education

Фундаментальные задачи по массивам

Семинар

Илья Почуев

Старший backend-разработчик,
Вконтакте

education

Цель занятия:

Освоить ключевые алгоритмы для эффективной обработки массивов.

Научиться применять различные подходы к решению типовых задач с массивами, фокусируясь на оптимизации по времени и памяти.

Развить навыки "in-place" модификации массивов и работы без дополнительных аллокаций.

Подготовиться к решению алгоритмических задач, связанных
с массивами, на интервью и в реальной разработке.

Сумма всех чисел

1 1000000

Two sum

- Дан отсортированный по возрастанию массив целых чисел и некоторое число target.

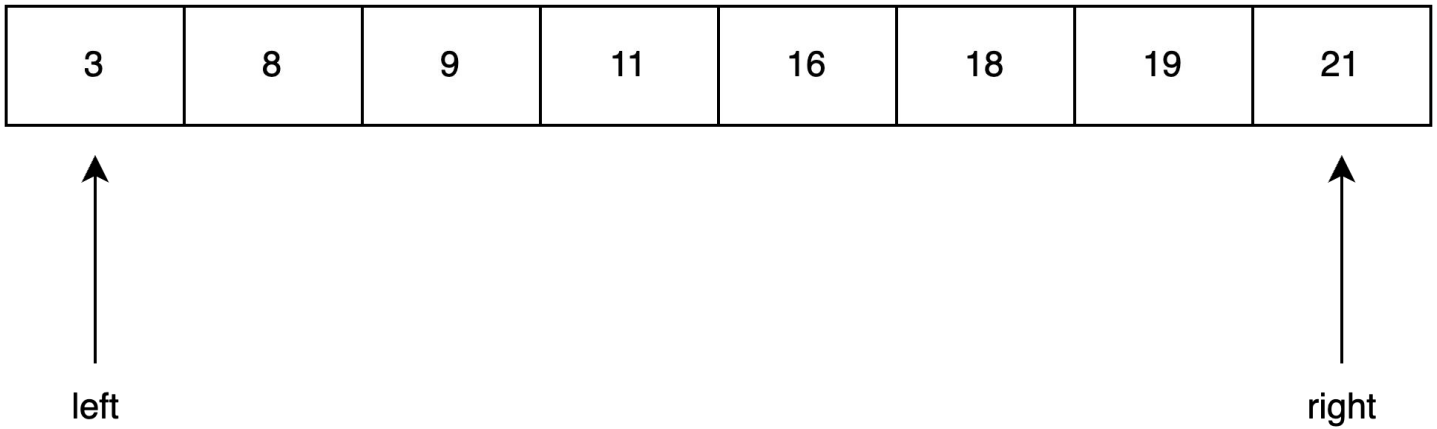
- Необходимо найти два числа в массиве, которые в сумме дают заданное значение target, и вернуть их индексы.

Two sum

target = 25

Two sum

target = 25



Two sum

```
function twoSum(nums, target) {
    left = 0;
    right = len(nums) - 1;

    while (left < right) {
        sum = nums[left] + nums[right];
        ...
    }

    return [];
}
```

Two sum

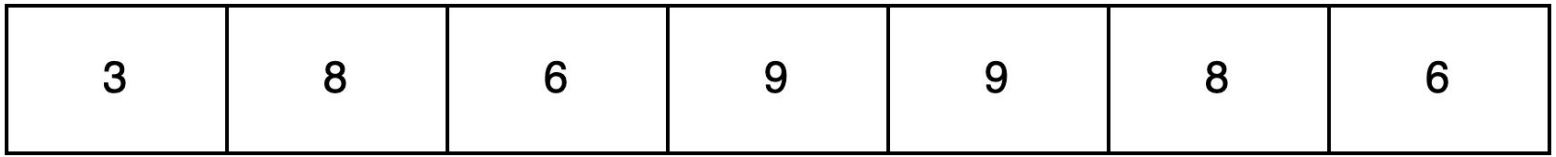
- Применяем метод двух указателей
- Суммируем значения по левым и правым индексами
- Если полученная сумма меньше исходного значения, то двигаем левый указатель, так как необходимо увеличить сумму. В противном случае двигаем правый

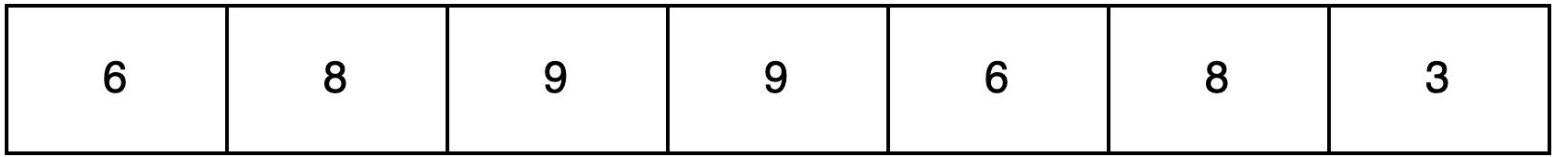
```
function twoSum(nums, target) {
    left = 0;
    right = len(nums) - 1;

    while (left < right) {
        sum = nums[left] + nums[right];

        if (sum == target) {
            return [left, right];
        } else if (sum < target) {
            left++;
        } else {
            right--;
        }
    }

    return [];
}
```

Развернуть массив

- Дан массив целых чисел.
- Необходимо развернуть его.
- Сделать это надо за линейное время без дополнительных аллокаций.

Развернуть массив





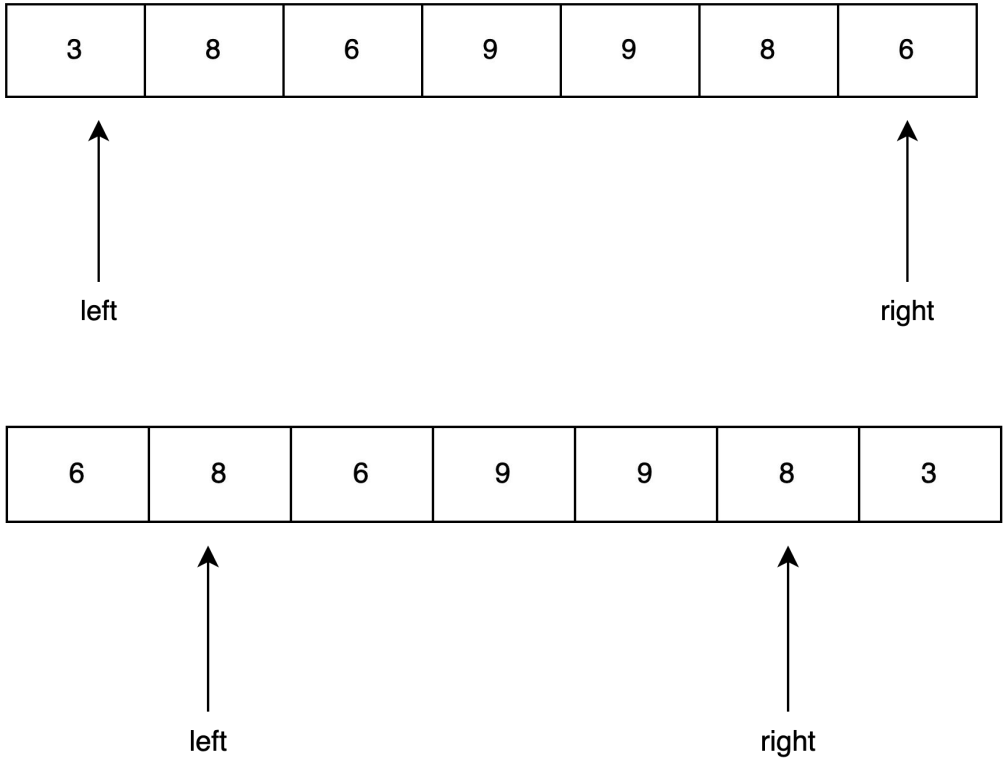
Развернуть массив

- Дан массив целых чисел.
- Необходимо развернуть его.
- Сделать это надо за линейное время без дополнительных аллокаций.

```
function reverseArray(arr) {


  return arr

}
```

Развернуть массив

- Дан массив целых чисел.
- Необходимо развернуть его.
- Сделать это надо за линейное время без дополнительных аллокаций.



