Introduction To Arrays

* what is an Array

→ collection of similar type of data / Homogenous data

→ data structure which stores a collection of Homogenous (similar) items.

→ Arrays have zero based indexing.

→ Contiguous Memory Allocation.

* Representation of an Array

```
int[] ages;
String[] names;
float weights[];
```

```
DataType ArrayName[];
```

→ we can keep the square brackets infront or after the ArrayName.

* Syntax for Array Declaration

→
```
int[] ages;
ages = new int[size];
```

→ `int[] ages = new int[size];`

→ `float ages[] = new float[size];`

```
DataType ArrayName[] = new DataType[size];
```

* Array literal

with curly braces we can initialize the Array and add value to it during initialization without defining the size. (Another way of declaring single dimensional Arrays)

```
int[] a = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
```

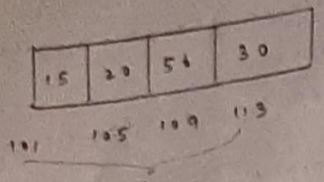
```
DataType[] ArrayName = {};
```

* Memory Allocation in Arrays

→ int[] age = new int[3];

1 Integer → 4 bytes

3 → 12 bytes

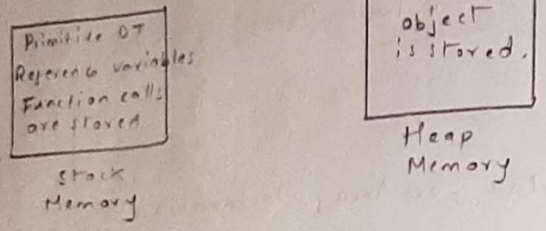
→ 

Total Memory = 4 Integers × 4 bytes = 16 bytes

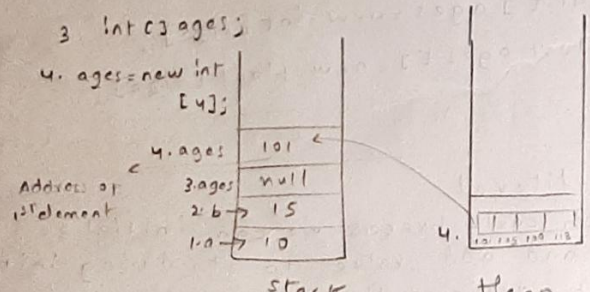
Memory addresses are Hexadecimal values

→ So Arrays have contiguous Memory Allocations because they are 101, 105, 109, 113.
105 because Int has 4 bytes
So on.

* stack and heap memory

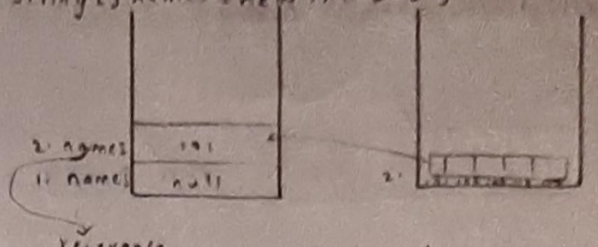


eg- 1 int a = 10;
2 int b = 15;
3 int[] ages;
4. ages = new int [4];



reference variables are in stack i.e a=10, b=15
& Actual Memory is in Heap i.e ages = new int[4]; (object)

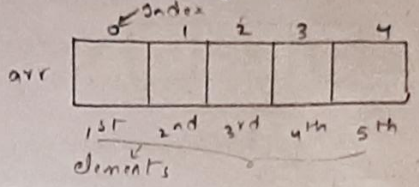
eg- 1 String[] names;
2 String[] names = new int[5];



str is pointing to some memory Allocated in the heap

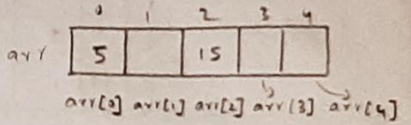
* Accessing element in 1-D Array

→ 0-based Indexing int arr[] = new int[5];



→ How to Access the elements

arr[0] = 5
arr[2] = 15



→ using System.out.println(arr[0]); we can print the element i.e 5 → (arr[0])

