Лабораторная работа 1
Асимптотическая сложность

Финоченко Александр Викторович Б02-201

Цель работы: проверить прямыми измерениями времени асимптотическую сложность алгоритмов по времени в зависимости от объёма данных.

Поиск

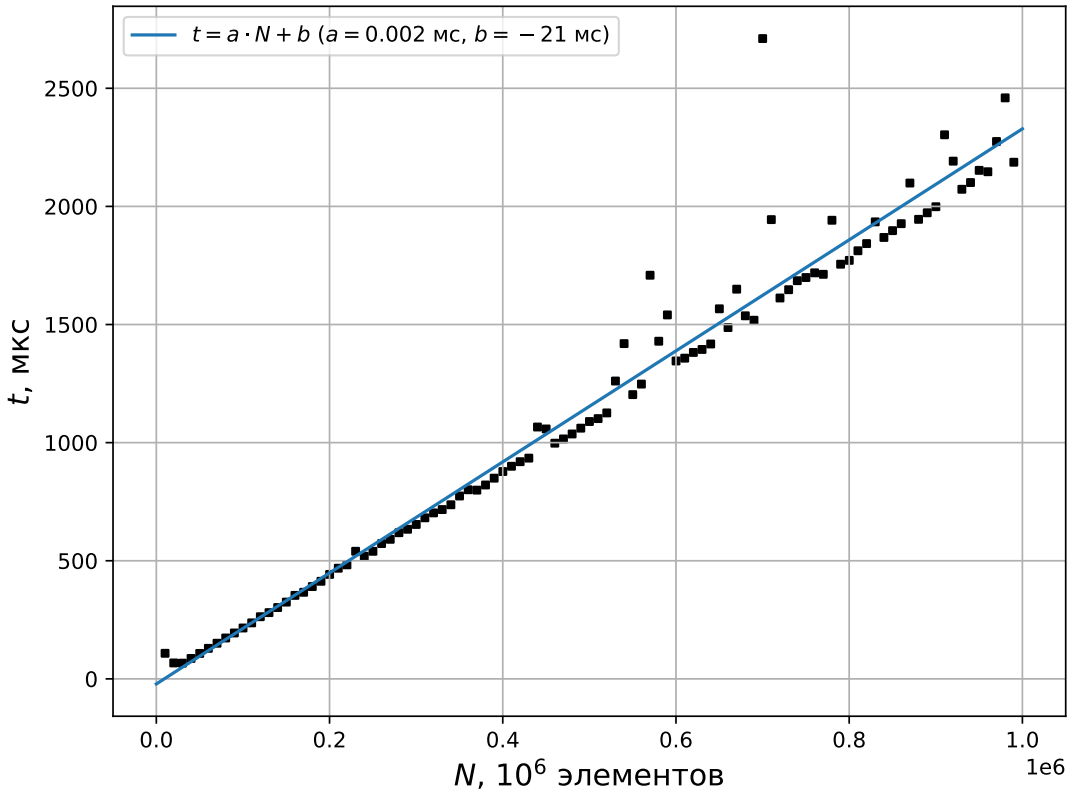
Полный перебор (папка exhaustive_search)

Сначала было измерено время для полного перебора (папка exhaustive_search). Для каждого N сразу в программе находилось среднее время для 1000 измерений.

Код функции полного перебора:

```
int exhaustive_search(int* A, int N, int value) {
  for (int i = 0; i < N; i++) {
    if (A[i] == value)
      return i;
  }
  return -1;
}
```

Чтобы построить график воспользуемся МНК. Если график представим в виде $t = aN + b$, то по МНК



Получаем хорошую линейную зависимость времени от объёма данных.

