Упражнение 1 Временная сложность

NESTED_LOOPS.cpp

```cpp
int sum = 0;

for (size_t i = 1; i < N; i *= 2) {
    for (size_t j = N; j > 0; j /= 2) {
        for (size_t k = j; k < N; k += 2) {
            sum += (i + j * k);
        }
    }
}
```

С помощью пошаговой трассировки составить точное выражение для функции временной сложности $T(N)$, а также оценить порядок роста этой функции

Упражнение 2 Линейный поиск

Вход: Последовательность ключей $\boldsymbol{A = \langle a_1, \ldots, a_n \rangle}$ и значение $\boldsymbol{v}$.

Выход: Индекс $\boldsymbol{1 \leq i \leq n}$, для которого $\boldsymbol{A[i] = v}$ или сообщение **NOT FOUND**, если такого индекса не существует.

1. Разработать циклический алгоритм линейного поиска

3. Оценить сложность $\boldsymbol{T(n)}$ разработанного алгоритма

Упражнение 3 Бинарный поиск

Вход: Последовательность ключей $\boldsymbol{A = \langle a_1, \ldots, a_n \rangle}$, для которой $\boldsymbol{a_1 \leq a_2 \leq \cdots \leq a_n}$, и значение $\boldsymbol{v}$.

Выход: Индекс $\boldsymbol{1 \leq i \leq n}$, для которого $\boldsymbol{A[i] = v}$ или сообщение **NOT FOUND**, если такого индекса не существует.

1. Разработать циклический алгоритм бинарного поиска.

3. Оценить сложность $\boldsymbol{T(n)}$ разработанного алгоритма.

Упражнение 4 Сортировка пузырьком

BUBBLE_SORT.cpp

```cpp
void bubbleSort(std::vector<int> arr) {
    size_t N = arr.size();
    for (size_t i = 0; i < N; ++i) {
        for (size_t j = 0; j < N - i - 1; ++j) {
            if (arr[j] > arr[j + 1]) {
                std::swap(arr[j], arr[j + 1]);
            }
        }
    }
}
```

2. Оценить сложность $\boldsymbol{T(N)}$ алгоритма. Выделить худший и лучший случай по временной сложности.