Arrays - I

Topics in Today's class

Array Basics

Search in an Array

Delete an Element from an Array

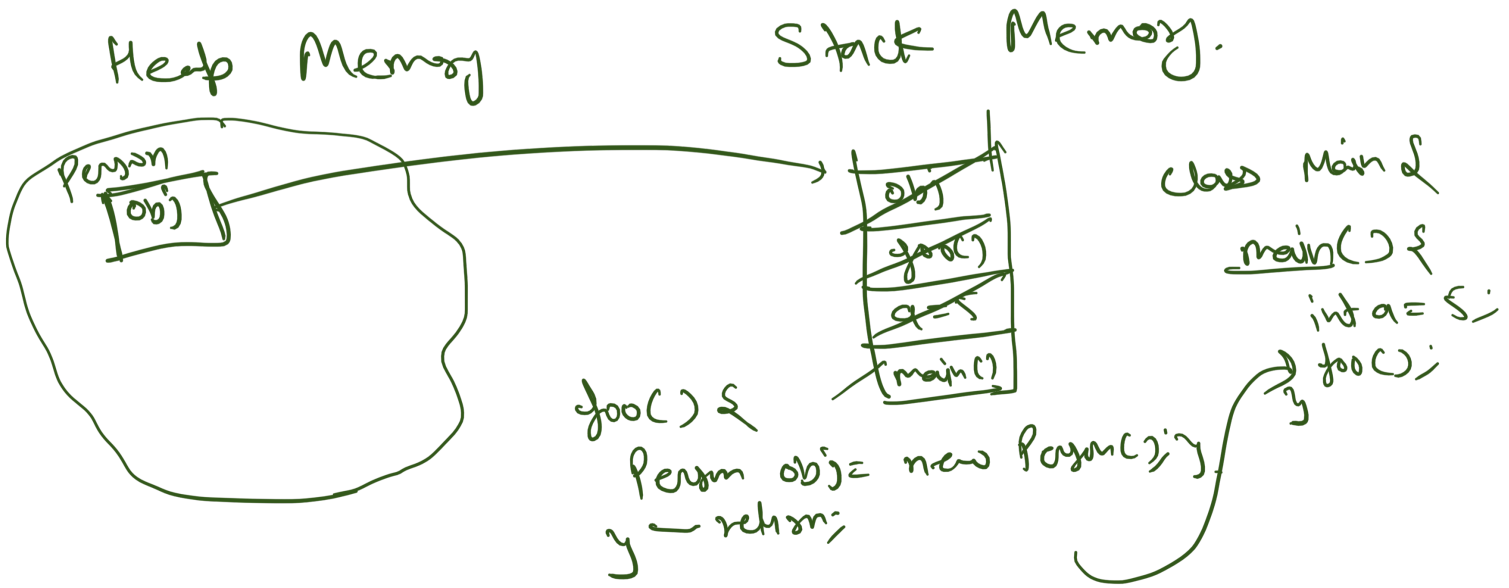
Find the largest element in an Array

Find the second largest element in an Array

Remove duplicates from a sorted Array

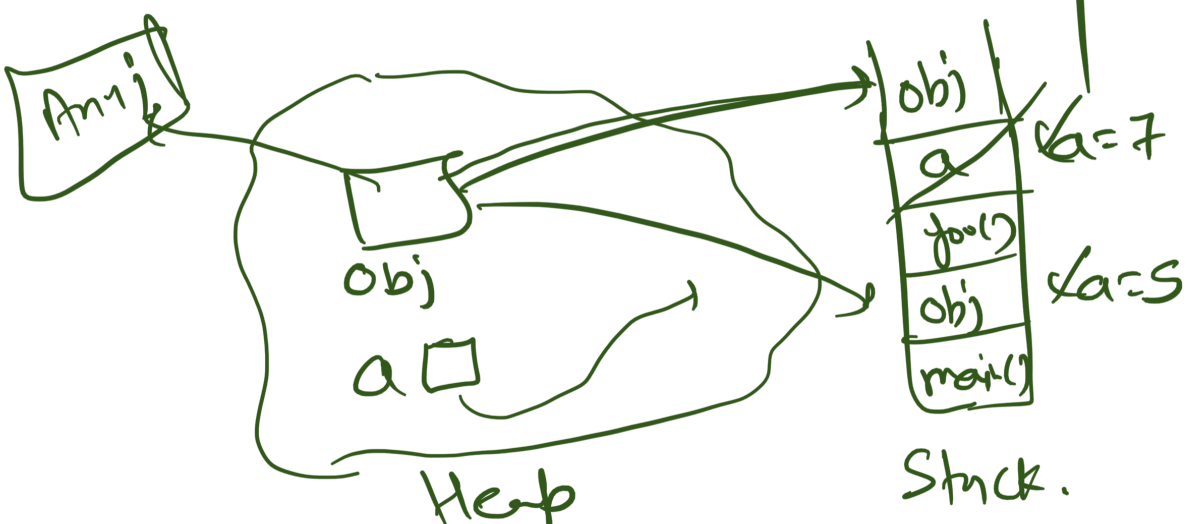
Array Basics

1. Majority element
2. longest 0s and 1s
3. Max area in a histogram.



```
main () {
  Person obj = new Person();
  a = 5;
  foo (a  obj);
  Print (obj.name);
}
```

```
foo ( int a  Person obj) {
    a = 7;
    obj.name = "Anuj";
}
```



[Always Pass by value]

new

```
class A {
    int a;
}

main ()
    A obj = new A()
    obj.a = 5;
    foo(obj)

}
```

```
foo(A obj){
    obj.a = 7

}
```

Search an Element in an Array

Delete an Element from an Array

Largest Element in an Array

Second Largest Element in an Array

Remove Duplicates from a Sorted Array

Practice Problems

