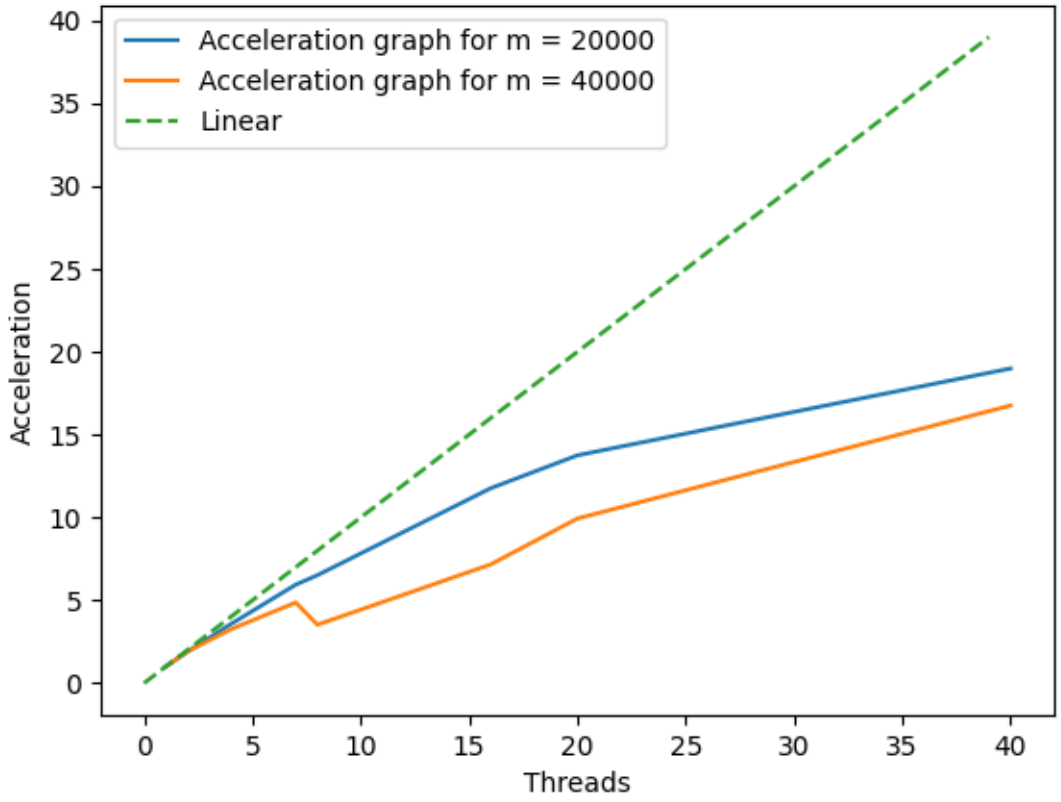
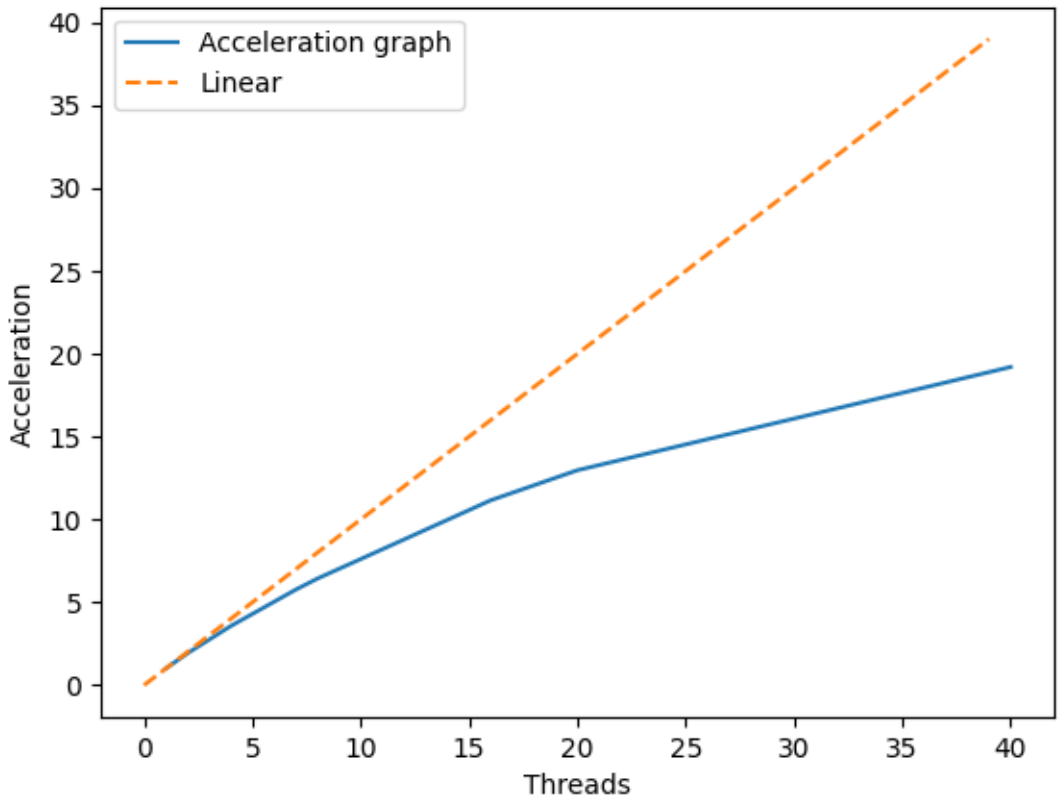
Задание 1

