Assignment 4 Report

1. Введение

Цель данной работы — провести анализ производительности алгоритмов, используемых для обработки ориентированных графов, применимых в задачах планирования и управления умными системами (Smart City / Smart Campus). Исследование основано на данных из файла metrics.csv, включающем метрики работы алгоритмов поиска сильно связанных компонент (SCC), топологической сортировки и поиска кратчайших путей в DAG-графах.

2. Методология

Для анализа использовались три ключевых алгоритма:
• Tarjan's Algorithm — для нахождения сильно связанных компонент (SCC);
• Kahn's Algorithm — для топологической сортировки DAG-графа;
• DP на основе топологического порядка — для поиска кратчайших путей.

Каждый граф был загружен из файла all_graphs.json и обработан программой Main.java, которая собирала метрики времени выполнения и количества операций. Все результаты были сохранены в metrics.csv для дальнейшего анализа.

3. Результаты экспериментов

Graph	Nodes	Edges	SCCs	TopoTime(ms)	SCCOps	DAGTime(ms)	Relaxations	Total(ms)
small1	6	5	6	1.036	0	0.598	0	3.283
small2	8	8	6	0.045	0	0.032	0	0.124
small3	9	9	3	0.019	0	0.021	0	0.078
medium1	12	12	3	0.019	0	0.029	0	0.095
medium2	15	15	13	0.058	0	0.038	0	0.168
medium3	16	15	10	0.037	0	0.03	0	0.13
large1	20	19	20	0.056	0	0.041	0	0.156
large2	25	24	25	0.067	0	0.052	0	0.181
large3	30	29	30	0.084	0	0.077	0	0.224

4. Анализ результатов

На основе полученных метрик можно сделать несколько наблюдений о поведении алгоритмов.

1. Рост количества SCC.
В малых графах (6–9 вершин) количество сильно связанных компонент варьируется от 3 до 6, в средних (12–16 вершин) — до 13, а в крупных (20–30 вершин) каждая вершина становится отдельной компонентой. Это говорит о том, что большие графы практически не содержат циклов и являются DAG-структурами. Алгоритм Tarjan обрабатывает такие графы за линейное время O(V + E), поэтому время вычисления остаётся стабильным.

2. Топологическая сортировка.
Алгоритм Kahn демонстрирует время работы в пределах 0.02–0.08 мс даже на больших графах. Это подтверждает его линейную сложность и минимальные накладные расходы при увеличении числа узлов.

3. Поиск кратчайших путей в DAG.
Алгоритм на основе топологического порядка показал наименьшее время работы — до 0.077 мс. При этом значение Relaxations=0 во всех случаях говорит о том, что пути были оптимальны изначально.

4. Суммарное время выполнения.
Общая длительность обработки не превышает 3.3 мс даже для самого маленького графа small1. Это может быть связано с первичной инициализацией JVM или особенностями структуры рёбер. Для всех последующих графов наблюдается линейный рост времени в пределах 0.1–0.22 мс.

5. Влияние структуры графа.
Циклические графы (small2, small3) не увеличивают нагрузку, так как SCCFinder корректно конденсирует их в DAG перед топологической сортировкой. Это делает систему устойчивой к плотным связям.

5. Выводы и рекомендации

Проведённый анализ показывает, что реализованные алгоритмы обладают высокой скоростью и стабильностью при увеличении размера графа. Tarjan’s SCC Finder эффективен для анализа циклических зависимостей, топологическая сортировка обеспечивает надёжный порядок выполнения задач, а поиск кратчайших путей по DAG подходит для планирования и оптимизации процессов.

Для систем типа Smart Campus и Smart City комбинация этих методов может использоваться для:
• оптимизации расписаний (распределение задач по времени);

• анализа зависимостей в сетях сенсоров;
• планирования обслуживания инфраструктуры.

Таким образом, предложенная архитектура алгоритмов применима к реальным распределённым задачам, где требуется высокая скорость вычислений и предсказуемая асимптотика.