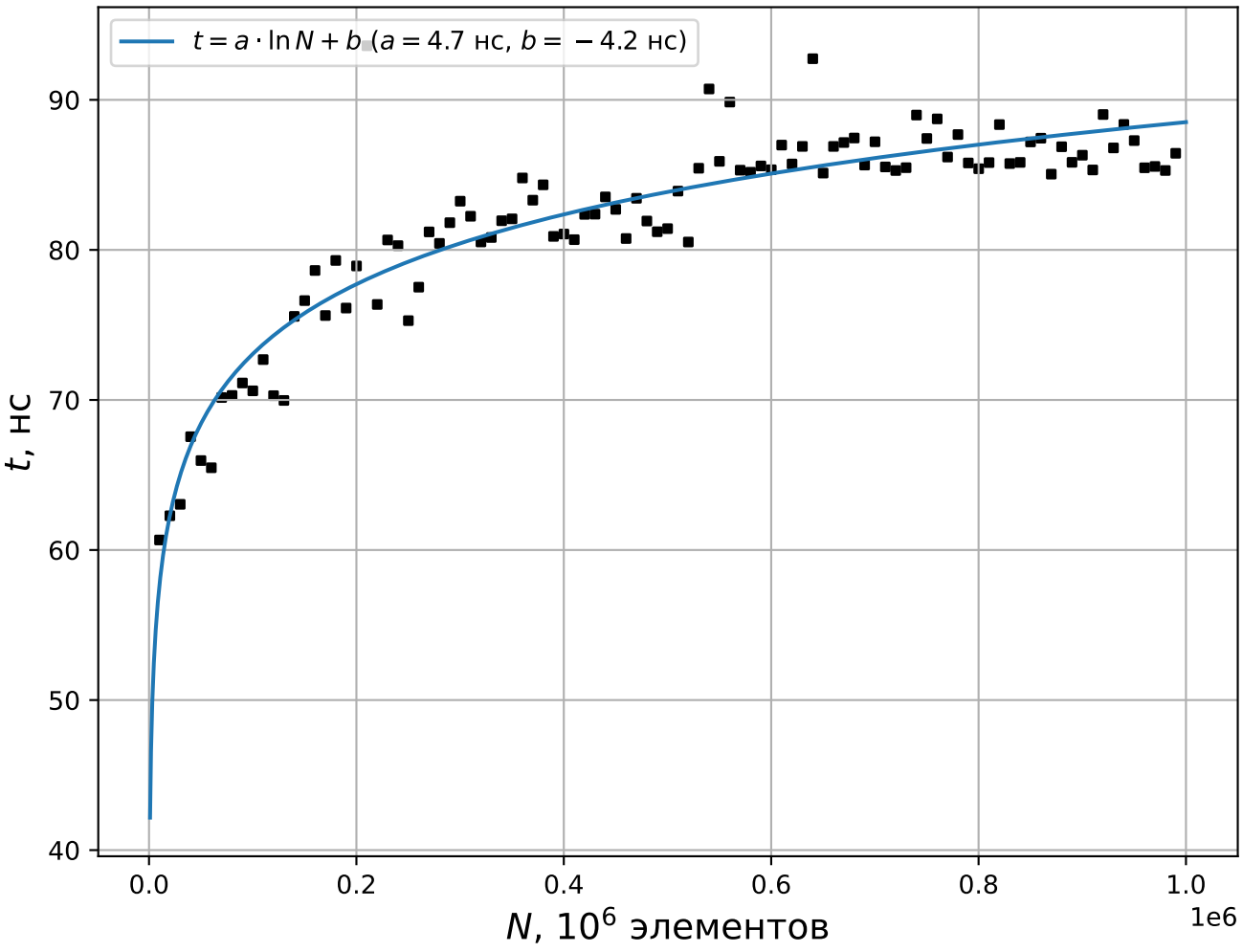
Бинарный поиск (папка binary_search)

Для измерения использовалась программа из папки binary_search. При малом количестве повторений программа выдавала сильно разные значения времени, поэтому было выбрано делать для одного N 100000000 измерений в одной программе.

Код функции бинарного поиска имеет вид

```
int binary_search(int* A, int N, int value) {
  int a = 0, b = N - 1, mid;
  bool flag = false;
  while ((a <= b) && (flag == false)) {
    mid = (a + b) / 2;
    if (A[mid] == value) return mid;
    if (A[mid] > value) b = mid - 1;
    else a = mid + 1;
  }
  return -1;
}
```

Если график представим в виде $t = a \log_2 N + b$, то по МНК



Получаем нормальную логарифмическую зависимость времени от объёма данных.

