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]);
            }
        }
    }
}
```

1. Докажите корректность алгоритма сортировки с помощью инварианта.
2. Оцените сложность $T(N)$ алгоритма.
3. Выделите худший и лучший случай по временной сложности.

Как мы можем **оптимизировать** работу алгоритма BUBBLE SORT?

1) Возьмём следующий инвариант внешнего цикла: после i-й итерации цикла последние i элементов массива будут являться i самыми наибольшими элементами массива и они будут находиться в отсортированном порядке.

Докажем по индукции.

База ($i = 0$): ничего не сделано, очевидно последние 0 элементов отсортированы.

Шаг: предположим, что после i-й итерации верно, что последние i элементов – наибольшие и отсортированы. На $(i+1)$-й итерации внешний цикл выполняет один проход внутреннего цикла, который сравнивает пары соседних элементов и больший из них ставит правее. Очевидно, что в таком случае наибольший элемент $a_k \mid 0 \le k < n-i$ – индекс эл-та до начала внешнего цикла, будет двигаться вправо до $n-(i+1)$. Так образуются уже $i+1$ наибольших элементов в конце. И отсортированы они будут, т.к. a_k элемент будет двигаться вправо среди последних $i+1$ элементов, пока не станет на свою позицию, чтобы $i+1$ последних э-тов стали отсортированы.

По индукции следует, что инвариант выполняется на каждой итерации внешнего цикла. Ч.т.д.

2) Пусть N – длина массива.

строка	к-во итер-ий	затраты
1)	1	C_1
2)	N	C_2
3)	$\frac{N(N-1)}{2}$	C_3
4)	$\frac{N(N-1)}{2}$	C_4
5)	$\sum_{i=0}^{N-1} t_i$	C_5

3) $\sum_{i=0}^{N-1} N-i-1 = \sum_{m=0}^{N-1} m = \frac{N(N-1)}{2}$

5) t_i – сколько раз выполнится условие в строке 4 на i-м шаге внешнего цикла.

$T(N) = C_1 + C_2 \cdot N + (C_3 + C_4) \cdot \frac{N(N-1)}{2} + C_5 \cdot \sum_{i=0}^{N-1} t_i$

3)

Худший случай $t_i = N - i - 1$, тогда $T(N) = c_1 + c_2 \cdot N + (c_3 + c_4 + c_5) \cdot \frac{N(N-1)}{2}$ – квадратичная ф-я

Лучший случай $t_i = 0$, тогда $T(N) = c_1 + c_2 \cdot N + (c_3 + c_4) \cdot \frac{N(N-1)}{2}$ – квадратичная ф-я.

Вывод: $T(N) = \Theta(N^2)$

Для оптимизации алгоритма можно ввести флаг который будет считать количество обменов. Если обменов будет 0, то массив уже отсортирован и можно закончить сортировку. В некоторых случаях это может дать линейное время работы. К примеру, если дан уже отсортированный массив.