Развернуть массив

- Дан массив целых чисел.
- Необходимо развернуть его.
- Сделать это надо за линейное время без дополнительных аллокаций.

```
function reverseArray(arr) {
  left = 0;
  right = len(arr) - 1;

  return arr
}
```

Развернуть массив

- Дан массив целых чисел.
- Необходимо развернуть его.
- Сделать это надо за линейное время без дополнительных аллокаций.

```
function reverseArray(arr) {
  left = 0;
  right = len(arr) - 1;

  while (left < right) {
    swap(arr[left], arr[right]);
    //arr[left], arr[right] = arr[right], arr[left]
    left++;
    right--;
  }

  return arr
}
```

Развернуть часть массива

- Дан массив целых чисел.
- Необходимо повернуть (сдвинуть) справа налево часть массива, которая указана вторым параметром.
- Сделать это надо за линейное время без дополнительных аллокаций
- Исходный массив: 1, 2, 3, 4, 5, 6, 7
- k = 3
- Результат: 5, 6, 7, 1, 2, 3, 4

Развернуть часть массива

- Дан массив целых чисел.
- Необходимо развернуть справа налево часть массива, которая указана вторым параметром.
- Сделать это надо за линейное время без дополнительных аллокаций

https://leetcode.com/problems/rotate-array/description/

```
function reverseArray(arr, left, right) {
  while (left < right) {
    swap(arr[left], arr[right])
    left++
    right--
  }
}

function solution(arr, k) {
  n = len(arr)


  return arr
}
```

Развернуть часть массива

- Дан массив целых чисел.
- Необходимо развернуть справа налево часть массива, которая указана вторым параметром.
- Сделать это надо за линейное время
  без дополнительных аллокаций

1, 2, 3, 4, 5, 6, 7

7, 6, 5, 4, 3, 2, 1

5, 6, 7, 4, 3, 2, 1

5, 6, 7, 1, 2, 3, 4

```
function reverseArray(arr, left, right) {
  while (left < right) {
    swap(arr[left], arr[right])
    left++
    right--
  }
}

function solution(arr, k) {
  n = len(arr)

  return arr
}
```

Развернуть часть массива

- Дан массив целых чисел.
- Необходимо развернуть справа налево часть массива, которая указана вторым параметром.
- Сделать это надо за линейное время без дополнительных аллокаций

1, 2, 3, 4, 5, 6, 7

5, 6, 7, 4, 3, 2, 1

// разворачиваем первые k чисел

5, 6, 7

// разворачиваем оставшийся массив

1, 2, 3, 4, 5

Если k > n?

```
function reverseArray(arr, left, right) {
  while (left < right) {
    swap(arr[left], arr[right])
    left++
    right--
  }
}

function solution(arr, k) {
  n = len(arr)

  reverseArray(arr, 0, n - 1) //7, 6, 5, 4, 3, 2, 1
  ...

  return arr
}
```

Развернуть часть массива

- Дан массив целых чисел.
- Необходимо развернуть справа налево часть массива, которая указана вторым параметром.
- Сделать это надо за линейное время без дополнительных аллокаций

1, 2, 3, 4, 5, 6, 7

5, 6, 7, 4, 3, 2, 1

// разворачиваем первые k чисел

5, 6, 7

// разворачиваем оставшийся массив

1, 2, 3, 4, 5

```
function reverseArray(arr, left, right) {
  while (left < right) {
    swap(arr[left], arr[right])
    left++
    right--
  }
}

function solution(arr, k) {
  n = len(arr)

  reverseArray(arr, 0, n - 1)
  reverseArray(arr, 0, k - 1)
  reverseArray(arr, k, n - 1)

  return arr
}
```

Развернуть часть массива

Если k > n?

```
function reverseArray(arr, left, right) {
  while (left < right) {
    swap(arr[left], arr[right])
    left++
    right--
  }
}

function solution(arr, k) {
  n = len(arr)

  reverseArray(arr, 0, n - 1)
  reverseArray(arr, 0, k - 1)
  reverseArray(arr, k, n - 1)

  return arr;
}
```

Развернуть часть массива

Если k > n?

```
function reverseArray(arr, left, right) {
  while (left < right) {
    swap(arr[left], arr[right])
    left++
    right--
  }
}

function solution(arr, k) {
  n = len(arr)

  reverseArray(arr, 0, n - 1)
  reverseArray(arr, 0, k mod n - 1)
  reverseArray(arr, k mod n, n - 1)

  return arr;
}
```

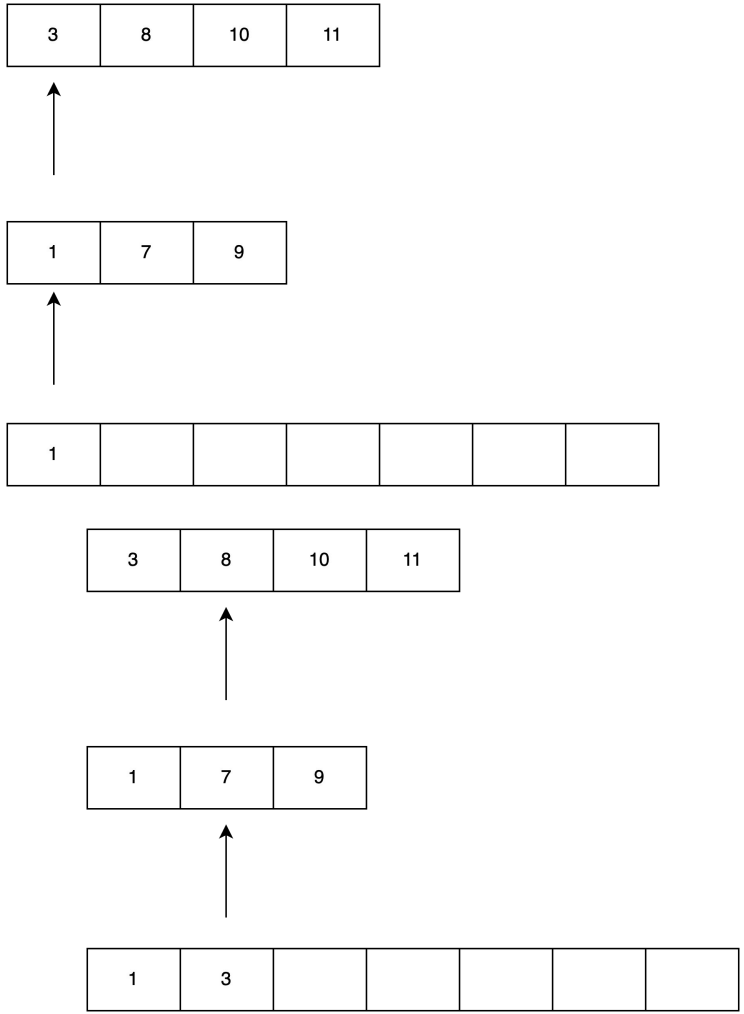
Слияние двух отсортированных массивов

- Дано два отсортированных массива.
- Необходимо написать функцию которая объединит эти два массива в один отсортированный.

Слияние двух отсортированных массивов

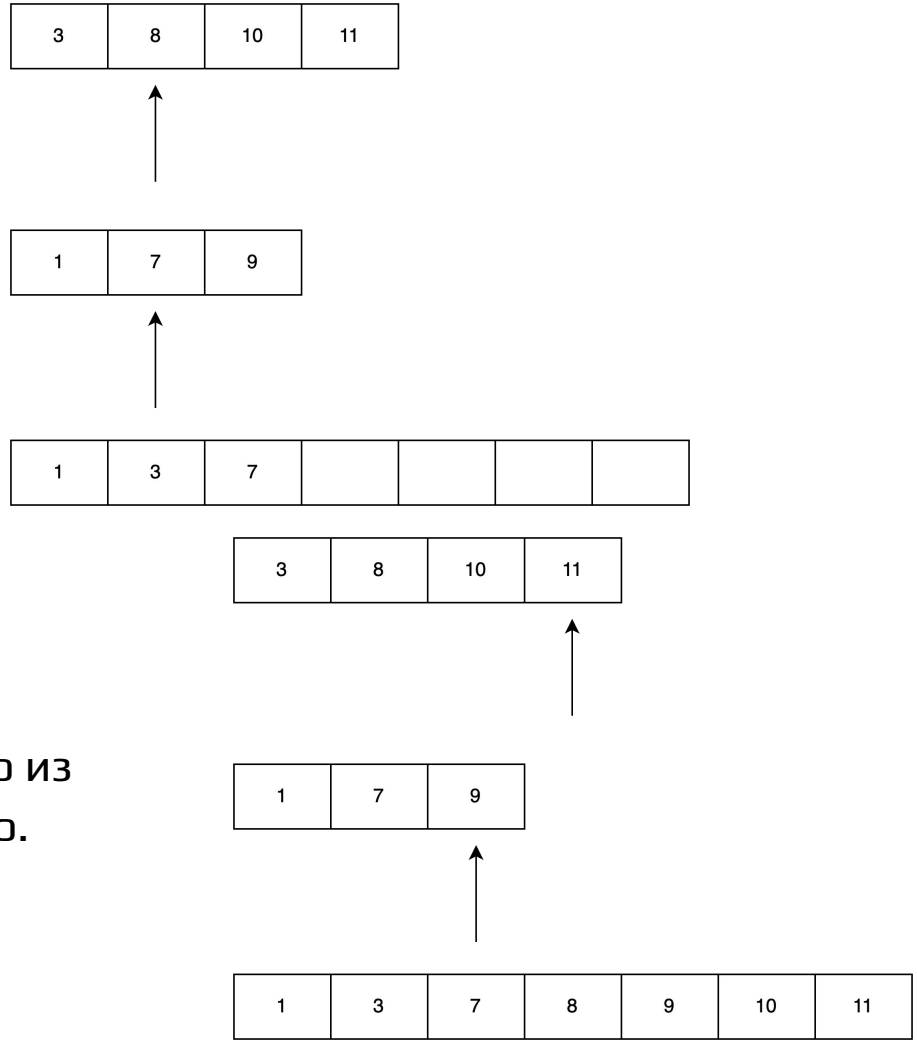
Слияние двух отсортированных массивов

- Создаем два указателя i и j.
- В цикле сравниваем значения под этими индексами.
- В зависимости от результата сравнения, в результирующий массив записываем элемент либо из первого массива, либо из второго.



