Сортировки

Формулировка задачи. Устойчивость

Сортировка – процесс упорядочивания элементов массива.

Устойчивость (stability) – устойчивая сортировка не меняет взаимного расположения равных элементов.

Квадратичные сортировки: сортировка выбором

- Во время работы алгоритма:

• Массив разделен на 2 части: левая – готова, правая – нет.

- На одном шаге:

1) ищем минимум в правой части,

2) меняем его с первым элементом правой части,

3) сдвигаем границу разделения на 1 вправо.

Все еще сортировка выбором

```
function selectionSort(T[n] a):
  for i = 0 to n - 2
    for j = i + 1 to n - 1
      if a[i] > a[j]
        swap(a[i], a[j])
```

- O(n^2) swap

```
function selectionSort(T[n] a):
   for i = 0 to n - 2
     min = i
     for j = i + 1 to n - 1
       if a[j] < a[min]
         min = j
     swap(a[i], a[min])
```

- O(n) swap

Устойчивая?

Квадратичные сортировки: сортировка вставками

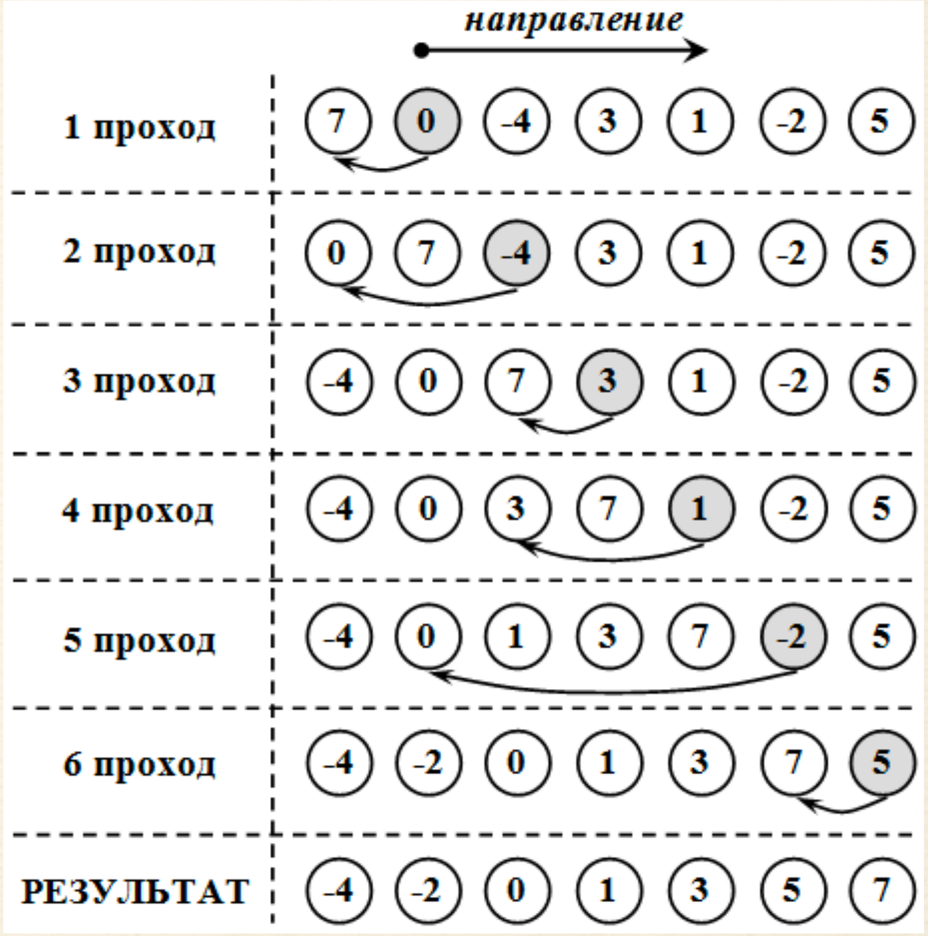
- Во время работы алгоритма:

• Массив разделен на 2 части: левая – готова, правая – нет.

