теориz графов

1. Что такое граф?

Граф – это структура данных, состоящая из множества вершин (точки) – V; и ребер (линии, соединяющие вершины) – E.

2. Максимальное количество ребер в простом графе:

Простой граф (без кратных ребер и петель).

Неориентированный граф:

$$\frac{n \cdot (n-1)}{2}$$

где n – количество вершин. Каждая из n вершин может быть соединена с $(n-1)$ вершинами, и так как каждое ребро считается дважды, делим на 2.

Ориентированный граф:

$$n \cdot (n-1)$$

Ребер становится в 2 раза больше, так как появляется направление.

3. Как по заданной матрице смежности быстро проверить граф на ориентированность?

Ориентированный граф – граф, ребра которого имеют направление (аналогия с односторонним движением).

Матрица смежности – матрица размерами $n \times n$, где n – количество вершин графа. Элемент матрицы $[i][j]$ равен 1, если есть ребро из i в j, и 0, если нет.

Таким образом, граф неориентированный, если матрица смежности симметрическая. Т.к. если на $[i][j]$ есть ребро, то и на $[j][i]$ оно есть в случае неориентированного графа. И это выполняется для всех элементов матрицы.

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

Список ребер:

Если в обычном графе у нас был простой список пар вершин (u, v) – ребро, то во взвешенном графе у нас появится третий компонент – вес: (u, v, price).

Обычный список ребер:

$$[(1, 2), (2, 3), (1, 3)]$$

Во взвешенном графе:

$$[(1, 2, 3), (1, 3, 5)]$$

Из вершины 1 можно попасть в вершину 2 по цене 3; и в вершину 3 по цене 5.

Список смежности:

Обычный список смежности:

$$\{1 : [3, 2], 2 : [1, 3], 3 : [1, 2]\}$$

Во взвешенном графе:

$$\{1 : [(3, 5), (2, 3)], \dots\}$$

Из вершины 1 можно попасть в вершину 2 по цене 3; и в вершину 3 по цене 5.

5. Что такое компонента связности графа? Каким может быть максимальное и минимальное количество компонент в одном графе?

Компонента связности графа – максимальный по размеру связный подграф графа. (т.е. если взять еще вершину, то подграф не будет связным).

Связный граф – граф, в котором возможно добраться в любую вершину из любой вершины по какому-либо пути.

Если n вершин:

- Минимальное количество компонент $= 1$, когда граф связный.
- Максимальное количество компонент $= n$, когда граф состоит из точек (вершин), т.е. нет ребер, только вершины.

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

Конечно, да, но не очень эффективно. В BFS мы идем по уровням с некоторой начальной вершины. Если в процессе находится вершина, которую мы уже посещали, то цикл найден.

DFS лучше в поиске циклов, так как при обходе графа мы спускаемся вглубь и сразу же находим зацикливания. Таким образом, DFS быстрее находит обратные ребра и занимает относительно меньше памяти.

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

Этот алгоритм является «жадным». Т.е. как только находится кратчайший путь до вершины, считается, что это лучший вариант. Алгоритм ищет ближайшую непосещённую вершину на всех шагах и считает его лучшим. Так что если в графе появятся отрицательные ребра, то принцип может нарушиться. Так как более длинный маршрут из отрицательных ребер (А-Б-С-Д-А) может оказаться более выгодным, так как снизит стоимость.

Скорее всего, его нельзя заставить работать с отрицательными весами, потому что у него есть принципы. Если при обработке вершины обнаруживается, что расстояние до неё может быть улучшено через отрицательное ребро, то вершину можно добавить обратно в очередь. Но это уже как будто не Дейкстра.

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

$N - 1$ раз хватает, чтобы найти циклы, НО если запустить алгоритм еще раз, то алгоритм попытается еще раз улучшить свои оценки. Может выясниться, что существует отрицательный ранее не найденный цикл. Так как если на N-м шаге расстояние может быть улучшено, это значит, что существует отрицательный цикл. Алгоритм не завершится, а числа все будут убывать... А это как раз ответ на второй вопрос. Не существует нижней границы стоимости из-за отрицательного цикла (если он достижим из начальной точки).

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

Конечно же, если граф с отрицательными ребрами. Дейкстра тут вообще не применим.

Если ребер мало, т.е. граф разреженный (E почти столько же, как V). Будет $O(V^2)$.

Дейкстра: $O((V + E) \log V)$.

Беллман может завершиться раньше, чем за $V - 1$ итераций.

Если на какой-то итерации расстояния не меняются, можно завершить алгоритм.

Создаем очередь с обновленными расстояниями.

Проверять только ребра с возможностью улучшения.

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

DFS:

Рекурсия:

- Плюсы:
  - Относительно простой код и короткий.
  - Когда у нас небольшой граф.
  - Нужно найти обратный путь.
- Минусы:
  - Большие графы (переполнение стека).
  - Проигрываем в памяти для больших графов.

Итерация:

- Плюсы:
  - Выигрыш в памяти.
  - Приоритетная очередь.
  - Большие графы.

- Минусы:
  - Сложно.
  - Управление стеком.

Использование списка в качестве стека эффективно: $O(1)$ добавлять и удалять.

BFS:

Рекурсия – не подходит.

Итерация:

- Плюсы:
  - Эффективно.
  - Использование очереди эффективно.
  - Эффективно по памяти.
- Минусы:
  - Управление очередью.

Использование списка в качестве стека неэффективно: $O(n)$ удалять. Лучше использовать двустороннюю очередь.