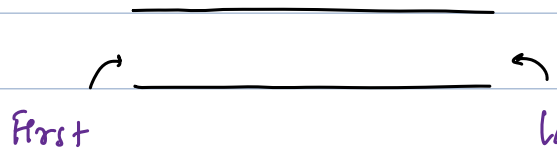
Today's Content

1. Deque Intro
2. Print Max of all subarrays of len = k
3. More questions.

Deque :



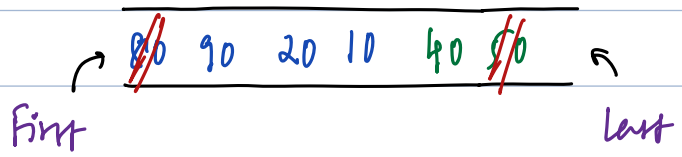
addFirst() addLast()

getFirst() getLast()

removeFirst(); removeLast()

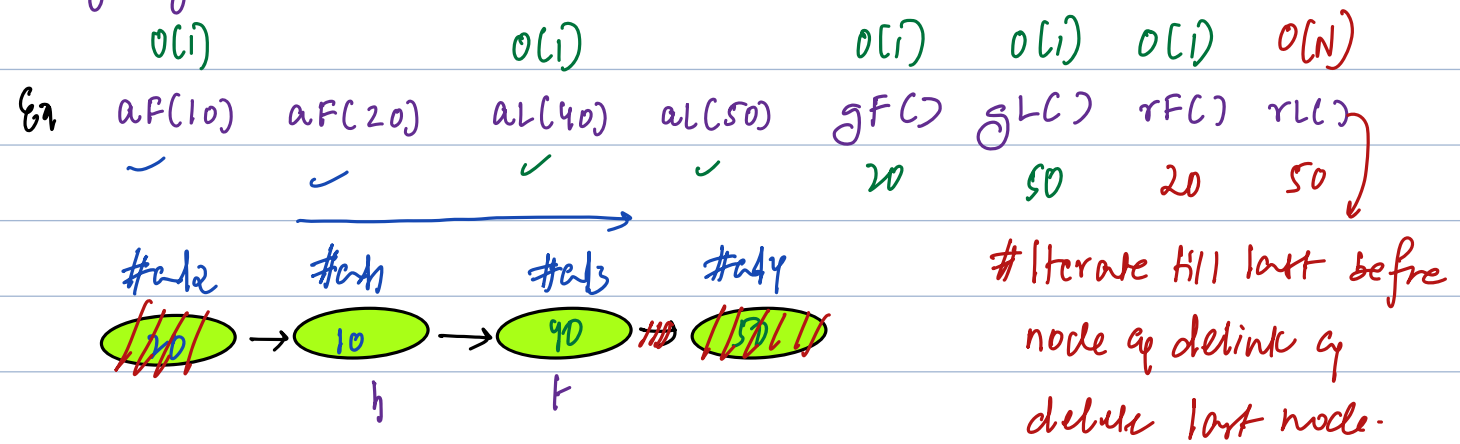
Dry Run:

Eg aF(10) ✓ aF(20) ✓ aL(40) ✓ aL(50) ✓ aF(80) ✓ rL() 50 rF() 80 aF(90) ✓



Implementation:

a. Using single linked list

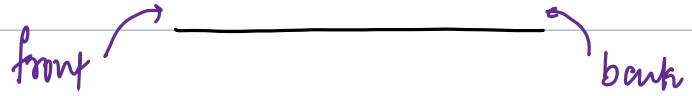


b. Using doubly linked list

In above dLL() takes O(N) but in dLL, dL() can be done in O(1) because we can traverse back in dLL().

Code : TODO

Inbuilt Syntax:



C++

```
deque<type> de;
de.push_front(x);      de.push_back(x);
de.front();            de.back();
de.pop_front();        de.pop_back();
```

Java : OCI

Not sure one check ?

```
Deque<Data_type> name = new ArrayDeque<>();
name.addFirst(x)       name.addLast(x)
name.getFirst()        name.getLast()
name.removeFirst();    name.removeLast()
```

python

```
de = deque();
de.appendleft(x);      de.append(x); #add last
de.popleft();          de.pop(); # delete last
de[0]                  de[-1]; #Acss last element
```

Given arr[N] & k
Print max element in every subarray of size = k

Eg:

	0	1	2	3	4	5	6	7	8
arr[] = {	10	1	9	3	7	6	5	11	8 }

k = 4

Output = 10 9 9 7 11 11

Idea1: Generate all subarrays of len = k
Iterate in subarray & calculate max
TC: O(N-k+1) * O(k) = O(N^2) SC: O(1)

Idea2: Sliding window? max

	0	1	2	3	4	5	6	7	8
arr[] = {	10	1	9	3	7	6	5	11	8 }

k = 4

Subarray max
{0..3} n = 10
{1-4} In n: delete arr[0] & add arr[4].
n+10; n-10 For add, inverse is sub
n*10; n/10 For mul inverse is /
max(n, 10) For max inverse doesn't exist

Note: We can add an element in max, but we cannot remove it.

Idea2: Sliding window + TreeMap.

0	1	2	3	4	5	6	7	8
10	1	9	3	7	6	5	11	8

arr[] = {10 1 9 3 7 6 5 11 8}

k = 4

~~10~~	1	9	3
	↑		

Operations	HashMap()	TreeMap
getMax()	O(N)	O(logN)
delete(n)	O(1)	O(logN)
Insert(n)	O(1)	O(logN)

TC: O(N-K+1) * logN ≈ O(NlogN) SC: O(N)

Idea3: Sliding Window + Deque

Eg:

0	1	2	3	4	5	6	7	8	9	10	11
~~15~~	~~12~~	~~8~~	~~4~~	~~10~~	~~9~~	7	12	10	7	14	3

arr[] = {15 12 8 4 10 9 7 12 10 7 14 3}

k = 5

#Already deleted

Idea2: #All possible answers

Output

15
12
10
12
12
12
14
14

~~15~~ → ~~12~~ ~~8~~ ~~4~~ ~~10~~ ~~9~~ ~~7~~ ~~12~~ ~~10~~ ~~7~~ 14 3

Operations: deque()

a. addLast()

b. getLast()

c. popLast()

d. getFirst()

e. popFirst()

```
void maxSubarray(vector<int> &arr){ TC: O(N) SC: O(N)
    int N = arr.size();
    deque<int> dq;
    for(int i=0; i<k; i++){          # 1st subarray [0..k-1]
        #insert arr[i];
        while(dq.size()>0 && dq.back()<arr[i]){
            dq.pop_back()
        dq.push_back(arr[i]);
    }
    print(dq.front());
    int s=1, e=k;    # 2nd subarray [1..k]
    while(e<N){
        #Remove arr[s-1] add arr[e]          s-1  s        e-1  e
        if(arr[s-1]==dq.front()){
            dq.pop_front();

        while(dq.size()>0 && dq.back()<arr[e]){
            dq.pop_back()
        dq.push_back(arr[e]);

        print(dq.front());
        s++;
        e++;
    }
}
```