M = N	Количество потоков														
	2			4		7		8		16		20		40	
	T1	T2	S2	T4	S4	T7	S7	T8	S8	T16	S16	T20	S20	T40	S40
20000	1.259	0.662	1.901	0.353	3.561	0.212	5.945	0.193	6.515	0.107	11.765	0.092	13.752	0.066	18.997
40000	5.124	2.690	1.905	1.577	3.250	1.052	4.869	1.453	3.527	0.715	7.168	0.516	9.938	0.306	16.763



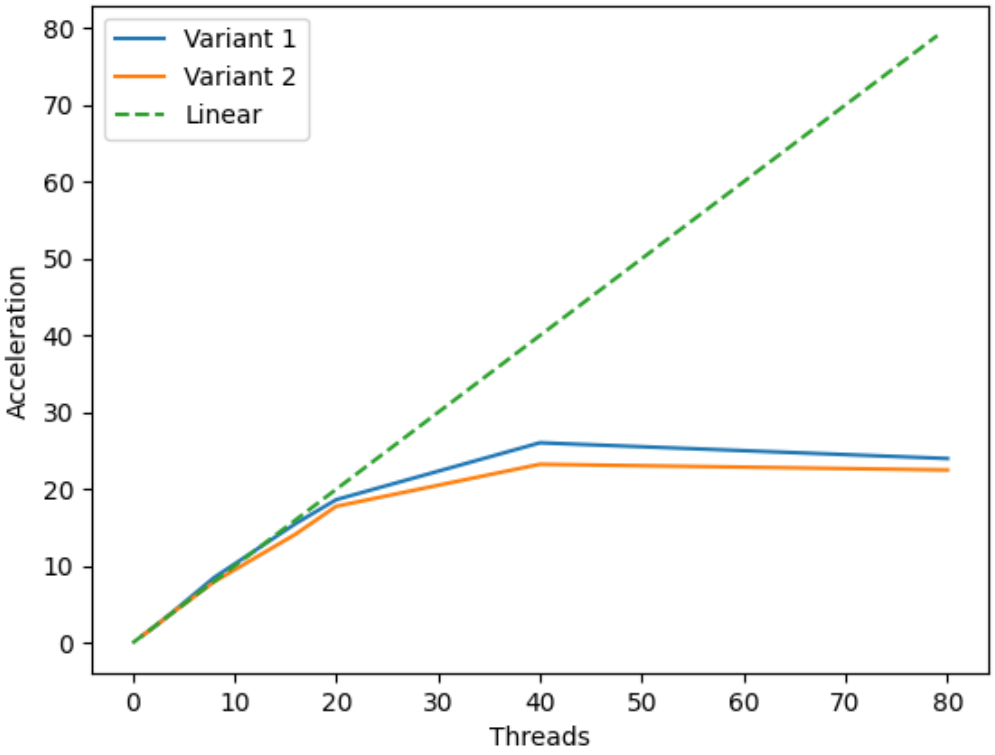
Вывод: по таблице и графику можно заметить, что после 16 потоков скорость выполнения программы увеличивается незначительно, а значит, смысла увеличивать количество потоков выше 40, а возможно и 20 почти нет. Также можно заметить, что программа, на которую подали больший объем данных, ускорилась меньше, чем та, на которую подали меньший. Однако, при увеличении количества потоков ситуация начала выправляться, из чего можно предположить, что при большем количестве потоков ускорение будет примерно одинаково.

