Лабораторная работа I. Асимптотическая сложность.

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

Поиск (3 - 5)

Напишите две функции поиска значения в массиве целых чисел (тип **int**): функцию поиска полным перебором для массива с произвольными данными и функцию бинарного поиска для упорядоченного массива. Убедитесь прямыми измерениями времени, что для функции с полным перебором время растёт линейно с объёмом данных, а для функции бинарного поиска - логарифмически. Следует учесть, что время одного запуска функции может быть небольшим и может зависеть от планировщика операционной системы. Необходимо для заданного числа элементов в массиве произвести множество запусков и измерить общее время. Для того, чтобы измерить время наихудшего случая, необходимо всегда выбирать значение, которое не содержится в массиве. Для проверки среднего случая можно многократно запустить функцию, выбирая элемент случайным образом. Чтобы произвести замер времени можно воспользоваться следующим кодом:

```
#include <iostream>
#include <chrono>

void func() {
    std::cout << "Hello world" << '\r';
}

int main() {
    auto begin = std::chrono::steady_clock::now();
    for (unsigned cnt = 100000; cnt != 0; --cnt)
        func();
    auto end = std::chrono::steady_clock::now();
    auto time_span =
    std::chrono::duration_cast<std::chrono::milliseconds>(end - begin);

    std::cout << "\n\n";
    std::cout << time_span.count() << std::endl;

```

```
    return 0;
}
```

Чтобы выбрать случайное целое число в заданном диапазоне можно воспользоваться следующим кодом:

```
#include <iostream>
#include <random>

int main() {
    int arr[] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 0};

    unsigned seed = 1001;
    std::default_random_engine rng(seed);
    std::uniform_int_distribution<unsigned> dstr(0,9);

    for (unsigned counter = 100; counter != 0; --counter)
        std::cout << arr[dstr(rng)] << '\n';
    std::cout << std::endl;

    return 0;
}
```

Для успешной сдачи лабораторной работы необходимо: произвести измерения времени работы функций для различного количества элементов N в диапазоне от 100 до $1'000'000$. Построить графики зависимости времени от количества элементов с использованием инструментов **matplotlib** и **numpy**.

Сумма двух (6 - 7)

В массиве требуется найти два различных элемента, которые в сумме дают заданное число, либо указать, что такой пары нет. Подтвердите прямыми измерениями, что алгоритм полного перебора имеет квадратичную асимптотическую сложность от N, количества элементов в массиве ($O(N^2)$). Для упорядоченного массива существует алгоритм, который работает с линейной асимптотикой. Реализуйте этот алгоритм и подтвердите асимптотику прямыми измерениями.

Часто используемый элемент (8 - 10)

В первой задаче мы подразумевали, что при поиске мы в качестве ключа передаём произвольные элементы. Однако, часто в жизни данные оказы-

ваются далеко не случайные и некоторые элементы мы ищем чаще других. Попытаемся оптимизировать наш массив для таких случаев. Если в массиве при поиске мы находим некоторый элемент, то будем продвигать его ближе к началу, чтобы при следующем поиске найти быстрее. Рассмотрим три стратегии. **Стратегия A**: если мы нашли в массиве некоторый элемент и он не является первым, то обменяем его с первых элментом. **Стратегия B**: если мы нашли в массиве некоторый элемент и он не является первым, то переставим его с левым соседом, элементом с индексом на единицу меньше. **Стратегия C**: если мы нашли в массиве некоторый элемент, то увеличим счётчик успешных поисков этого элемента (для этого нам потребуется дополнительный массив счётчиков), если элемент не первый и его счётчик превышает счётчик элемента слева, то поменяем их местами. Во всех трёх стратегиях ничего не предпринимаем, если элемент в массиве не найден.

Проверьте прямыми измерениями времени:

- изменится ли асимптотика поиска, если при поиске запросы будут равномерно распределены по множеству значений, включая неудачный поиск (нет такого элемента);
- изменится ли асимптотика поиска, если распределение не равномерно (какие-то данные при поиске встречаются гораздо чаще);
- есть ли различия в асимптотической сложности этих стратегий при равномерном и неравноверном распределении данных.