Welcome to CS 112!

CS 112 - Recitation 1

Types, Arrays, & Big O

Contact Info

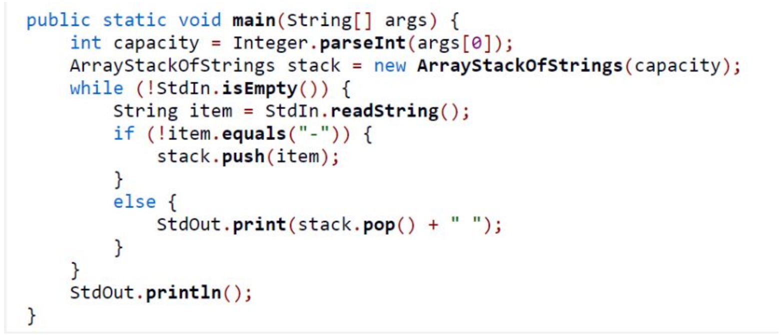
Email: ah1058@scarletmail.rutgers.edu

80% attendance
20% quiz
Recitation: 100 pts out of 1000
Camera needs to be turned on

All information please visit: https://ds.cs.rutgers.edu/

Attendance

Primitive vs Reference Types (and Pointers)



```
public static void main(String[] args) {
    int capacity = Integer.parseInt(args[0]);
    ArrayStackOfStrings stack = new ArrayStackOfStrings(capacity);
    while (!StdIn.isEmpty()) {
        String item = StdIn.readString();
        if (!item.equals("-")) {
            stack.push(item);
        }
        else {
            StdOut.print(stack.pop() + " ");
        }
    }
    StdOut.println();
}
```

Arrays

How do we declare/instantiate an array? Access an element?

Q1: Reverse an Array

Write a method that reverses a string and analyze the time and space complexity. The input string is given as an array of characters char[].

You may assume all the characters consist of printable ascii characters.

Example 1:

Input: ["h","e","l","l","o"]

Output: ["o","l","l","e","h"]

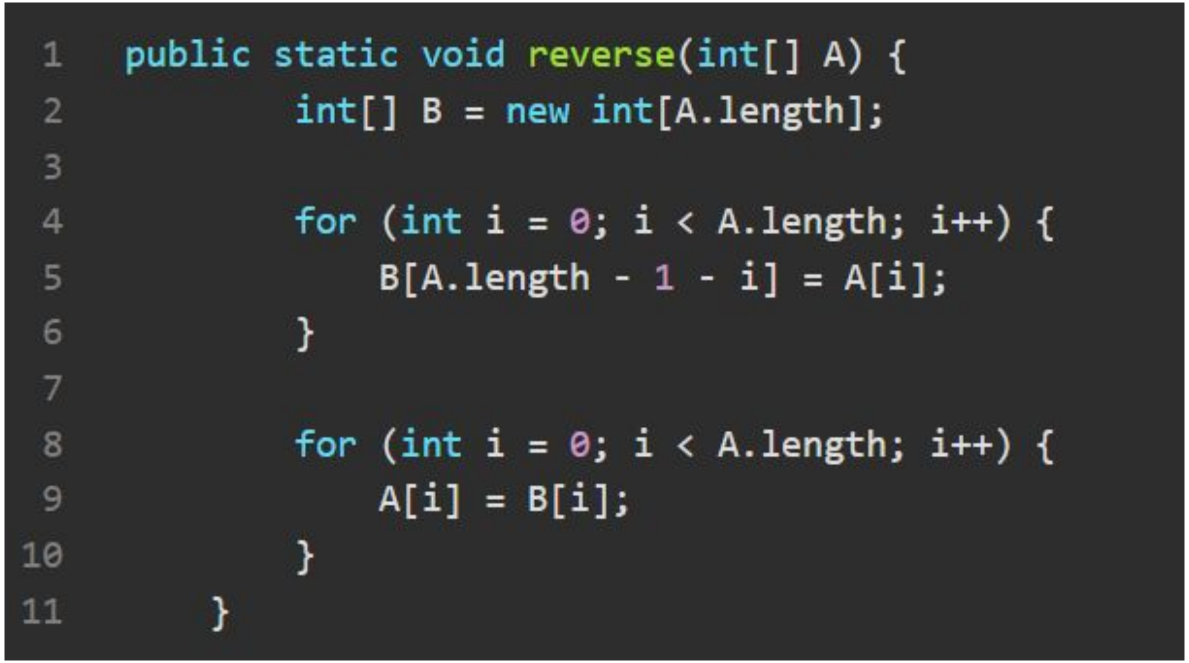
Example 2:

