ARRAYS - 1

General Steps to solve DSA questions:-

1. Understand the problem statement without any ambiguity.
2. Come up with a brute force approach.
3. List down the observations that can help us optimize time and space complexity.
4. Come up with the optimized approach.

Q Given an integer array A, find max value of →

$f(i,j) = A[i] - A[j] \quad \forall (i,j)$

Sol
1. Find the minimum of the array.
2. Find the max of the array.
3. Subract.

} TC: O(N) SC: O(1)

1. Sort the array
2. Take the difference b/w the 0th & last element

} TC: O(NlogN)
SC: O(~~NlogN~~) for sorting

```
public int solution(int[] A, int n){
        int  ma = Integer.MIN_VALUE;
        int  mi = Integer.MAX_VALUE;
        for(int i : A){
                mi = Math.min(mi, Ai);
                ma = Math.max(ma, i);
        }
        return (ma-mi);
} // TC: O(N)  SC: O(1)
```

Q. Given an array of integers find the max value of the fn $|A[i] - A[j]| + |i-j|$ where $|x| \to \begin{cases} \geq x & \text{if } x \geq 0 \\ \geq -x & \text{if } x < 0 \end{cases}$

Sol $|A[i] - A[j]| + |i-j|$

Without losing loss of generality we can assume $i \geq j$

Since $f(i,j) = f(j,i)$

So, we can remove the mod for the right side

$|A[i] - A[j]| + i - j$

Two cases here $\to$ $A[i] > A[j]$ or $A[j] > A[i]$

ie., $A[i] - A[j]$ could either be +ve or -ve.

Case (i)

$A[i] - A[j] + i - j$

$\underbrace{(A[i] + i)}_{\text{MAXIMUM}} - \underbrace{(A[j] + j)}_{\text{MINIMUM}}$

Case (ii)

$|A[j] - A[i]| + i - j$

$A[j] - A[i] + i - j$

$\underbrace{(A[j] - j)}_{\text{MAXIMUM}} - \underbrace{(A[i] - i)}_{\text{MINIMUM}}$

return max (case (i) & case (ii))

Note

We can assume $i > j$ because $f(i,j) = f(j,i)$ and we don't need to compute the same thing twice.

TC = O(N)

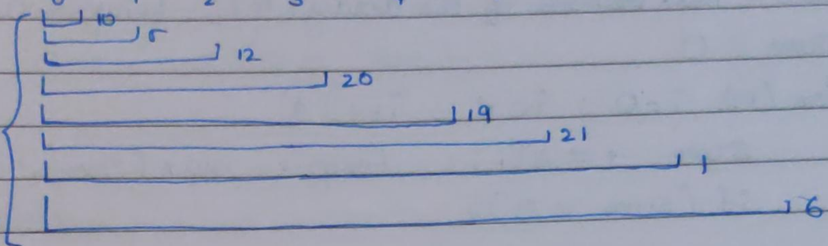
SC = O(1)

Q. Given an integer array A, find max subarray sum that starts from index 0.

Sol:- Subarray → continous part of the array.

A → [10, -5, 7, 8, -1, 2, -20, 5]
0 1 2 3 4 5 6 7

Subarrays under consideration



In the example above, the answer should be 21.

Thoughts :-
- Keep track of curr sum and max.
- Solve it in O(N), O(N) TC & SC respectively
- This idea is called prefix sum.

```
int curr_sum =0, max_sum = A[0];
for (int i=0; i<N; i++){
    curr_sum += A[i];
    if (curr_sum > max_sum){
        max_sum = curr_sum;
    }
}
```

NOTE: Possible no. of subarrays that could be formed with an array of size N

$$= \Sigma N = \frac{N(N+1)}{2}$$

Page No.
Date

Q Find max subarray sum in the given array

Sol Kadane's Algorithm

$A \rightarrow [10, -5, 7, 8, -1]$

```
ans = max element of the array (A); temp = ans
sum = 0
for (int i=0 ; i<N ; i++) {
    sum += A[i];  temp = max(sum, temp);
    if (sum < 0){
        sum = 0; }
    ~~ans temp = max(sum, temp);~~
return max(temp, ans);
```

X

```
ans = INT_MIN;
sum = 0;
for ( int i=0 ; i<N ; i++) {
    sum += A[i];
    ans = max(sum, ans);
    if (sum < 0)
        sum = 0;
}
return ans;
```

TC = O(N)
SC = O(1)

Brute Force :- $\underbrace{\forall \text{ subarrays}}_{O(N^2)}$, $\underbrace{\text{calculate sum}}_{O(N)}$ of each subarray $= O(N^3)$

Page No.
Date

Optimize it further to make it $O(N^2)$

```
ans = INT_MIN
for(int i=0; i<N; i++){
    sum = 0;
    for(int j=i; j<N; j++){
        sum += A[j];
        ans = max(sum, ans);
    }
}
return ans;
```

TC = $O(N^2)$
SC = $O(1)$

Observations:-

1) If all the elements are +ve, ans = sum of all the elements.

2) If all the elements are -ve, ans = max element of the array.

3) Cannot do sorting as it is not subset & we need indices for subarray.

How to find the subarray itself?

```
if (sum < 0) {
    start = i; end = i;
    sum = 0;
} else {
    end ++;
}
```

Page No.
Date

*** Q Given a binary array A, $\forall i$ A[i] = 0 or 1, find maximum count of 1's we can achieve in the array by flipping the bits of atmost on subarray.

Sol <u>Bruteforce :-</u>

$\underbrace{\forall \text{ subarrays,}}_{O(N^2)}$ $\underbrace{\text{flip bits}}_{O(N)}$ and calculate 1s $\rightarrow O(N^3)$

We can keep counters and further optimize this to $O(N^2)$

<u>Observations</u>

1) $\forall i$, A[i] = 1 => ans = N
2) $\forall i$, A[i] = 0 => ans = N (flip all the elements)
3) Flipping a 0 contributes +1 to the ans.

4) But flipping a ~~zero~~ one is not taking away. It is just reducing the 1.

5) What if we could find a subarray that contributes the most?
Flip 1's to -1 & 0's to 1's.

$$ANS = (\text{no. of 1s initially}) + \underbrace{(\text{max contribution})}_{\text{Maximum Subarray Sum}}$$

```
if (A[i] == 1)
    sum += -1
else
    sum += 1
```

Page No.
Date

**Q. Given an integers array A, $\forall i$ $A[i] = 0$;
Return the final subarray after performing all queries

Queries $\rightarrow (i, x) \Rightarrow$ Add x to all element from $A[i]$ --- $A[N-1]$
$0 <= i < N$

Sol $Q \rightarrow \{(1, 3), (4, 2), (3, 1)\}$

$A \rightarrow [0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0]$

Use the concept of prefix sum.

$A \rightarrow [0 \quad 3 \quad 0 \quad 1 \quad 2 \quad 0 \quad 0]$

prefix $\rightarrow [0 \quad 3 \quad 3 \quad 4 \quad 6 \quad 6 \quad 6]$

TC $\rightarrow O(Q + N)$
SC $\rightarrow O(1)$

Q. Queries $\rightarrow (i, j, x)$ Add x to all elements from $A[i]$ to $A[j]$

Sol 1. Do the same thing as above
$\rightarrow$ Add x at i in A
2. Subtract
$\rightarrow$ Sub x at $j+1$ in A
check for bounds here.
3. Prefix Sum

TC $= O(Q+N)$
SC $= O(1)$