Добавляем элемент в конец списка

```
/home/lxuser/.pyenv/versions/3.12.9/bin/python /home/lxuser/Documents/mai/05_devops/mai-devops/lab04/main.py
---------------------------------------

Добавляем 1000 элементов в конец списка:
Время выполнения:  0  ms

---------------------------------------

Добавляем 10000 элементов в конец списка:
Время выполнения:  6  ms

---------------------------------------

Добавляем 100000 элементов в конец списка:
Время выполнения:  51  ms

---------------------------------------

Добавляем 1000000 элементов в конец списка:
Время выполнения:  516  ms

---------------------------------------


Process finished with exit code 0
```

Добавляем элемент в начало списка

```
/home/lxuser/.pyenv/versions/3.12.9/bin/python /home/lxuser/Documents/mai/05_devops/mai-devops/lab04/add_to_begin.py
----------------------------------------

Добавляем 1000 элементов в начало списка:
Время выполнения:  0  ms

----------------------------------------

Добавляем 10000 элементов в начало списка:
Время выполнения:  15  ms

----------------------------------------

Добавляем 100000 элементов в начало списка:
Время выполнения:  983  ms

----------------------------------------

Добавляем 1000000 элементов в начало списка:
Время выполнения:  92694  ms

----------------------------------------


Process finished with exit code 0
```

Временная сложность

1)Добавление элемента в конец списка (list.append(x))

- **Временная сложность: O(1) (амортизированная)**
- Список в Python реализован как динамический массив, и добавление в конец обычно выполняется быстро. В редких случаях, когда требуется увеличение размера массива, происходит переаллокация, но в среднем это всё равно O(1).

2)Добавление элемента в начало списка (list.insert(0, x))

- **Временная сложность: O(n)**
- При вставке в начало все существующие элементы сдвигаются на одну позицию вправо, что требует линейного времени.

Сортировка пузырьком

```
/home/lxuser/.pyenv/versions/3.12.9/bin/python
Сортируем пузырьком список размера 1000:
Время выполнения:  44  ms

---------------------------------------

Сортируем пузырьком список размера 10000:
Время выполнения:  5506  ms

---------------------------------------

Сортируем пузырьком список размера 100000:
Время выполнения:  554649  ms

---------------------------------------


Process finished with exit code 0
```

```python
def bubble_sort(arr):  3 usages    new *
    n = len(arr)
    for i in range(n):
        for j in range(0, n - i - 1):
            if arr[j] > arr[j + 1]:
                arr[j], arr[j + 1] = arr[j + 1], arr[j]
    return arr
```

Временная сложность алгоритма сортировки пузырьком $O(n^2)$

Стандартная сортировка

```
Сортируем список размера 1000:
Время выполнения:  0  ms

---------------------------------------

Сортируем список размера 10000:
Время выполнения:  1  ms

---------------------------------------

Сортируем список размера 100000:
Время выполнения:  12  ms

---------------------------------------

Сортируем список размера 1000000:
Время выполнения:  192  ms

---------------------------------------


Process finished with exit code 0
```

Временная сложность стандартной сортировки в python O(nlog(n))

Вопросы

1) **Что такое временная сложность алгоритма, как её определить?**

Временная сложность T(N) — это функция от количества входных данных, равная максимальному времени работы алгоритма на данном входе. Используется в качестве одного из критериев оценки алгоритма.

2) **Какой способ оптимизации времени выполнения сортировки списка применён в задании №4?**

Изменили алгоритм сортировки пузырьком с сложностью O(n^2) на стандартную функцию сортировки с меньшей временной сложностью O(n log n)

3) **Как можно оптимизировать добавление элементов в начало списка?**

Заменить list на коллекцию, в которой метод вставки элементов в начало имеет меньшую временную сложность. Так как метод insert() коллекции list не оптимизирован для вставки элементов в начало списка (временная сложность O(n)), то верным решением будет заменить list на коллекцию deque, у которой есть оптимизированный метод вставки в начало appendleft() со сложностью выполнения O(1).