- На одном шаге:

1) берем первый элемент правой части,

2) вставляем его на подходящее место в левой части.



Все еще сортировка вставками

```
function insertionSort(a):
  for i = 1 to n - 1
    j = i - 1
    while j ⩾ 0 and a[j] > a[j + 1]
      swap(a[j], a[j + 1])
      j--
```

- O(n^2) сравнений
- O(n^2) swap

```
function insertionSort(a):
  for i = 1 to n - 1
    j = i - 1
    k = binSearch(a, a[i], 0, j)
    for m = j downto k
      swap(a[m], a[m+1])
```

- O(nlogn) сравнений
- O(n^2) swap

Устойчивая?

Сортировка слиянием.

1) Если в рассматриваемом массиве один элемент, то он уже отсортирован — алгоритм завершает работу.

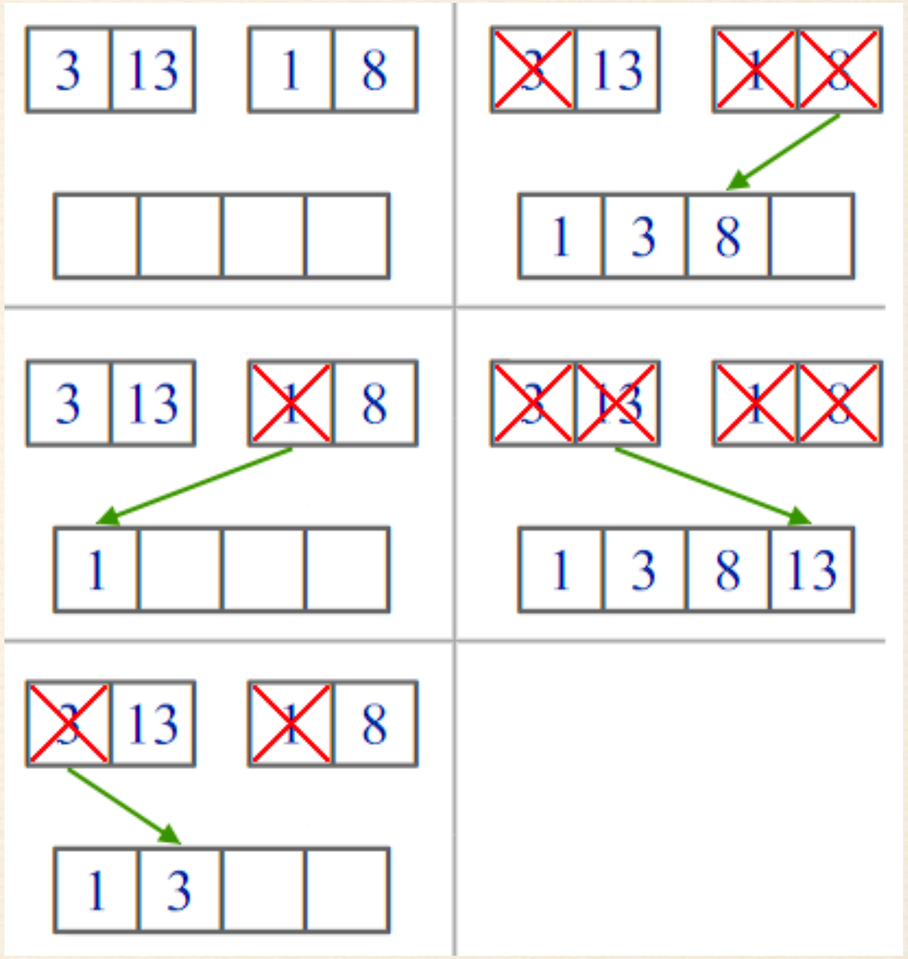
2) Иначе массив разбивается на две части, которые сортируются рекурсивно.

