное невозможно. Используем другие алгоритмы

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

N раз показывает существует ли отрицательный цикл в графе: за N-1 раз прохода алгоритма все возможные релаксации пройдут и каждая вершина получит налучший возможный

вес по тому как до нее можно найти в предположении что отрицательных циклов нет. И если в N раз какое-либо значение вершины поменяется=>есть отрицательный цикл, который заклеймил нашу вершину и сделал в целом задачу о нахождении кратчайшего пути неккоректной, так как при следующих проходах алгоритма значение этой вершины(и других 'зараженных') будет убывать до бесконечности(не достигая ее).

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

Успел написать только то, что у Форда Беллмана в целом асимптотика хуже чем у Дейкстры, но если учесть константы - простота алгоритма Форда Беллмана(3 вложенных цикла по сравнению с Дейкстрой) может затащить на разреженных графах.

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?

Рекурсия очень долго работает, когда стек вызовов переполняется, итеративный вариант в этом случае работает эффективнее и надежнее. При этом рекурсия пишется проще. Надо искать баланс в зависимости от задачи. Насчет списка в качестве стека. В целом использовать его эффективно так как есть команды append и pop как раз удаляющие и добавляющие элементы в конце списка(что и требуется в стеке). Так же список прост и интуитивно понятен в работе. Однако эффективность пропадает, когда необходимо добавлять или удалять не в конце. **А теперь вы наконец-то можете отдохнуть и насладиться прекрасным Ревашолем**