education

Массивы. Часть 1

Алгоритмы и структуры данных
Илья Почуев

Что будет на занятии

- Что такое массив
- Структура массива
- Основные операции над массивом и их сложность
- Задача на два указателя
- Задача на разворот массива

education

Особенности массива и его структура

Особенности массива

- Непрерывная область памяти заданного размера
- Массив хранит однотипные данные, которые расположены друг за другом в памяти
- Доступ к элементу массива выполняется с помощью целочисленного индекса
- Обращение к ячейке по индексу происходит за константное время
- Индексация массива начинается с нуля
- Размер массива должен быть известен заранее

Структура массива

size — количество элементов в массиве

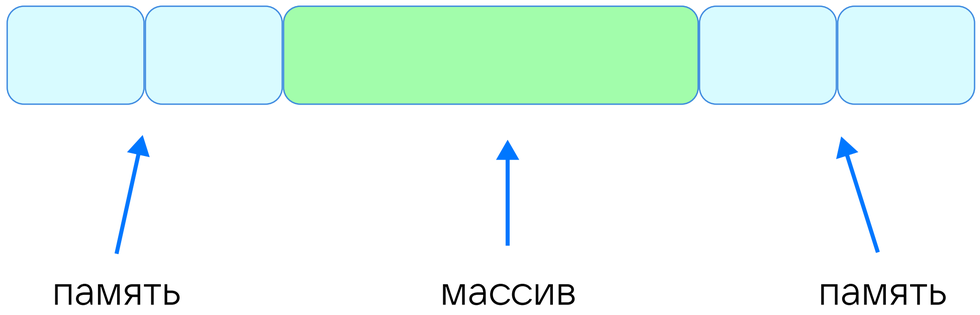
capacity — объём массива, количество выделенной памяти (size ≤ capacity)

type — тип данных

```
Array:
        size int
        capacity int
        type int
```

Как выделяется память под массив

Для массива из 8 элементов с типом int64 при условии, что на моей архитектуре 1 элемент с типом int64 занимает 8 байт, в памяти будет последовательно аллоцировано 64 байта



education

Основные операции над элементами массива и их сложность

Основные операции над элементами массива и их сложность

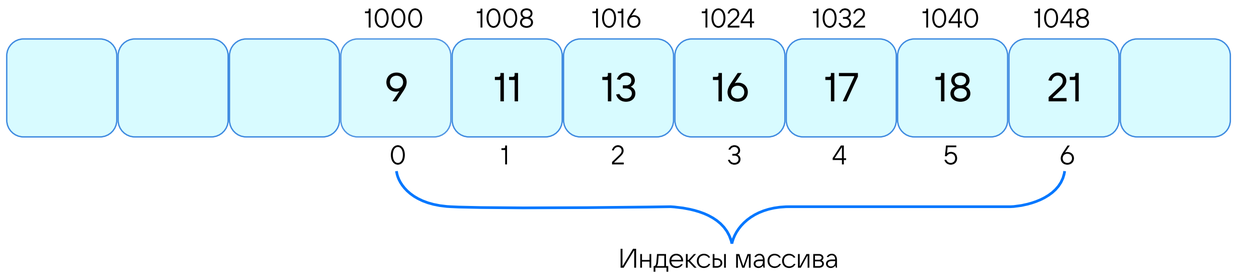
`get(index)` — получение элемента по индексу O(1)

`append()` — вставка в конец O(1)

`remove(index)` — удаление элемента из массива O(n)

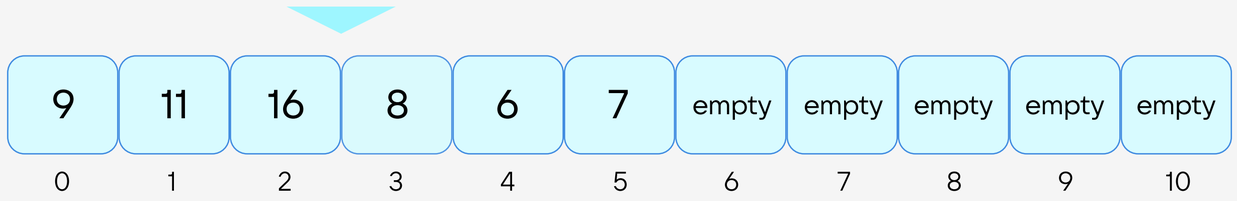
Как получить элемент массива по индексу

- Допустим, нулевой элемент начался с ячейки памяти с номером 1000
- Тогда байты с 1000 по 1007 будут принадлежать первому элементу массива.
  Байты с 1008 – второму и т.д.
- Зная, что все элементы располагаются последовательно, легко получить доступ к каждому из них
- Элемент номер 4 можно найти по формуле: нулевой элемент плюс произведение индекса на размер каждой ячейки: 1000 + 4*8 = 1032

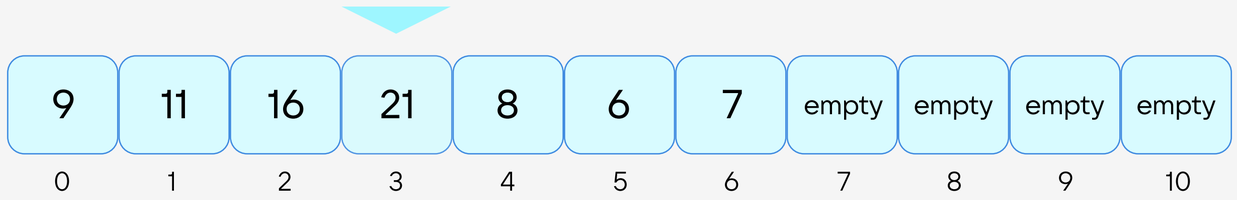


Как добавить элемент в середину массива

Задача: вставить число 21 после элемента массива с индексом 2

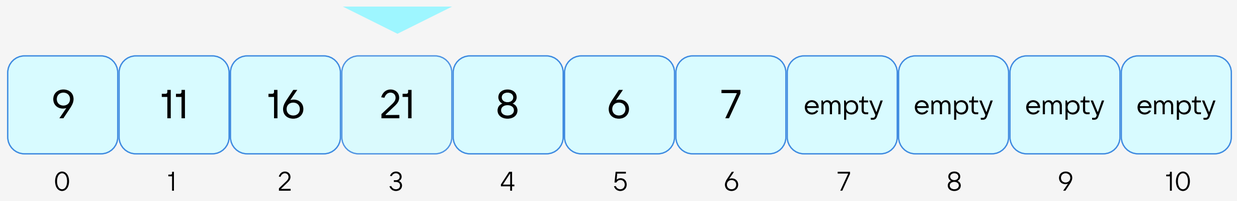


Решение: сдвинуть все элементы, начиная с элемента с индексом 3, на одну позицию вправо

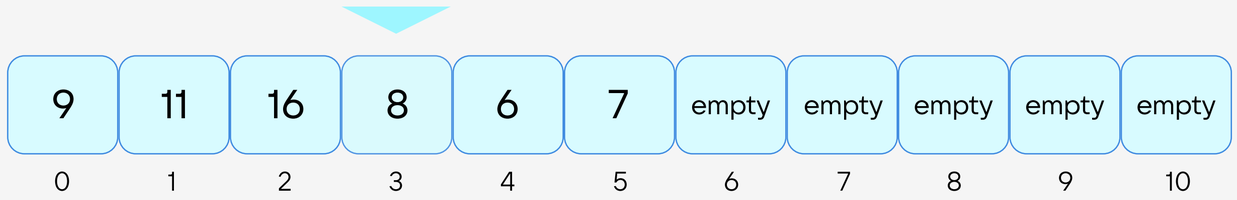


Как удалить элемент из середины массива

Задача: удалить число 21



Решение: сдвинуть все элементы, начиная с элемента с индексом 4, на одну позицию влево



Резюме

1. Массив отлично подходит, когда операций на получение элемента значительно больше, чем вставок, идеально для readonly* хранилища
2. Мы заранее должны знать его размер
3. Не подходит в ситуациях, когда необходимо производить много вставок в середину

* readonly означает, что после того как значение (данные) было записано в переменную или хранилище, оно больше не может быть изменено

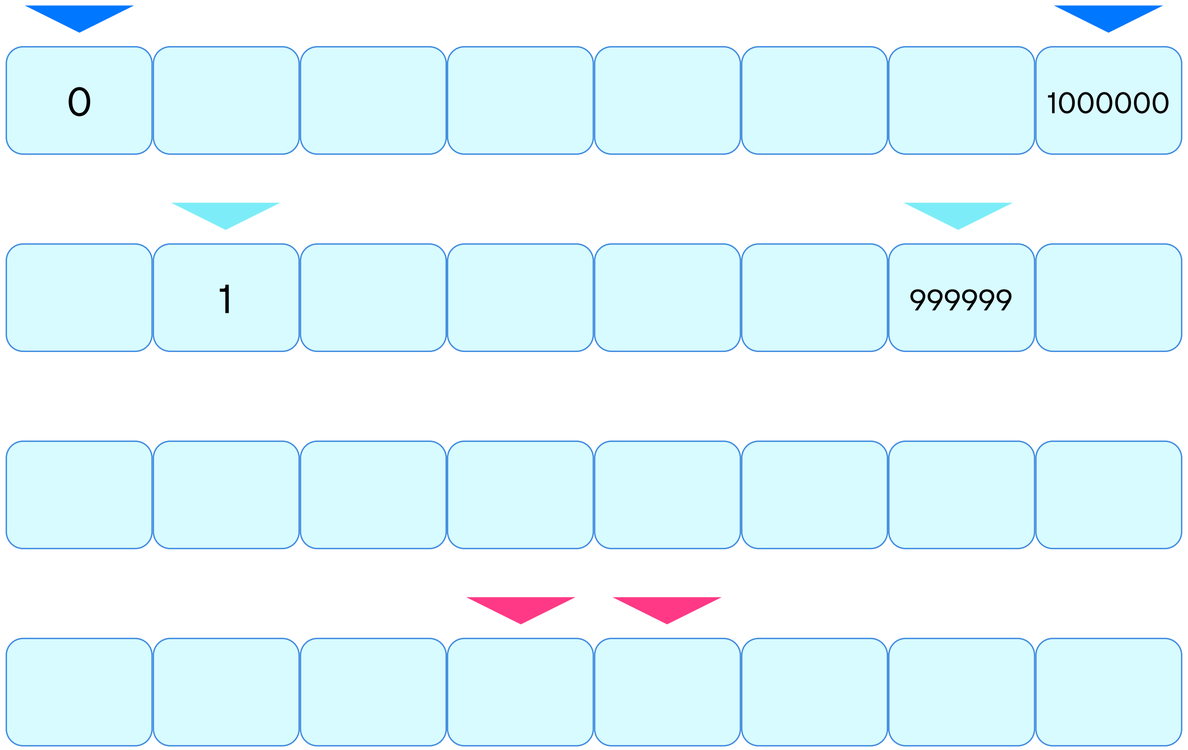
- Пример: организация хранения товаров в памяти
- Один раз добавили все товары
- Чтобы узнать цену товара, вы производите только выборку
- Операции вставки происходят гораздо реже

Как сложить все числа от 0 до 1000000

Складывать их последовательно

Как сложить все числа от 0 до 1000000

Складывать 0 + 1000000; 1 + 999999 и так далее



education

Задача на два указателя

Задача на два указателя (two pointers)

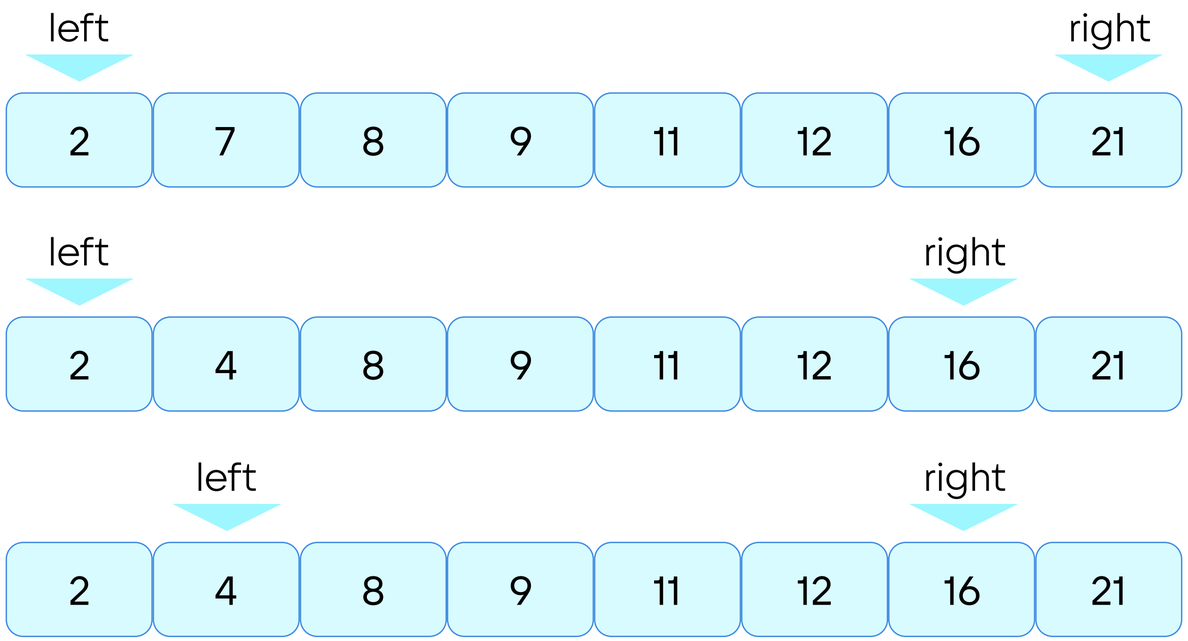
Дано:

- массив целых чисел, отсортированных по возрастанию,
- некоторое число sum.

Задача: написать функцию, которая возвращает два числа из заданного массива, в сумме дающие sum. Если таких двух чисел в массиве нет, должно возвращаться null.

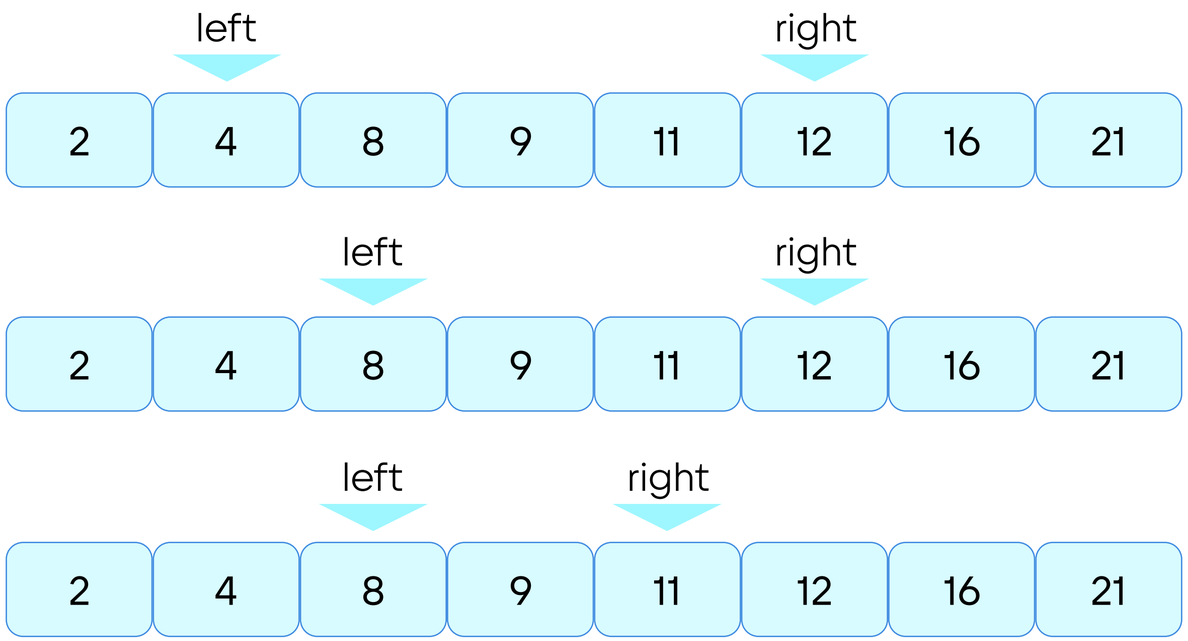
Ограничение: в массиве может быть только одна пара чисел, которая соответствует условию.

Решение задачи на два указателя



sum = 19

Решение задачи на два указателя



sum = 19

Решение задачи на два указателя

- Переменная left указывает на начало массива
- Переменная right указывает на конец массива
- Продолжаем цикл, пока эти две переменные не пересекутся

```
function twoSum (arr, sum) {
    left = 0;
    right = len (arr) - 1;
    while (left != right) {
        ...
    }
    return null;
}
```

Решение задачи на два указателя

```
function twoSum (arr, sum) {
        left = 0;
        right = len (arr) - 1;
        while (left != right) {
                tmp = arr [left] + arr [right];
                 if (tmp == sum) {
                     return [arr[left], arr [right]];
                 }
         ...
         }
         return null;
}
```

- В цикле складываем элементы под индексами left и right
- В случае если сумма равна искомому числу, возвращаем два наших элемента
- В противном случае двигаем необходимый указатель и продолжаем итерироваться

Решение задачи на два указателя

```
function twoSum (arr, sum) {
        left = 0;
        right = len (arr) - 1;
        while (left != right) {
                tmp = arr [left] + arr [right];
                if (tmp == sum) {
                    return [arr [left], arr [right]];
                }
                if (tmp < sum) {
                    left++;
                    continue;
                }
                right- -;
         }
         return null;
}
```

- В случае если результат сложения меньше sum, двигаем левый указатель вправо
- Иначе двигаем правый указатель влево

Сложность O(n)

Как неправильно решать эту задачу

```
function twoSum (arr, sum) {
        for (i = 0; i < len (arr); i++) {
            for (j = 0; j < len (arr); j++) {
                if (arr [i] + arr [j] == sum) {
                    return [arr [i], arr [j]];
                }
            }
        }
        return null;
}
```

Цикл в цикле

Сложность O(n^2)

education

Задача на разворот массива

Задача на разворот массива

Дано: массив целых чисел.

Задача: развернуть массив, то есть вывести его в обратном порядке.

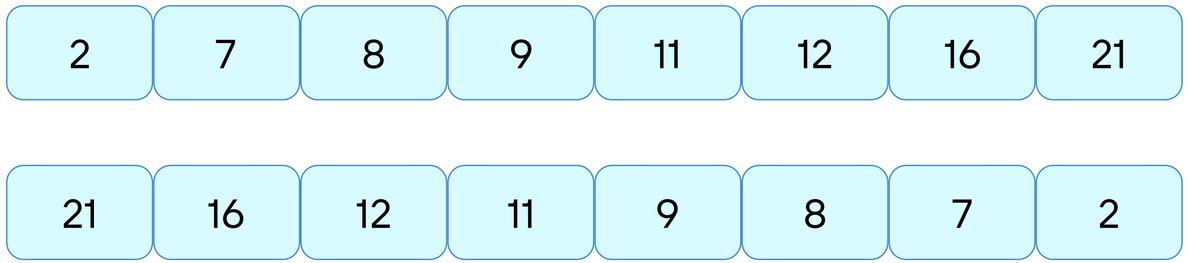
Ограничение: линейное время без дополнительных аллокаций памяти.

Задача на разворот массива

Дано: массив целых чисел.

Задача: развернуть массив, то есть вывести его в обратном порядке.

Ограничение: линейное время без дополнительных аллокаций памяти.

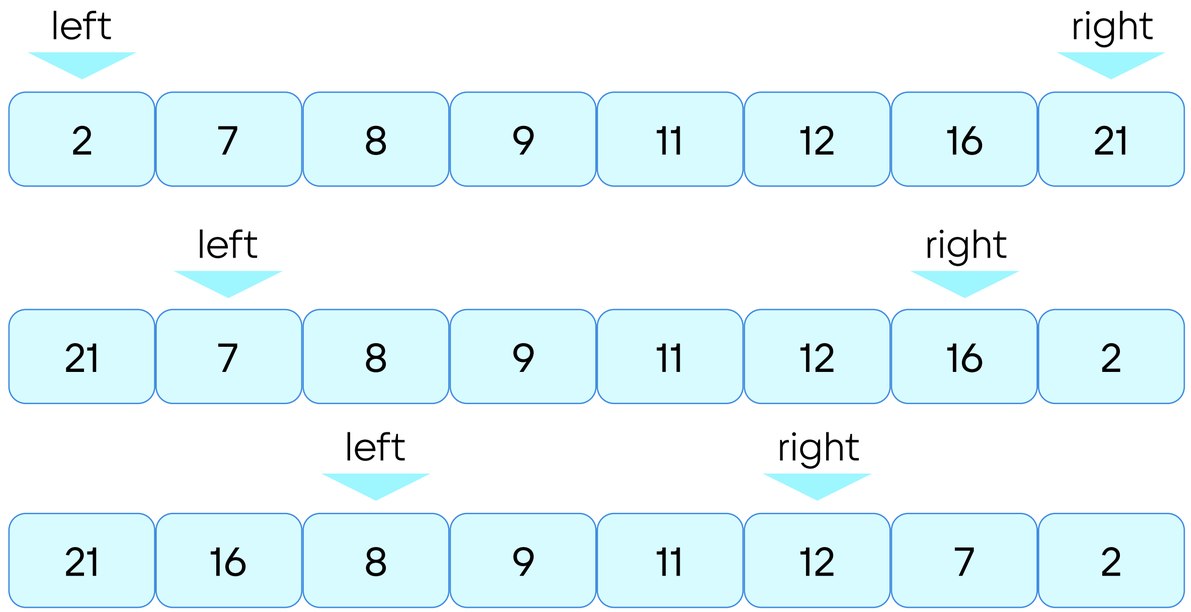


Задача на разворот массива

Дано: массив целых чисел.

Задача: развернуть массив, то есть вывести его в обратном порядке.

Ограничение: линейное время без дополнительных аллокаций памяти.



Задача на разворот массива

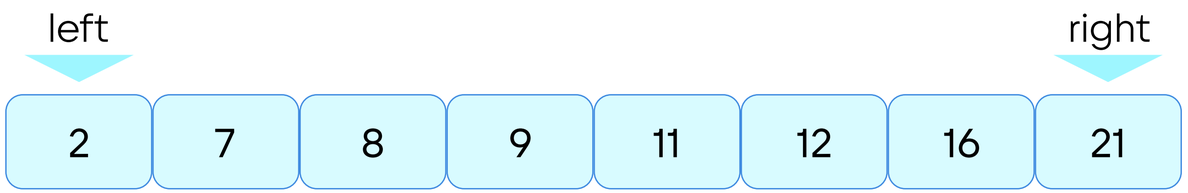
Дано: массив целых чисел.

Задача: развернуть массив, то есть вывести его в обратном порядке.

Ограничение: линейное время без дополнительных аллокаций памяти.

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

        return arr
}
```



Развернуть массив

swap ~ arr[left], arr[right] = arr[right], arr[left]

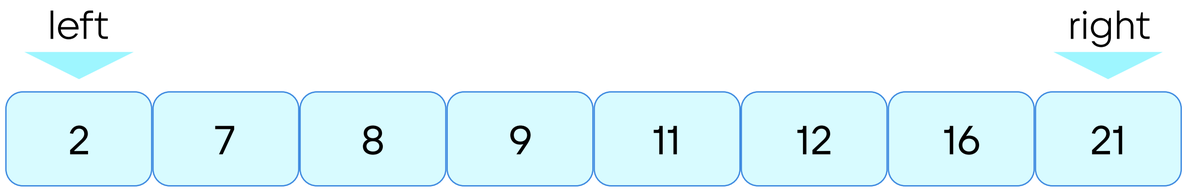
```
function reverseArray (arr) {
        left = 0;
        right = len (arr) - 1;
        while (left < right) {
                swap (arr [left], arr [right]);
                ...
        }
        return arr
}
```

Задача на разворот массива

Дано: массив целых чисел.

Задача: развернуть массив, то есть вывести его в обратном порядке.

Ограничение: линейное время без дополнительных аллокаций памяти.




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

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

         return arr
}
```

Резюме

- Организация массива в памяти
- Основные операции над ним
- Рассмотрели подход к решению задачи через два указателя
- Разобрали пример использования двух указателей

education

Спасибо
за внимание