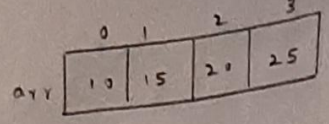
* refer demoArrays() Method in VS code

* Array Types

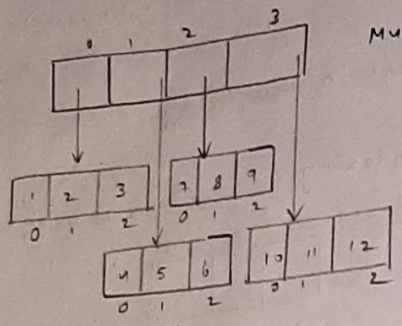
1 single Dimensional Array

2 Multi Dimensional Array

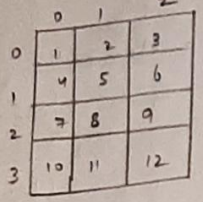
* An Array of Arrays is called as Multidimensional Array.



1-D Array

* 

Multidimensional Array



* Initializing & Accessing the elements of 2-D Array

→ int[][] a = new int[4][3];

Size = 12 elements (4×3)
48 bytes → 12×4
size of main Array
size of inside Array
int size

Datatype [][] ArrayName = new Datatype [size][size];

size of the Array's
Mandatory
optional

→ int[] a = new int[4];

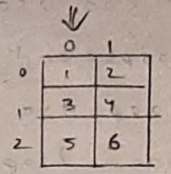
Square bracket representing that we are making an Array. & the Datatype represents Array of Integer type

→ int[][] a = new int[4][3];

sq bracket representing that we are making an Array.
int[] represents that we are making Array of Arrays.

* 2D-Array Literal

int[][] multiArray = {{1,2},{3,4},{5,6}};



* Accessing Elements of 2D Array

multiArray[0][0] = 1
multiArray[0][1] = 2
:

* refer multiArrays.cs Method in Vscode

* 3D-Array example -

int[][][] arr = new int[5][3][2];

3 elements
120 bytes

size of Main Array
size of inside Array
size of inside of inside Array

Access - arr[0][0][0] = ~;

* length operation in Arrays

eg- String name[] = new string[3];

Sout(name.length);

o/p - 3

ArrayName.length

* refer lengthu method in vscode

* Traversing through the Array

we can use loops to traverse through the Array

1. For loop
2. For each loop // it prints every value (limitation)
3. while loop

→ refer codes in vscode

* we can use length operator in loops condition block, for outer loop use ArrayName.length for inner loop use ArrayName[i].length
(i → index)

* if we have row/column of different sizes we call them as stagged Array.

* calculate the sum of all the elements in the given Array

refer code

* calculate the Maximum value out of all the elements in an Array

refer code

* Search the given element x in the Array - if present return index else return -1

refer code

- pdf Notes
- Assignment Questions

Basic Array Problems

* Taking Array Input in Java

```
Sout("Enter Array elements: ");
for(int i=0; i<arr.length; i++){
    arr[i] = sc.nextInt();
}
```

```
Sout("enter size of Array: ");
int n = sc.nextInt();
int[] arr = new int[n];
```

// Displaying elements

```
for(int i=0; i<arr.length; i++){
    Sout(arr[i] + " ");
}
```

refer code

* Array Reference in java

→ creating one more Array & copying arr elements to arr2.

→
```
int arr_2[] = arr
Sout("The copied Array elements are:");
printArray(arr_2);
```

let arr values are - 1, 2, 3, 4, 5
so, arr_2 values o/p is 1, 2, 3, 4, 5

→ we are changing arr_2 values

```
arr_2[0] = 9;
arr_2[3] = 90;
Sout("arr elements after changing arr2:");
printArray(arr);
Sout("arr_2 elements after changing arr_2");
printArray(arr_2);
```

