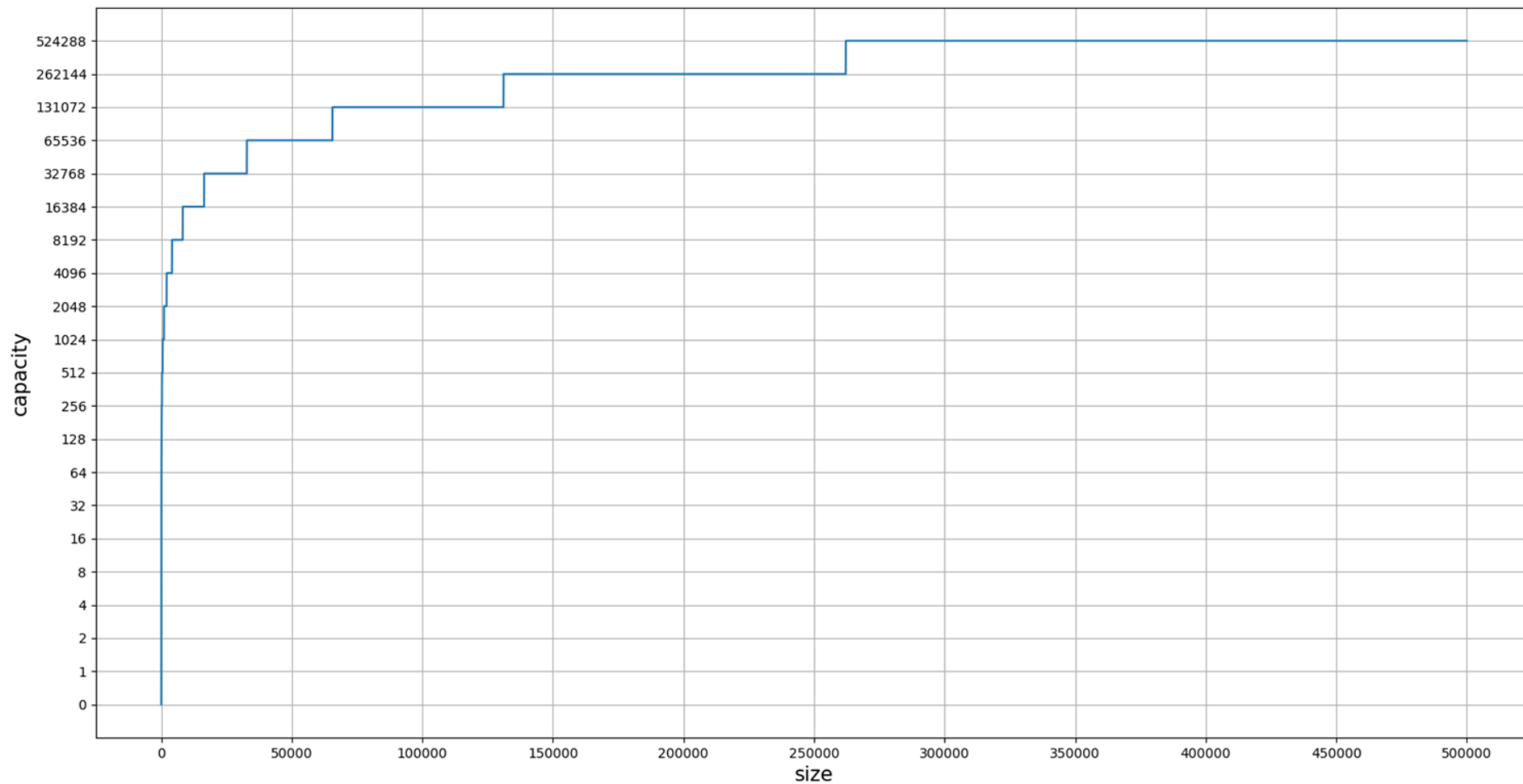


Лабораторная работа:

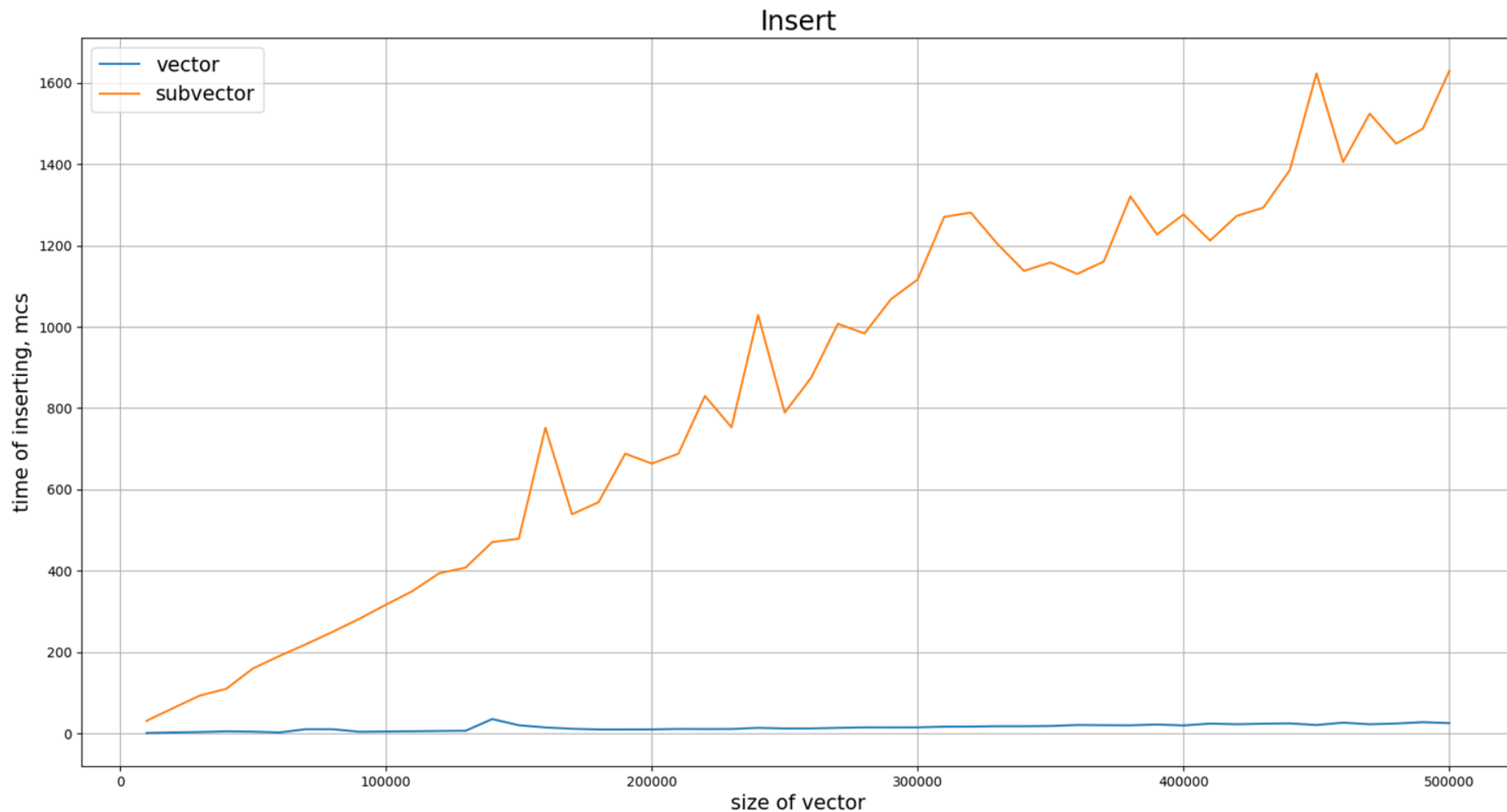
Сравнение контейнеров ручной работы и контейнеров из stl