Слияние двух отсортированных массивов

- Создаем два указателя i и j.
- В цикле сравниваем значения под этими индексами.
- В зависимости от результата сравнения, в результирующий массив записываем элемент либо из первого массива, либо из второго.



Слияние двух отсортированных массивов

```
function merge_sorted_arrays(arr1, arr2) {
  merged_array = []

  return merged_array
}
```

Слияние двух отсортированных массивов

- Создаем два указателя i и j.
- В цикле сравниваем значения под этими индексами.
- В зависимости от результата сравнения, в результирующий массив записываем элемент либо из первого массива, либо из второго.

```
function merge_sorted_arrays(arr1, arr2) {
  merged_array = []
  i = 0; j = 0

  while i < len(arr1) and j < len(arr2) {

  }

   return merged_array
}
```

Слияние двух отсортированных массивов

- Создаем два указателя i и j.
- В цикле сравниваем значения под этими индексами.
- В зависимости от результата сравнения, в результирующий массив записываем элемент либо из первого массива, либо из второго.

```
function merge_sorted_arrays(arr1, arr2) {
  merged_array = []
  i = 0; j = 0

  while i < len(arr1) and j < len(arr2) {
    if arr1[i] < arr2[j] {
      merged_array.append(arr1[i])
      i++
    } else {
      merged_array.append(arr2[j])
      j++
    }
  }

  return merged_array
}
```

Слияние двух отсортированных массивов

- Создаем два указателя i и j.
- В цикле сравниваем значения под этими индексами.
- В зависимости от результата сравнения, в результирующий массив записываем элемент либо из первого массива, либо из второго.

```
function merge_sorted_arrays(arr1, arr2) {
  merged_array = []
  i = 0; j = 0

  while i < len(arr1) and j < len(arr2) {
    if arr1[i] < arr2[j] {
      merged_array.append(arr1[i])
      i++
    } else {
      merged_array.append(arr2[j])
      j++
    }
  }

  // Если в каком-то из массивов остались
  // элементы, добавляем их в конец слияния
  merged_array.append(arr1[i:])
  merged_array.append(arr2[j:])

  return merged_array
}
```

Какой недостаток?

Какой недостаток?

Сложность по памяти

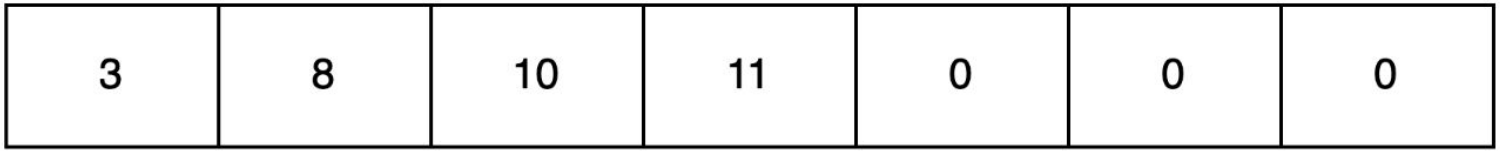
Слияние двух отсортированных массивов. Без дополнительных аллокаций

- Дано два отсортированных массива. Необходимо написать функцию которая объединит эти два массива в один отсортированный.

- Первый массив имеет размер, равный результирующему массиву, значения в конце равны нулям. Мы не должны создавать третий массив.

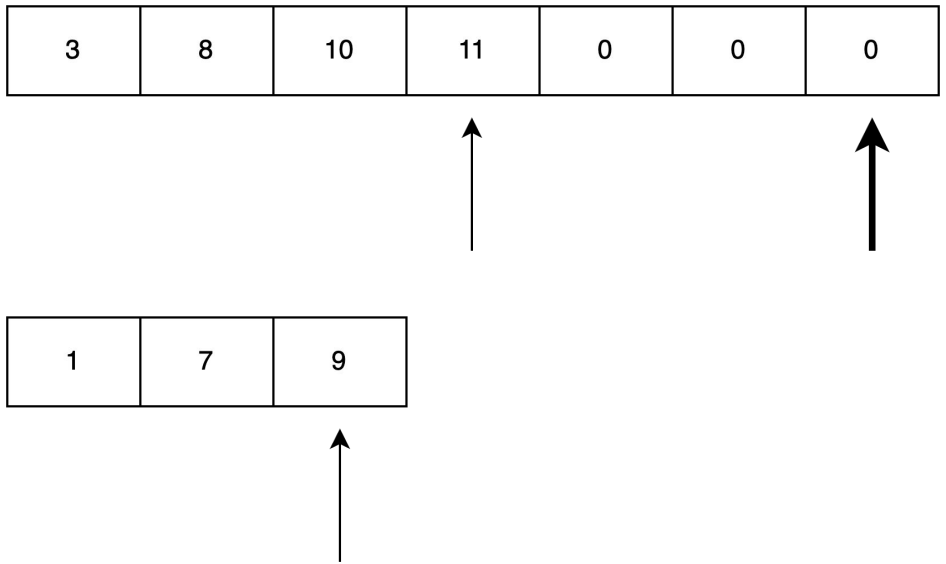
https://leetcode.com/problems/merge-sorted-array/description/

Слияние двух отсортированных массивов



Слияние двух отсортированных массивов

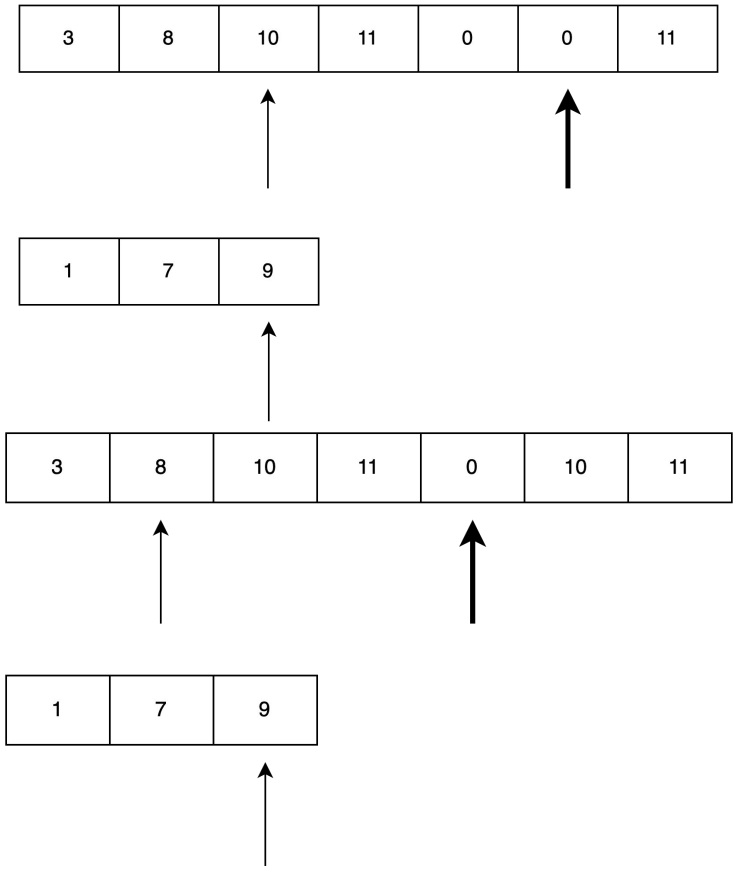
- Указатели будут идти не слева направо, а справа на лево.
- Теперь у нас появится третий указатель, в который мы будем писать результирующие значения.



Слияние двух отсортированных массивов

- Указатели будут идти не слева направо, а справа на лево.

