String str1 = "Karthik" , a+b , "47"

String str2 = "Rai" , b+a , "74"

str1.compareTo(str2);

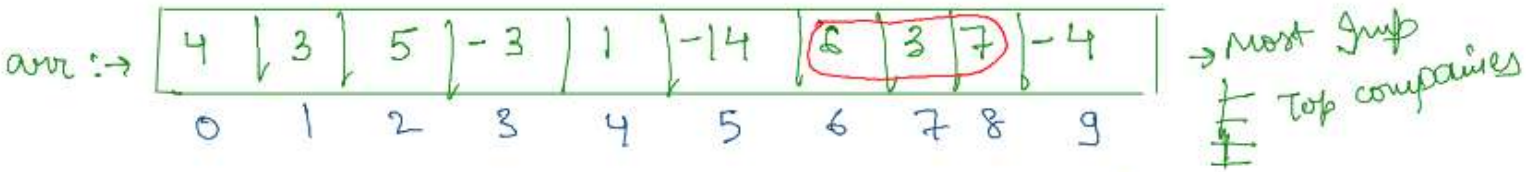
↳ It is compare both the strings in lexographically order

Question Maximum Sum Subarray.

-1 | 2 3 | -2 1

To find subarray with maximum sum

Ans 5



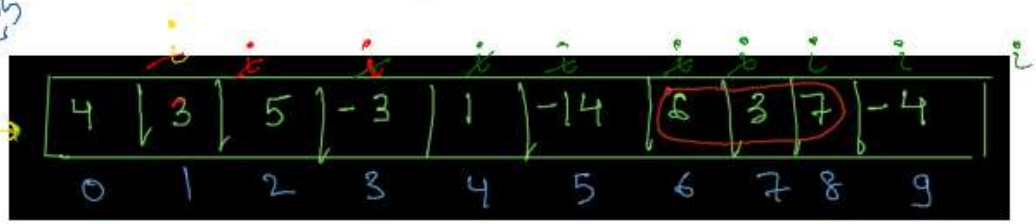
Naive method → { Lets find all the subarrays first and then find max sum from that }

↳ $O(n^2) + O(n)$

⇒ $TC = O(n^2)$

TC = O(n)

optimized solution :⇒ Kadan's algorithm



(sum till this index) csum → 4

(maximum sum) osum → 4

csum = ~~7~~ ~~12~~ ~~9~~ ~~10~~ ~~-4~~ ~~6~~ ~~9~~ ~~16~~ 12

~~4, 3, 5, -3, 1, -14,~~

6, 3, 7, -4

Ans

osum = ~~4~~ ~~7~~ ~~12~~ 16

~~4, 3, 5~~

6, 3, 7

```
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<arr.length;i++){
        arr[i]= scn.nextInt();
    }
    int csum=arr[0];
    int osum = arr[0];

    for(int i=1;i<arr.length;i++){
        if(csum>=0){
            csum +=arr[i];
        }else{
            csum=arr[i];
        }

        osum = Math.max(osum,csum);
    }
```

~~16~~ 12

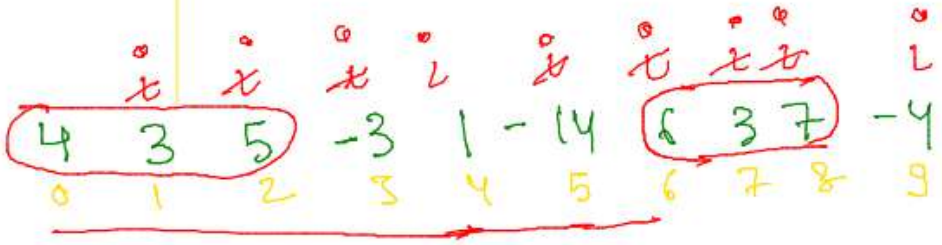
~~9~~

csum = ~~6~~

osum = ~~4~~

~~7~~ ~~12~~

16



Kadane's algorithm

Qun Maximum product Subarray:→

=

arr = (2 3) -2 4

sub → 6

naive approach → find all subarrays and calculate their max product

$O(n^2)$

O(n) Optimized approach ⇒

max = arr[0] = ~~2~~ ~~8~~ ~~4~~ 4

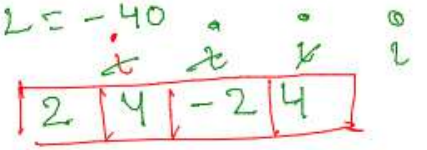
min = arr[0]; = ~~2~~ ~~-8~~ ~~-16~~ -64

ans → = arr[0]; = ~~2~~ (8)

max = ~~-10~~ ~~20~~ * -2 = 20

min = -10 * -2 = -40

20



for(i = 1 to n){

{if(arr[i] < 0){ swap(max, min); }}

max(4, -8)

min(4, -64)

max = Math.max(arr[i], max * arr[i]);

min = Math.min(arr[i], min * arr[i]);

ans = Math.max(ans, max);