Input: ["H","a","n","n","a","h"]

Output: ["h","a","n","n","a","H"]

LeetCode 344

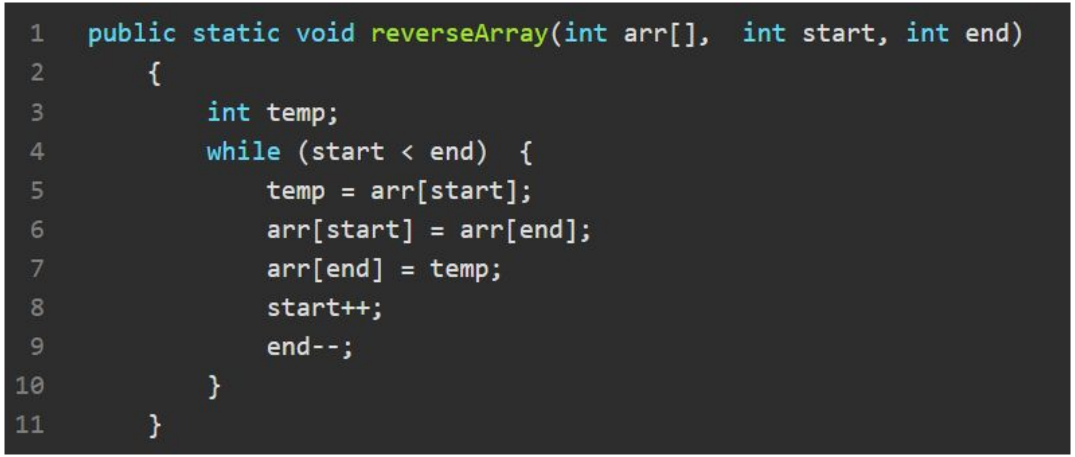
Reverse an Array: Naive Solution



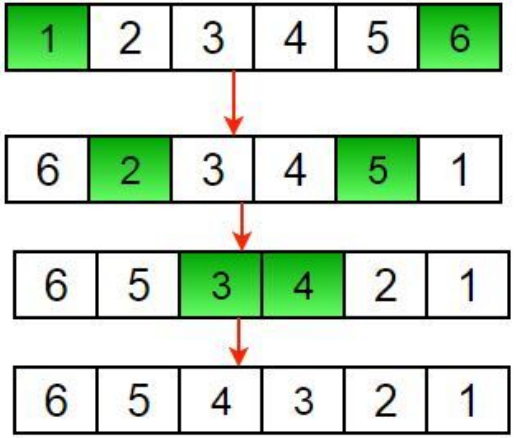
```java
public static void reverse(int[] A) {
        int[] B = new int[A.length];

        for (int i = 0; i < A.length; i++) {
            B[A.length - 1 - i] = A[i];
        }

        for (int i = 0; i < A.length; i++) {
            A[i] = B[i];
        }
    }
```

Time Complexity: O(n), Space Complexity: O(n)

Reverse an Array: Solution 2: Iterative Pointers



```java
public static void reverseArray(int arr[],  int start, int end)
    {
        int temp;
        while (start < end)  {
            temp = arr[start];
            arr[start] = arr[end];
            arr[end] = temp;
            start++;
            end--;
        }
    }
```



Time Complexity: O(n) , Space Complexity: O(1)

Q2: Two Sums (Breakout)

Given an array of integers that is already ***sorted in ascending order***, find two numbers such that they add up to a specific target number.