3) После сортировки двух частей массива к ним применяется процедура слияния, которая по двум отсортированным частям получает исходный отсортированный массив.

Слияние

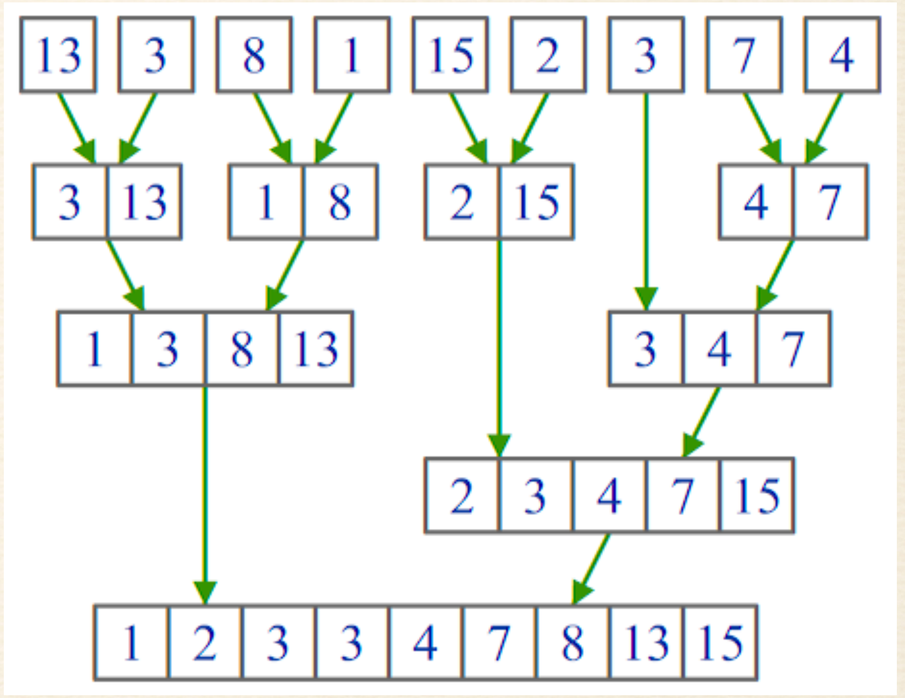
Эта процедура заключается в том, что мы сравниваем элементы массивов (начиная с начала) и меньший из них записываем в финальный. И затем, в массиве у которого оказался меньший элемент, переходим к следующему элементу и сравниваем теперь его. В конце, если один из массивов закончился, мы просто дописываем в финальный другой массив. После мы наш финальный массив записываем заместо двух исходных и получаем отсортированный участок.

```
function merge(a : int[n]; left, mid, right : int):
    it1 = 0
    it2 = 0
    result : int[right - left]

    while left + it1 < mid and mid + it2 < right
        if a[left + it1] < a[mid + it2]
            result[it1 + it2] = a[left + it1]
            it1 += 1
        else
            result[it1 + it2] = a[mid + it2]
            it2 += 1

    while left + it1 < mid
        result[it1 + it2] = a[left + it1]
        it1 += 1

    while mid + it2 < right
        result[it1 + it2] = a[mid + it2]
        it2 += 1

    for i = 0 to it1 + it2
        a[left + i] = result[i]
```



