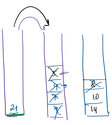
Deque
Wednesday, 11 May 2022 8:24 PM

18) Implement Queue using Stacks
→ to push(), pop(), top(), size(), is Empty()

5 4 7 7 8 8 10 7 7 14 7 7 21
deque 4 3 1 8

Enque() : O(1)
Deque() : O(n) ? [Transfering to q from]
rear() : O(1)
front() : O(N) [Transfer to q for]
size() : O(1)



Enque() : Push to S1 O(1)
Deque() : → O(1) if S2 is empty
transfer S1 → S2
pop data to S2
→ Best Case : Iterations 1
→ Worst Case : Iterations N
O(N)

front() : O(1)

rear() : if S1.top()
if S1 is empty()
last Element S2
Keep a temp variable & update when ever new value is Inserted
O(1)

size() : S1.size() + S2.size()
= O(1)

→ Best of Cases & Best >> Worst of Worst Worst

Amortized Analysis
Best : 1 Worst : N
Best case occurs a lot of times when compared to worst case
O(1)

⇒ For any data point to que
→ push to S1
→ pop to S1
→ push to S2
→ pop to S2
→ for a single element during push entire cause it will only take 4 steps

→ N Elements : 4N Iterations
⇒ Per Element : 4 Iterations : O(1)

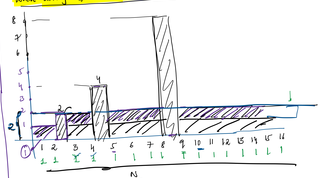
Vectors
→ Arrays?
→ int ar[10]

To traverse from 10 ⇒ O(1) time
We want to traverse 1 ⇒ 22 Iterations
int tmp[11]
Copy all data from ar[] → tmp[]
int ar[11]
Copy all data from tmp[] → ar[]

We want to traverse 1 → 24
int tmp[24]

⇒ To traverse 1 Element → O(N)

→ Arrays
→ int ar[2]
When array size is full double it



total Iterations = 2N
on an average per elem = 2 : O(1)

Deque :
→ double Ended Queue

Functions :
push_front()
pop_front()
push_rear()
pop_rear()
front(), rear()
size()

Implementation :
Double Linked List
→ predefined library
C++ / Java / Python
which support all these functions

18) Given ar[N] & K, find max element in every window of size K.

	0	1	2	3	4	5	6	7	8	9	10	11	12
ar[] :	3	15	6	12	4	2	10	9	13	7	2	5	3

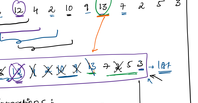
K = 4

Solutions :
1) Max of all windows
(N-K+1) → O(K)
TC: O(N²)

→ Hint :
→ Sorted map
→ Tree Map
→ keep are in sorted order
→ log n
2) Sliding Window using Tree Map
(N-K+1) → [Delete / Insert / get Max] ;
updating window
sliding to new window
(N-K+1) → log K
TC: N log N SC: O(N)

	0	1	2	3	4	5	6	7	8	9	10	11	12
ar[] :	3	15	6	12	4	2	10	9	13	7	2	5	3

K = 4



→ left

operations :
pop_front()
pop_back()
push_back()
back()
front()
Deque

15
15
12
12
10
13
13
13
13
7

Ans: 10 10 9 8 5
K = 4

10 9 8 5

10
10

3) Declare deque dq & int

for(int i = 0; i < N; i++){
// ar[i] we insert
while(dq.size()>0 && ar[i] > dq.rear())
// delete from back
dq.pop_back()
dq.push_back(ar[i])
}
print(dq.front())
for(int i = K; i < N; i++){
// ar[i] is insert, ar[i-K] is outgoing element
while(dq.size()>0 && ar[i] > dq.rear())
dq.pop_back();
if(dq.front() == ar[i-K]) dq.pop_front()
print(dq.front())
}

Observation :
To get window Max :
keep your Deque in Decreasing & can have some elements
TC: O(N) SC: O(K)

// To get window Min :
keep your deque in Increasing order elements
can repeat

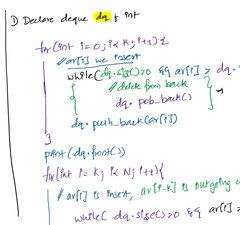
20) Given an A[N], find min index i such that,
sum variable from [i to i-1] should be >0 & Never negative
sum variable from [i to i-1] all elements

Ex: 0 1 2 3 4 5
2 -3 4 -2 -1 3
2 -1 [4 2 1 4 3]
i

Ans2: 0 1 2 3
2 1 -4 3
[3 5 6 2]
All Non Negative

Ans: 0 1 2 3 4 5
3 1 -1 -4 5 -1
3 0 -1 -4 5 -1
3 2 1 -3 2 1

→ Hint :



→ Append n-1 element to array
→ Calculate prefix[]
→ For min of every window of size n - previous array sum ≥ 0 → sum from (0, i-1)
if >=0, Then i is my answer

Solutions :
Iterate from every Index
TC: O(N²) SC: O(1)

2nd Approach :
if sum of all array value is -ve
No ans is possible.

ar[] : 3 -1 -2 -2 -5
3 2 0 -2 -7
3 2 1 -1 -6

3 4 -2 -3 2
3 7 5 2 4

Idea :
sum = 0 // Iterate & get sum of all array elements
sum < 0 ; return -1;
sum = 0 ; start = 0
for(int i = 0; i < N; i++){
sum = sum + ar[i];
if(sum < 0){
start = i+1;
sum = 0;
}
}

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