1. Find the smallest element in the given Array.
2. Find the Third Largest Element in an Array.
3. Check if the Array is Sorted.
4. Reverse the given Array
5. Write a program to replace every element with the greatest element on its right side.

    Expected Output :

    The given array is : 7 5 8 9 6 8 5 7 4 6

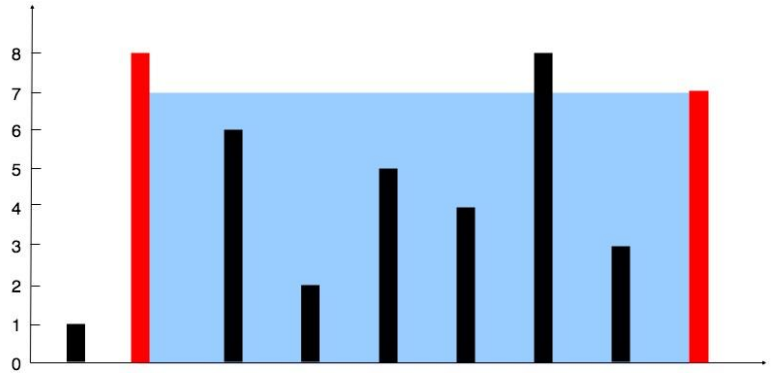
    After replace the modified array is: 9 9 9 8 8 7 7 6 6 0

Arrays - II

Leaders in an Array

Maximum Sum Subarray

Container with most Water



Rotate Array by k steps

a[] = {1, 2, 3, 4, 5, 6, 7, 8}

k = 3.

output: {4, 5, 6, 7, 8, 1, 2, 3}

{2, 3, 4, 5, 6, 7, 8, 1}

{3, 4, 5, 6, 7, 8, 1, 2}

{4, 5, 6, 7, 8, 1, 2, 3}

leftRotate

```
firstElement = a[0]
for (int i=0; i<n-1; i++){
    a[i] = a[i+1];
}
a[n-1] = firstElement;
```

```
for(int i=0; i<k; i++){
    leftRotate( )
}
```

O(n.k)

a[] = {1, 2, 3, 4, 5, 6, 7, 8}

k = 3.

1. Reverse k elements
2. Reverse remaining elements
3. Reverse whole array.

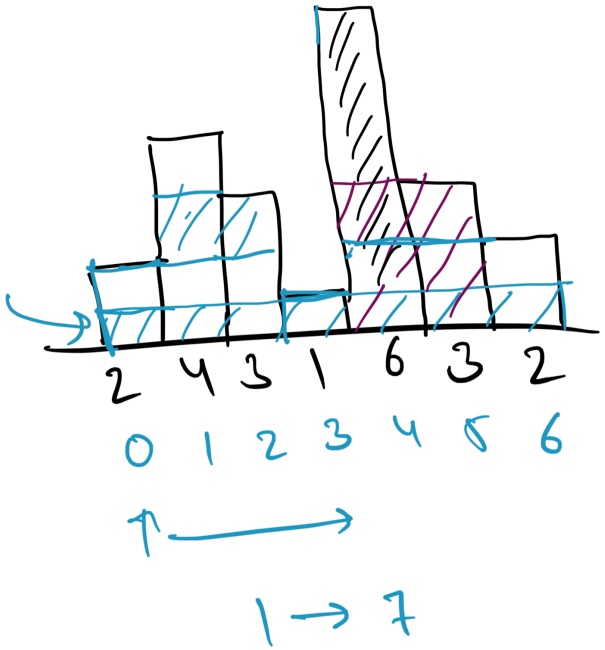
{1, 2, 3, 4, 5}
↓ Reverse
{5, 4, 3, 2, 1}

{3, 2, 1, 8, 7, 6, 5, 4}
↓
{4, 5, 6, 7, 8, 1, 2, 3}

Trapping Rain Water Problem

Q Largest Area in a Histogram :

Stack.



7x1 → 7

height = Y ≥ 3

1 → 7

Practice Problems

1. Print frequency of all the elements in a sorted Array.
2. In an Array of all 0s & 1s, find the longest length of all consecutive 1s.

   Input: 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1

   Output: 4

3. Move all 0s to the end of the given Array.

   Input: 8, 0, 1, 3, 0, 0, 5

   Output: 8, 1, 3, 5, 0, 0, 0

4. Trapping Rain water Problem in O(1) space complexity.
5. Minimum Sum Subarray Problem

a[] = 1 1 1 0 0 1 0 1 1 1 1 0 1

maxCount = 0 1 4

curCount = 0 1 2 3

0 1

0 1 2 3 4

0 1

Practice Problems

6. Print the elements in the maximum sum subarray .

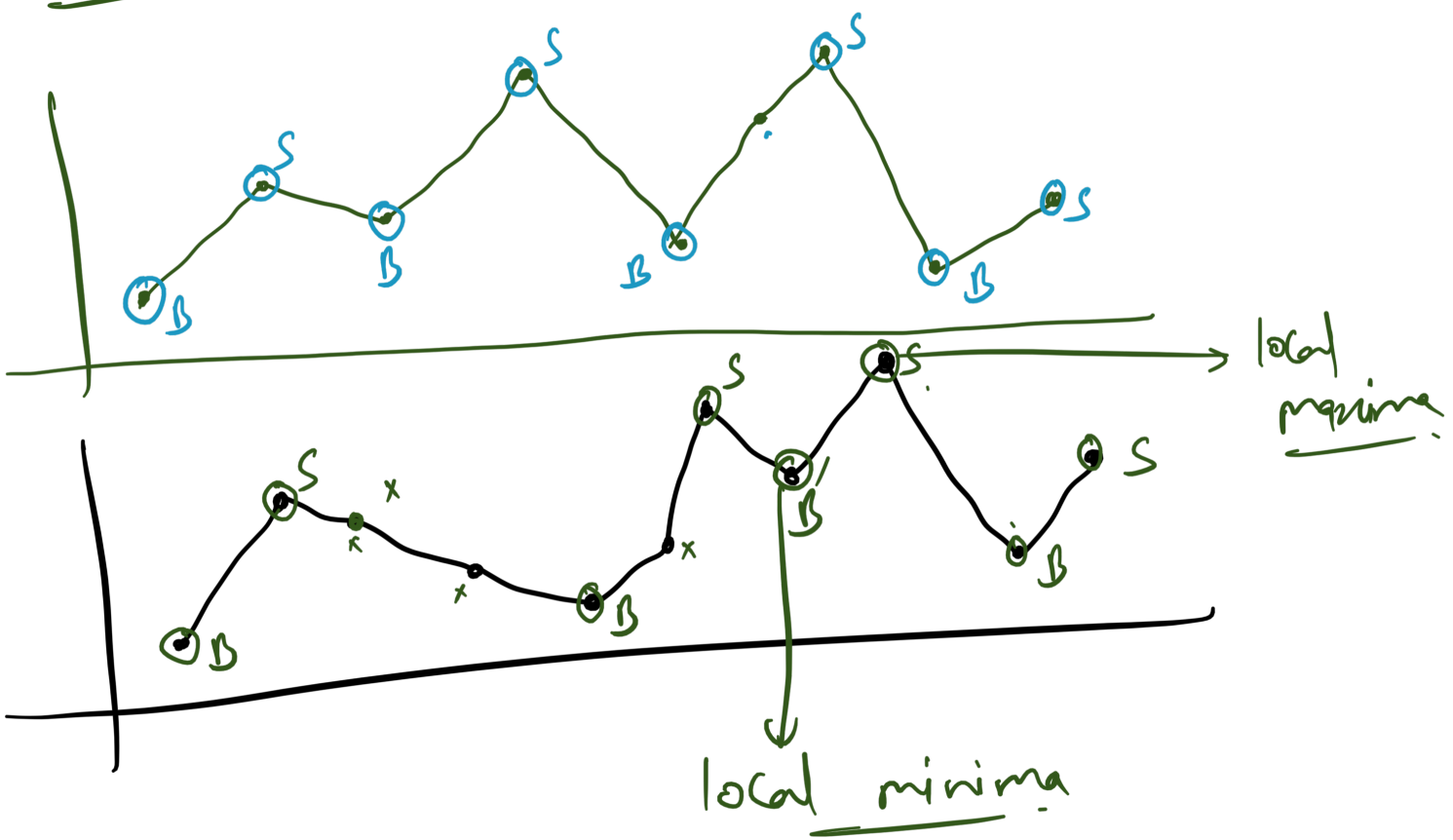
7. Stocks Buy & Sell problem

    https://leetcode.com/problems/best-time-to-buy-and-sell-stock/

    https://leetcode.com/problems/best-time-to-buy-and-sell-stock-ii/

a[] = {1, 4, 3, 7, 2, 5, 6, 1, 2}

maximum profit



a[] = { 1, 4, 3, 7, 2, 5, 6, 1, 2 }

maximum profit

buy = ~~1 3 2~~ 1

sell = ~~4 7 6~~ 2

profit = 3 + 4 + 4 + 1

→ 12.

Arrays - III

Find if there is a subarray with 0 sum

Transpose of a Matrix

Rotate a Matrix 90° clockwise

Search Element in Row-wise and Column-wise Sorted Matrix

Practice Problems

1. Split array in three equal sum subarrays.
2. Find the largest subarray with equal numbers of 0s and 1s
3. Majority Element Problem - Find the element that repeats more than n/2 times.
4. Left Rotate an Array by 1 step.
5. Left Rotate an Array by k steps.

a[] = { 0, 0, 0, 0, 1, 0, 1, 1, 0, 1 }

O(N²) — int ans=0;

```
for (int i=0; i<n; i++) {
    int zeroes
    int ones
    for (int j = i+1; j<n; j++) {
        if (a[j] = =0) zeroes++
        else ones++;
        if (zeroes == ones) {
            ans = Math.max(ans, 2* zeroes);
        }
    }
}
```

a[] = { 0, 0, 0, 0, 1, 0, 1, 1, 0, 1 }

4
-4
0

0 1 2 3 4 5 6 7 8 9

Prefix { -1, -2, -3, -4, -3, -4, -3, -2, -3, -2 }. Why?

{ 2 3, -4, 1, 2, -1 }

Sum → { 2, 5, 1, 2, 4, 3 }

-1 → 0
-2 → 1
-3 → 2
-4 → 3

0, 1
-1
0

ans = 8.

Majority Element

$a[] = \{2, 1, 3, 1, 2, 2, 2\}$

$n = 7$

$> \frac{n}{2}$

$\frac{7}{2} = 3$

HashMap

$2 \rightarrow 4$

$1 \rightarrow 2$

$3 \rightarrow 1$

Space $\rightarrow O(n)$

$O(1)$.

→ Moore's Voting Algorithm

2, 3

(7)

$\{2, 3, 2, 2, 2, 3, 3\}$

$(4 > \frac{7}{2})$ **O(N)**

[curElement = 2
count = 1]

O(nlogN).

① O(N²)

② Sort → O(NlogN).

③ HashMap → Space → O(N)

④ Moore's voting algorithm.

{1, 1, 2, 2, 2, 2, 3, 3}

count
→ curElement

Arrays - IV

Trapping Rainwater using O(1) Space Complexity

Given an array arr[], find the maximum j – i such that arr[j] > arr[i]

Find Two Numbers in a sorted Array with the Given Sum

Find Three Numbers in an unsorted Array with the Given Sum

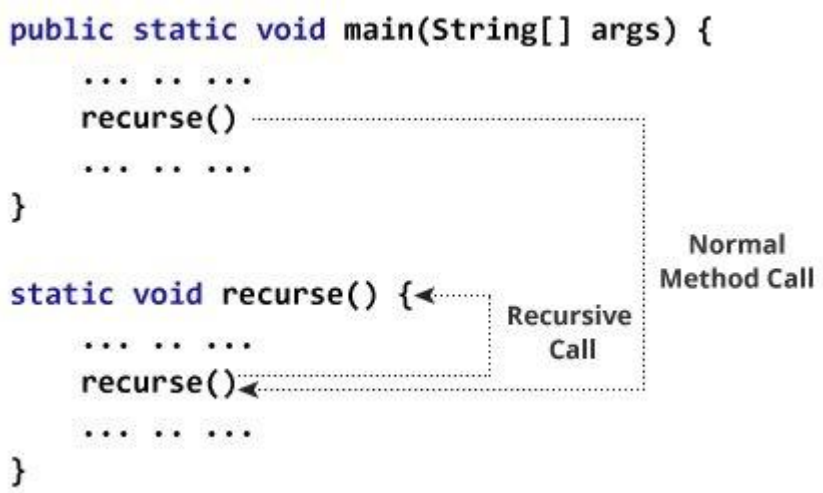
Practice Problems

https://practice.geeksforgeeks.org/explore?page=1&category[]=Arrays&sortBy=submissions

Recursion
& Backtracking - I

Recursion

The process in which a function calls itself directly or indirectly is called recursion and the corresponding function is called a recursive function. Using a recursive algorithm, certain problems can be solved quite easily.



Sum of n Natural Numbers using Recursion

Advantages and Disadvantages of Recursion

When a recursive call is made, new storage locations for variables are allocated on the stack. As, each recursive call returns, the old variables and parameters are removed from the stack. Hence, recursion generally uses more memory and is generally slow.

On the other hand, a recursive solution is much simpler and takes less time to write, debug and maintain.

Find the Power of a Number using Recursion

Find the Number of paths in an nxm Matrix

Practice Problems

1. Check if an array is a Palindrome using Recursion.
2. Factorial of a Number using Recursion
3. Find the sum of square of N Natural Numbers using Recursion.
4. Find Greatest common divisor of two numbers (GCD using Euclid Formula)
5. More Recursion Problems:
   https://www.geeksforgeeks.org/recursion-practice-problems-solutions/