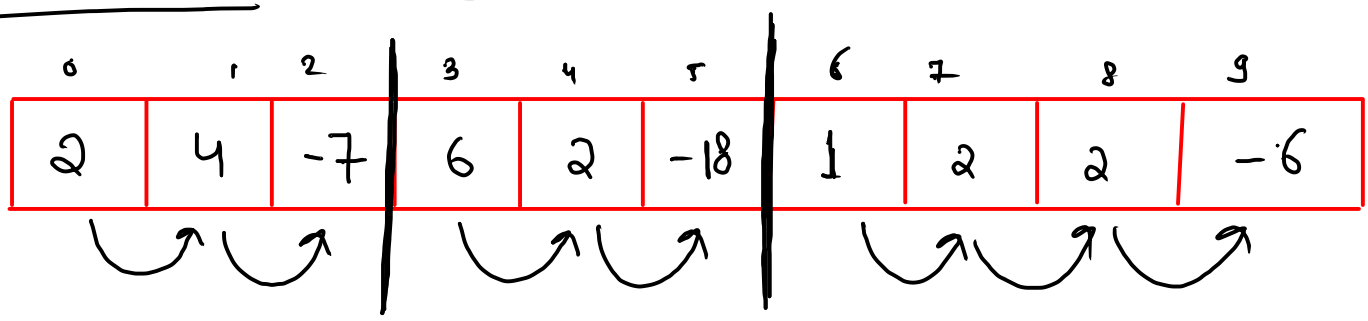
Revision :-

- Sorting , lambda function
- arrays, subarray, Kadane's algo
- 2 pointers
- Prefix array
- Arrays as hashmap
- 2d array
- String & substring
- Binary Search (BSLB & BSUB)
- ArrayList
- Stacks
- Hashmap
- Queue
- PQ

Kadane's algo



maximum_sum = ~~-∞~~ ~~2~~ ~~6~~ 8

sum_so_far = ~~2~~ ~~6~~ ~~-1~~ ~~6~~ ~~8~~ ~~-10~~ ~~1~~ ~~3~~ ~~5~~ -1

code

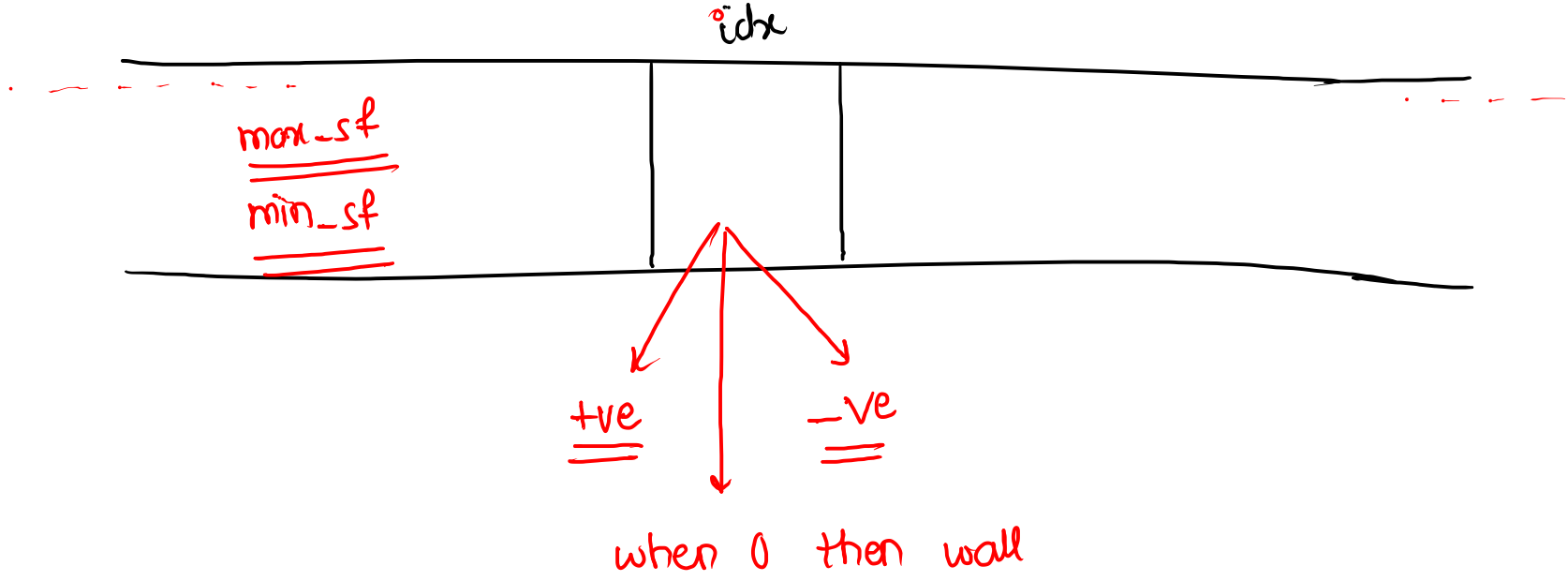
```java
public static int kadanesAlgo(int[] arr, int n) {
    int maxSum = Integer.MIN_VALUE;
    int sumsf = 0;
    for (int i = 0; i < n; i++) {
        if (sumsf < 0) {
            sumsf = arr[i];
        } else {
            sumsf = sumsf + arr[i];
        }
        if ( sumsf > maxSum ) {
            maxSum = sumsf;
        }
    }
    return maxSum;
}
```

make it starting

make it better

Maximum Product Subarray 2

arr [2 , 3 , -4 , 5]



Code

```java
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int n = scn.nextInt();
    int[] arr = new int[n];
    for (int i = 0; i < n; i++) {
        arr[i] = scn.nextInt();
    }

    System.out.println(kadanesAlgo(arr, n));
}

public static int kadanesAlgo(int[] arr, int n) {
    int maxsf = 1;
    int minisf = 1;
    int result = Integer.MIN_VALUE;
    for (int i = 0; i < n; i++) {
        int temp = maxsf;
        maxsf = Math.max( arr[i], Math.max( maxsf * arr[i], minisf * arr[i] ) );
        minisf = Math.min( arr[i], Math.min( temp * arr[i], minisf * arr[i] ) );
        result = Math.max( result, maxsf );
    }
    return result;
}
```

(curr , curr * max, curr * min)

⇒ 2 pointers

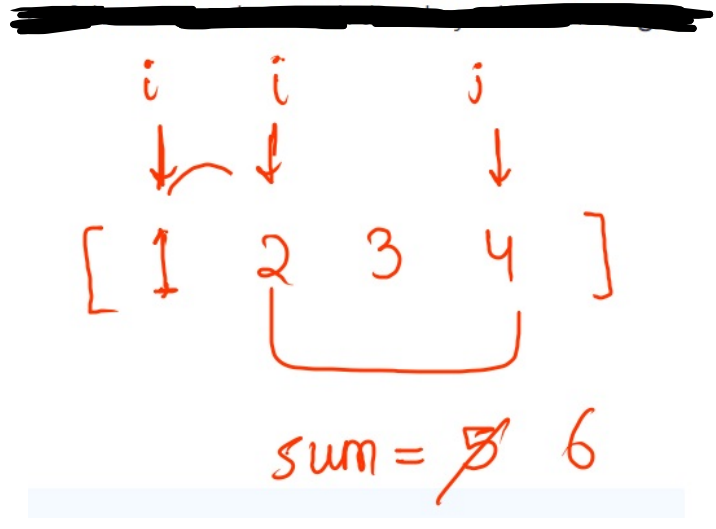
↳ sort 01

↳ sort 012

target = 6

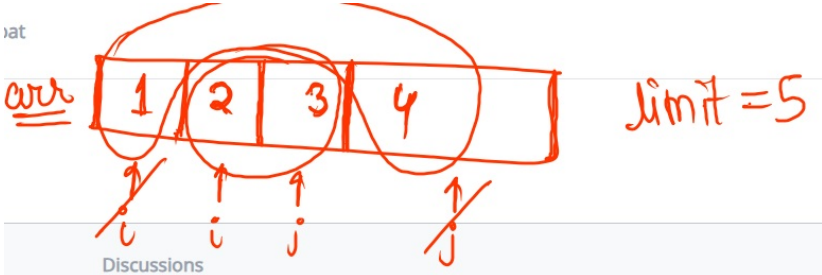
Discussions

Reach Target



Code

Count boat

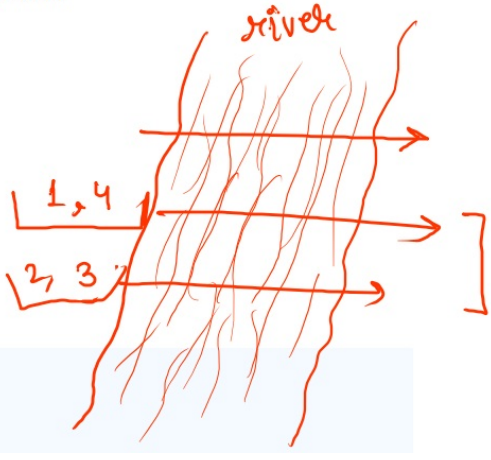


Discussions

veight of the **ith** person, and an infinite number of
mit. Each boat carries at most **two people** at the
le is at most limit.

n person.

of array.



Submissions: 60
Max Score: 10
Difficulty: Medium

Rate This Challeng

More

```java
public static int countBoats(int[] arr, int n, int limit) {
    Arrays.sort(arr);
    int i = 0;
    int j = n - 1;
    int count = 0;
    while (i <= j) {
        int sum = arr[i] + arr[j];
        if ( sum > limit ) {
            j--;
        } else {
            i++;
            j--;
        }
        count++;
    }
    return count;
}
```

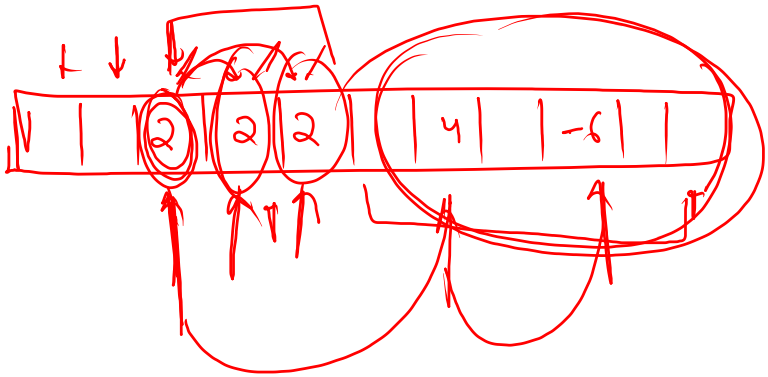
3 Sum

$$arr[i] + arr[j] + arr[k] == 0$$

$$arr[i] + arr[j] == -1 * arr[k]$$

target

sorted

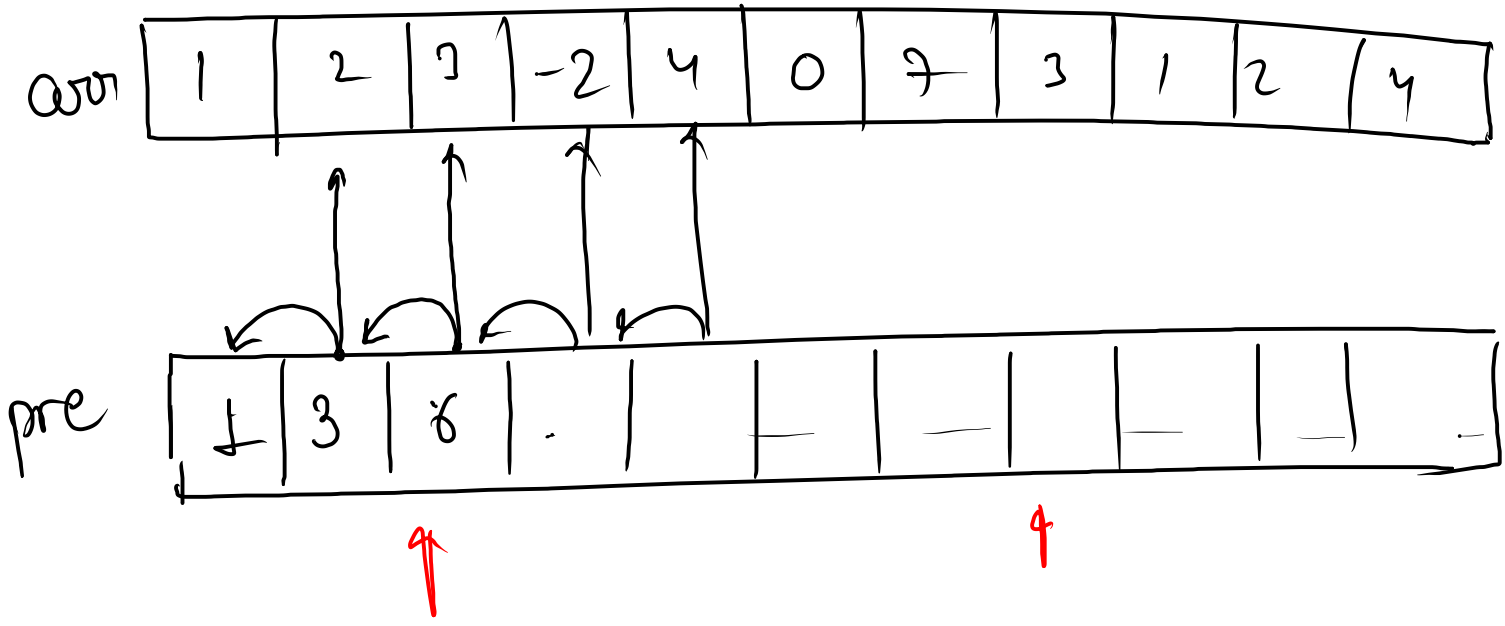


code

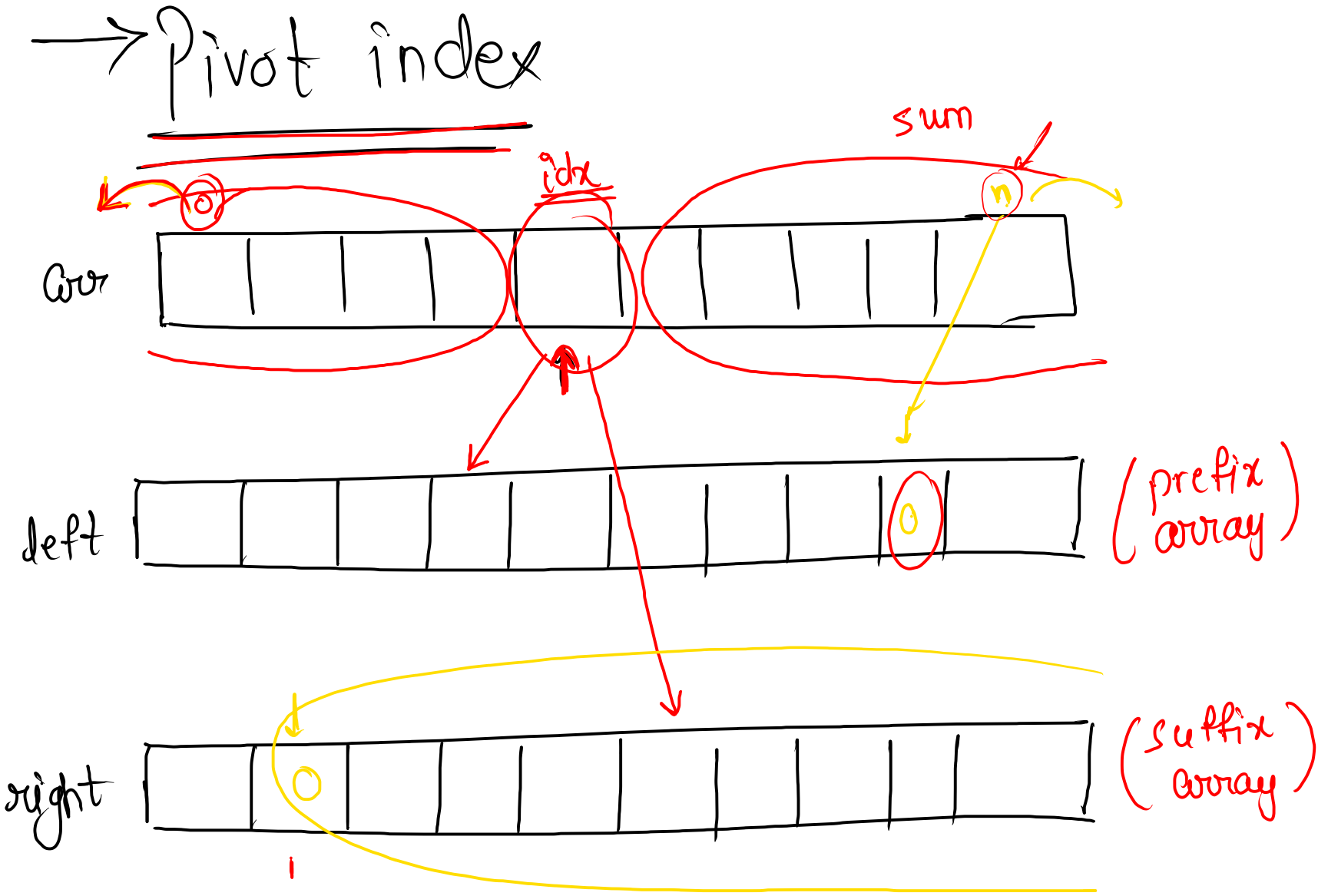
```java
public static void threeSum(int[] arr, int n) {
    Arrays.sort(arr);
    for (int k = 0; k < n; k++) {
        int target = -1 * arr[k];
        int i = k + 1;
        int j = n - 1;
        while ( i < j ) {
            int sum = arr[i] + arr[j];
            if ( sum == target ) {
                System.out.println(arr[k] + " " + arr[i] + " " + arr[j]);
                i++;
                j--;
            } else if (sum < target) {
                i++;
            } else {
                j--;
            }
        }

        while ( k + 1 < n && arr[k] == arr[k + 1] ) {
            k++;
        }
    }
}
```

to handle duplicates

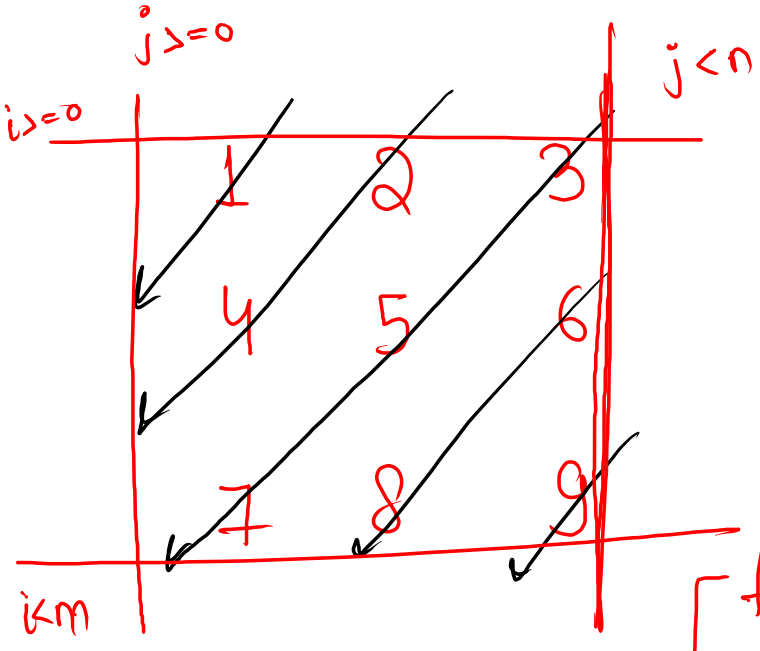
⇒ prefix sum array



→ Pivot index



Print the matrix left-diagonal wise



(0,0)

(0,1)

(0,2)

i j

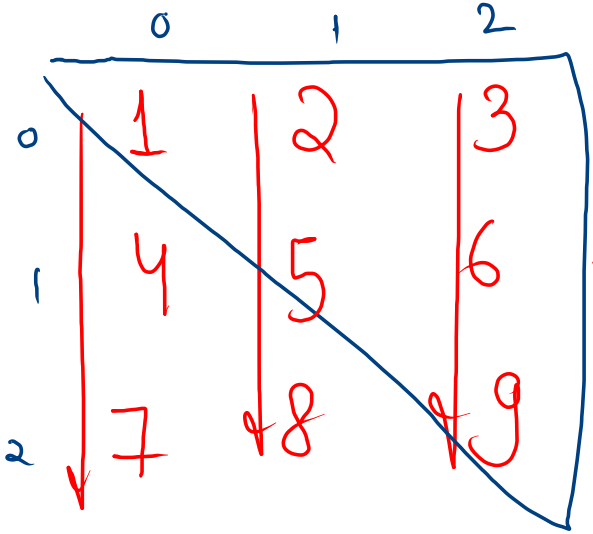
```
for ( int gap = 0 ; gap < arr[0].len ; gap++){
    for( int i=0, j=gap; j>=0; i++, j--){

    }
}
```

Transpose of Matrix of N*N

arr

transpose

```
for(i = 0 → N)
    for(j = 0 → N)
        if(i <= j)
```



1 4 7

2 5 8

3 6 9

i <= j

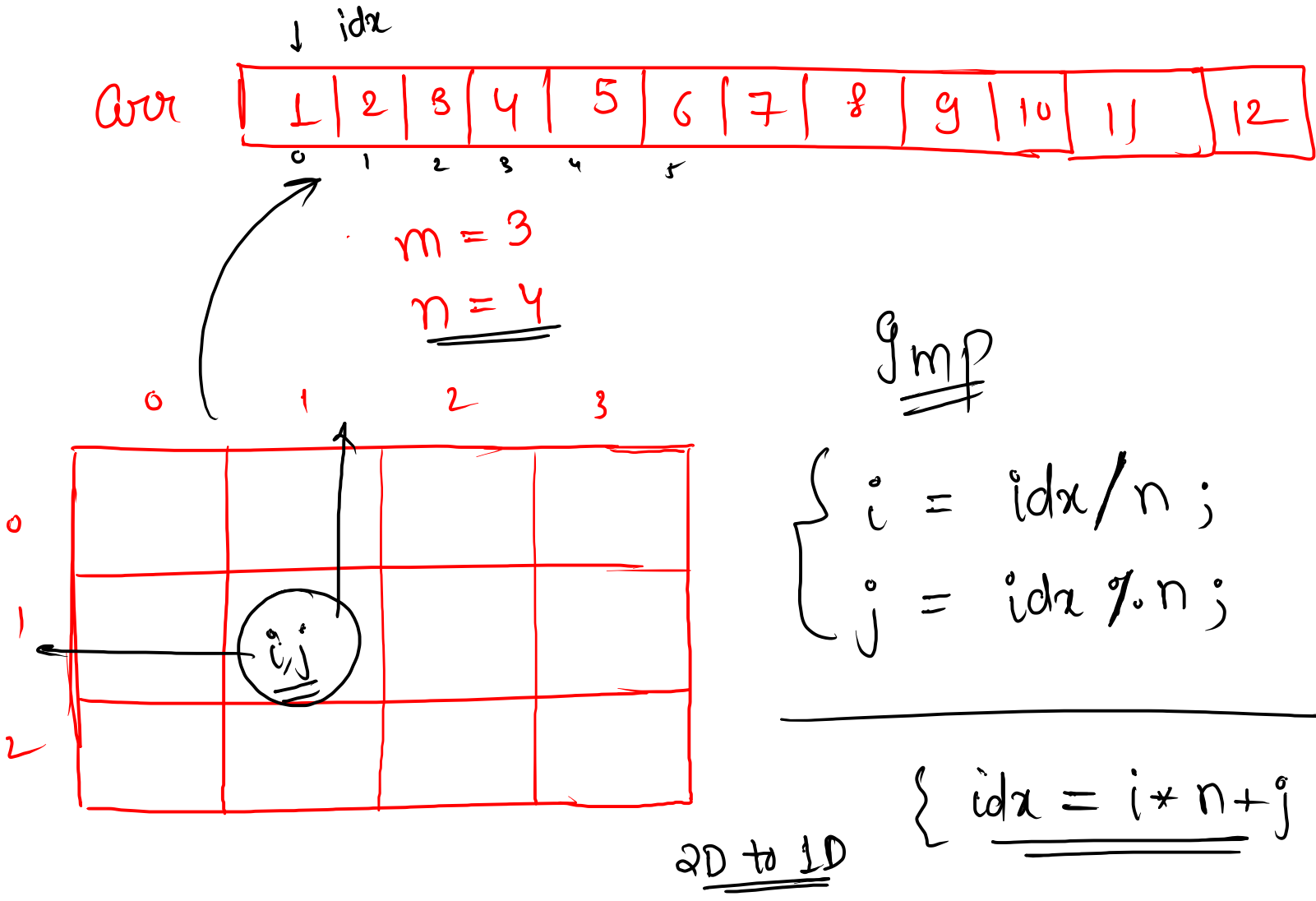
i j

[0,0 1,1 2,2

0,1 1,2

0,2]

Convert 1-D Array to 2-D Array



Imp

$$\begin{cases} i = idx / n; \\ j = idx \,\%\, n; \end{cases}$$

$$\{ idx = i * n + j$$

2D to 1D

StringBuilder sb = new StringBuilder();

- sb.append(x);
- sb.delete(charAt(idx));
- sb.reverse();
- sb.charAt(idx);

Locate the Target String

Imp

str = "geekster"

target = "st"

```
public static int locateTarget(String str, String tar) {
    for (int i = 0; i <= str.length() - tar.length() ; i++) {
        for (int j = 0; j < tar.length(); j++) {
            if ( tar.charAt(j) != str.charAt(i + j) ) {
                break;
            }

            if ( j == tar.length() - 1 ) {
                return i;
            }
        }
    }
    return -1;
}
```