Зайцев Артем, 3822Б1ПР1

В Таблице 1 приведено сравнение алгоритмов умножения для структур хранения FlatMatrix и CRSMatrix при одинаковых входных данных (Flat_CRS_comparison()). Size – порядок матрицы, Frequency – отношение числа ненулевых элементов к числу Size * Size, т. е. процент ненулевых элементов матрицы. Здесь и далее время указывается в миллисекундах.

	Size					
	100	**200**	**300**	**400**		
FlatMatrix	18.1361	126.855	449.756	1087.44	**5%**	**Frequency**
	23.8592	128.083	483.652	1177.24	**10%**	
	15.6629	122.445	549.486	1073.55	**50%**	
	20.1755	163.013	674.595	1111.71	**100%**	
CRSMatrix	3.8881	27.0121	76.0727	209.919	**5%**	**Frequency**
	9.9724	51.9625	152.917	339.224	**10%**	
	37.2674	207.071	661.519	1619.48	**50%**	
	47.7975	299.825	1065.57	1912.99	**100%**	

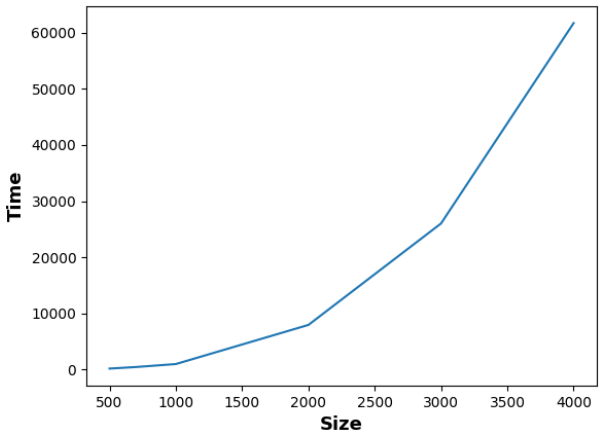
В Таблице 2 приведено среднее время работы алгоритма умножения для структуры хранения CRSMatrix по результатам 10 проходов алгоритма (test_average_time_CRS(10, {10, 100, 1000}, {1, 5, 10, 20, 50})).

		Size		
		10	**100**	**1000**
Frequency	**1%**	0.03781	1.26041	1024.08
	5%	0.04082	3.96043	3027.68
	10%	0.04982	6.95452	5347.13
	20%	0.05813	14.7771	10055.90
	50%	0.14883	31.6094	24133.82

В Таблице 3 приведено среднее время работы алгоритма умножения для структуры хранинния CRSMatrix для Frequency = 1 (test_average_time_CRS(2, {500, 700, 1000, 2000, 3000, 4000}, {1}))

Size	500	700	1000	2000	3000	4000
Time, ms	167.645	446.255	974.794	7945.35	26022.4	61724.3

Полученный график похож на параболу, тогда предположим, что сложность алгоритма умножения в CRSMatrix есть $O(n^2)$. Построим еще один график зависимости, где элементу Size[i] поставим в соответсвие элемент Time[i] / (Size[i] ** 2).



Видим линейную функцию малого роста, при изменении аргумента на Δsize = 3500, значение функции изменяется всего лишь на Δtime = - 0.00175, что означает, что предположение о сложности $O(n^2)$ условно справедливо при плотности матрицы 1%.