Зависимость объема выделенной памяти от размера вектора при push_back

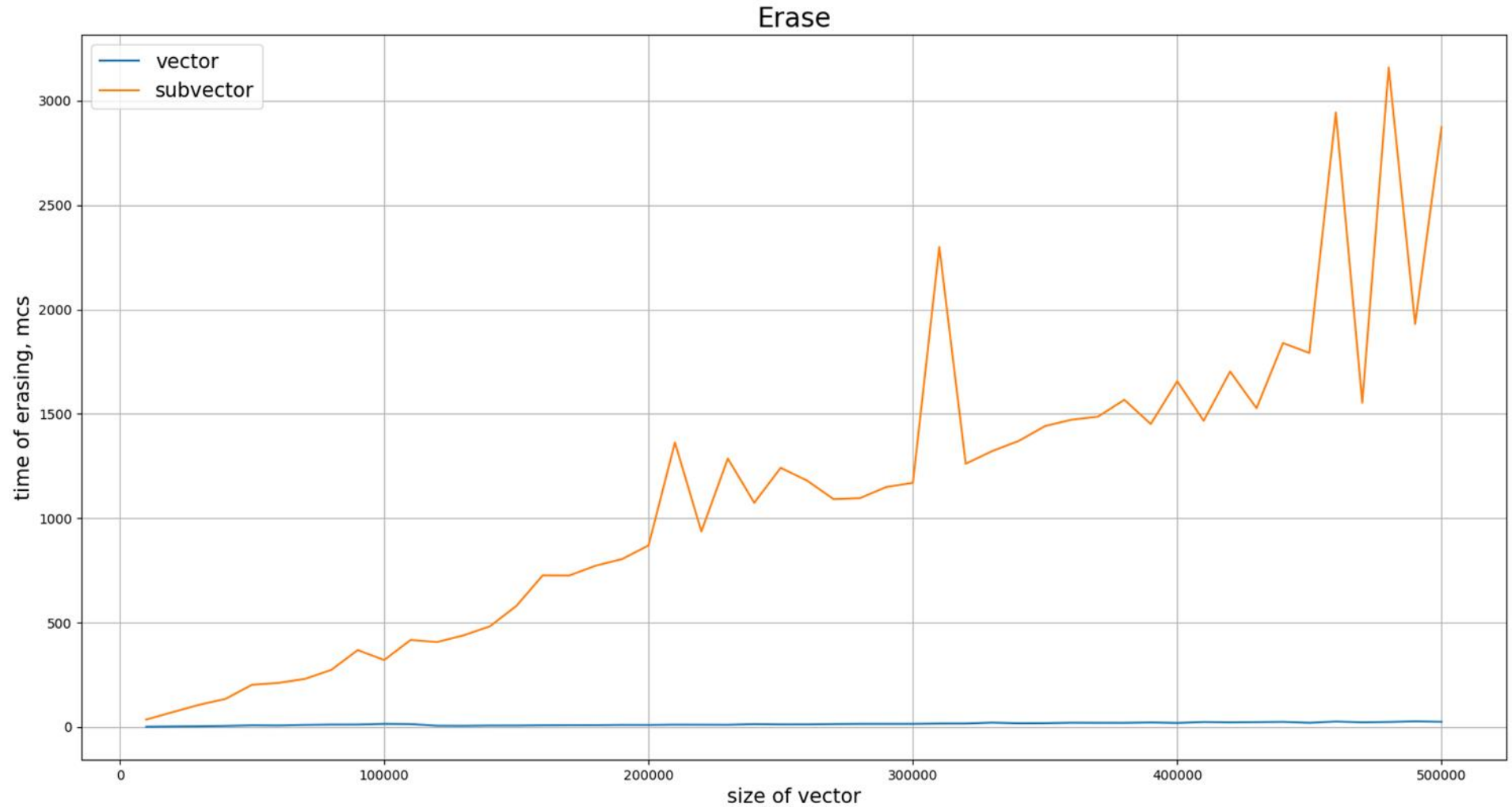


•Каждый раз, достигая степени двойки, вектор из stl выделяет в 2 раза больше уже используемой памяти в расчете на дальнейшее заполнение

Зависимость среднего времени вставки в произвольное место вектора от его размера



Зависимость среднего времени удаления элемента из произвольного места вектора от его размера