→ we want the o/p to be

arr → 1, 2, 3, 4, 5

arr-2 → 1, 2, 3, 99, 5

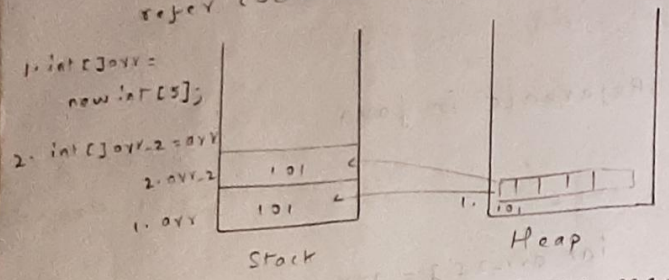
But arr & arr-2 both the elements changes after changing the values of arr-2 i.e

arr → 9, 2, 3, 99, 5

arr-2 → 9, 2, 3, 99, 5

* After changing Array2 (arr-2) the values of Array (arr) also changes because arr-2 & arr in stack are referencing to same address in heap so after changing arr2, arr also changes this is also known as shallow copying.

refer code (2)



Both are referencing to same memory in heap so after changing arr-2 elements i.e

arr-2[0] = 9;

arr-2[3] = 99;

the values of arr also changes because arr is also pointing to same memory.

int arr-2[] = arr → In this situation the copy of reference is made not the Actual Array. To prevent this use Inbuilt functions like ArrayName.clone() / Arrays.copyOf(the Array we want to copy, length of required Array) / Arrays.copyOfRange(the Array we want to copy, start index, end index)

* Pass by value difference in Array & Normal Variable

refer Vscode (4)

* using Inbuilt functions because After changing the values of Array2 Array(Actual Array) changes to solve this use Inbuilt functions.

refer code (3)

* Count the number of occurences of a particular element x.

- Take Input Array & the element x & Increment the count if x is found & return count.

(refer code)

* Find the last occurence of an element x in a given Array

- Input Array & the element x. If x is found assign that Index value to the Varible & return Variable

(refer code)

* Count the number of elements strictly greater than x.

(refer code)

* check if given Array is sorted or not.

If the current element is less than the previous element it is not sorted.

* Find Smallest & largest element of an Array with return type Array.
  - sort the Array with inbuilt function
    Arrays.sort(arr)
  - initialize the 1st & last element in an Array.
  - return Array.

HW
- Find the kth smallest & kth largest element of an Array with return type Array

* static methods can only Access static variables
* A static method can only call other static method
* static methods cant refer to non static variables or methods.

eg-

```
public class Main {
    int a = 10;
    psvm(String args[]) {
        sout(a);   //we cant Access
    }
}

public class Main {
    static int a = 10;
    psvm(String[] args) {
        Sout(a); //we can Access
    }
}
```

o/p = 10

- pdf Notes
- Assignment Questions

18

Intermediate Array Problems

Target Sum

* Find the total number of pairs in the Array whose sum is equal to the given value X.

4	6	3	5	8	2

Target 7

Ans = 4+3 = 7, 5+2 = 7 ↓ 2 pairs

hint:
- compare Array first element & remaining values.

* Count the number of triplets whose sum is equal to the given value x.

1	4	5	6	3

Target = 12

Ans = 1+5+6 = 12, 4+5+3 = 12 ↓ 2 pairs

→ 1+4+5 = 10
1+4+6 = 11
1+4+3 = 8
1+5+6 = 12
1+5+3 = 9
1+6+3 = 10
→ 4+5+6 = 15
So on

Array Manipulation

* Find the unique number in a given Array where all the elements are being repeated twice with one value being unique (only +ve elements are given in an Array)

next page

Smallest & 2nd smallest element

1	2	3	4	2	1	3
0	1	2	3	4	5	6

arr

Ans = 4

Algorithm:

- Compare first value with remaining values: if they are equal assign some value, after that the number which is not assigned print that value. (only +ve values are given in an Array so better Assign -ve value)

* Find the second largest element in the given Array

→

9	8	9	6	9	5	8
0	1	2	3	4	5	6

Max = 9
Second Max = 8

→

0	-2	0	-3	0	-4
0	1	2	3	4	5

Max = 0
Secondmax = -2

* Integer.MIN_VALUE => -2147483648 (-∞)
* Integer.MAX_VALUE => 2147483647 (∞)

These are Smallest & largest Integer values

Algorithm -

1. Find Max
2. Mark all Maximums as -∞
3. Find Max again (for second Max)

* Given an Array 'a' consisting of Integers Return the first value that is repeating in this Array if no value is being repeated, return -1.