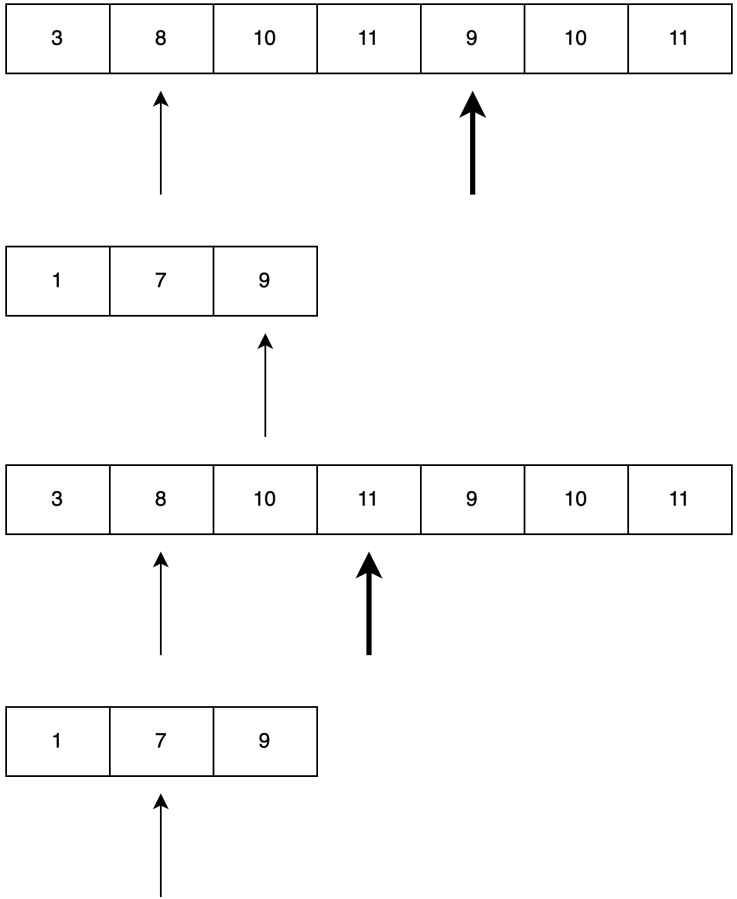
- Теперь у нас появится третий указатель, в который мы будем писать результирующие значения.



Слияние двух отсортированных массивов

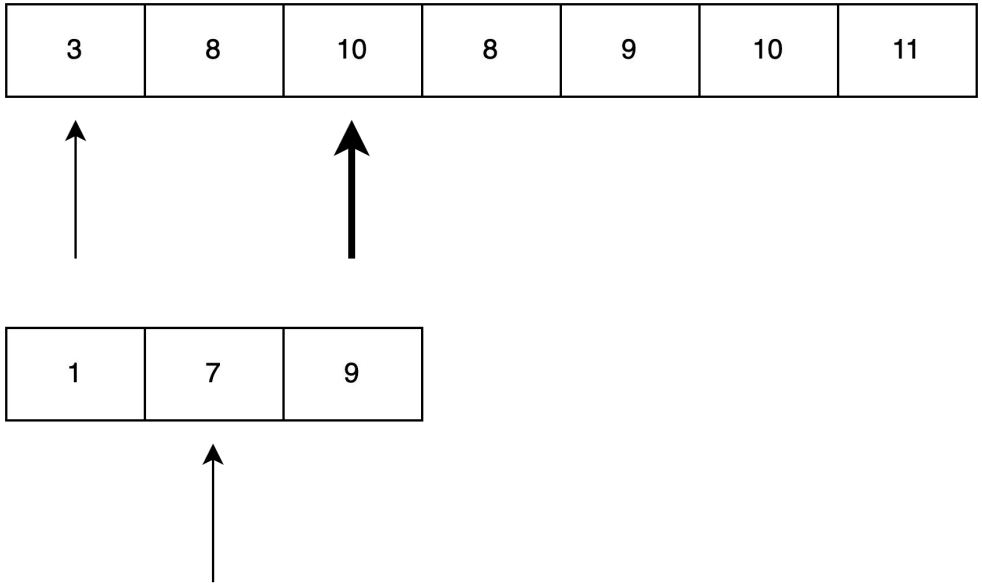
- Указатели будут идти не слева направо, а справа на лево.

- Теперь у нас появится третий указатель, в который мы будем писать результирующие значения.

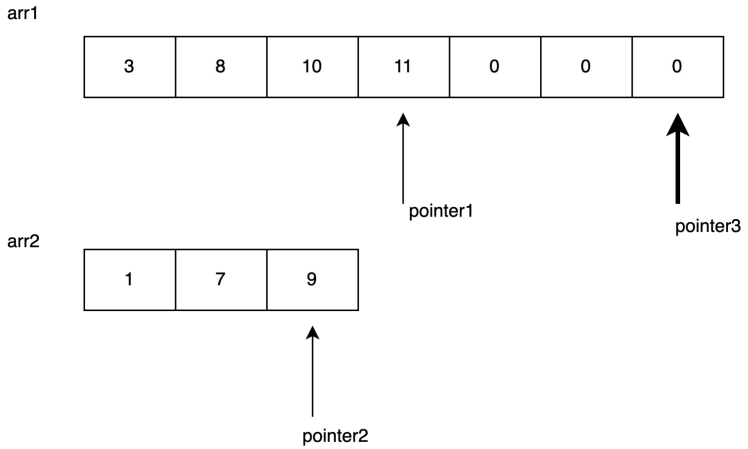


Слияние двух отсортированных массивов

- Указатели будут идти не слева направо, а справа на лево.
- Теперь у нас появится третий указатель, в который мы будем писать результирующие значения.

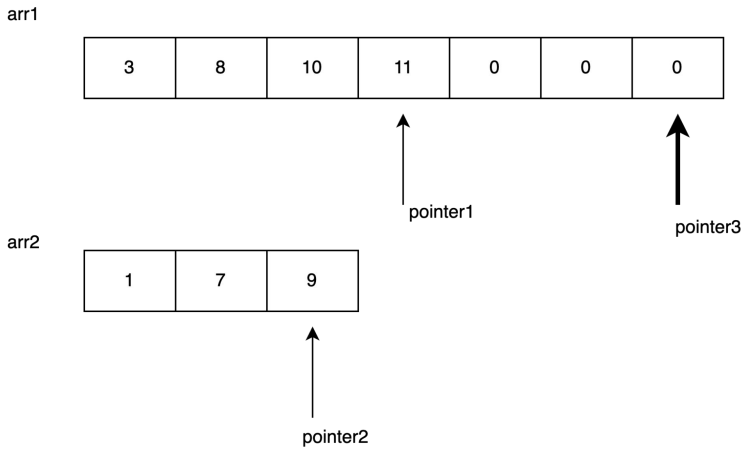


Слияние двух отсортированных массивов



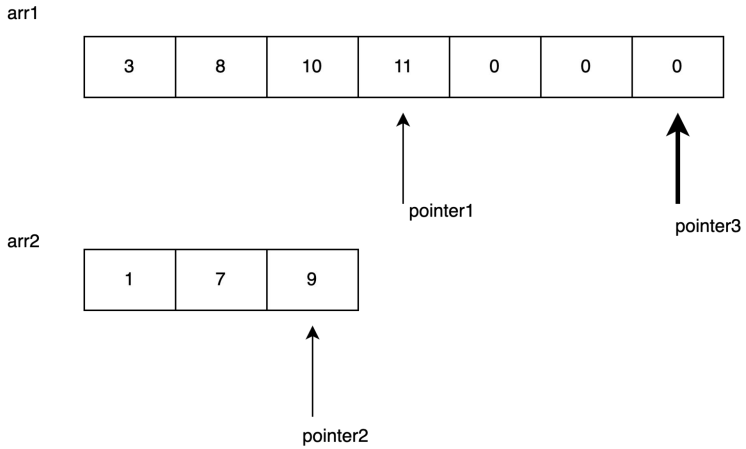

```
function merge(arr1, arr2) {


        return arr1

}
```

Слияние двух отсортированных массивов



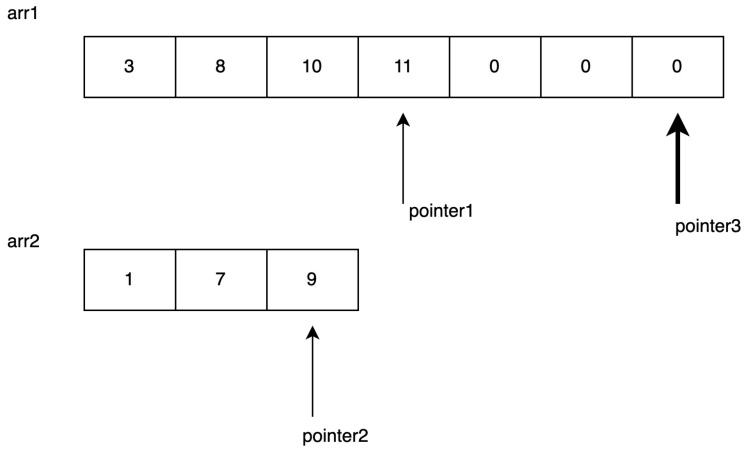

```
function merge(arr1, arr2) {
        pointer1 = len(arr1) - len(arr2) - 1
        pointer2 = len(arr2) - 1
        pointer3 = len(arr1) - 1



        return arr1
}
```

Слияние двух отсортированных массивов




```
function merge(arr1, arr2) {
        pointer1 = len(arr1) - len(arr2) - 1
        pointer2 = len(arr2) - 1
        pointer3 = len(arr1) - 1

        while pointer2 >= 0 {

        }

        return arr1
}
```

Слияние двух отсортированных массивов




```
function merge(arr1, arr2) {
  pointer1 = len(arr1) - len(arr2) - 1
  pointer2 = len(arr2) - 1
  pointer3 = len(arr1) - 1

  while pointer2 >= 0 {
    if pointer1 >= 0
     and arr1[pointer1] > arr2[pointer2] {
      arr1[pointer3] = arr1[pointer1]
      pointer1--
    } else {
      arr1[pointer3] = arr2[pointer2]
      pointer2--
    }
    pointer3--
  }

  return arr1
}
```

Сортировка массива из 0 и 1

- Дан массив, содержащий только 0 и 1.
- Напишите функцию, которая сортирует его так, чтобы все нули оказались в начале, а все единички — в конце.