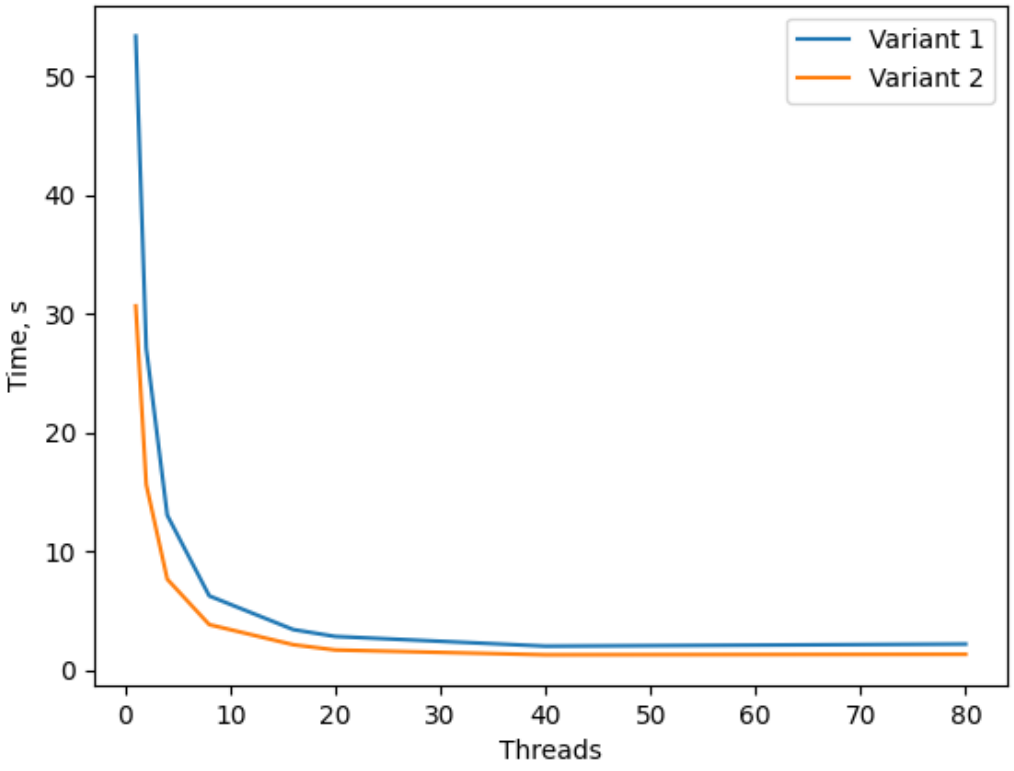
Задание 2:

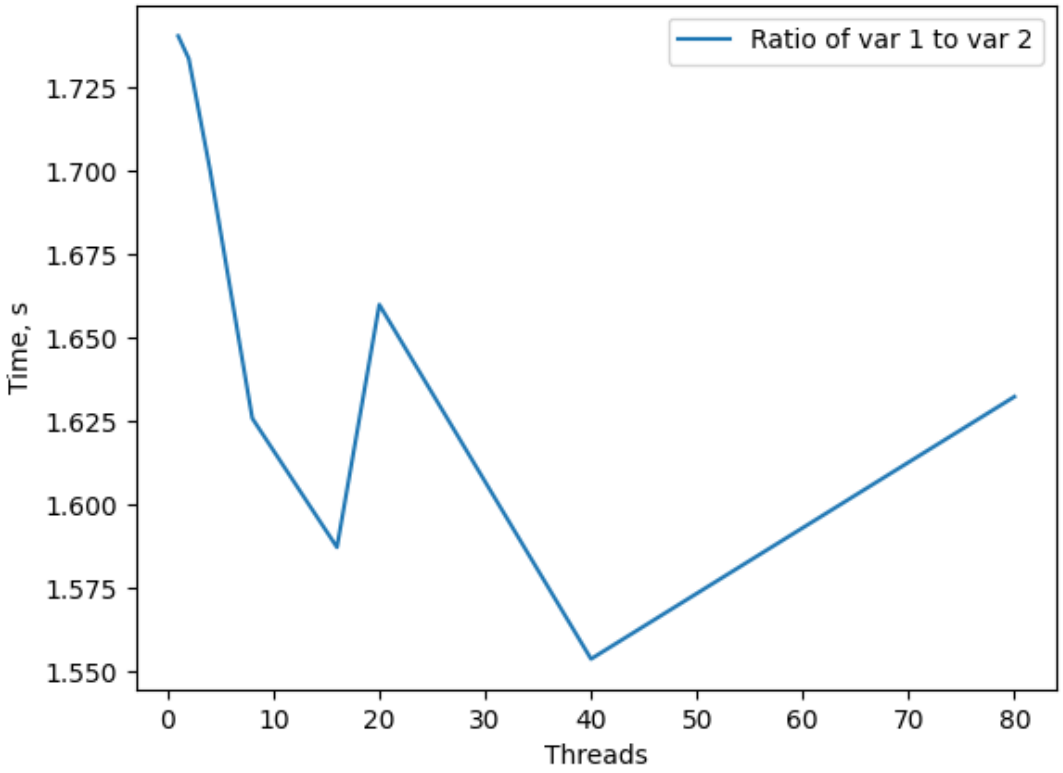


Вывод: по графику можно заметить, что ускорение работы программы при росте числа потоков стремительно падает, однако разница в ускорении между 20 и 40 потоками все еще значительна. Значит, есть смысл увеличивать количество потоков до 40, однако поднимать это значения еще выше смысла не много.

Задание 3:


Variant 1
Variant 2
Linear
Acceleration
Threads
0
10
20
30
40
50
60
70
80

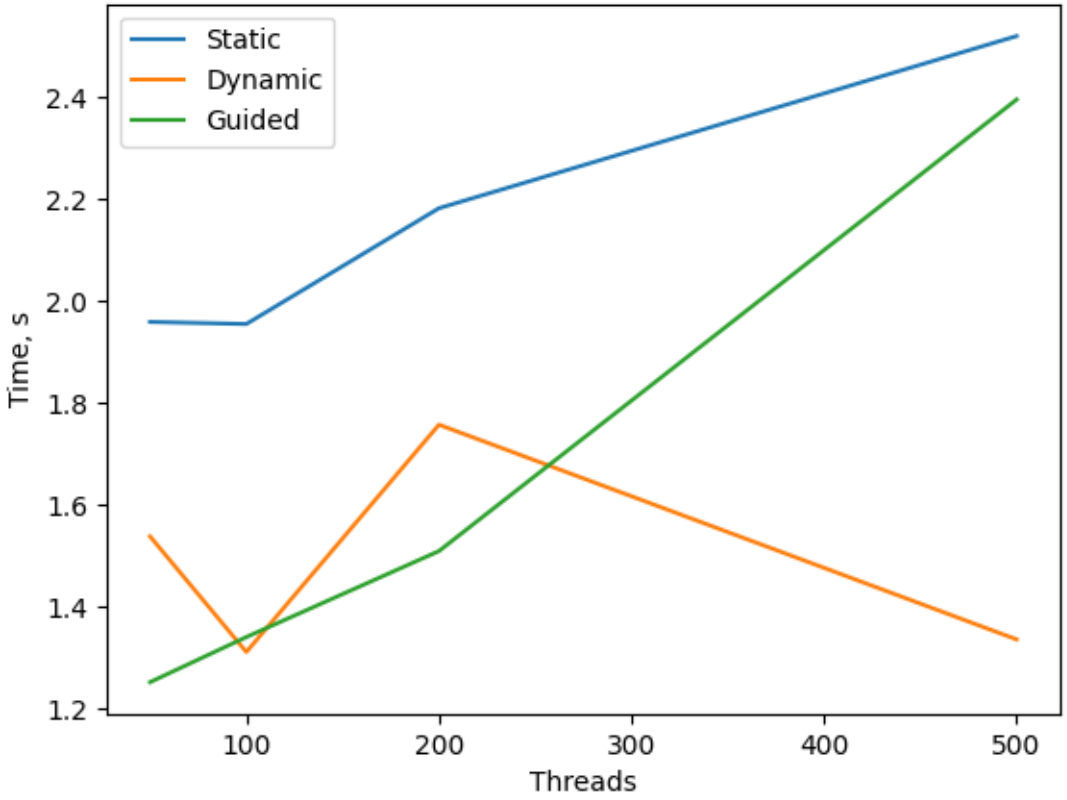

Variant 1
Variant 2
Time, s
Threads
0
10
20
30
40
50
60
70
80



Вывод: по графикам видно, что программа, реализованная первым способом, лучше ускоряется при увеличении числа потоков, однако в целом, она работает гораздо медленнее, чем второй вариант. Следовательно, эффективнее программа, реализованная вторым способом. Стоит отметить, что оба варианта программы при увеличении числа потоков до 80 работают медленнее, чем при 40 потоках, вероятно, это происходит из-за сложности распределения вектора по потокам.

Отдельно хочу отметить, что ранняя версия моей программы, которая работала не на безопасных векторах, а на приятных malloc, работала примерно в три раза быстрее. В качестве иллюстрации этого тезиса, привожу такой пример: моя прежняя программа за приблизительно то же время, которое работает второй вариант программы, обрабатывала в разы больший объем данных (сейчас m = 30000, тогда m = 50000).

Эксперимент с schedule



Вывод: эффективнее использовать schedule dynamic с k == 100.