Сортировки

Сортировка Шелла и сортировка вставками

Сортировка вставками

- Сортировка вставками – это алгоритм сортировки, в котором элементы входной последовательности просматриваются по одному, и каждый новый поступающий элемент размещается в нужное место среди ранее упорядоченных элементов.
- Т.е. создаётся подтип, в котором элемент всегда добавляется в конец списка, а затем перемещается по списку до тех пор, пока он меньше предыдущего элемента.

- Посмотрим, как это работает: у нас есть последовательность из 5 чисел. Этот алгоритм изначально выбирает первый элемент и помещает его в подмассив, после этого он переходит к следующему элементу и так как 33 больше чем 14, это число остаётся в конце подмассива.
- Следующее число у нас идёт 10, оно помещается в подмассив и внутри подмассива сравнивается поочерёдно, сначала с числом 33, а затем с числом 14. И так как 10 самое маленькое, это число идёт в самое начало списка.
- После этого идёт число 35. Оно помещается в подмассив, и так как оно среди них самое большое, то оно остаётся на месте.
- После этого идёт число 27. Это число сравнивается сначала с 35, потом с 33, а потом с числом 14. Т.к оно больше 14, то это число помещается после 14.

- Итог: массив отсортирован.
- Теперь перейдём к реализации данного алгоритма на языке python. Создадим функцию, в качестве параметра которой будет выступать последовательность. Теперь создадим цикл for, чтобы пройти по всем элементам последовательности.
- Внутри цикла создадим две переменные: current_value, который в нашем случае будет отвечать за текущий элемент и position – которому будет присвоен индекс текущего элемента.
- Теперь создадим цикл while, который будет сортировать внутри подмассива элементы и он будет работать, пока текущий индекс элемента будет больше 0 и пока последний элемент в этом подмассиве будет больше текущего элемента.

- Внутри цикла while мы будем перемещать текущий элемент внутри подмассива, пока он не встанет в нужную позицию.
- А также в конце цикл while должен передать следующее значение в наш подмассив.
- И функция должна возвращать нам отсортированный массив.

- Пример рабочего кода be like:

```python
def create_array(size=10, max=50):
    from random import randint
    return [randint(0, max) for _ in range(size)]
```

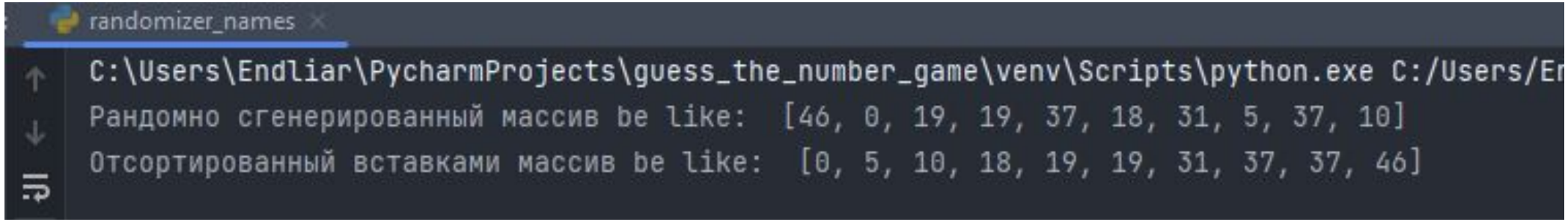
```python
def insertion_sort(lst):
    for item in range(1, len(lst)):
        current_value = lst[item]
        position = item

        while position > 0 and lst[position - 1] > current_value:
            lst[position] = lst[position - 1]
            position -= 1
        lst[position] = current_value
```

- Код вывода результата be like:

```
124
125     lst = create_array()
126     print("Рандомно сгенерированный массив be like: ", lst)
127     lst = insertion_sort(lst)
128     print("Отсортированный вставками массив be like: ", lst)
129
```

- Вывод:



Сортировка Шелла

- Сортировка Шелла – улучшенный алгоритм сортировки вставками. Идея метода Шелла состоит в сравнении элементов стоящих не только рядом, но и на определённом расстоянии друг от друга.
- Давайте более подробно рассмотрим следующий пример, чтобы понять, как работает данный алгоритм сортировки.

- Итак, у нас есть несортированный массив и мы разбиваем его на подмассивы, выбирая не рядом стоящие элементы, а через определённый диапазон, который рассчитывается в зависимости от длины массива. После разбиения на подмассивы, мы сравниваем два элемента между собой и при необходимости меняем местами, а именно: у нас есть массив из 10 переменных, диапазон определяется таким образом – мы длину массива будем нацело делить на два.
- Мы берём и в диапазоне от нулевого индекса до 4. Сравниваем крайние значения между собой, они стоят в нужном порядке, дальше мы сдвигаем вправо к следующему элементу массива следующее значение под индексом 1 и, соответственно, значение под индексом 5. И опять сдвигаем данный диапазон вправо. И так до конца.

- После этого мы должны диапазон, через который выбираются элементы, опять разделить на два. Так как у нас предыдущее значение диапазона было = 4. Мы делим его на два, и получается 2.
- Соответственно сравниваем нулевой индекс и второй. Меняем их местами, сдвигаем диапазон вправо. И опять сравниваем дальнейшие элементы. И так до конца.
- После этого мы опять должны данный диапазон поделить на два. Т.е диапазон будет уже равен единице – а значит сравнивать элементы мы будем уже рядом стоящие.
- В конечном счёте получаем отсортированный массив.

- Перейдём к реализации данного алгоритма на языке python.
- Создадим функцию, которая принимать будет неотсортированный массив снова в качестве параметра. Всё по классике.
- Затем создадим переменную get, которая будет отвечать за диапазон, между которым будут сравниваться два элемента.
- Теперь создадим цикл while, условием работы которого будет: пока значение get > 0.
- Создадим цикл for внутри цикла while.
- Создадим переменную current_value и присвоим ей значение текущего элемента.
- И также создадим переменную position, которая будет отвечать за значение индекса элемента.

- Теперь создадим еще один цикл while, условием работы которого будет: пока значение индекса будет >= get и при этом элемент сравниваемый с текущим значение будет больше него.
- Внутри цикла мы будем менять сравниваемые элементы местами.
- И за пределами цикла while мы должны переменную get опять разделить нацело пополам.

- Пример рабочего кода:

```python
def create_array(size=10, max=50):
    from random import randint
    return [randint(0, max) for _ in range(size)]
```

```python
def shell_sort(lst):
    get = len(lst) // 2
    while get > 0:
        for value in range(get, len(lst)):
            current_value = lst[value]
            position = value

            while position >= get and lst[position - get] > current_value:
                lst[position] = lst[position - get]
                position -= get
                lst[position] = current_value
        get //= 2
    return lst
```

- Результат работы кода: