1D Arrays:

P1
- Beggers Problem part 1 ✓
- Beggers Problem part 2 ✓

P2
- Max absolute difference part 1 ✓
- Max absolute difference part 2 ✓

P3
- Max subarray sum part 1 ✓
- Max subarray sum part 2 ✓ (Kaden's Algorithm)

* the pace of class is faster in advanced DSA

* we do not always write all the codes. it will be left as assignments.

P1.1 Given an integer array A, (A[i] is initialy 0 for all i) return the final array after processing multiple queries:

prequel

Query(i, x) → add x to all the numbers from index i to n-1

ex a[]:

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

Query

(i, x)

(1, 3)

(4, 2)

(3, 1)

output: 0 3 3 4 6 6 6

(1, 3)
(3, 1)
(4, 2)

$Q \log Q + N$

idea1

TC: $O(Q \times N)$ SC: $O(1)$

idea2

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

Query

(i, x)

(1, 3)

(4, 2)

(3, 1)

I	X	
1	3	0
4	2	1
3	1	2

```
int[] Qrery(int a[], int I[], int X[]){
    n=a.len
    int q = I.len   // or X.len
    for(i=0; i<q; i++){
        ind = I[i];  x = X[i]
        a[ind] += x
    }
    for(i=1; i<n; i++){
        a[i] += a[i-1]
    }
    ret a
}
```

TC: $O(Q+N)$

SC: $O(1)$

P1.2 Given an integer array A, (A[i] is initialy 0 for all i) return the final array after processing multiple queries:

Query(i, j, x) → add x to all the numbers from index i to j

ex a[]:

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

Queries

(1, 3, 2)

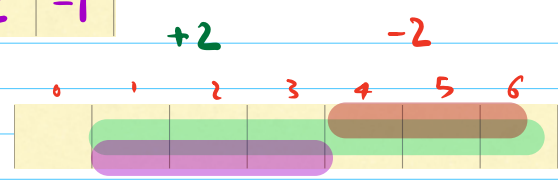
(2, 5, 3)

(5, 6, -1)

output

0	2	5	5	3	2	-1

(i, j, x) → (i, x), (j+1, -x)



TC: O(2Q+N) ~ O(Q+N)

SC: O(1)

P2.1 Given an integer array A, find max value of $f(i,j)$

max $f(i,j) = A[i] - A[j]$ for all i, j

ex $a[] = \{ \overset{0}{1} \quad \overset{1}{3} \quad \overset{2}{-2} \}$

i	j	A[i]-A[j] → f(i,j)
0	0	0
0	1	-2
0	2	3
1	0	2
1	1	0
1	2	5 → ans
2	0	-3
2	1	-5
2	2	0

idea1

TC: $O(n^2)$

SC: $O(1)$

idea2 $f(i,j) = A[i] - A[j]$

one for loop

find max

min

TC: $O(N)$

SC: $O(1)$

$|-3| = 3$

$|+7| = 7$

$$|x| = \begin{cases} x & x >= 0 \\ -x & x < 0 \end{cases}$$

P2.2 Given an integer array A, find max value of $f(i,j)$

max $f(i,j) = |A[i]-A[j]| + |i-j|$ for all i, j

ex $a[] = \{ \overset{0}{1} \quad \overset{1}{3} \quad \overset{2}{-2} \}$

brute force

TC:

i	j	① $\lvert A[i]-A[j]\rvert$	+ ② $\lvert i-j\rvert$	$f(i,j)$
0	0	0	0	0
0	1	+2	1	3
0	2	3	2	5
1	0	2	1	3
1	1	0	0	0
1	2	5	1	6
2	0	+3	2	5
2	1	+5	1	6
2	2	0	0	0

ans

observations

① $i == j \rightarrow f(i,j) = 0$

② $f(i,j) == f(j,i)$

(*) $i < j$

MAX

optimize: $f(i,j) = |A[i] - A[j]| + (i-j) \quad i > j$

$A[i] >= A[j]$

$f(i,j) = A[i] - A[j] + i - j$

$= \underbrace{A[i] + i}_{X_i} - \underbrace{(A[j] + j)}_{X_j}$

$A[k] + k = X_k$

$\max f(i,j) = X_i - X_j$

$X_i \to$ max, $X_j \to$ min

$A[i] < A[j]$

$f(i,j) = -A[i] + A[j] + i - j$

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

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

$Y_k = A[k] - k$

$f(i,j) = Y_j - Y_i$

$Y_j \to$ max, $Y_i \to$ min

```
for (k=0; k<n; k++){
    X = a[k]+k;     Y=a[k]-k
    maxX = math.max(X, maxX)
    minX = math.min(X, minX)
    maxY = math.max(Y, maxY)
    minY = math.min(Y, minY)
}
ret math.max(maxX-minX, maxY-minY)
```

TC: O(n)

SC: O(1)

break?

P3.1 Given an integer array a[], find the max subarray sum for subarrays starting from index 0.

ex a[] = { 10 | -5 | 7 | 8 | -1 | 2 }

0	1	2	3	4	5
10	-5	7	8	-1	2

10

5

12

20

19

21 → ans

```
int maxSubarraySum1(int a[])
    n = a.Len;   sum = 0;   ans = INT_MIN
    for(i=0; i<n; i++){ // Carry forw
        sum += a[i]
        ans = math.max(ans, sum)
    }
    ret ans
}
```

TC: O(n)
SC: O(1)

P3.2 Given an integer array a, find the max subarray sum for all the subarrays.

ex $a[] = \{$

0	1	2	3	4	5
10	-5	7	8	-1	2

$\}$

observations

① if a[ ] for all i $a[i] >= 0 \rightarrow \sum a[i]$ ✓

② if a[ ] for all i $a[i] < 0 \longrightarrow$ max of $a[i]$ ✓

-3 -1 -7

③ $\boxed{\underbrace{\ldots, 3, -5}}, \boxed{7, 8, 1, 2} \ldots$

Kaden's Algorithm

i

	0	1	2	3	4	5	6	7	8	9	10	11	12
	10	-5	7	8	-11	2	-20	10	-3	-10	15	8	-10
sum	10	5	12	20	9	11	-9 / 0	10	7	-3 / 0	15	23	13
ans	10	10	12	20	20	20	20	20	20	20	20	23	23 → ans

```
int maxSubarraySum2(int a[])
    n = a.Len
    ans = a[0]
    sum = 0
    for(i=0; i<n; i++){
        sum += a[i]
        ans = max.max(ans, sum)
        if(sum<0) sum = 0
    }
ret ans;
}
```

TC:

SC:

$A = [-3, -8, -1, -5]$

		0	1	2	3
sum	0	~~-3~~ 0	~~-8~~ 0	~~-1~~ 0	~~-5~~ 0
ans	-3	-3	-3	-1	-1

i