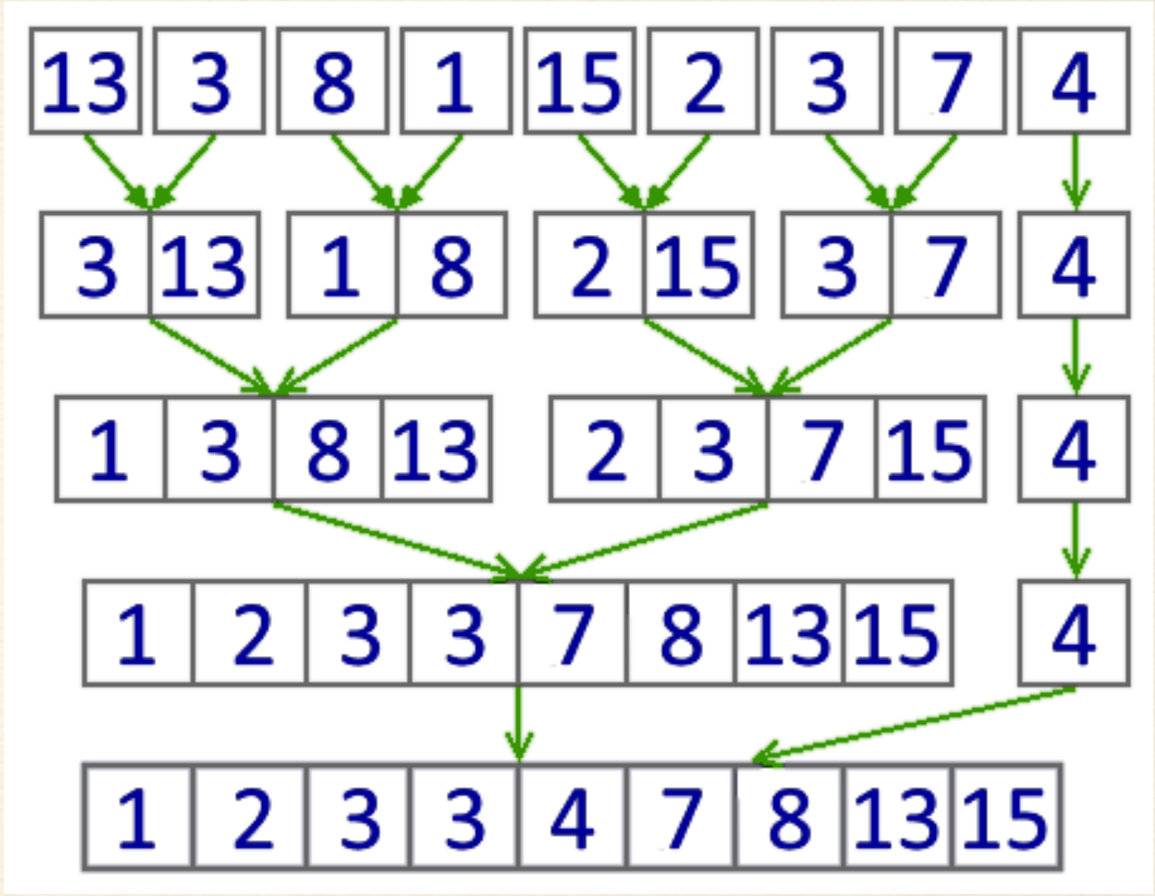
Сортировка слиянием. Рекурсивный алгоритм

```
function mergeSortRecursive(a : int[n]; left, right : int):
    if left + 1 >= right
        return
    mid = (left + right) / 2
    mergeSortRecursive(a, left, mid)
    mergeSortRecursive(a, mid, right)
    merge(a, left, mid, right)
```



Сортировка слиянием. Итеративный алгоритм

```
function mergeSortIterative(a : int[n]):
    for i = 1 to n, i *= 2
        for j = 0 to n - i, j += 2 * i
            merge(a, j, j + i, min(j + 2 * i, n))
```



Время работы

Чтобы оценить время работы этого алгоритма, составим рекуррентное соотношение. Пускай $T(n)$ — время сортировки массива длины n, тогда для сортировки слиянием справедливо $T(n) = 2T(n/2) + O(n)$

$O(n)$ — время, необходимое на то, чтобы слить два массива длины n. Распишем это соотношение:

$$T(n) = 2T(n/2) + O(n) = 4T(n/4) + 2O(n) = \cdots = T(1) + \log(n)O(n) = O(n\log(n)).$$

Устойчивая?