1	5	3	4	6	2	4
0	1	2	3	4	5	6

Ans = 3

1	2	4	6	3

Ans = -1

→ If we write return statement in an method it ends the method

HW
* Return the last value that is repeating in the above problem

HW
* find second smallest element

Reversing & Rotating An Array

Basic Terminologies

* Sample Input/o/p
* Test case
* Dry-run
* optimized code → Better time & space complexity
* In-place (without using extra memory in the same Array itself)
* Edge cases
* static methods can only call other static methods

* Given 2 Integers a and b. Swap the 2 given values using temporary Variables

```
a=5 b=6
c=a
a=b
b=c
```

* Swap without using third Variable (sum & difference method)

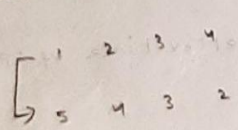
```
a=a+b   // 4+2 = 6
b=a-b   // 6-4 = 2
a=a-b   // 6-2 = 4
```

(Assignment operator Associativity is right to left)

o → original
c → changed

$a_c = a_o + b_o$
$b_c = a_c - b_o$ // $(a_o + b_o) - b_o = a_o$
$a_c = a_c - b_c$ // $(a_o + b_o) - a_o = b_o$

* Reverse an Array consisting of Integer values



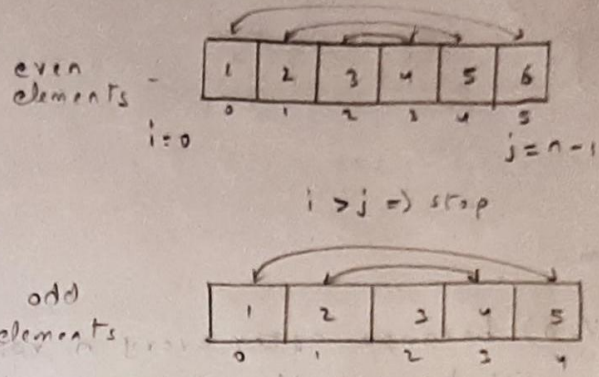
method 1 - Storing the reversed Array in New Array

- Declare the new Array & Initialize a Variable as 0. i.e j=0
- traverse the original Array from last & store the values of original Array in new Array & print the new Array.

method 2 - reversing the Array in same Array.

- traverse the Array from last & print the Array.

method 3 - reversing the Array using swapping method. (reversing in same Array)

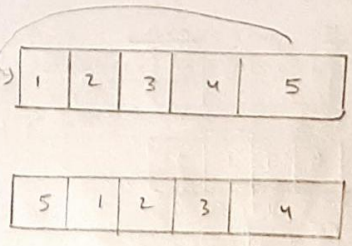


- Initialize i=0 for first element & j=n-1 for last
- Iterate a loop for the condition i<j.
- swap the values of i & j & Increment i & j.

* Rotate the Array 'a' by k steps, where k is non-negative.

note: k can be greater than n as well.

eg -



K = 2 | 4 | 5 | 1 | 2 | 3 |

K = 3 | 3 | 4 | 5 | 1 | 2 |

K = 4 | 2 | 3 | 4 | 5 | 1 |

K = 5 | 1 | 2 | 3 | 4 | 5 |

K = 6 | 5 | 1 | 2 | 3 | 4 |

.