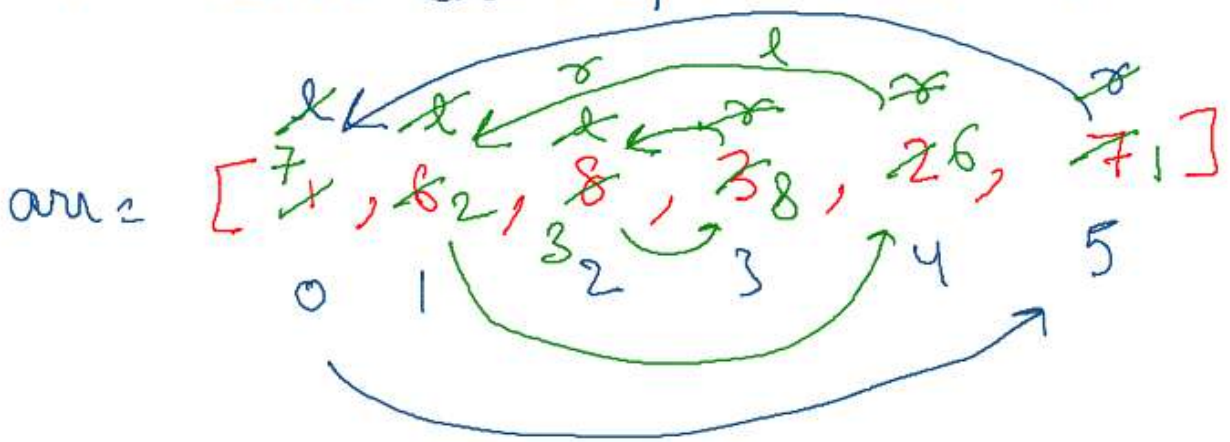
Ques Reverse an array :→

arr = [1 , 6 , 8 , 3 , 2 , 7]

arr = [7 , 2 , 3 , 8 , 6 , 1] ↵

→ solution.

condition → no extra space allowed




```
while(l<r){
swap(l, r);
l++;
r--;
}
```

```
l++;
r--;
```

ans = [7 2 3 8 6 1] ↵

```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();
        }

        reverse(arr);

        for(int i=0;i<n;i++){
            System.out.println(arr[i]+" ");
        }
        /* Enter your code here. Read input from STDIN. Print output
    }

    public static void reverse(int[] arr){
        int l = 0;
        int r= arr.length-1;

        while(l<r){
            int temp = arr[l];
            arr[l]=arr[r];
            arr[r]=temp;

            l++;
            r--;
        }
    }
}
```

Qus 4 Rotate right

arr = [1, 2, 3, 4, 5, 6, 7]

k = 2

arr = [6, 7, 1, 2, 3, 4, 5] k = 9

extra space not allowed

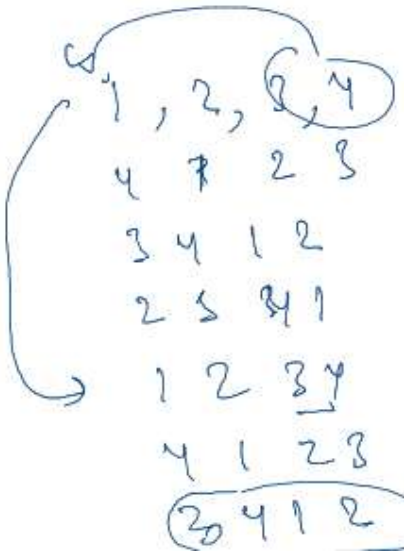
arr = [1, 2, 3, 4, 5, 6, 7]

arr = [5, 4, 3, 2, 1, 6 7]

arr = [5, 4, 3, 2, 1, 7 6]

arr = [6 7 1 2 3 4 5]

Solution



k=6 , k=-2

1, 2, 3, 4, 5