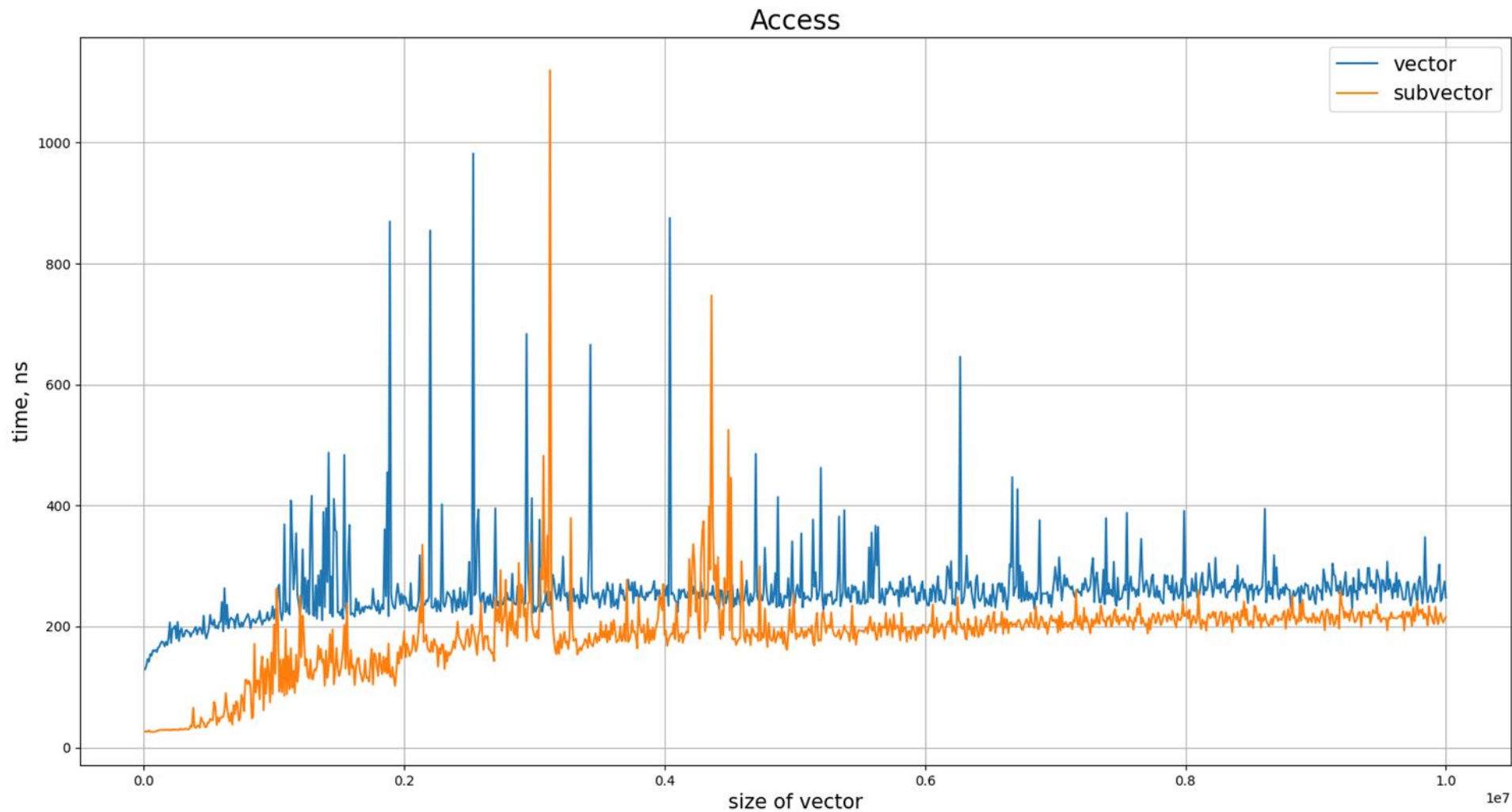


• Вектор ручной работы не оптимизирован, поэтому он создает новый вектор, копируя в него элементы старого, чтобы добавить/удалить один элемент. А вектор из `stl` увеличивает `capacity` с запасом, как видно из первого графика, и добивается константного времени работы.

