Arrays — Linear DS.

Insert (0,0).

Contiguous Memory Allocations

Random Access : O(1)
Insert at end : O(1)
Insert at beginning : O(n)
Delete at end : O(1)
Delete at beginning : O(n)

arr: [2, 3, 4]

o/p = 1

arr: [2, 3, 5]

Simple search: O(n)

BF: TC: O(n²)

arr: [0, -10, 1, 3, -20] sort → [-10, -10, 0, 1, 3]

TC: O(n log n)
AS: O(1)

arr: [0, -10, 1, 3, -20]

isPresent:

TC: O(n)
AS: O(n)

n → neg [1, n+1]

arr: [-2, -5, -3, -2, -4, 1, -7]

Index for 1
Index for 2
Index for 7

n = 7

i is the index for (i+1)

TC: O(n)
AS: O(1).

n = 5

[1, n]

arr: [0, -10, 1, 3, -20].

O3 & -ves with neg

arr: [6, 6, -1, 3, 4]

TC: O(n+m)
AS: O(n+m)

TC: O(n*m)
AS: O(1)

TC: O(n*m)
AS: O(1)

[2, 3, 5, 1, 7, 8]

Substring: [2, 3, 5] [5, 1] [7] [] [5] [8] [5, 8]

Subsequence: [2, 5] [5, 8] [2, 3, 5], [], [7]

[1, 2, 3, -2, 5]: [] [1] [1,2] [2] [2,3] [3] [1,2,3] [3,-2] [2,-2]

```
for i in [0, n)
    sum=0
    for j in [i, n)
        sum += arr[j]
        res = max (res, sum)
```

TC: O(n²)
AS: O(1)

[1, 2, 3, -2, 5]

maxSum: 1 3 6 4 9

TC: O(n)
AS: O(n)

maxSum[i] = (arr[i], maxSum[i-1] + arr[i])

[1, 2, 3] → [1, 2, 3, 1, 2, 3]

[-1, 2, 3, -5] → 5

case-1: Circular property never used : Kadane's → sum1

case-2: Circular property is used → sum2

TC: O(n)
AS: O(1)

[8, -8, 9, -9, 10, -11, 12]

[-8, 8, -9, 9, -10, 11, -12]

maxSum = 11

minSum = -11

overall Sum = 11

sum2 = overallSum − minSum
= 11 − (-11) = 22