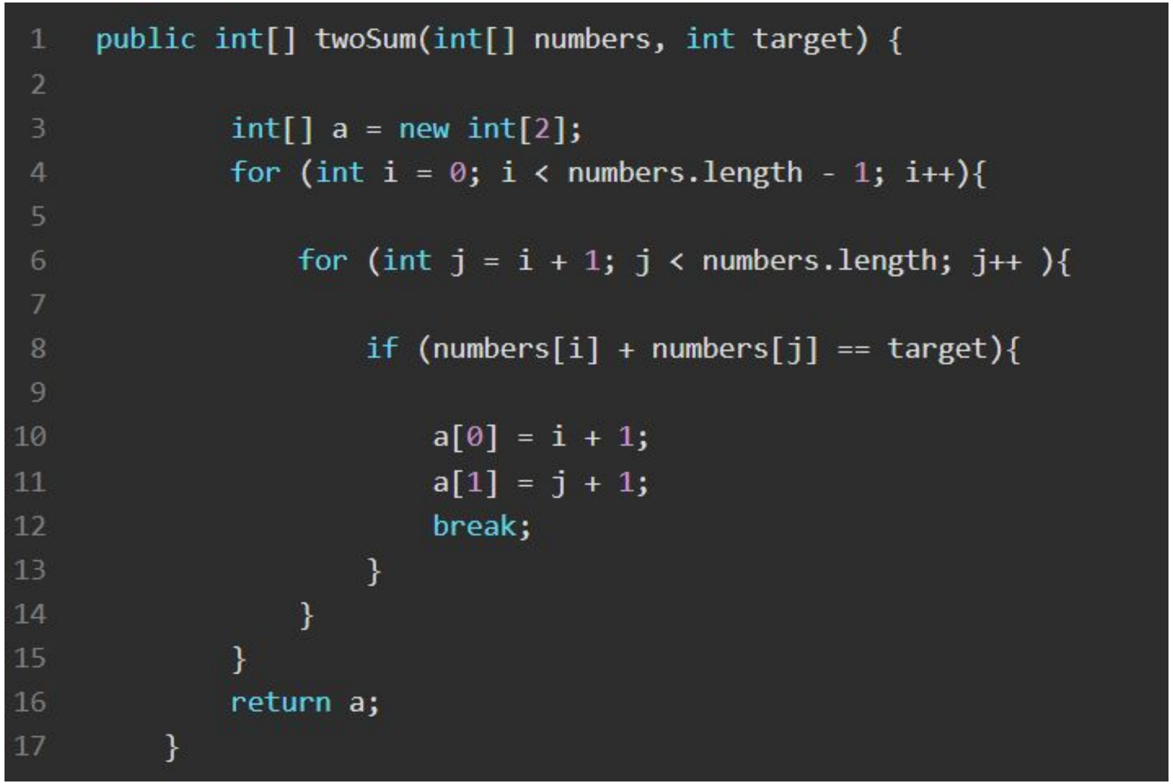
Example 1:

Input: numbers = [2, 7, 11, 15], target = 9

Output: [1,2]

Explanation: The sum of 2 and 7 is 9. Therefore index1 = 1, index2 = 2.

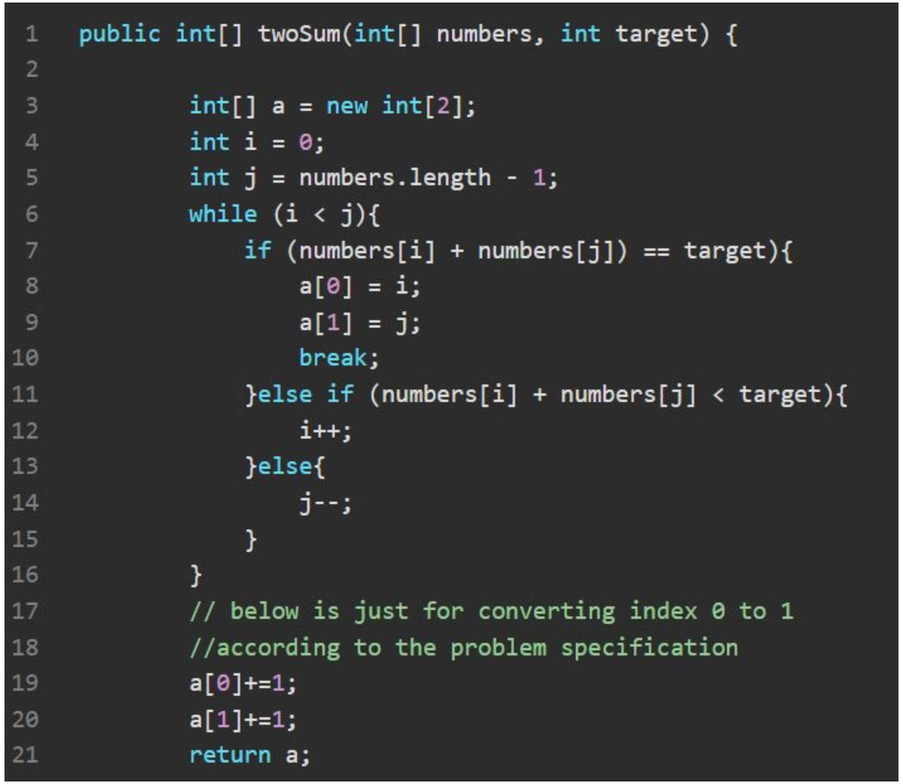
Q2: Two Sums: Naive Solution



```java
public int[] twoSum(int[] numbers, int target) {

        int[] a = new int[2];
        for (int i = 0; i < numbers.length - 1; i++){

            for (int j = i + 1; j < numbers.length; j++ ){

                if (numbers[i] + numbers[j] == target){

                    a[0] = i + 1;
                    a[1] = j + 1;
                    break;
                }
            }
        }
        return a;
    }
```

Time Complexity: O(n^2) ,
Space Complexity: O(1)

Q2: Two Sums: 2 Pointer Solution



```java
public int[] twoSum(int[] numbers, int target) {

        int[] a = new int[2];
        int i = 0;
        int j = numbers.length - 1;
        while (i < j){
            if (numbers[i] + numbers[j]) == target){
                a[0] = i;
                a[1] = j;
                break;
            }else if (numbers[i] + numbers[j] < target){
                i++;
            }else{
                j--;
            }
        }
        // below is just for converting index 0 to 1
        //according to the problem specification
        a[0]+=1;
        a[1]+=1;
        return a;
```

Time Complexity: O(n)
Space Complexity: O(1)

Q3: Merge Two Sorted Arrays (Optional)

Given two sorted integer arrays nums1 and nums2, merge nums2 into nums1 as one sorted array. Analyze the time and space complexity.

The number of elements initialized in nums1 and nums2 are m and n respectively. You may assume that nums1 has a size equal to m + n such that it has enough space to hold additional elements from nums2.

No need to code! Just think through the algorithm...

Example:

Input: nums1 = [1,2,3,0,0,0], m = 3, nums2 = [2,5,6], n = 3

Output: [1,2,2,3,5,6]

Hint: **Think about how you can do this in constant space.**

LeetCode 88

Merge Two Sorted Arrays: Naive Solution 1

1. Create a new temporary array
2. Put all the elements of array 1 into the temp array, and then do the same with all the elements of array 2
3. Sort the temporary array
4. Copy the elements of the temporary array into the original array 1

`Time Complexity: O((n+m)log(n+m))`
`Space Complexity: O(n+m)`

`How can we improve both time and space complexity?`

Merge Two Sorted Arrays: Slightly Better Solution (2 Ptrs)

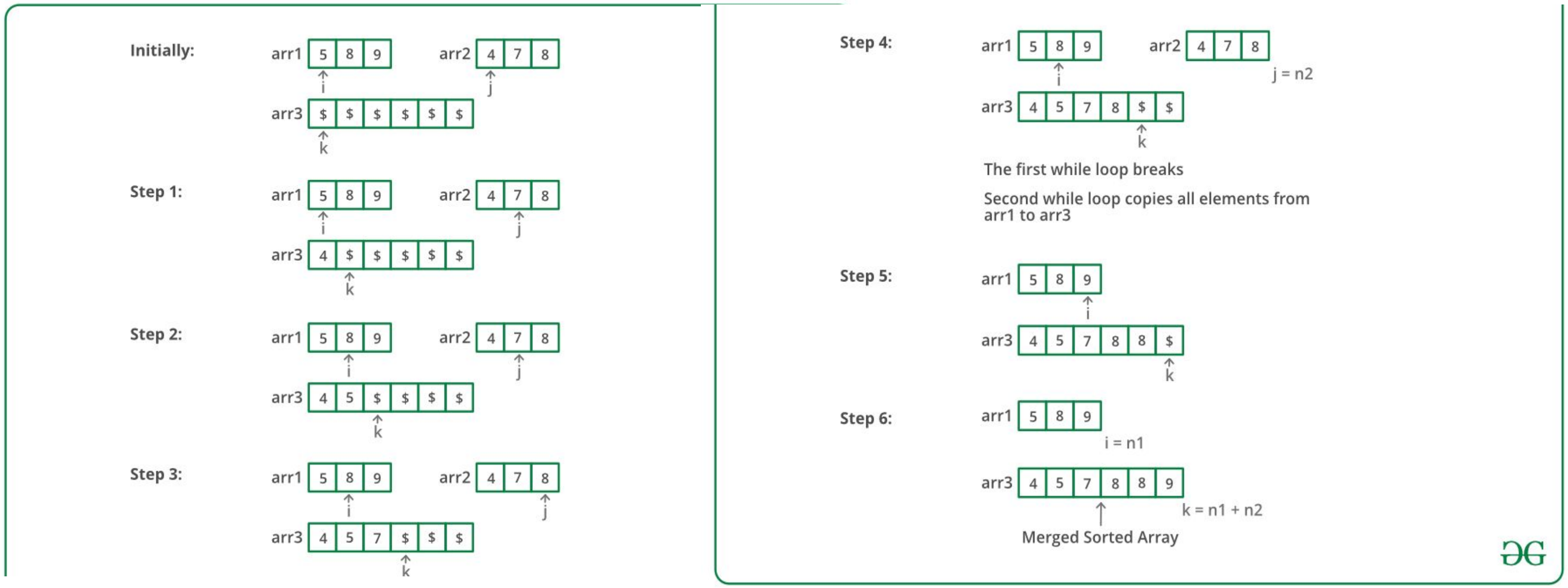
1. Create a new temporary array
2. Have one pointer for element 0 of array 2, one pointer for element 0 of array 2
3. Compare the two elements, and insert the smaller one into the temporary array
4. Move the pointer of the element that was smaller
5. Repeat until one array's elements are completely in the temp array
6. Put the rest of the other array's elements into the temp array
7. Copy the elements of the temp array into the original array 1

