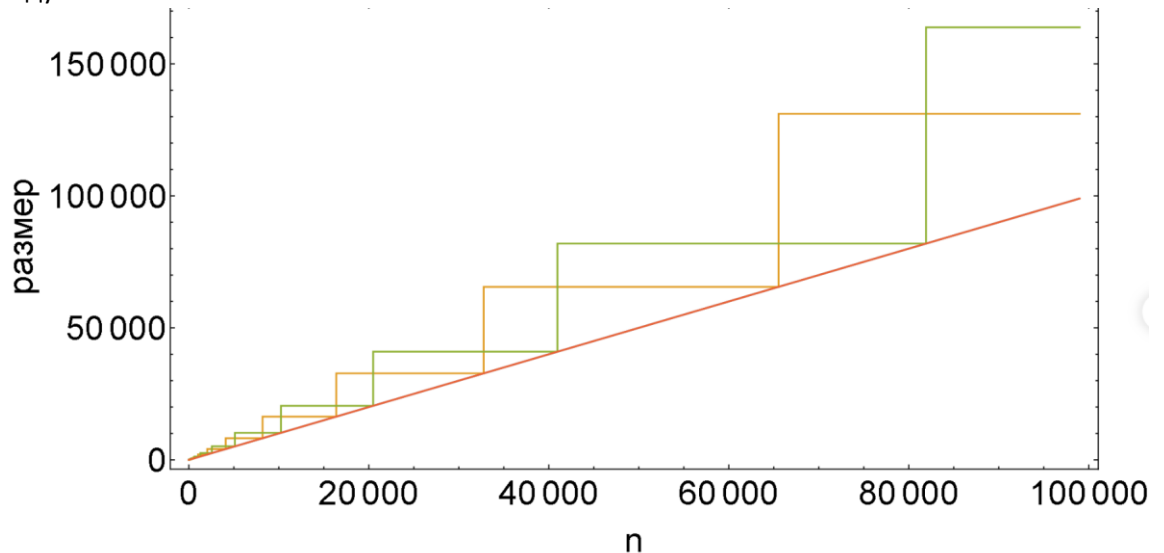
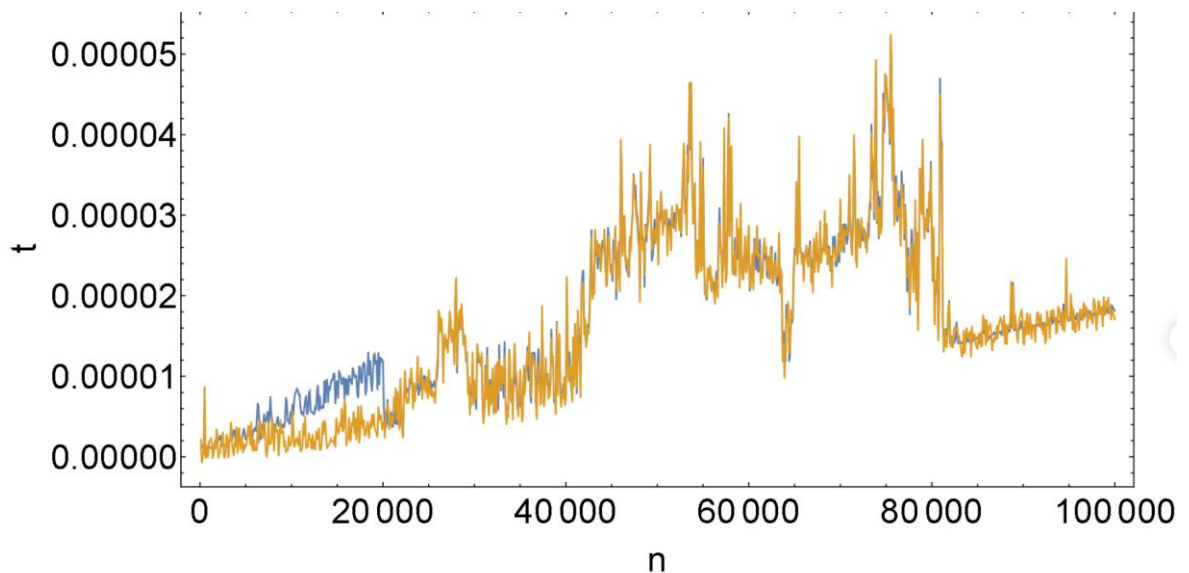


## ЛАБА ПО КОНТЕЙНЕРАМ STL

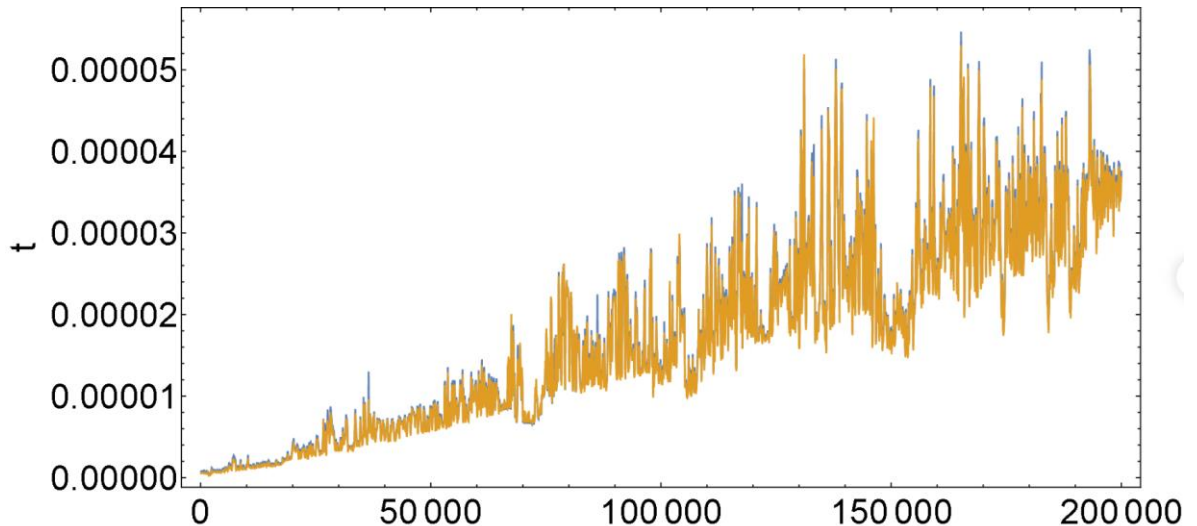
- 1) Нужно было вывести capacity и size в зависимости от количества элементов при последовательном push\_back. Микропояснение за реализацию: создаем 2 файла в которые будет записываться значение cap и size для сабвектора и вектора. Далее ставим границу на количество итераций n и просто с начального размера нашего вектора увеличиваем до конечного n. Все это время каждый момент записываем в соответствующие файлы значения cap и size для сабвектора и вектора. Получается что-то в духе



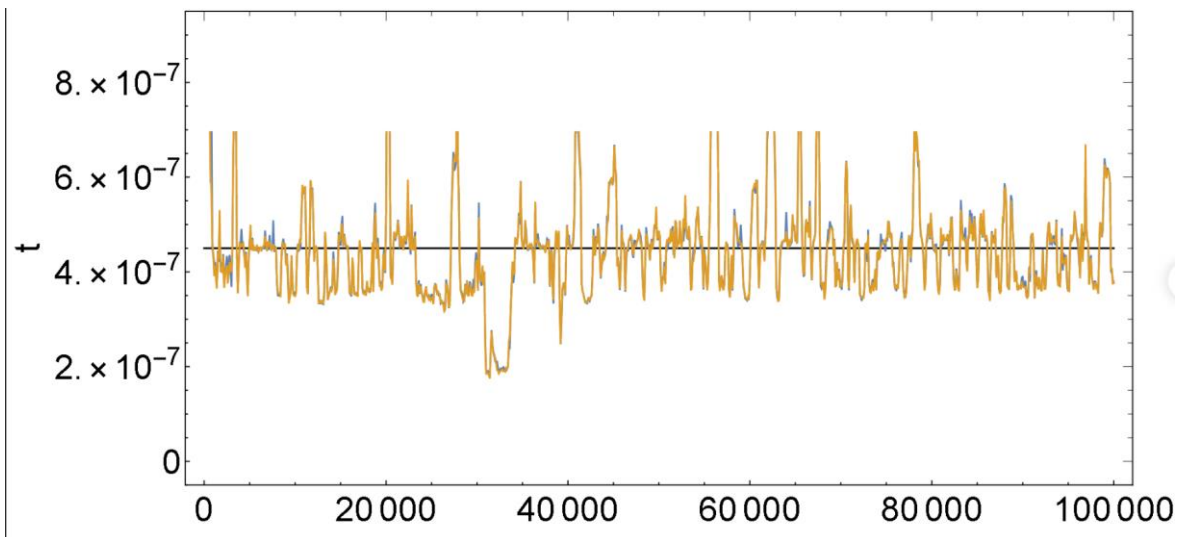
- 2) Далее надо было построить среднее время вставки элемента в произвольное место. Мы произвольно генерим вектор и сабвектор, генерим произвольный индекс, генерим произвольное значение и засекаем время на insert в это произвольное место этого произвольного значения для сабвектора и вектора. В конце концов нужно обязательно удалить какой-то любой элемент, чтобы не было  $n+=2$  за итерацию. Делаем для группы элементов (в среднем 100 или 500), считаем суммарное время, делим на количество раз.



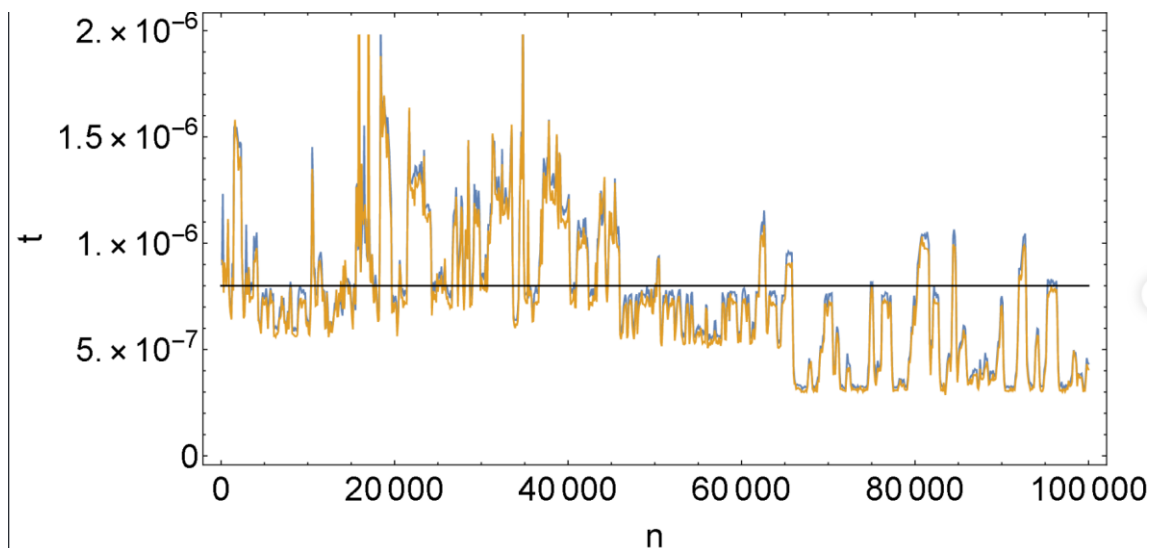
- 3) Далее надо было построить среднее время удаления элемента из произвольного места. Мы произвольно генерим вектор и сабвектор, генерим произвольный индекс и засекаем время на erase из этого произвольного места для сабвектора и вектора. В конце концов нужно обязательно добавить какой-то любой элемент, чтобы не было  $n+2$  за итерацию. Делаем для группы элементов (в среднем 100 или 500), считаем суммарное время, делим на количество раз.



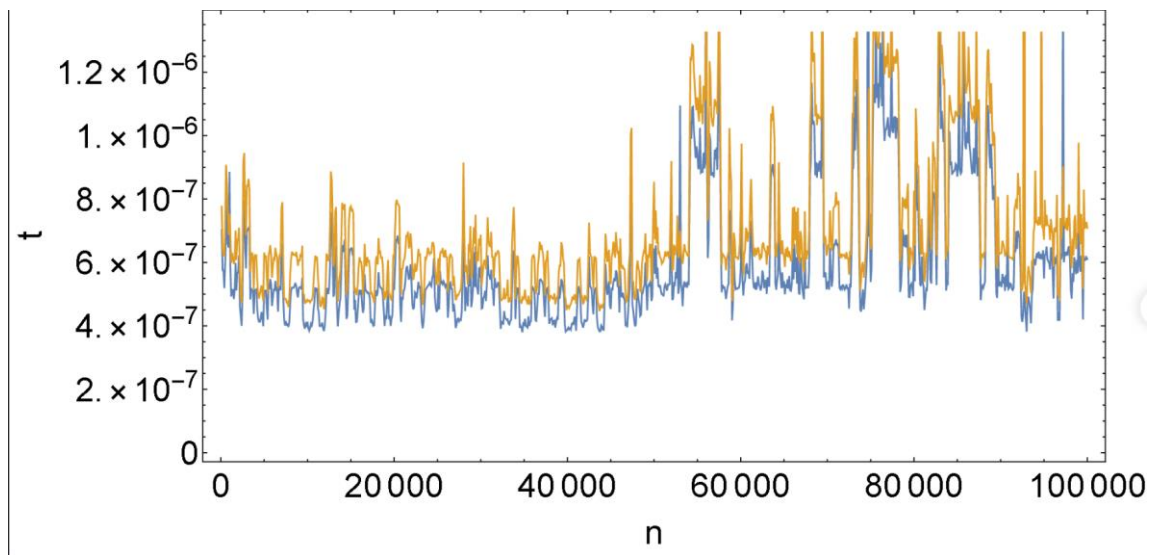
- 4) Далее надо было построить среднее время доступа к элементу из произвольного места. Мы произвольно генерим вектор и сабвектор, генерим произвольный индекс и засекаем время на ++ этого элемента для сабвектора и вектора. Делаем для группы элементов (в среднем 100 или 500), считаем суммарное время, делим на количество раз. Асимптотика  $O(1)$ .



- 5) Дальше определить среднее время добавления в начало односвязного списка. Заменяем вектор на fl, сабвектор на sfl и меняем в рандомном заполнении на заполнение фл и сфл (а то у меня ошибка там была и я сидела не понимала в чем прикол такой). В конце обязательно нужно удалить любой в принципе элемент, чтобы не было  $n=+2$ ; Дальше засекаем время добавления в начала у фл, у сфл, усредняем по 100 операций.



- 6) Дальше определить среднее время удаления из начала односвязного списка. Заменяем вектор на fl, сабвектор на sfl и меняем в рандомном заполнении на заполнение фл и сфл (а то у меня ошибка там была и я сидела не понимала в чем прикол такой). В конце обязательно нужно добавить любой в принципе элемент, чтобы не было  $n=+2$ . Дальше засекаем время удаления из начала у фл, у сфл, усредняем по 100 операций.



7) Дальше определить среднее время прохода по разным контейнерам. У меня в принципе написано для vector, forward\_list, map и set. К листам нужно общаться по-особому, т.к. у них нет понятия [], поэтому было написано через el. По логике тот же проход, только описан по-другому. В случае set'a нужно было делать обход по итератору и сначала не работала штука с \*it++, т.к. сначала добавляло 1, а потом переименовывалось. Ну и не считалось в конечном счете. В итоге просто добавляем значение каждого элемента set к сумме определенной. В итоге получилась вот такая штука:

