

Сортировка списка

-- упорядочивание элементов в списке.

1. выбором:

- а. простым выбором,
- б. пирамидальная*;

2. обменные методы:

- а. пузырьек,
- б. пузырьек с флагом,
- с. расческой*,
- д. шейкер;

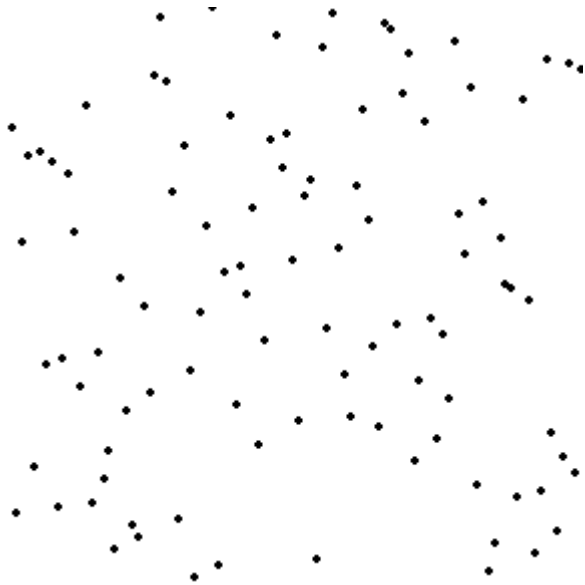
3. вставками:

- а. простыми вставками,
- б. вставками с барьером,
- с. вставками с бинарным поиском,
- д. гномья*,
- е. Шелла;

4. быстрая (один из обменных методов).

Сортировка выбором

- 1) выбирается наименьший элемент и ставится в позицию 1,
- 2) из оставшихся выбирается наименьший и ставится в позицию 2,
- 3) продолжать до конца списка.

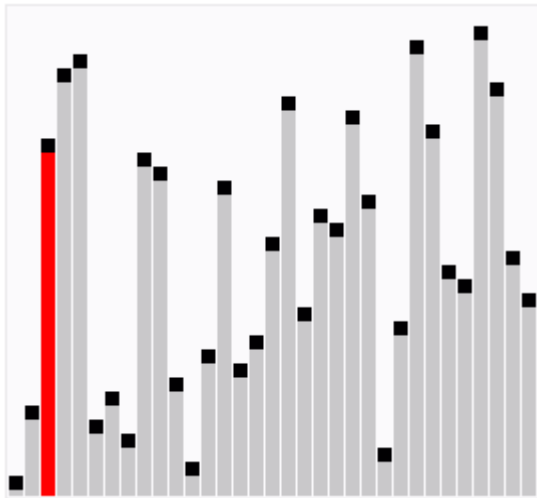


```
def selectionsort(arr):  
    for i in range(len(arr)):  
        minind = i  
        for j in range(i+1,len(arr)):  
            if arr[j] < arr[minind]:  
                minind = j  
        arr[i],arr[minind] = arr[minind],arr[i]  
        print(i,' : ', *arr)  
    return arr
```

```
start list: 4 8 2 5 9 1  
0 : 1 8 2 5 9 4  
1 : 1 2 8 5 9 4  
2 : 1 2 4 5 9 8  
3 : 1 2 4 5 9 8  
4 : 1 2 4 5 8 9  
5 : 1 2 4 5 8 9  
sorted list: 1 2 4 5 8 9
```

Пузырьковая сортировка

- 1)сравниваются 2 соседних элемента;
- 2)если нарушен порядок, то они обмениваются.



```
def bubblesort(arr):  
    n = len(arr)  
    for i in range(n-1):  
        for j in range(n-1-i):  
            if arr[j]>arr[j+1]:  
                arr[j],arr[j+1] = arr[j+1],arr[j]  
            #print(arr)  
        print(i, ' : ', *arr)  
    return arr
```

```
start list: 4 8 2 9 5 1  
0 : 4 2 8 5 1 9  
1 : 2 4 5 1 8 9  
2 : 2 4 1 5 8 9  
3 : 2 1 4 5 8 9  
4 : 1 2 4 5 8 9  
sorted list: 1 2 4 5 8 9
```

Пузырьковая с флагом

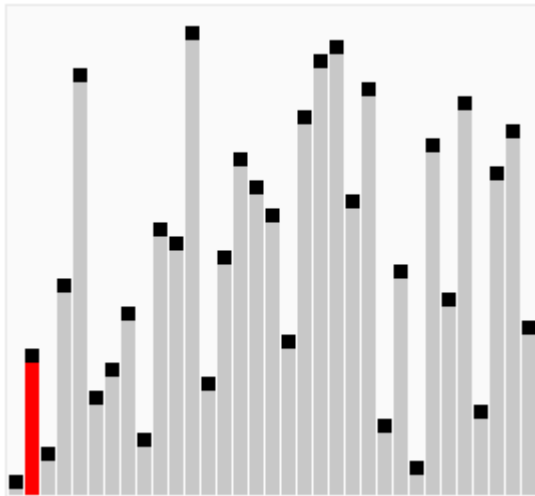
```
def bubblesortwithflag(arr):  
    n = len(arr)  
    for i in range(n-1):  
        flag = True  
        for j in range(n-1-i):  
            if arr[j]>arr[j+1]:  
                arr[j],arr[j+1] = arr[j+1],arr[j]  
                flag = False  
        if flag:  
            break  
        print(i, ' : ', *arr)  
    return arr
```

```
start list: -4 -3 -2 -1 4 8 2 9 5 1  
0 : -4 -3 -2 -1 4 2 8 5 1 9  
1 : -4 -3 -2 -1 2 4 5 1 8 9  
2 : -4 -3 -2 -1 2 4 1 5 8 9  
3 : -4 -3 -2 -1 2 1 4 5 8 9  
4 : -4 -3 -2 -1 1 2 4 5 8 9  
sorted list: -4 -3 -2 -1 1 2 4 5 8 9
```

Шейкер-сортировка

Это пузырьковая сортировка, работающая в 2 прохода:

- прямой,
- обратный.



```

def shakersort(arr):
    left = 0
    right = len(arr) - 1
    print(left, right)
    while left < right:
        r_new = left
        for i in range(left, right):
            if arr[i] > arr[i+1]:
                arr[i], arr[i + 1] = arr[i + 1], arr[i]
                r_new = i
        right = r_new
        print(*arr, ' | ', left, right)
        l_new = right
        for i in range(right - 1, left - 1, -1):
            if (arr[i] > arr[i+1]):
                arr[i], arr[i + 1] = arr[i + 1], arr[i]
                l_new = i
        left = l_new
        print(*arr, ' | ', left, right)

    return arr

```

start list: 10 -4 -3 -2 -1 4 8 2 9 5 1

-4 -3 -2 -1 4 8 2 9 5 1 10 | 0 9

-4 -3 -2 -1 1 4 8 2 9 5 10 | 4 9

-4 -3 -2 -1 1 4 2 8 5 9 10 | 4 8

-4 -3 -2 -1 1 2 4 5 8 9 10 | 5 8

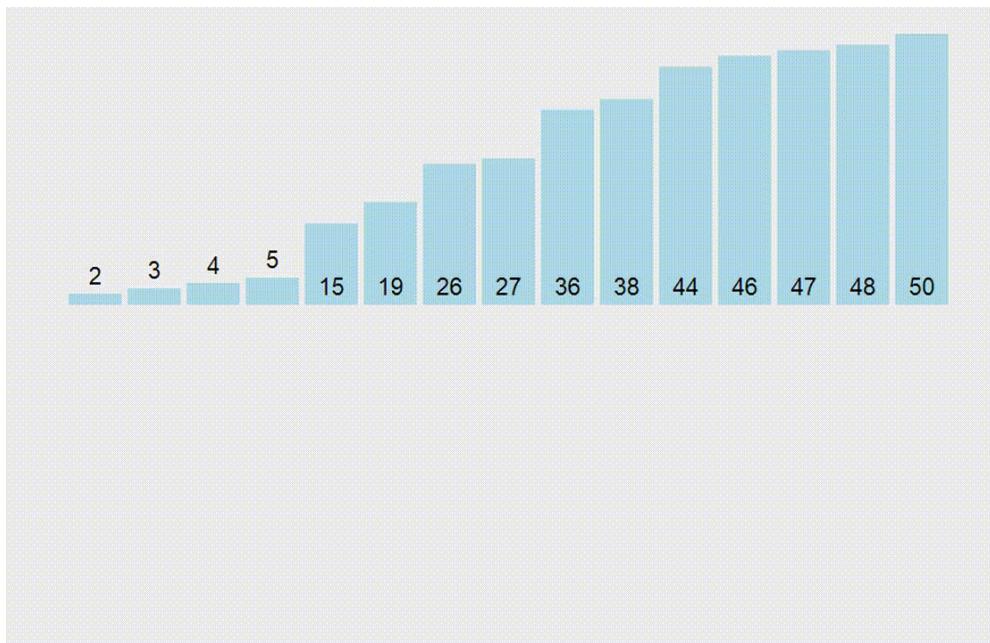
-4 -3 -2 -1 1 2 4 5 8 9 10 | 5 5

-4 -3 -2 -1 1 2 4 5 8 9 10 | 5 5

sorted list: -4 -3 -2 -1 1 2 4 5 8 9 10

Сортировка вставками

- 1) часть массива, расположенная левее текущего элемента, считается отсортированной;
- 2) в отсортированной части массива ищется место для вставки текущего элемента;
- 3) вставка элемента в найденную позицию.



```

def insertionsort(arr):
    for i in range(1, len(arr)):
        cur = arr[i]
        j = i - 1
        while (j >= 0 and cur < arr[j]):
            arr[j + 1] = arr[j]
            j = j - 1
        arr[j + 1] = cur
        print(i, ' : ', arr[:i], ' | ', arr[i:])
    return arr

```

```

start list: 10 -4 -3 -2 -1 4 8 2 9 5 1
1 : [-4] | [10, -3, -2, -1, 4, 8, 2, 9, 5, 1]
2 : [-4, -3] | [10, -2, -1, 4, 8, 2, 9, 5, 1]
3 : [-4, -3, -2] | [10, -1, 4, 8, 2, 9, 5, 1]
4 : [-4, -3, -2, -1] | [10, 4, 8, 2, 9, 5, 1]
5 : [-4, -3, -2, -1, 4] | [10, 8, 2, 9, 5, 1]
6 : [-4, -3, -2, -1, 4, 8] | [10, 2, 9, 5, 1]
7 : [-4, -3, -2, -1, 2, 4, 8] | [10, 9, 5, 1]
8 : [-4, -3, -2, -1, 2, 4, 8, 9] | [10, 5, 1]
9 : [-4, -3, -2, -1, 2, 4, 5, 8, 9] | [10, 1]
10 : [-4, -3, -2, -1, 1, 2, 4, 5, 8, 9] | [10]
sorted list: -4 -3 -2 -1 1 2 4 5 8 9 10

```

Вставками с барьером

```
def insertionsortwithbarrier(arr):  
    arr = [0] + arr  
    for i in range(1, len(arr)):  
        arr[0] = arr[i]  
        j = i - 1  
        while (arr[0] < arr[j]):  
            arr[j + 1] = arr[j]  
            j = j - 1  
        arr[j + 1] = arr[0]  
        print(i, ' : ', arr[:i], ' | ', arr[i:])  
    return arr[1:]
```

```
start list: 10 -4 -3 -2 -1 4 8 2 9 5 1  
1 : [10] | [10, -4, -3, -2, -1, 4, 8, 2, 9, 5, 1]  
2 : [-4, -4] | [10, -3, -2, -1, 4, 8, 2, 9, 5, 1]  
3 : [-3, -4, -3] | [10, -2, -1, 4, 8, 2, 9, 5, 1]  
4 : [-2, -4, -3, -2] | [10, -1, 4, 8, 2, 9, 5, 1]  
5 : [-1, -4, -3, -2, -1] | [10, 4, 8, 2, 9, 5, 1]  
6 : [4, -4, -3, -2, -1, 4] | [10, 8, 2, 9, 5, 1]  
7 : [8, -4, -3, -2, -1, 4, 8] | [10, 2, 9, 5, 1]  
8 : [2, -4, -3, -2, -1, 2, 4, 8] | [10, 9, 5, 1]  
9 : [9, -4, -3, -2, -1, 2, 4, 8, 9] | [10, 5, 1]  
10 : [5, -4, -3, -2, -1, 2, 4, 5, 8, 9] | [10, 1]  
11 : [1, -4, -3, -2, -1, 1, 2, 4, 5, 8, 9] | [10]  
sorted list: -4 -3 -2 -1 1 2 4 5 8 9 10
```

Вставками с бинарным поиском

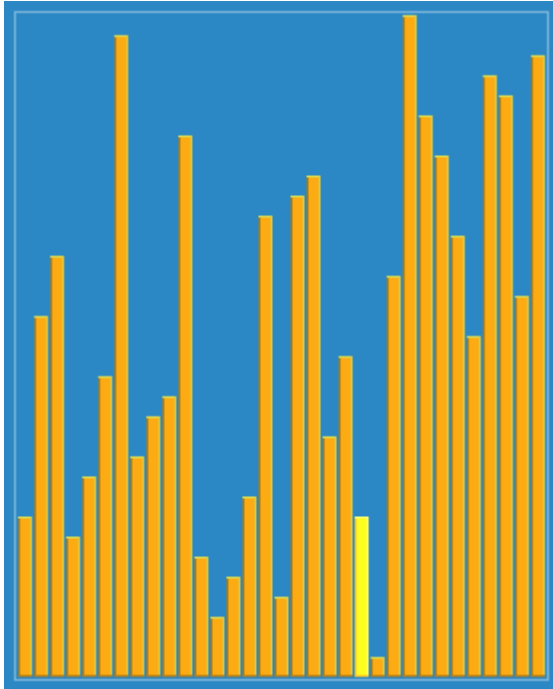
```
def insertion_sort_with_binsearch(arr):
    for i in range(1, len(arr)):
        cur = arr[i]
        lo = 0
        hi = i
        if lo == hi: lo += 1
        else:
            while lo < hi:
                mid = (lo + hi) // 2
                if cur < arr[mid]: hi = mid
                else: lo = mid + 1

        j = i
        while (j > lo and j > 0):
            arr[j] = arr[j-1]
            j = j - 1
        arr[lo] = cur
        print(i, ' : ', *arr)
    return arr
```

```
start list: 4 8 2 9 5 1
1 : 4 8 2 9 5 1
2 : 2 4 8 9 5 1
3 : 2 4 8 9 5 1
4 : 2 4 5 8 9 1
5 : 1 2 4 5 8 9
sorted list: 1 2 4 5 8 9
```

Сортировка Шелла

сортировка вставками, осуществляемая с определенным шагом.



```

def Shellsort(arr):
    inc = len(arr) // 2
    while inc:
        for i, el in enumerate(arr):
            while i >= inc and arr[i - inc] > el:
                arr[i] = arr[i - inc]
                i -= inc
            arr[i] = el
            print(i, ' : ', *arr, ' | ', inc)
        inc = 1 if inc == 2 else int(inc * 5.0 / 11)
    return arr

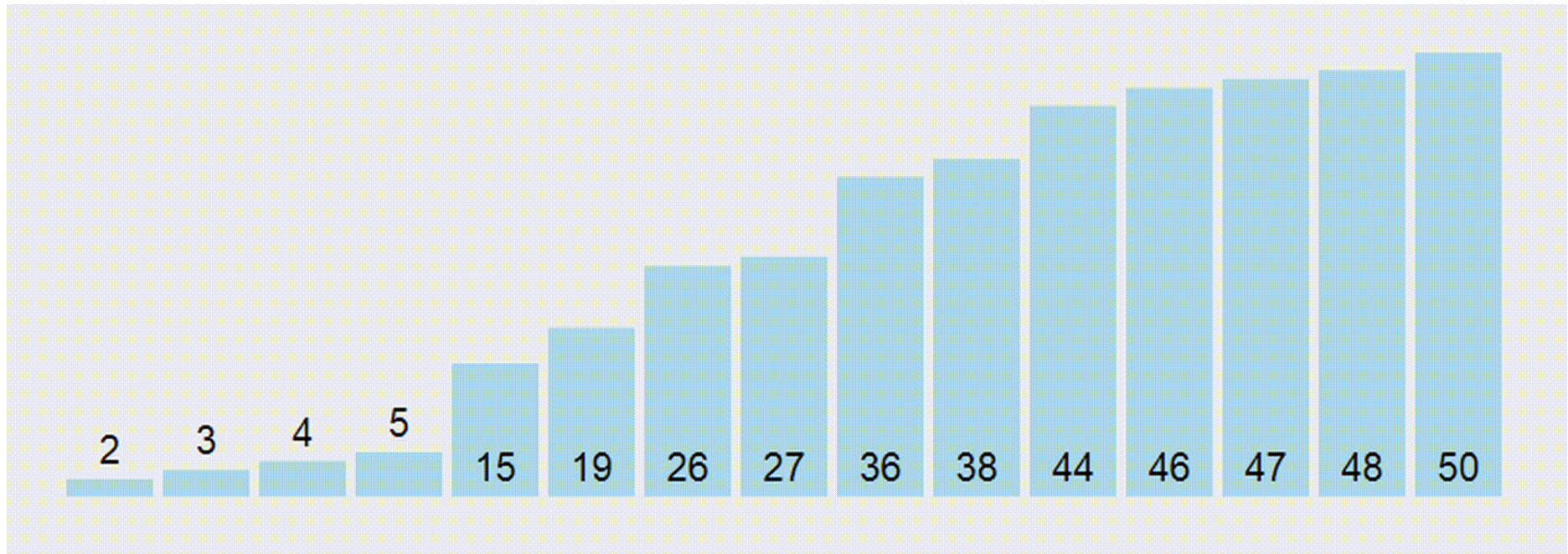
```

```

start list: 4 8 2 9 5 1
0 : 4 8 2 9 5 1 | 3
1 : 4 8 2 9 5 1 | 3
2 : 4 8 2 9 5 1 | 3
3 : 4 8 2 9 5 1 | 3
1 : 4 5 2 9 8 1 | 3
2 : 4 5 1 9 8 2 | 3
0 : 4 5 1 9 8 2 | 1
1 : 4 5 1 9 8 2 | 1
0 : 1 4 5 9 8 2 | 1
3 : 1 4 5 9 8 2 | 1
3 : 1 4 5 8 9 2 | 1
1 : 1 2 4 5 8 9 | 1
sorted list: 1 2 4 5 8 9

```

Быстрая сортировка



```

def quicksort(arr, start = 0, end = None):
    if len(arr) == 0:
        return arr
    pind = r.randint(start, end-1)
    pivot = arr[pind]
    left = [x for x in arr if x < pivot]
    right = [x for x in arr if x > pivot]
    print('pivot: ', pind, pivot)
    print('left: ', left)
    print('right: ', right)

    return quicksort(left, 0, len(left)) +
[pivot] + \
    quicksort(right, 0, len(right))

```

```

start list: 4 8 2 9 5 1
pivot:
2 2
left: [1]
right: [4, 8, 9, 5]
pivot: 0 1
left: []
right: []
pivot: 1 8
left: [4, 5]
right: [9]
pivot: 1 5
left: [4]
right: []
pivot: 0 4
left: []
right: []
pivot: 0 9
left: []
right: []
sorted list: 1 2 4 5 8 9

```