1. Алгоритм AD

Алгоритм AD реализован на языке C++. На вход подается бинарная матрица и минимальная поддержка. В текущей реализации используется не чисто бинарная матрица (как у П. А. Прокофьева в RUNC-M), а матрица из байтов. Пример работы алгоритма:

1.1. Пример 1

Вход:

$$\begin{pmatrix} 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 \end{pmatrix} \tag{1}$$

Минимальная поддержка s = 3.

Выход:

Всего наборов: 11

1
1 3
1 3 4
1 4
2 3
2 3 4
2 4
3
3 4
4

Максимальных наборов: 2

1 3 4
2 3 4

1.2. Пример 2

Вход:

$$\begin{pmatrix} 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \tag{2}$$

Минимальная поддержка s = 3.

Выход:

Всего наборов: 9

0

0 5

3

3 4

3 4 5

3 5

4

4 5

5

Максимальных наборов: 2

0 5

3 4 5

2. Скорость счёта

Множество всех s-совместимых наборов строилось строго по алгоритму. Множество максимальных наборов строилось тогда, когда алгоритм не мог найти следующий совместимый столбец. В этом случае максимальность проверялась по базе данных (модификация была предложена Н. А. Драгуновым в курсовой работе). Результаты счёта приведены в таблице:

	n = 20	**n = 30**
m = 100	747, 72 мс	3400, 489 мс
m = 1000	1136, 764 мс	4042, 3649 мс

Здесь m - число транзакций (строк), n - число атрибутов (столбцов). Минимальная поддержка s = 0.1. В первой строке указано число всех максимальных частых наборов, во второй строке - время работы алгоритма. Сравнивая с результатами Н. А. Драгунова можно сделать вывод, что алгоритм AD работает быстрее DepthProject без модификации, но медленнее DepthProject с модификацией. Однако текущая реализация алгоритма AD далека от оптимальной, и есть надежда, что результаты можно улучшить.