• Асимптотика `subvector`: `erase` и `insert` – $O(n)$

• Асимптотика `vector`: `erase` и `insert` – $O(1)$

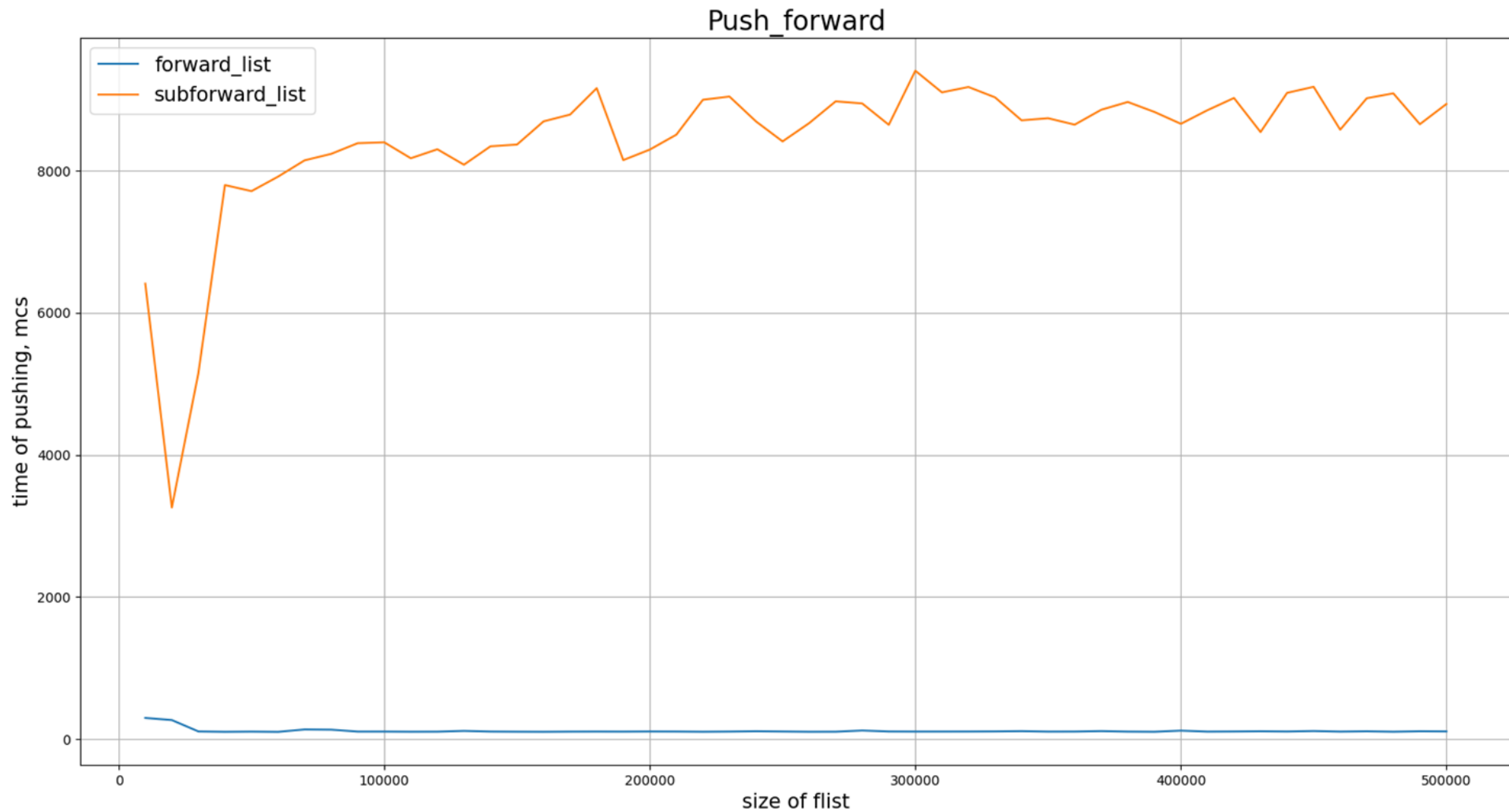
Зависимость среднего времени доступа к произвольному элементу вектора от его размера



- Т.к. все элементы в векторе лежат подряд, то доступ к ним возможен за константу, в отличие от, например, списка