Сумма двух

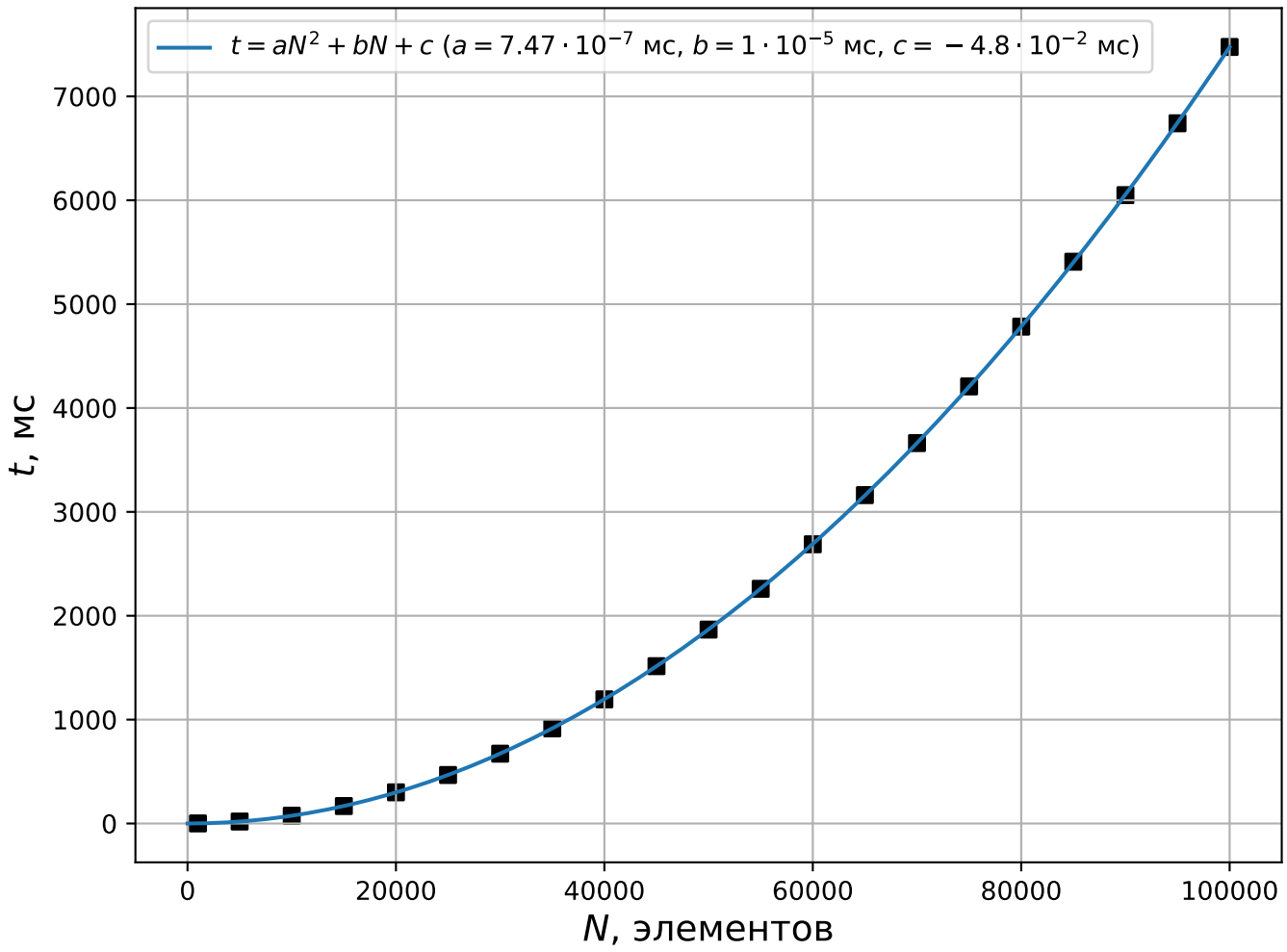
Полный перебор $O(N^2)$ (папка sum_of_two)

Было измерено время для полного перебора с вложенным циклом . Для каждого N сразу в программе находилось среднее время для 1000 измерений.

Код функции полного перебора представлен ниже

```
void sum_search(int* A, int N, int value) {
  for (int i = 0; i < N - 1; i++) {
    for (int j = i + 1; j < N; j++) {
      if (A[i] + A[j] == value) {
        cout << i << " " << j << "\n";
        return;
      }
    }
  }
}
```

Если график представим в виде $t = a \log_2 N + b$, то по МНК $a = 7.47 \cdot 10^{-7}$ мс, $b = 1 \cdot 10^{-5}$ мс, $c = -4.8 \cdot 10^{-2}$ мс.



Получаем хорошую квадратичную зависимость времени от объёма данных.

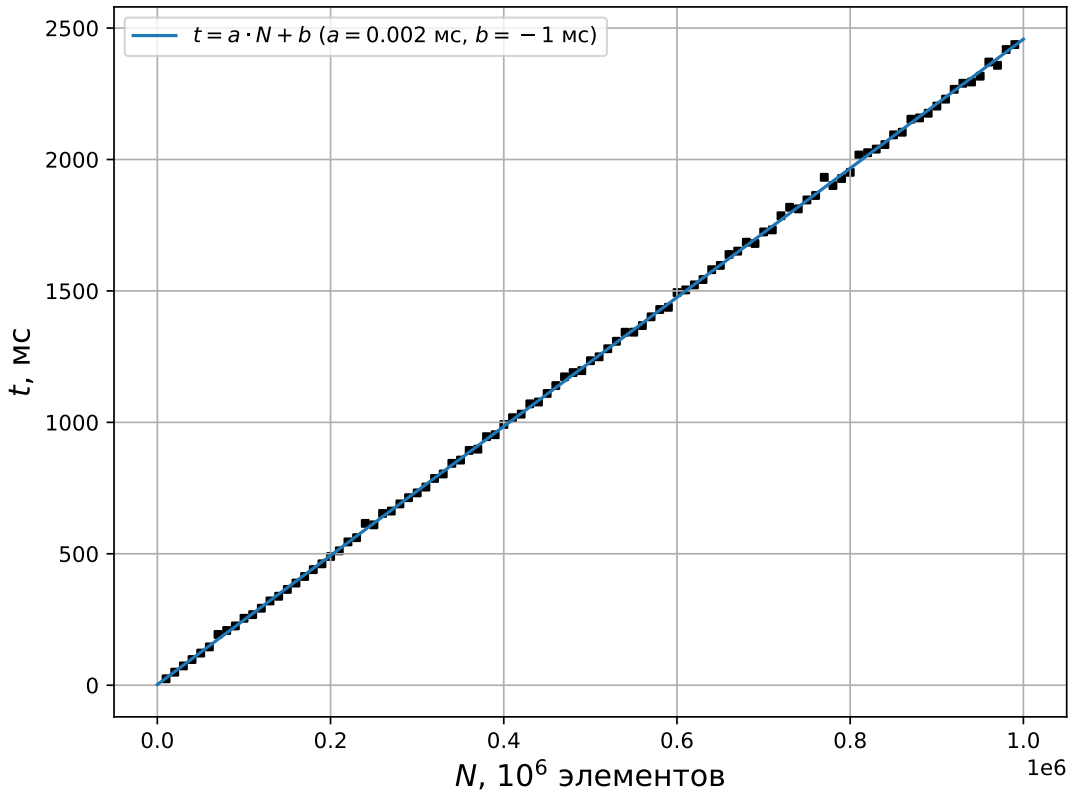
Алгоритм поиска $O(N)$ (папка sum_ON)

Код функции, работающей по алгоритму с асимптотикой $O(N)$, представлен ниже

```
void sum_of_two(int* A, int N, int value) {
  int i = 0, j = N - 1;
  while (i != j) {
    if (A[i] + A[j] == value) {
      cout << i << " " << j << "\n";
      return;
    }
    if (A[i] + A[j] > value)
      j--;
    else
      i++;
  }
}
```

Берутся два крайних индекса, если сумма значений больше, то уменьшаем верхний индекс, меньше – увеличиваем нижний, если равна – выводим значения.Заметим, что этот алгоритм корректно работает. Для каждого N сразу в программе находилось среднее время для 1000 измерений.

Чтобы построить график воспользуемся МНК. Если график представим в виде $t = aN + b$, то по МНК



Часто используемый элемент

Приведем три стратегии

```
void strategy_A(int* A, int N) {
  for (int i = 1; i < N; i++) {
    if (A[i] != A[0]) {
      int temp;
      temp = A[0];
      A[0] = A[i];
      A[i] = temp;
    }
  }
}
```

