5. Arrays

Friday, February 4, 2022 2:49 AM

https://docs.oracle.com/javase/7/docs/api/java/util/Arrays.html

https://docs.oracle.com/javase/7/docs/api/java/lang/String.html

- organizes items sequentially in memory
  index 0 -> n
- Good for iterating if you know the indices
* optimal for access, indexing (access element by index number), O(1)
  • insert/delete/search is O(n)

// Array's push is O(1)

- Treat strings as array of characters

2 Types

1) Static - fixed size

2) Dynamic - reserved size for additional elements and can grow in size

String reversal practice

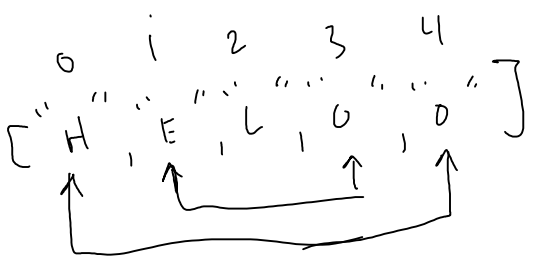
Input = ["H","E","L","L","O"]

output = ["O","L","L","E","H"]

Approach:

2 pointer approach

needs: temp value for storage, divide loop length by 2




```
class Solution {
  public void reverseString(char[] s) {
    for (int i = 0; i < s.length / 2 ; i++) {
      char temp = 'a'; // Saves placeholder value for swapping
        temp = s[i];
```

```
        s[i] = s[s.length - (1 + i)];
        s[s.length - (1 + i)] = temp;
    }
  }
}
```

Time: O(n/2) -> O(n), because we have 1 for loop that is dependent on size of input

Space: O(1), because we create a constant space char

Merge Sorted Arrays Practice

Input:

- 2 integer arrays, nums1 and nums2, sorted in non-decreasing order
- 2 integers, m and n, representing number of elements in nums1 and nums2 respectively

Output: Merge nums1 and nums2 into a single array sorted in non-decreasing order

Assume: final sorted array should be stored inside nums1. nums1 has a length of m+n, where m are the elements that should be merged, and the last n elements are arbitrary and set to 0. nums2 is length of n

Example 1:

Input: nums1 = [1,2,3,0,0,0] m = 3

nums2 = [2,5,6], n = 3

output: nums1 = [1,2,2,3,5,6]

Approach 1:

create temp array with size m, do sort & insert of nums1[i] and nums2[i] into temp array then copy into nums1

Time: O(n), b/c 1 for loop

Size: O(n), b/c we're creating an array dependent on input size

---

Approach 2: fill values starting the back of nums1 to avoid shifting elements

3-pointer approach, each pointer at the end of both arrays,

Compare values at pointers and insert the larger value, decrement the pointer that was inserted

Edge case: pointers < 0, to avoid out of bounds error if any of the arrays are empty or when pointer goes to end of array

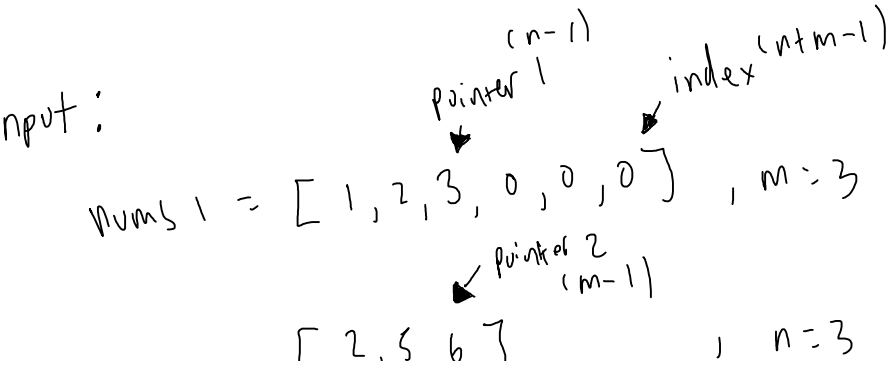
e.g. input: nums1 = [1], m=1 | nums2 = [], n=0

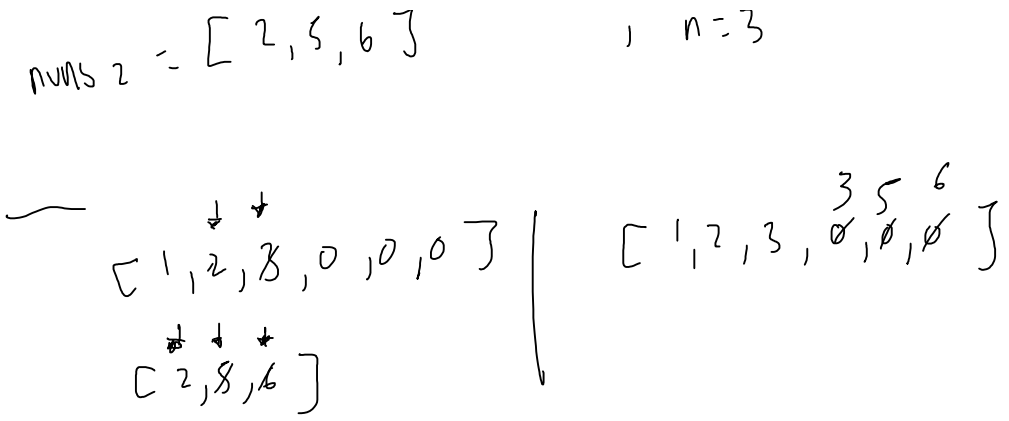
output: 0

input: nums1[0], m=0 | nums2 = [1], n=1

output: [1]

Input:





```
class Solution {
   public void merge(int[] nums1, int m, int[] nums2, int n) {
      // 3 Pointer Approach
      int pointer1 = m-1;
      int pointer2 = n-1;
      int index = m + n -1;

      while (index >= 0) {
         // Edge case check if one of the arrays are empty or when
pointer goes to the end of an array
         if (pointer1 < 0 ) {
            nums1[index] = nums2[pointer2--];
            }
         else if (pointer2 < 0){
            nums1[index] = nums1[pointer1--];
         }

         // Comparison because we are not the end of either arrays
         else{
            if (nums1[pointer1] > nums2[pointer2]){
               nums1[index] = nums1[pointer1--];
            }
            else{
               nums1[index] = nums2[pointer2--];
            }
         }
         // Decrement after checks
         index--;
      }
   }

   // O(n+m) time, because it is dependent if n>m or n<m, O(max(n,m))

   // O(1) space, because space created is constant
```