- Асимптотика: $O(1)$

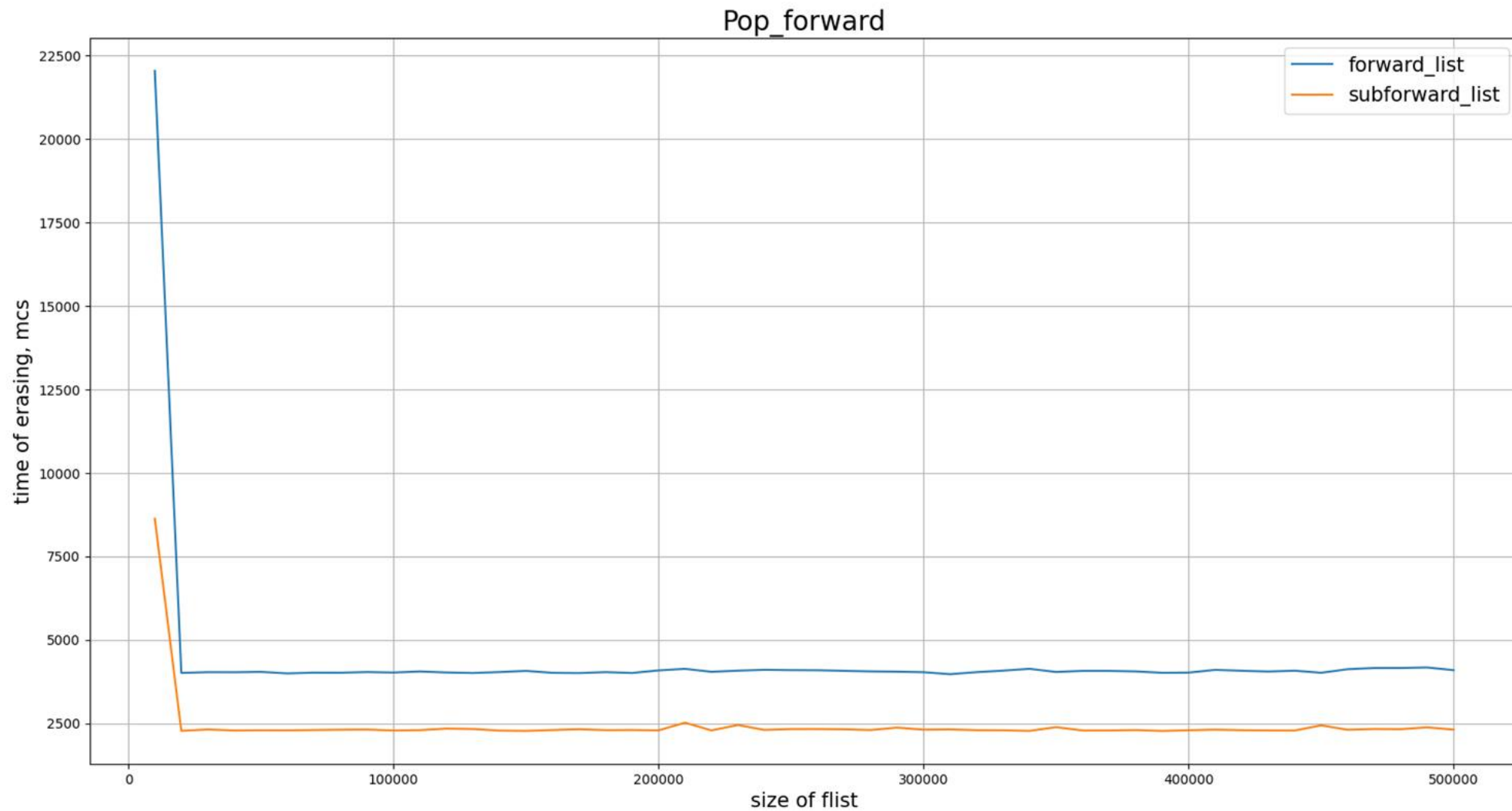
Зависимость времени добавления элемента в начало списка от его размера



- Список из `std` при добавлении нового элемента выделяет памяти с запасом, а `subforwardlist` выделяет для каждого элемента отдельно.