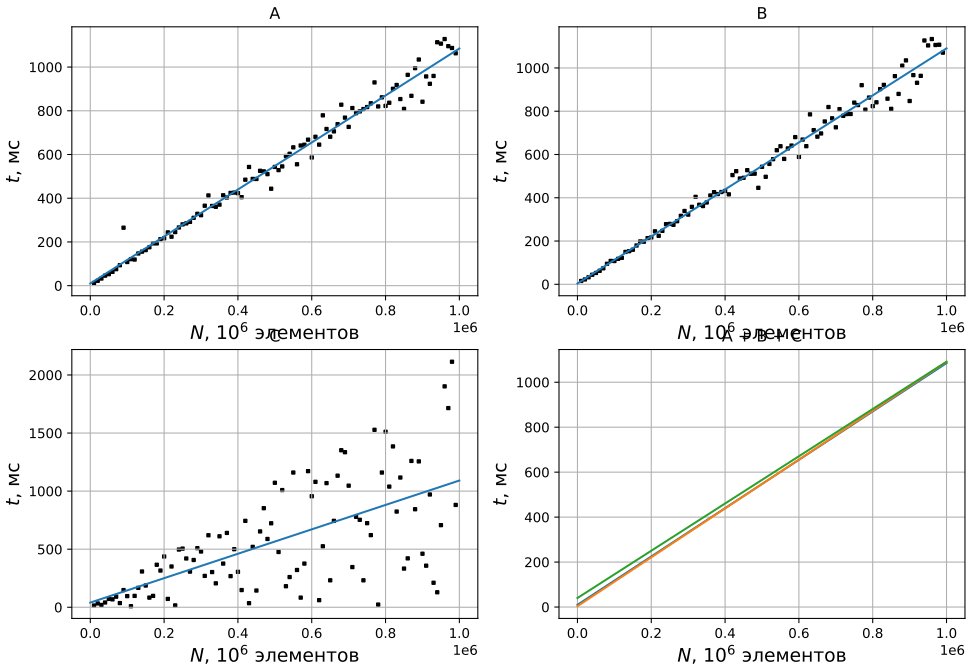
```
void strategy_B(int* A, int N) {
  for (int i = 1; i < N; i++) {
    if (A[i] != A[0]) {
      int temp;
      temp = A[i-1];
      A[i-1] = A[i];
      A[i] = temp;
    }
  }
}
```

```
void strategy_C(int* A, int N) {
  int carr[MAX_RAND + 1] = {0};
  for (int i = 0; i < N; i++) {
    carr[A[i]]++;
  }
  for (int i = 1; i < N; i++) {
    if (carr[A[i]] > carr[A[i - 1]]) {
      int temp = A[i - 1];
      A[i - 1] = A[i];
      A[i] = temp;
    }
  }
}
```

Равномерный массив

Равномерный массив был создан просто как последовательность чисел от 1 до N. Т.к. у равномерного массива все числа встречаются 1 раз, то для поиска было выбрано случайное число.

Приведём графики

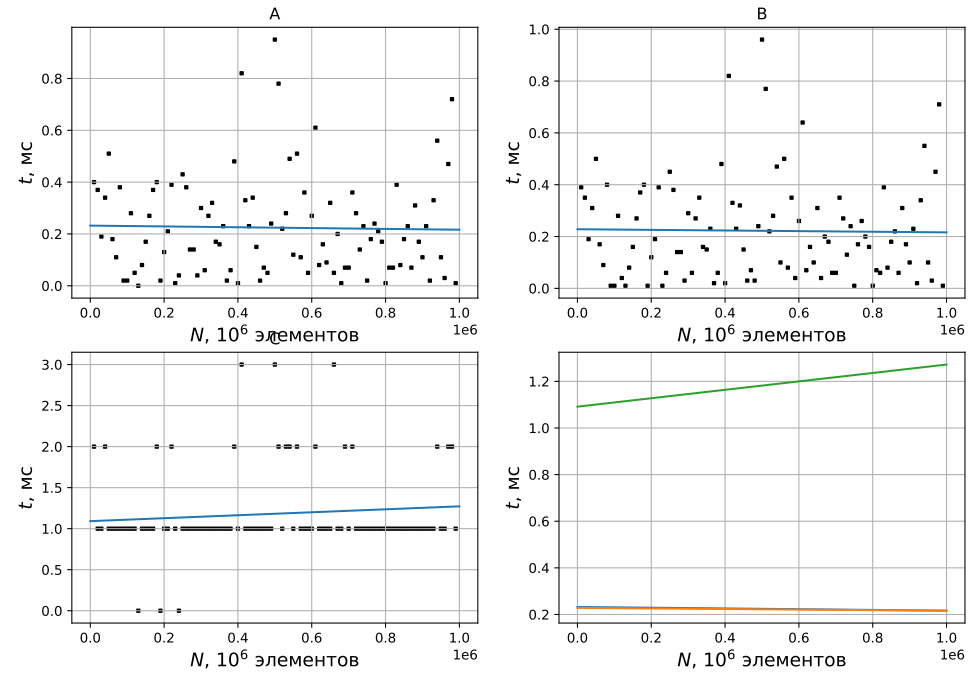


Видно, что асимптотика не зависит от стратегии.

Неравномерный массив

Неравномерный массив был создан просто как случайных чисел от 1 до MAX_RAND. Т.к. у неравномерного массива числа могут встретиться несколько раз, для поиска выбирается наиболее распространенное число число.

Приведём графики



Видим, что стратегии A и B дают асимптотику $O(1)$. Стратегия C всё равно даёт $O(N)$.