**Ответы на вопросы про статью**

**0. Да**

1. Статья про алг. Флойда-Уоршелла и про то, как сделать его эффективнее. Улучшение заключается в ускорении алгоритма при работе с большими разреженными графами.

2. Классический алгоритм всегда работает за O(n^3), где n - количество вершин в графе. Если Граф разреженный (количество ребер примерно равно количеству вершин), то алгоритм проделывает много бесполезных сравнений, тратя время впустую, от чего можно избавиться улучшенной версией из этой статьи.

“*even for sparse graphs, i.e., graphs with vertices having very few outgoing and incoming edges, the second and the third loops will explore the whole matrix containing entries mostly with ∞ values. Since, the condition of the if statement will not be satisfied when either* [𝑖, 𝑘] or 𝐴[𝑘, 𝑗] *is ∞, most of these explorations will be a useless*”

3. Структура статьи удобная, автор часто приводит примеры и подробные описания работы всяких вспомогательных конструкций, описывает достаточно просто и понятно. По лексике было легко, хотя все еще читать статью на другом языке и вникать в нее посложнее, чем на родном. P.S. Про простоту языка и конструкций мнение очень субъективное, поскольку у меня достаточно хороший уровень английского, но другого мнения я не придумаю.

4. Опечаток не нашел, ошибок вроде нет, но я не всматривался в описание сложности других алгоритмов при сравнении. Может там закралась какая-то ошибка, но внешне выглядит правильно.

Имеется в виду вот это обоснование сложности алгоритма Джонсона:

*Thus, its complexity* is (𝑁𝑀 + 𝑁(𝑁𝑙𝑔𝑁 + 𝑀𝑙𝑔𝑁)) *(plus the additional small reweighting costs of the vertices with* 𝑁*).*

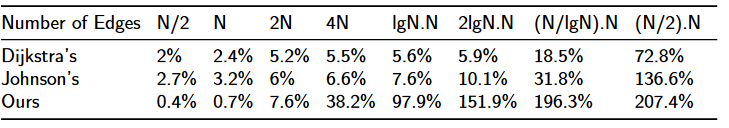
5. В статье есть: оригинальный алгоритм, первая версия алгоритма (без ручного выбора следующего индекса для проверки), и вставка кода с реализацией этого самого выбора с указанием куда его вставлять.

Вроде бы чтобы написать алгоритм этого хватает, в крайнем случае там есть еще код на C, который при желании можно прочитать, а если есть опыт C, то прочитать еще и довольно просто, он хорошо написан (хорошо для C, относительно кодов на питоне он выглядит крайне страшно, но так выглядят все коды на C/C++, по опыту они выглядят страшнее, чем читаются)

6. Прочитал я статью, мне не очень понравилось в два раза большее время работы на очень плотных графах.

Можно ли как-то перед запуском алгоритма проверять граф на плотность и понимать стоит ли использовать такой алгоритм, или нет?

(Знаю, что вопрос бесполезный, так как при возникновении в задаче опасности больших графов стоит просто изначально задуматься о написании более технически сложного, но быстрого алгоритма, но все же интересно)



Чем правее, тем хуже