So on

Algorithm -

- (K = K % n) because after every n rotation array will become the same as the initial Array

eg- K = K % n

(K = 0) 0 % 5 = 0 (K = 5) 5 % 5 = 0 (K = 10) 10 % 5 = 0

(K = 1) 1 % 5 = 1 (K = 6) 6 % 5 = 1 ; So on, 0

(K = 2) 2 % 5 = 2 (K = 7) 7 % 5 = 2 2

(K = 3) 3 % 5 = 3 (K = 8) 8 % 5 = 3 3

(K = 4) 4 % 5 = 4 (K = 9) 9 % 5 = 4 4

- arr: 1 | 2 | 3 | 4 | 5 | 6 | 7 (n = 7, K = 5; 0 to n-K-1; n-K; n-1)

n - K = 7 - 5 = 2 to n - 1 = 6

ans: 3 | 4 | 5 | 6 | 7 | 1 | 2

- Traverse the Array from n-K to n-1 & 0 to n-K-1 and store in the new Array & print it

- (n-K) → n = 5, K = 1

→ arr: 1 | 2 | 3 | 4 | 5 (0 1 2 3 4)

→ ans: 5 | 1 | 2 | 3 | 4 (0 1 2 3 4)

→ (n-K) 5 - 1 = 4 (last) It gives the element we want at the front, so think as n-1 gives the last index value so (n-1) (=) (n-K)

(n-2)
(n-3)
(n-4)
(n-5)

we want K steps backward so (n-K)

* Rotate the given Array 'a' by K steps, where K is non negative without using extra space (means without creating new Array to store rotate in same Array)

note: K can be greater than n as well

eg- n = 7, K = 5

arr: 1 | 2 | 3 | 4 | 5 | 6 | 7 (0 1 2 3 4 5 6)

1. P1 + P2

2 rev(p1) + rev(p2) → 2 1 7 6 5 4 3

3 rev(rev(p1)+rev(p2)) → 3 4 5 6 7 1 2

ans: 3 | 4 | 5 | 6 | 7 | 1 | 2

4 P2 + P1

Algorithm -

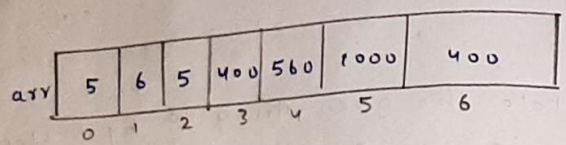
* (k=k%.n) because after every n rotation array will be same as the initial Array. become

* Divide the Array in 2 parts p1 & p2
  p1 => 0 to n-k-1
  p2 => n-k to n-1

* reverse p1 & reverse p2

* Lastly reverse the whole Array i.e 0 to n-1 (rev(rev(p1)+rev(p2)))

* Given q queries, check if the given number is present in the Array or not.
  Note: value of all the elements in the Array is less than 10 to the power 5



q=5

5	34	400	10000	560
↓	↓	↓	↓	↓
Yes	NO	Yes	No	Yes

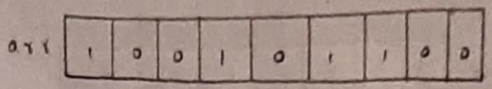
logic -

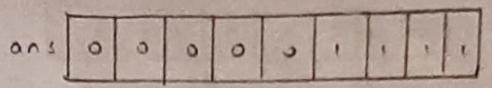
* Make a new Array name frequency of size 10^5
* for the values of original Array Increment frequency Array Indexes by 1
* enter number to be searched if the values of frequency is greater than 0 print Yes else No.

Sorting based Array Problems

* Problems based on two pointer approach

* Sort an Array consisting of only 0's & 1's. (without Inbuilt function)





method 1 -

- traverse the Array if the Array element is 0. Increment count
- Again traverse the Array & store the value 0 until the value i is less than the count value else store 1 & print the Array.

we are traversing 2 times for optimized approach use 2 pointers approach i.e

method 2 -

* Initialize the left most value as 0 & right most value as n-1. i.e
  left = 0
  right = n-1
* traverse the elements until left is less than right
* if arr[left] == 1 & arr[right] == 0 (swap) & decrement the elements & increment left++ & right--
* if arr[left] == 0 Increment left++
* if arr[right] == 1, decrement right--

* Given an Array of Integers 'a' move all the even Integers at the begining of the array followed by all the odd Integers. The relative order of odd or even Integers does not matter. return any Array that satisfies the condition.

arr | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

ans | 2 | 4 | 6 | 1 | 3 | 5 | 7 |

logic - 2 pointer Approach (refer above problem logic.

* Given an Integer Array 'a' sorted in non decreasing order, return an array of the squares of each number sorted in non decreasing order.

(non-decreasing / Increasing both are similarly same)

non decreasing order | 1 | 2 | 2 | 4 |

Increasing order | 1 | 2 | 3 | 4 |

non decreasing order elements are used to show that they can have repeated numbers

eg 1-

arr | 2 | 4 | 6 | 7 | 10 |

ans | 4 | 16 | 36 | 49 | 100 |

eg 2 -

arr | -10 | -9 | -3 | 8 | 12 | 52 |

Sq. arr | 100 | 9 | 4 | 1 | 16 | 25 |

0 1 2 3 4 5

ans | 1 | 4 | 9 | 16 | 25 | 100 |

* The largest square is present at the beginning of the Array or at the end compare these 2 values, which one is greater store that value in new Array & Increment that Array.

* finally sort the Array (i.e reverse the Array.

* use Math.abs() function because we also have negative values => abs(-x) = x
abs(x) = x

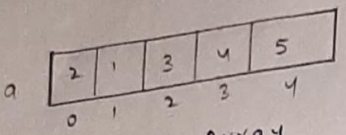
* Instead of reversing the Array we can also use Instead of k = 0 & Instead of k = n-1 or k++ use k--.

21

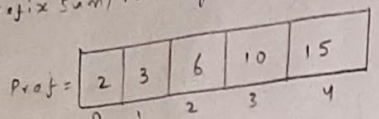
Prefix Sum Array Problems

☆ Problems based on prefix sum Approach

* Given an Integer Array 'a', return the prefix sum / running sum in the same array without creating a new Array



→ with creating a new Array
Prefix sum / running sum Array



* The prefix sum of 2nd index is -
  Index - 0 + 1 + 2
  2 + 1 + 3 = 6

  Ily 3rd index is -
  2 + 1 + 3 + 4 = 10

  and so on

pref[0] = a[0]

pref[1] = a[0] + a[1] pref[0]

pref[2] = a[0] + a[1] + a[2] pref[1]

pref[3] = a[0] + a[1] + a[2] + a[3]

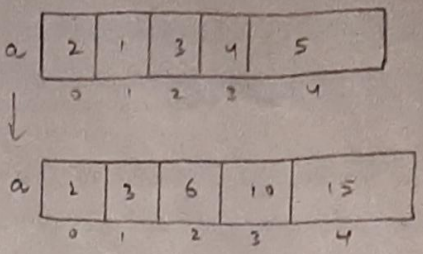
pref[4] = a[0] + a[1] + a[2] + a[3] + a[4]
= pref[3] + a[4]

pref[2]

For any i >= 1

$$pref[i] = pref[i-1] + a[i]$$

→ without creating a new Array.



a[0] = a[0]

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

a[2] = a[1] + a[2]

a[3] = a[2] + a[3]

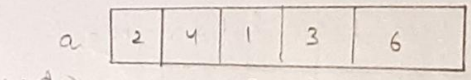
a[4] = a[3] + a[4]

For any i >= 1

$$a[i] = a[i-1] + a[i]$$

* Given an Array of Integers of size n. Answer q queries where you need to print the sum of values in a given string range of indices from l to r both included)

  Note: the values of l and r in queries follow 1-based Indexing.



(1-based Indexing)

a[0] = 0 (If exists) but the user doesn't take the I/p

1. l = 1, r = 3
   2 + 4 + 1 = 7

2. l = 2, r = 5
   4 + 1 + 3 + 6 = 14

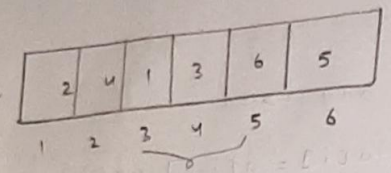
3. l = 2 r = 2
   4.

* Brute force Approach (Basic Approach)

```
for(i=l; i<=r; i++){
    sum = sum + arr[i];
}
```

This Answer gives us somew correct Answer but if we have somany queuries(eg) we have to traverse the Array 2 no of times O(q*n) complexity (It is not correct Approach) we have to think better approach

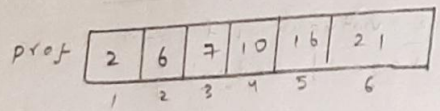
* optimized Approach



=> l=3 r=5

1+3+6=10

But the condition is without traversing we want the sum from l to r.



we want sum from l to r i.e -

Sum = pref[r] - pref[l-1]

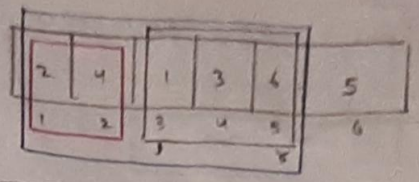
eg - pref[5] - pref[3-1]

16 - 6 = 10

-> 3 to 5

pref[5] - pref[2]

5 - 2 = 3

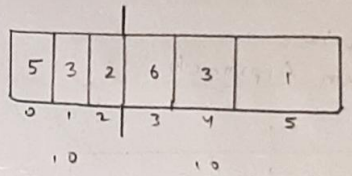


* we want the sum of Block part, so from blue part subtract red part to get the sum.

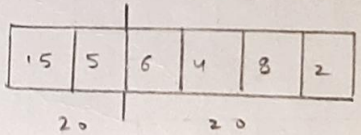
i.e

pref[r] - pref[l-1]

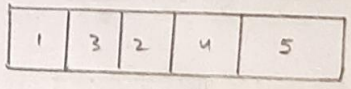
* check if we can partition the Array in to two subarrays with equal sum. More formally check that the prefix sum of a part of the array is equal to the suffix sum of rest of the Array.

a 

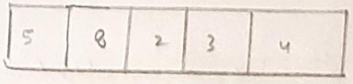
true

b 

true

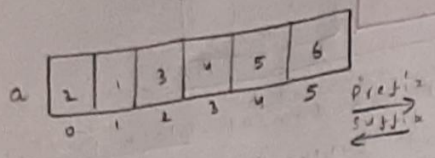
c 

false

d 

false

prefix sum & suffix sum Definition



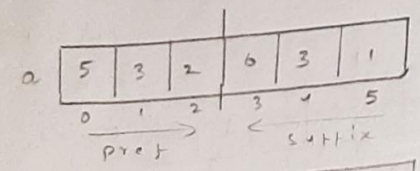
prefix sum of index $i = \sum_{j=0}^{i} pref[i] = a[0] + a[1] + \ldots a[i]$

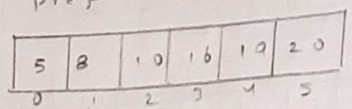
suffix sum of index $i = \sum_{j=i}^{n-1} suff[i] = a[i] + a[i+1] + \ldots a[n-1]$

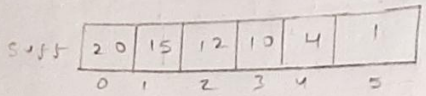
pref = 2 3 6 10 15 21

suff = 21 19 18 15 11 6

eg- prefix & suffix Sum Approach







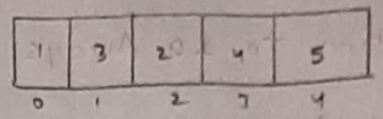
if pref[i] == suff[i+1]

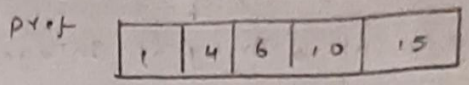
true

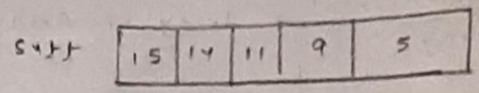
else

false

eg 2 -







pref[i] == pref[i+1]

=> false

Method 2 -

* we can find the suffix sum using

Suffix sum = Total sum - prefix sum

Algorithm -

* find the totalsum of the Array.
* traverse a loop and Add 1st element to the prefix sum i.e prefixSum = prefixSum + arr[i]
* find suffixSum i.e suffixSum = totalSum - prefixSum.
* If prefixSum == SuffixSum return true else return false.

HW

* Find suffix Sum Array

a = 2 5 6 1 3

Suffix Sum Array