`Time Complexity: O(n+m), Space Complexity: O(n+m)`

`Picture of Solution on Next Slide`

`How can we improve the space complexity?`

Merge Two Sorted Arrays: Slightly Better Solution (2 Ptrs)



Merge Two Sorted Arrays: Efficient Solution

Similar to the last solution EXCEPT… start merging from the end of the array!

Time Complexity: O(n+m), Space Complexity: O(1)

Video Explanation:
https://www.youtube.com/watch?v=P1Ic85RarKY

Extra Thinking (Optional):

1. **What is the worst case that will result in the slowest runtime? What is the best case that will result in the fastest runtime?**
2. **What if the original two arrays are unsorted?**

Warm-Up Questions

1. Why distinction between **primitive** and **reference** types? Why not just have reference types?
2. What is a **pointer**? Any example?
3. What is **null**? Any example?
4. We have seen **static method** (method that can be used without having to instantiate an object), what about **static variables**? What does static do?

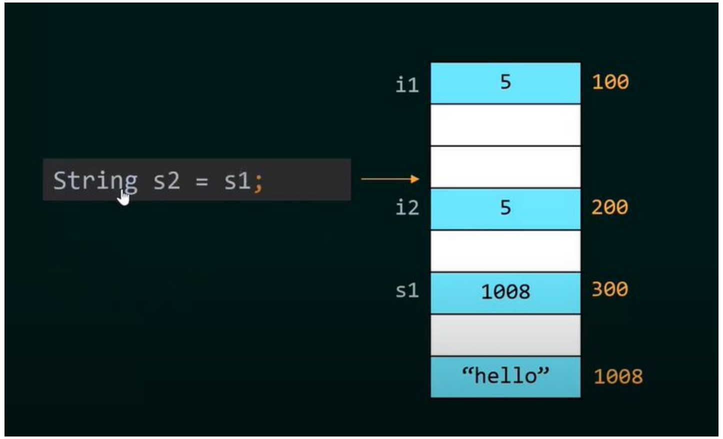
Warm-Up Questions Solution (See pg.110 - pg.113)

1. **Performance**. Primitive type data run faster than corresponding reference types.

Example: What's the output of the following code?

```
String string1 = "hello";
String string2 = string1;
string1 = "world";
StdOut.println(string1);
StdOut.println(string2);
```

Answer:
world
hello



```
String s2 = s1;
```

Warm-Up Questions Solution Cont. (See pg.110 - pg.113)

2. Similar to Java Reference, a pointer is a **machine address.** In Java, there is only one way to create a reference (new) and one way to change a reference (assignment statement, aka copy the reference).

    Example:
    ```
    Node head = new Node(d); // create a reference
    Node ptr = head; // assignment statement
    ```

3. A literal value that refers to **no object**. If you invoke a method using null reference will result a **NullPointerException**. (E.g. Node prev = null;)
4. A “global variable”. Like instance variables, static variables are accessible to **every** method but **not associated** with any object. (Rare to use in actual programming)

Good Work!

1. Log-in to DynRec
2. Enter Quiz Code: