数组

数组（array)：是一种线性表数据结构。它用一组连续的内存空间，来存储一组具有相同类型的数据。

元素：表示数组中的数据

索引(下标)：数组的索引从0开始，表示元素的位置。

数组访问(Access)：通过索引去访问某一个元素。

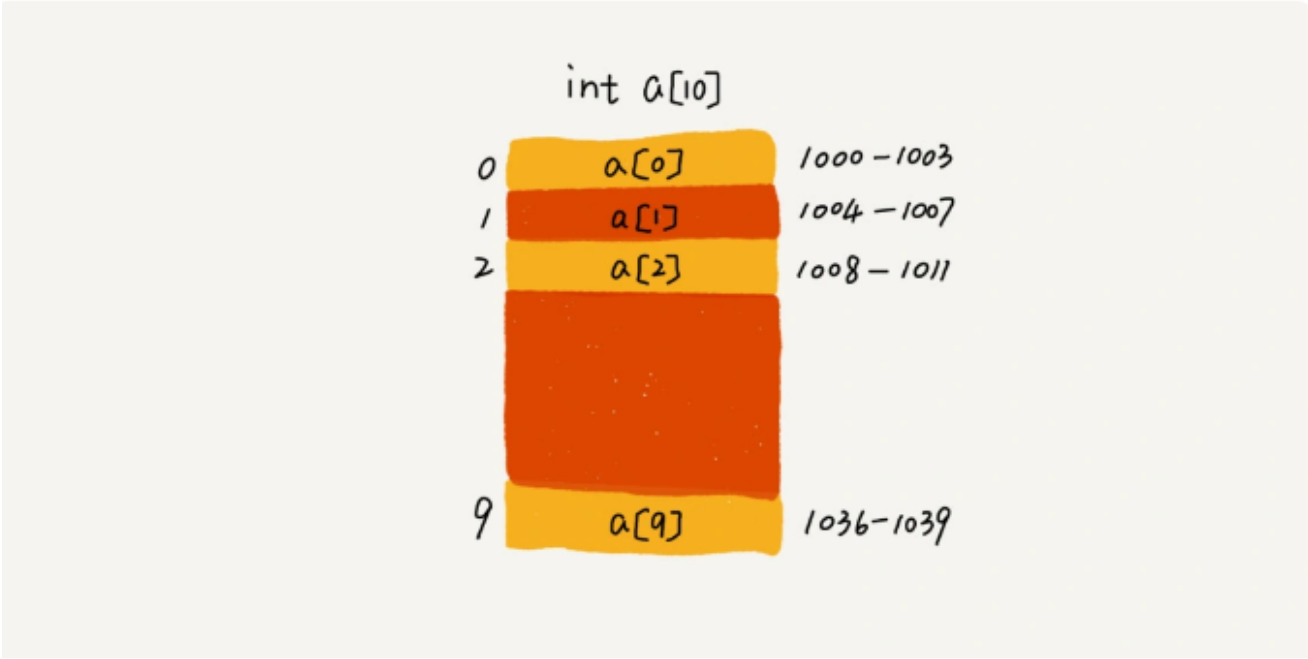
数组如何根据索引访问数组元素？

数组寻址公式:

```
a[i]_address = base_address + i * data_type_size

    a[i]_address:表示索引为i的元素的位置
    base_address：内存块的首地址
    data_type_size:数组中每个元素的大小
```

例子：我们拿一个长度为 10 的 int 类型的数组 int[] a = new int[10] 来举例。在我画的这个图中，计算机给数组 a[10]，分配了一块连续内存空间 1000～1039，其中，内存块的首地址为 base_address = 1000。



计算机会给每个内存单元分配一个地址，计算机通过地址来访问内存中的数据。当计算机需要随机访问数组中的某个元素时，它会首先通过下面的寻址公式，计算出该元素存储的内存地址。

为什么数组支持随机访问？

因为只要计算出base_address，数组其他元素基于偏移量很快就可以计算出来。

数组搜索(Search): 查找数组中存不存在某一个元素。

1. 访问 Access: O(1) 通过元素的索引和它在内存中起始的位置，以及元素所占的空间大小，可以得到元素在内存中的实际位置，不需要遍历整个数组。
2. 搜索 Search： O(N) 为了查找某一个元素是否在数组中，通常会遍历整个数组(最坏情况时间复杂度)

3. 插入 Insert： O(N) 通常在数组元素a的前面插入一个元素，由于数组是连续的，我们需要将元素a及其它后边所有的元素往后移动一个位置。假设需要在数组第一个元素之前插入一个元素，那么数组中所有的元素都需要往后移一个位置(最坏情况时间复杂度)。 当原来的内存空间不够插入时，我们需要开辟一片新的内存空间，将原数组中的数据复制到新的内存空间中。
4. 删除 Delete： O(N) 假设我们删除数组元素a的前一个元素，我们需要将元素a及其他后边的所有元素向前移动一个位置。 假设要删除数组中的第一个元素，那么所有元素都需要向前移动一个位置(最坏情况时间复杂度)。

特点：

- 适合读
- 不适合写

常用操作：

1. 创建数组

```
//Four solutions to create an array in
Java
        //Take [1,2,3] as example
        //Solution 1
        int[] a = {1,2,3};
        System.out.println("a: " +
Arrays.toString(a));

        //Solution 2
```

```
        int[] b = new int[]{1,2,3};
        System.out.println("b: "+
Arrays.toString(b));
//前两种情况是已知数组中的元素

        ***//Solution 3  已知数组中的长度
        int[] c = new int[3];
        for (int i = 0; i < a.length; i++)
{
            c[i] = i+1;
        }
        System.out.println("c: "+
Arrays.toString(c));

     ****//Solution 4  可以指定数组中的类型并
且不知道数组的长度
        /**int 为Java中的基本数据类型，integer
为int的对象类型(包装数据类型)*/
        ArrayList<Integer> arr = new
ArrayList<>();
        for (int i = 0; i < 3; i++) {
            arr.add(i+1);
        }
        //[1,2,3]
        System.out.println("arr: "+
arr.toString());
```

2. 添加元素

创建数组中的前三种方法，在添加元素时，需要重新创建一个新数组，并且将原数组的元素复制到新数组，因此使用第四种创建方法更合适。

```
//Add element
        //Time Complexity: O(1)    特殊情况，当且仅当数组所在内存空间仍有空闲区域
        arr.add(99);    //不指定索引，默认加在数组尾端
        //[1,2,3,99]
        System.out.println("arr: "+arr.toString());
        //Insert element
        //Time Complexity: O(N)
        arr.add(3,88);  //第一个参数，插入的索引，  第二个参数，插入的元素
        //[1,2,3,88,99]
        System.out.println("arr: "+arr.toString());
```

3. 访问元素

通过下标（索引）来访问元素

```
//Access element
        //Time Complexity: O(1)
        int c1 = c[1];
        int arr1 = arr.get(1);

        System.out.println("c1: "+c1);
        System.out.println("arr1: "+arr1);
```

4. 修改元素

对于方式三：可以直接通过索引修改元素

对于方式四：调用.set(n1,n2) 第一个参数为要修改的位置的索引，第二个参数为修改后元素的值。

```
//Update element
        //Time Complexity : O(1)
        c[1] = 11;
        arr.set(1,11);
        //1 - >11
        System.out.println("c1: "+ c[1]);
        System.out.println("c:" +
Arrays.toString(c));
        System.out.println("arr1: "+
arr.get(1));
        System.out.println("arr: "+
arr.toString());
```

5. 删除元素

删除元素，利用第四种方法比较合适.remove(n),方法中的参数是要删除数组中哪个元素的索引，由于删除了中间的某一个元素，所有后面的元素都需要向前移动所以时间复杂度是:O(N)

```
//Remove element
        //Time Complexity: O(N)
        arr.remove(2);
        System.out.println("arr: "+
arr.toString());
```

6. 遍历数组

通过for循环遍历数组，注意ArrayList获取元素使用的get()

Time Complexity: O(N)

```
        //Iterate c
        for (int i = 0; i < c.length; i++)
{
            int current = c[i];
            System.out.println("c at index
" + i +": " + current);
        }
        //Iterate arr
        for (int i = 0; i < arr.size();
i++) {
            int current = arr.get(i);
            System.out.println("arr at
index " + i +": " + current);
```

7. 查找元素

注意ArrayList中查找元素可以用contains (),返回值为Boolean类型

Time Complexity: O(N)

```
//Find an element
        //Find an element in c
        for (int i = 0; i < c.length; i++)
{
            if (c[i] == 3){
                System.out.println("We
found 3 in c!");
            }
        }
        //Find an element in arr
        boolean is3 = arr.contains(3);
        System.out.println("Are We found 3
in arr?" + is3);
```

8. 数组的长度

对于前三种方法，可以通过.length获取长度,第四种方法，通过.size()获取数组的长度。 Time Complexity: O(1)因为在创建数组的时候，会在内部保存一个count的变量每次增加或者删除，会修改count的变量值，从而获取数组 长度。

```
//The length of an array
        //Time Complexity: O(1)
        int CSize = c.length;
        int arrSize = arr.size();
        System.out.println("c.length: "
+CSize );
        System.out.println("arr.length: "
+arrSize );
```

9. 数组的排序 (内置的排序方法)

***重要： Time complexity: O(NlogN)

对于前三种创建方式，由于创建的是int类型的数组，可以将其改为Integer类型的数组，然后使用 Arrays.sort(T[], Collections.reverseOrder()); 进行排序

对于ArrayList 可以直接使用 Collections.sort(arr,Collections.reverseOrder()); 第一个参数是需要排序的数组，第二个参数意思是将数组反转

```
int[] c = new int[]{2, 3, 1};
        ArrayList<Integer> arr = new
ArrayList<>();
        arr.add(2);
        arr.add(3);
        arr.add(1);
        //[2,3,1]
        System.out.println("c: "+
Arrays.toString(c));
        System.out.println("arr: "+
arr.toString());

        //from small to big
        //Time complexity: O(NlogN)
        Arrays.sort(c);
        //[1,2,3]
        System.out.println("c: "+
Arrays.toString(c));
        Collections.sort(arr);
```

```
        System.out.println("arr: "+
arr.toString());

        //from big to small
        //Time complexity: O(NlogN)
        //For c,you can read an array in
reverse
        //Arrays.sort(T[],
Collections.reverseOrder());
        //For arr

 Collections.sort(arr,Collections.reverseO
rder());
        //[3,2,1]
        System.out.println("arr: "+
arr.toString());
```

leetcode: 485、283、27