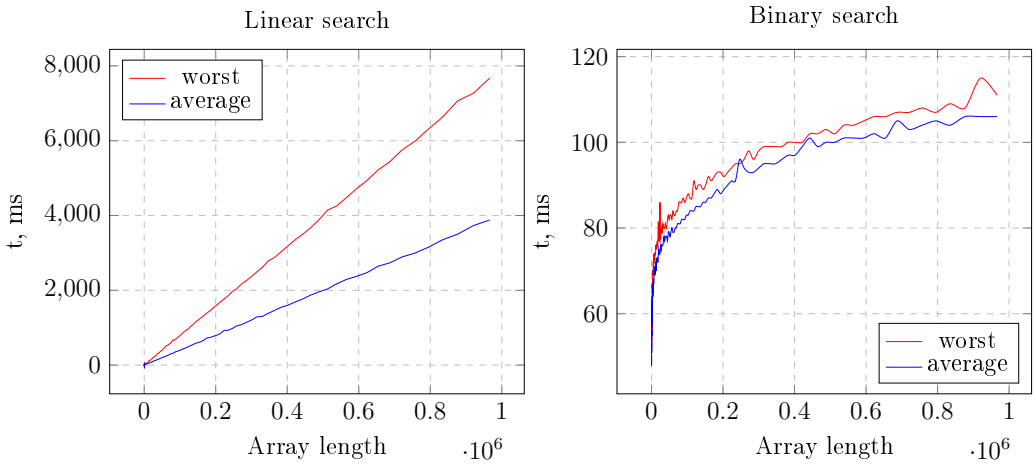
Асимптотическая сложность алгоритмов поиска

Для выполнения первого задания была написана программа search.cpp, в которой реализованы следующие методы:

1. linear_search – линейный поиск
2. binary_search – бинарный поиск
3. fill – заполнение массива строго возрастающими целыми положительными числами
4. generate_needed – генерация случайного числа (в массиве или нет, учитывая какой случай нужен)
5. print_arr – выводит весь массив, нужен для debug
6. timing – возвращает время, затраченное алгоритмом на выполнение, принимает в качестве аргументов саму функцию поиска, размер тестируемого массива, количество прогонок, флаг среднего или худшего случая.
7. run_auto – делает заданное число полных прогонов (для всех размеров массива), может быть необходимо для уменьшения влияния task handler операционной системы.

В результате прямых измерений с помощью кода получены следующие результаты:



Асимптотическая сложность алгоритмов поиска суммы

Для выполнения этой части аналогично была написана программа sum.cpp со следующим функционалом:

1. linear_sum_search – линейный (плохой) поиск

2. smart_sum_search – бинарный поиск

3. fill – заполнение массива строго возрастающими целыми положительными числами

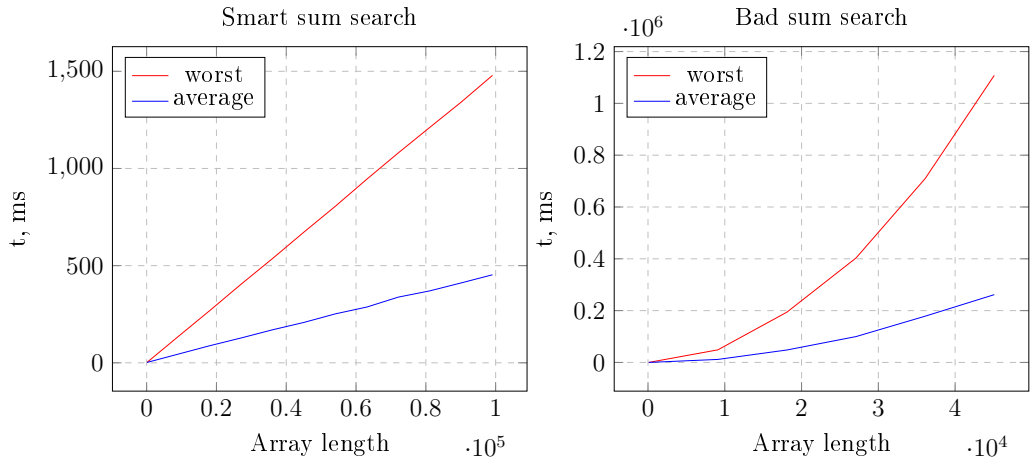
4. generate_needed – генерация случайного числа (в массиве или нет, учитывая какой случай нужен)

5. print_arr – выводит весь массив, нужен для debug

6. timing – возвращает время, затраченное алгоритмом на выполнение, принимает в качестве аргументов саму функцию поиска, размер тестируемого массива, количество прогонок, флаг среднего или худшего случая.

7. run_auto – делает заданное число полных прогонов (для всех размеров массива), может быть необходимо для уменьшения влияния task handler операционной системы.

В результате прямых измерений с помощью кода были получены следующие данные:



Сравнение сложности стратегий с обычным поиском