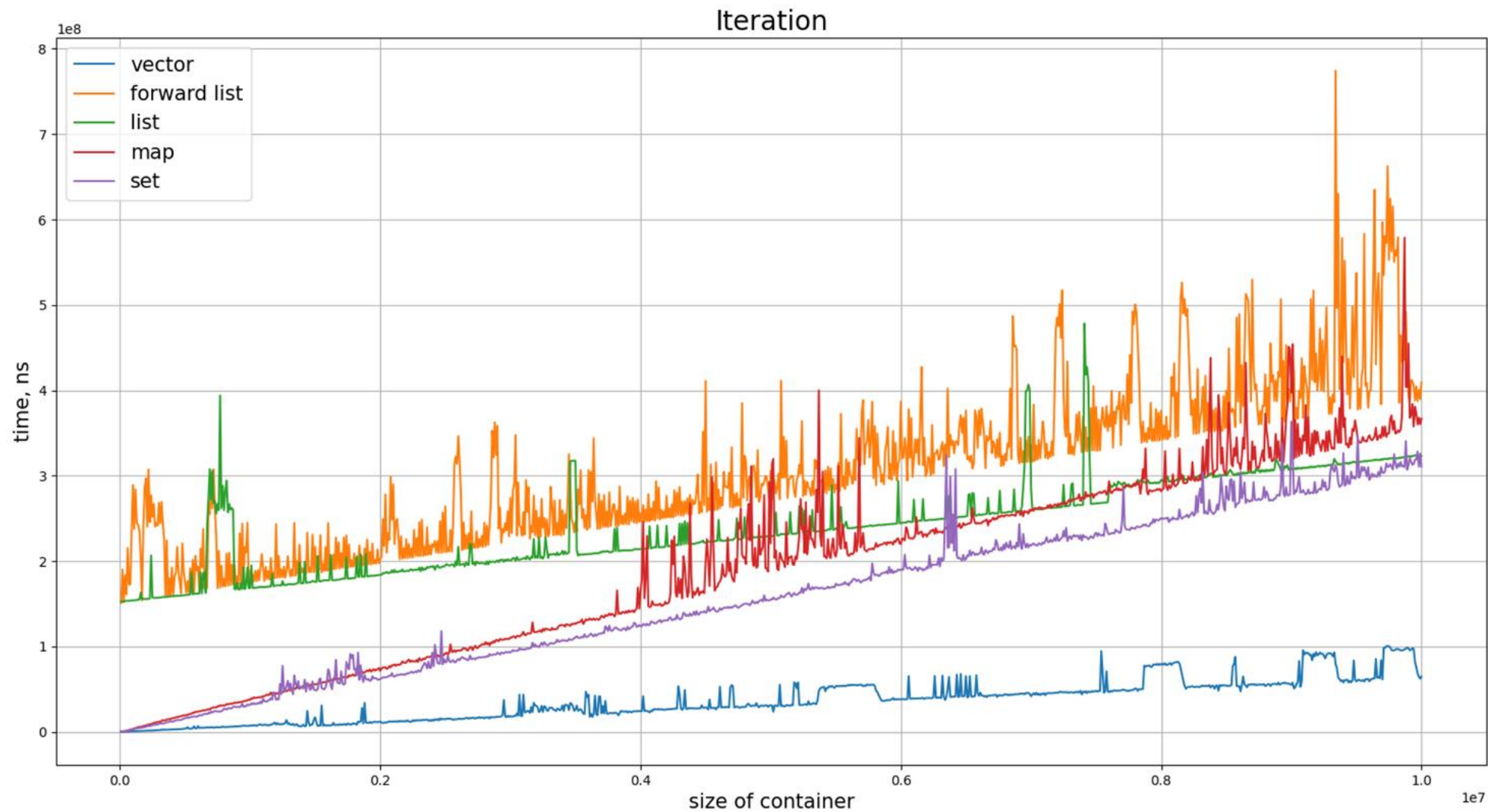
- Асимптотика: оба работают за $O(1)$, но к `subforwardlist` прибавляется еще константа выделения памяти

Зависимость времени удаления элемента из начала списка от его размера



•Асимптотика: работают за $O(1)$, но тонкости реализации неизвестны, поэтому непонятно, почему `list` из `stl` работает дольше

Зависимость среднего времени обхода контейнера от его размера



- Выполняем операцию присваивания значения элемента переменной, чтобы для всех контейнеров получить прибавление одинаковой константы и сравнить более честно
- Односвязный и двусвязные списки совершают обход одинаково долго, т.к. им приходится разыменовывать все указатели на элементы.
- Вектор работает быстро, потому что все его элементы лежат в куче подряд
- Асимптотика: $O(n)$