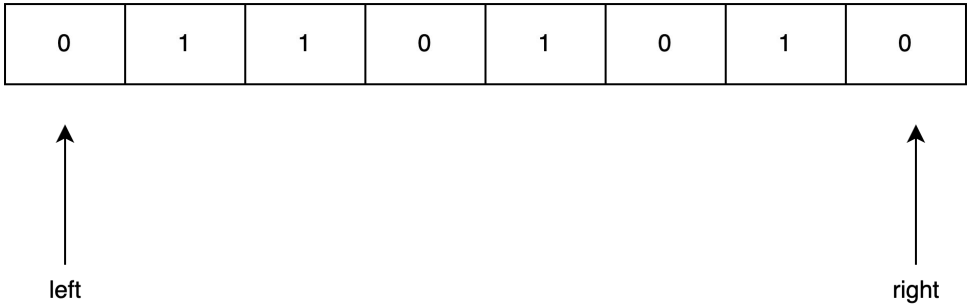
Решение должно быть in-place.

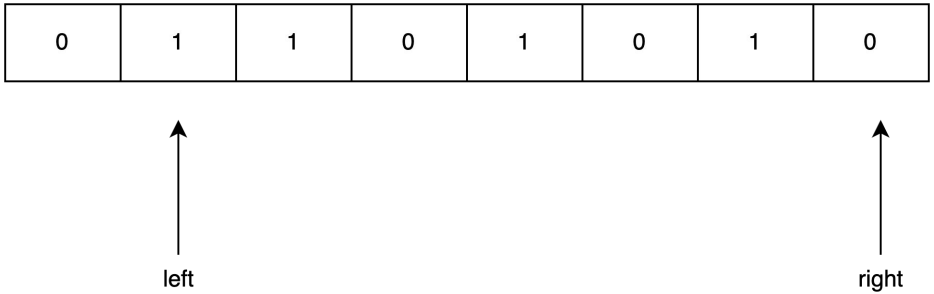
Сортировка массива из 0 и 1

0	1	1	0	1	0	1	0

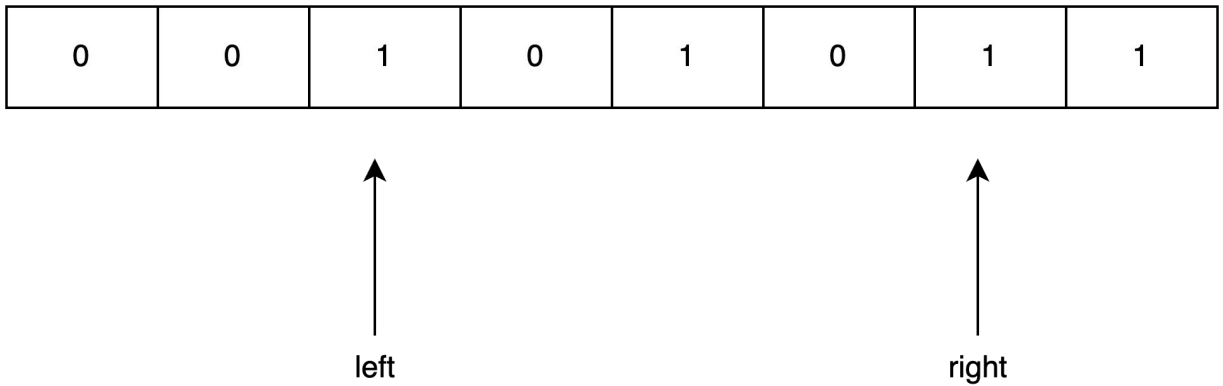
0	0	0	0	1	1	1	1

Сортировка массива из 0 и 1

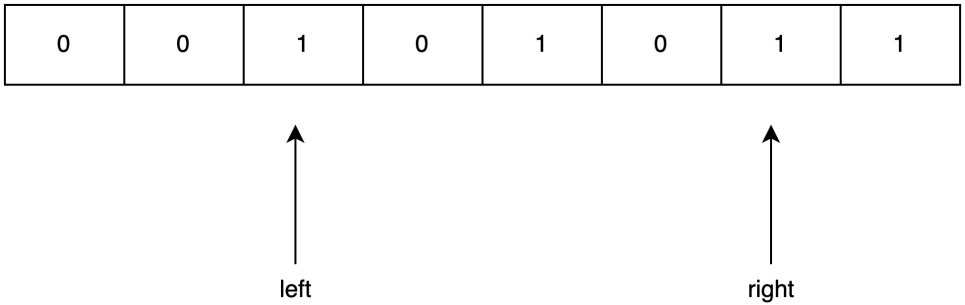


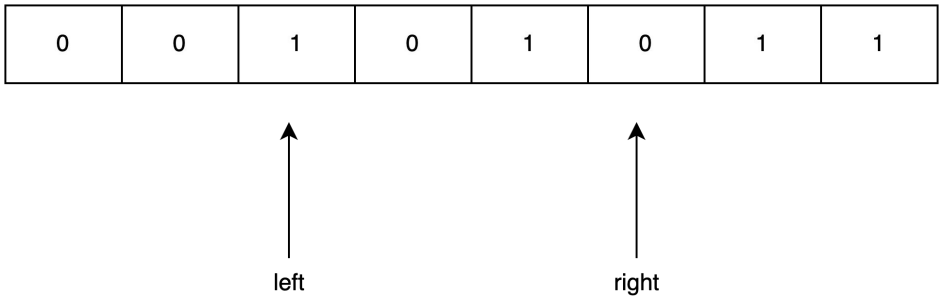


Сортировка массива из 0 и 1



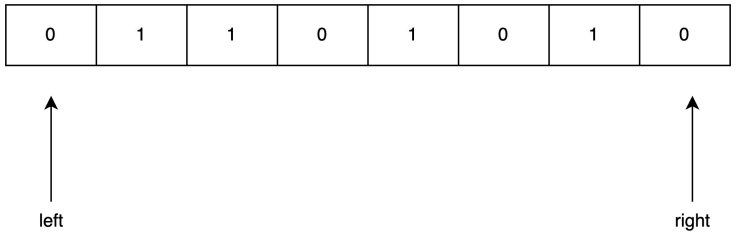
Сортировка массива из 0 и 1





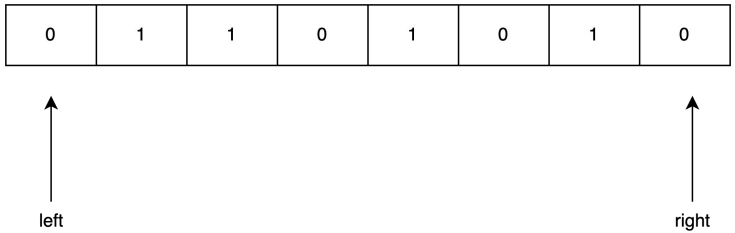
Сортировка массива из 0 и 1

Сортировка массива из 0 и 1



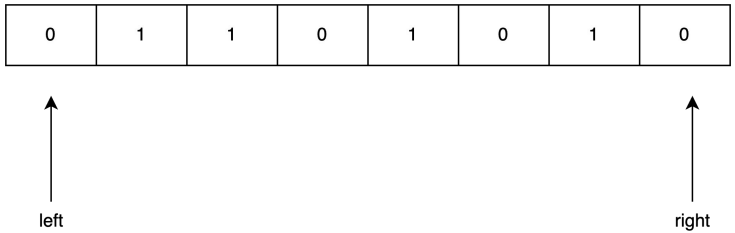

```
function sort_binary_array(arr) {
    left = 0  // Указатель для нулей
    right = len(arr) - 1  // Указатель для единиц

    while ??? {

    }

    return arr
}
```

Сортировка массива из 0 и 1



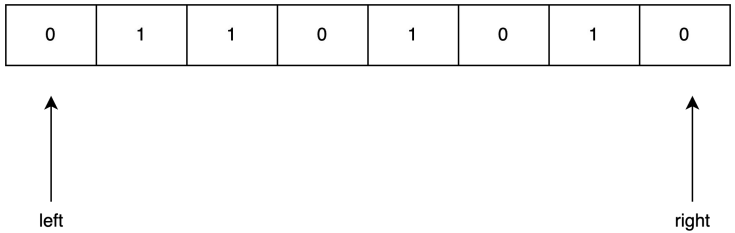

```
function sort_binary_array(arr) {
    left = 0
    right = len(arr) - 1

    while left < right {
        ....
    }

    return arr
}
```

Сортировка массива из 0 и 1



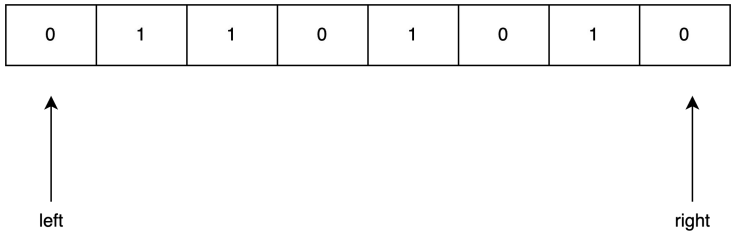

```
def sort_binary_array(arr) {
    left = 0
    right = len(arr) - 1

    while left < right:
        // если текущий элемент слева равен 0,
        // двигаем левый указатель вправо
        if arr[left] == 0 {
            left++
        }

    return arr
}
```

Сортировка массива из 0 и 1




```
function sort_binary_array(arr) {
    left = 0
    right = len(arr) - 1

    while left < right {
      if arr[left] == 0 {
          left++
       } elif arr[right] == 1 {
          // если текущий элемент справа равен 1
          right--
       } else {
          // если arr[left] == 1 и arr[right] == 0, меняем их
          // местами
    }
    return arr
}
```

Сортировка массива из 0 и 1




```
function sort_binary_array(arr) {
    left = 0
    right = len(arr) - 1

    while left < right {
      if arr[left] == 0 {
          left++
       } elif arr[right] == 1 {
          right-- // если текущий элемент справа равен 1
       } else {
         // если arr[left] == 1 и arr[right] == 0, меняем их
         // местами
          arr[left], arr[right] = arr[right], arr[left]
          left++
          right--
       }
    }
    return arr
}
```

```
function sort_binary_array(arr) {
    left, right = 0, len(arr) - 1
    while left < right {
        if arr[left] == 1{
            arr[left], arr[right] = arr[right], arr[left]
            right--
        } else {
            left++
        }
    }
    return arr
}
```

Задача флага Нидерландов

- Дан массив состоящий из нулей, единиц и двоек
- Необходимо его отсортировать за линейное время

https://leetcode.com/problems/sort-colors/description/

Задача флага Нидерландов

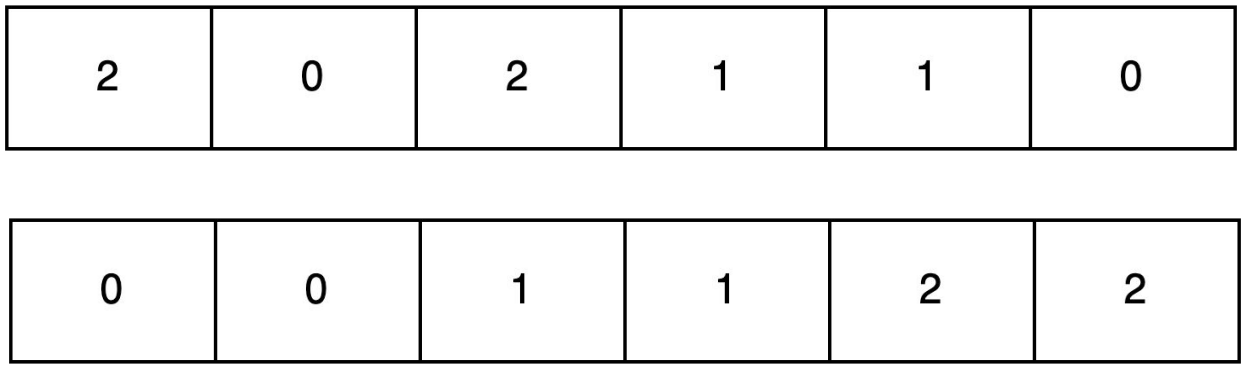
2	0	2	1	1	0

0	0	1	1	2	2

Задача флага Нидерландов

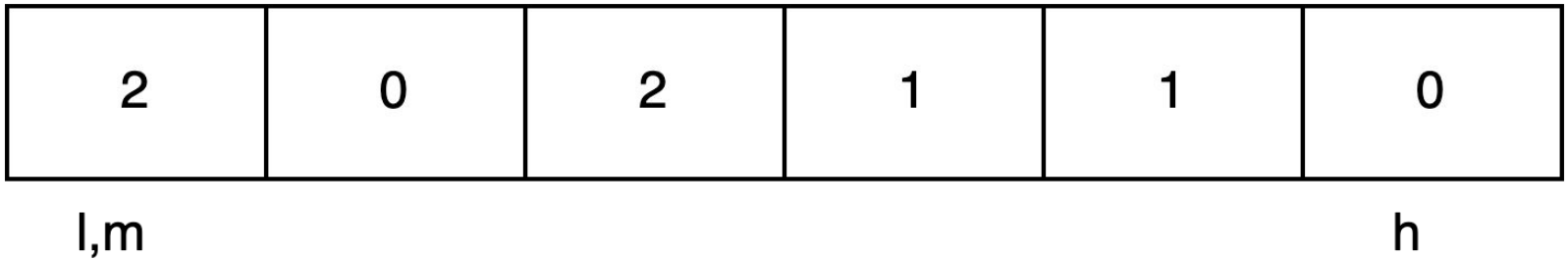
Три указателя:

- low — указывает на позицию, где должен находиться следующий 0.
- mid — текущий элемент, который мы проверяем.
- high — указывает на позицию, где должен находиться следующий 2



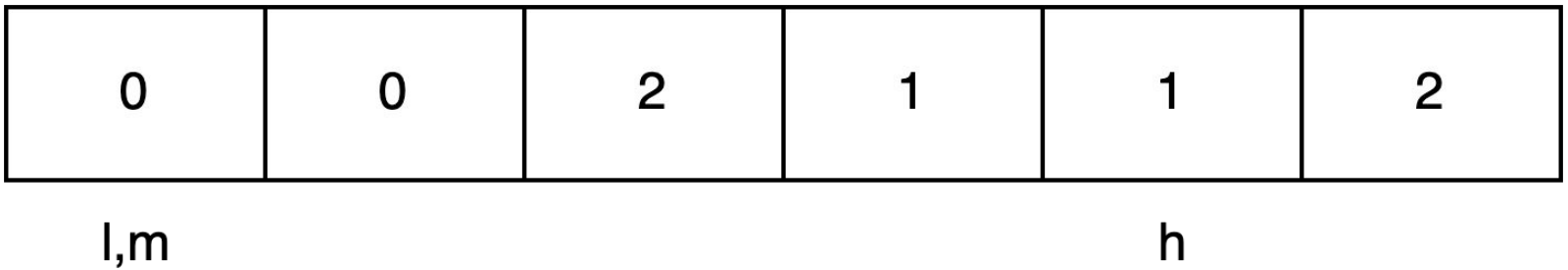
Задача флага Нидерландов

- Если значение под индексом m равно 2, то меняем его со значением под индексом h
- arr[m] == 2 -> Меняем arr[m] и arr[h] (2 <- -> 0)
- Уменьшаем high



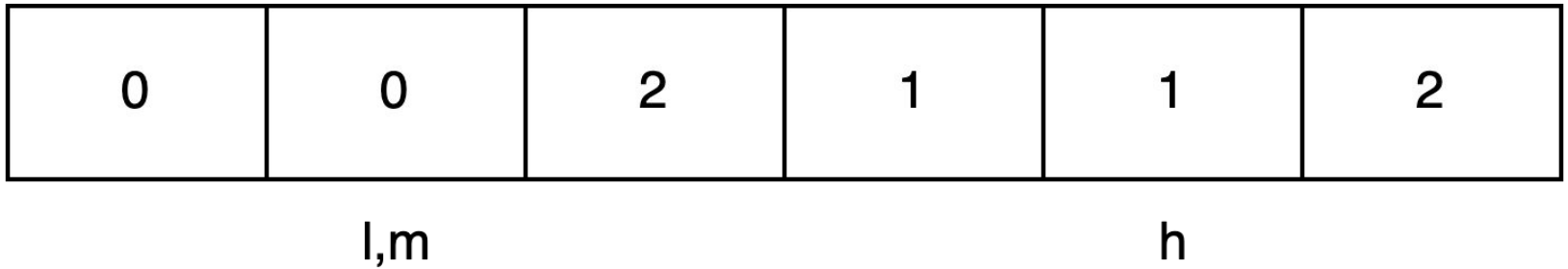
Задача флага Нидерландов

- Если текущий элемент 0, меняем его с элементом на позиции low
- arr[m] == 0 => Меняем arr[m] и arr[l] (0 <- -> 0)
- Увеличиваем low и mid



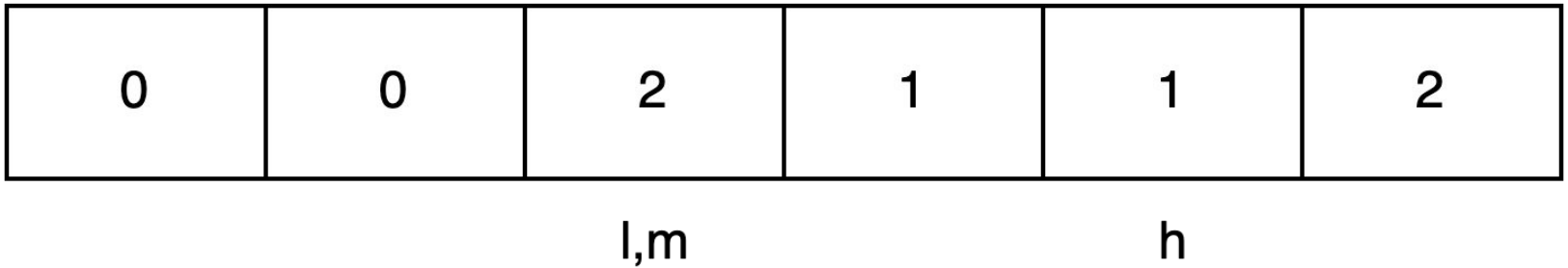
Задача флага Нидерландов

- Если текущий элемент 0, меняем его с элементом на позиции low
- arr[m] == 0 -> Меняем arr[m] и arr[l] (0 <- -> 0)
- Увеличиваем low и mid



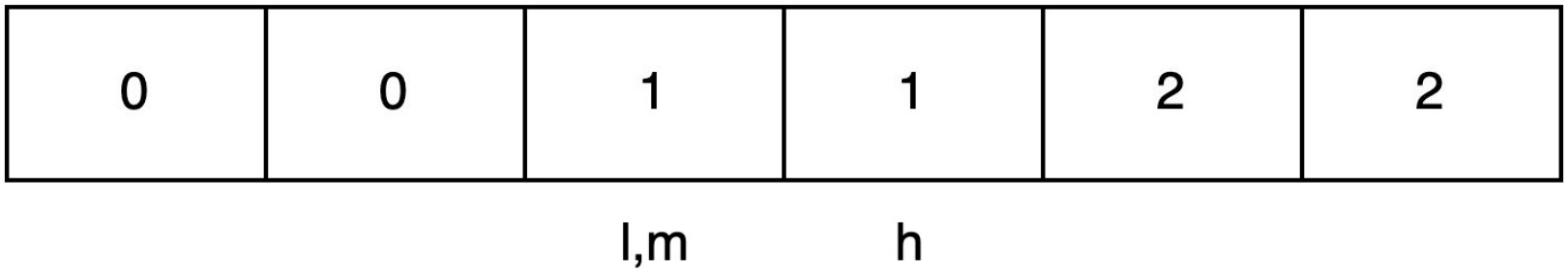
Задача флага Нидерландов

- Если значение под индексом m равно 2, то меняем его со значением под индексом h
- arr[m] == 2 => Меняем arr[m] и arr[h] (2 <- -> 1)
- Уменьшаем high



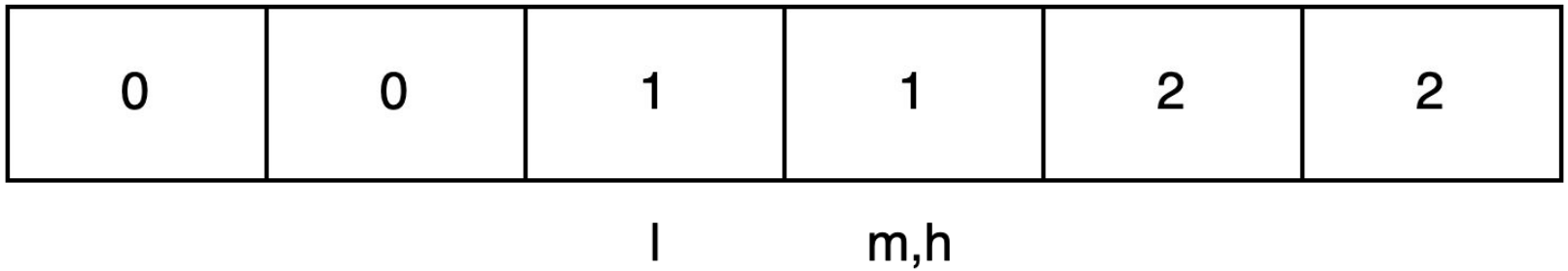
Задача флага Нидерландов

- Если значение под индексом m равно 1, то просто инкрементируем m
- arr[m] == 1 => ничего не меняем
- m++



Задача флага Нидерландов

➔ Продолжаем цикл пока выполняется условие m <= h



```
function sortColors(nums) {
    low = 0;
    mid = 0;
    high = len(nums) - 1;

    while (mid <= high) {
        ...
    }
}
```

```
function sortColors(nums) {
    low = 0;
    mid = 0;
    high = len(nums) - 1;

    while (mid <= high) {
        if (nums[mid] == 0) {
            nums[low], nums[mid] = nums[mid], nums[low];
            low++;
            mid++;
        }
        ...
    }
}
```

```
function sortColors(nums) {
    low = 0;
    mid = 0;
    high = len(nums) - 1;

    while (mid <= high) {
        if (nums[mid] == 0) {
            nums[low], nums[mid] = nums[mid], nums[low];
            low++;
            mid++;
        } else if (nums[mid] == 1) {
            mid++;
        } else if (nums[mid] == 2) {
            nums[mid], nums[high] = nums[high], nums[mid];
            high--;
        }
    }
}
```

Передвинуть четные числа вперед

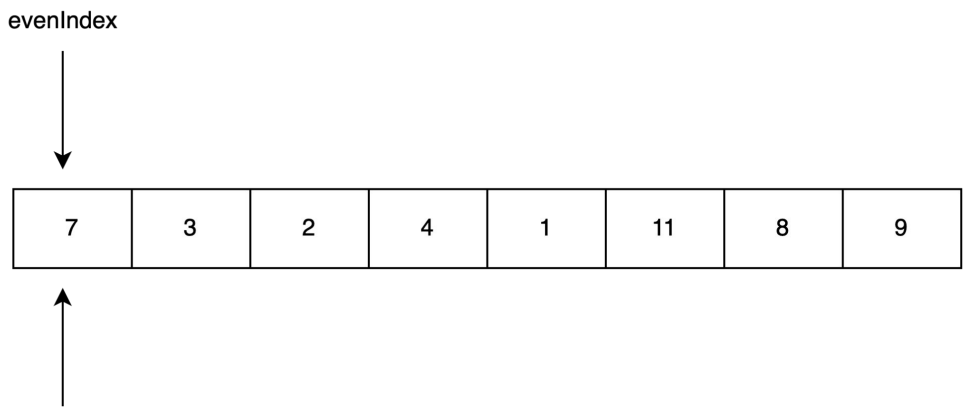
- Дан не отсортированный массив целых чисел
- Необходимо перенести в начало массива все четные числа, сохраняя их очередность
- То есть если 8 стояла после 2, то после переноса в начало, он по-прежнему должна стоять после 2

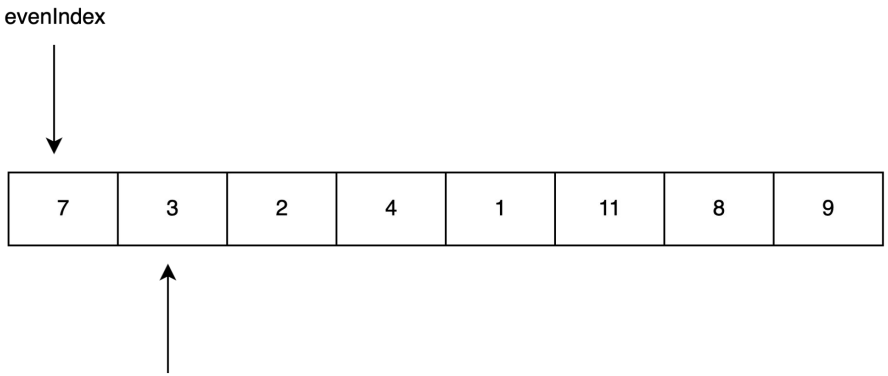
Пример ввода: [3, 2, 4, 1, 11, 8, 9]

Пример вывода: 2 4 8 1 11 3 9

Передвинуть четные числа вперед

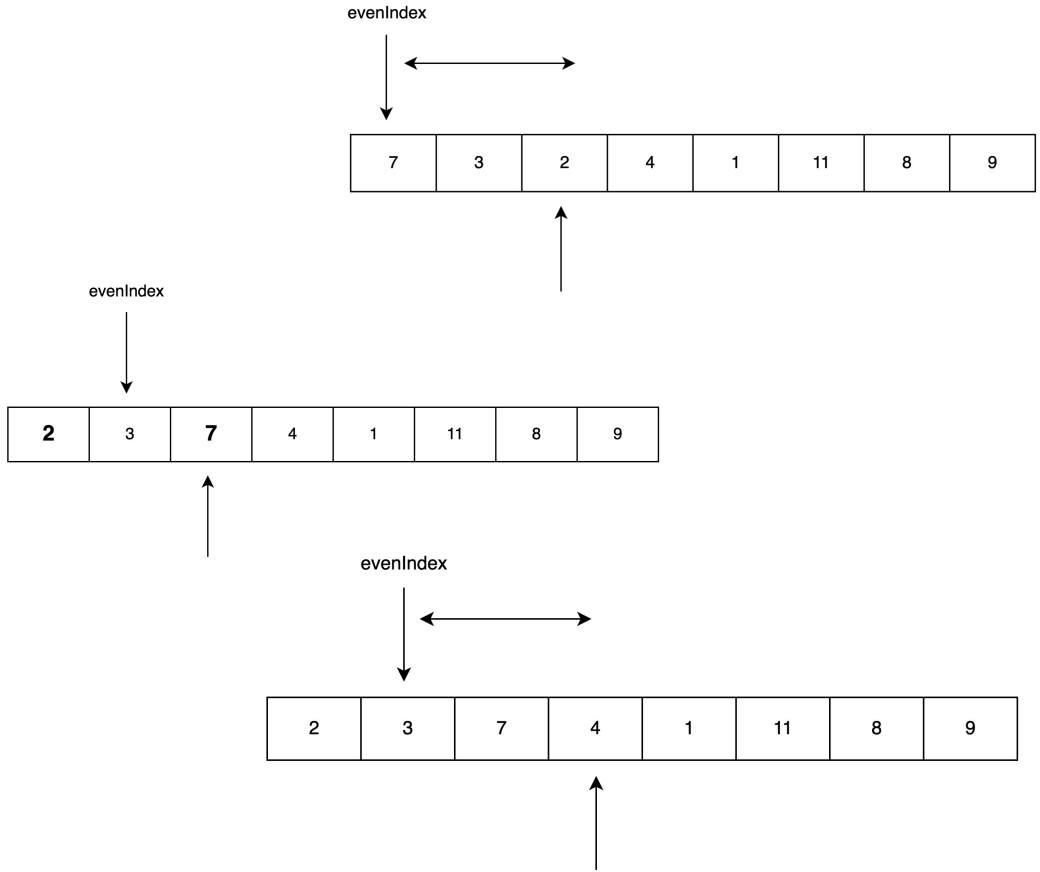
- Также будем использовать два индекса
- Один будет идти по всему массиву, а второй будет увеличиваться, только когда мы встретим четное число





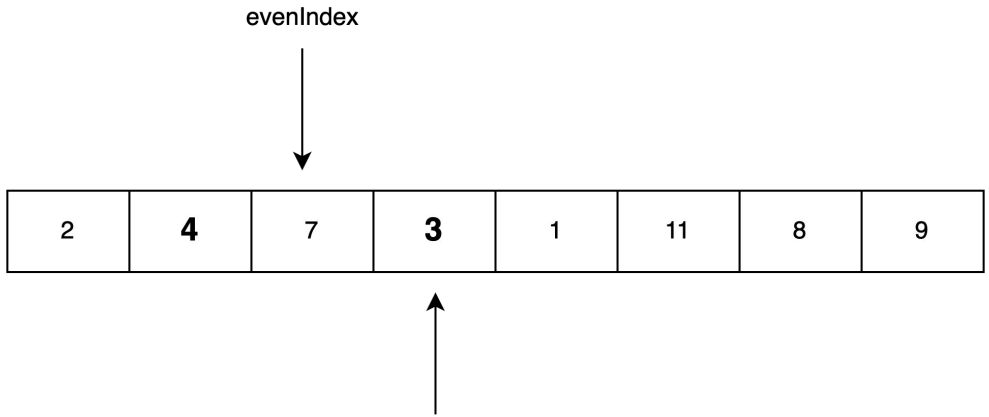
Передвинуть четные числа вперед

- Также будем использовать два индекса
- Один будет идти по всему массиву, а второй будет увеличиваться, только когда мы встретим четное число



Передвинуть четные числа вперед

- Также будем использовать два индекса
- Один будет идти по всему массиву, а второй будет увеличиваться, только когда мы встретим четное число



Передвинуть четные числа вперед

7	3	2	4	1	11	8	9

2	4	8	7	1	11	3	9

```
function evenFirst(arr) {


  return arr

}
```

Передвинуть четные числа вперед

- Также будем использовать два индекса
- Один будет идти по всему массиву, а второй будет увеличиваться, только когда мы встретим четное число

```
function evenFirst(arr) {
  evenIndex = 0
  for i = 0; i < len(arr); i++ {
  }

  return arr
}
```

Передвинуть четные числа вперед

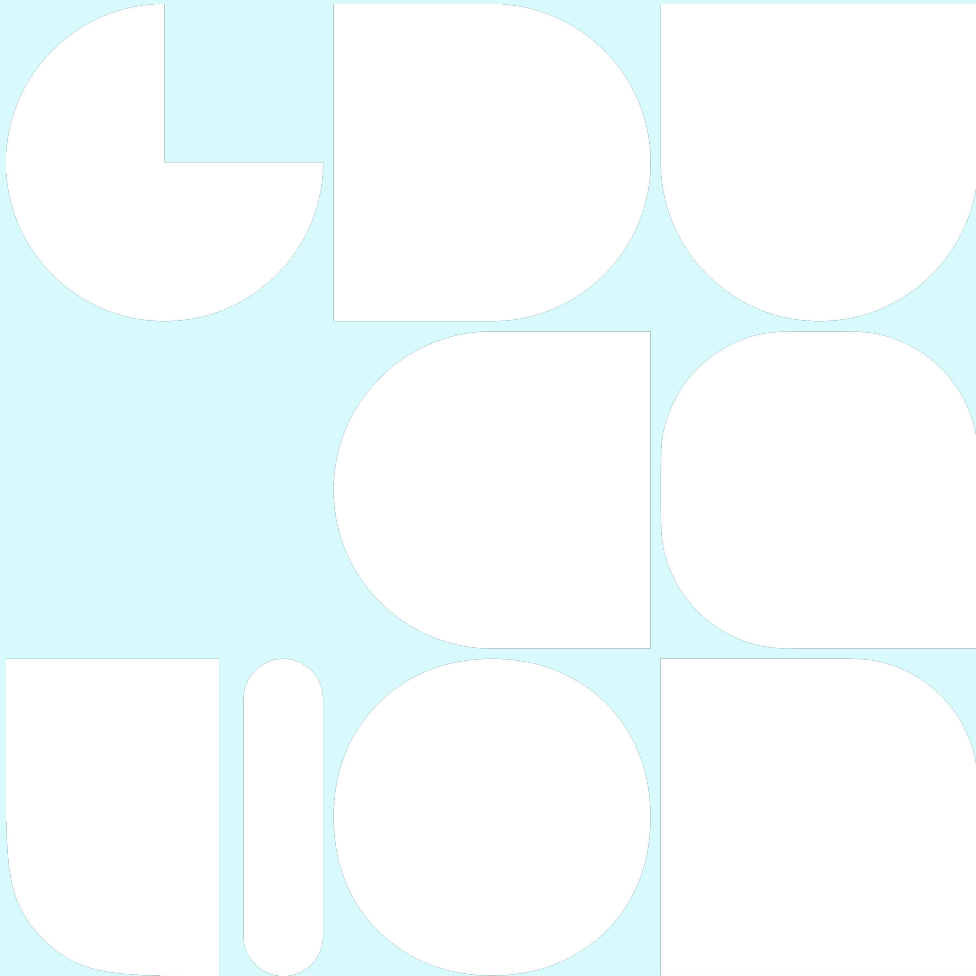
- Также будем использовать два индекса
- Один будет идти по всему массиву, а второй будет увеличиваться, только когда мы встретим четное число

```
function evenFirst(arr) {
  evenIndex = 0
  for i = 0; i < len(arr); i++ {
    if arr[i] % 2 == 0 {
      swap(arr[i], arr[evenIndex])
      evenIndex += 1
    }
  }

  return arr
}
```

Нули в конец

В школе прошел экзамен по математике. Несколько человек списали решения и были замечены. Этим школьникам поставил 0 баллов.

Задача: есть массив с оценками, среди которых есть 0. Необходимо все оценки, равные нулю перенести в конец массива, чтобы все такие школьники оказались в конце списка.



Нули в конец

Пример ввода: [0, 0, 1, 0, 3, 12]
Пример вывода: [1, 3, 12, 0, 0, 0]

Пример ввода: [0, 33, 57, 88, 60, 0, 0, 80, 99]
Пример вывода: [33, 57, 88, 60, 80, 99, 0, 0, 0]

Пример ввода: [0, 0, 0, 18, 16, 0, 0, 77, 99]
Пример вывода: [18, 16, 77, 99, 0, 0, 0, 0, 0]

Спасибо за внимание!

И хорошего вечера:))

education