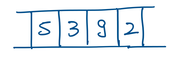
Iterative - no extra space

Tuesday, 5 October 2021 1:20 PM

(Q) Sort a queue without using extra space

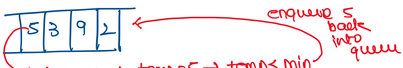
assuming ascending sort

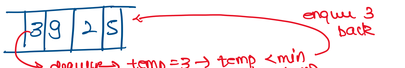
→ $O(n^2)$ time complexity, $O(1)$ space

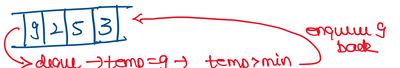


Now since only enqueue or dequeue operation is allowed, we will use them to find the 1st minimum element.

min = INT_MAX.







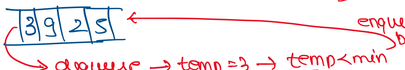
∴ Now we have traversed the entire queue once and found min ele = 2

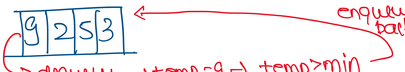
Now traverse the queue back and find 2nd minimum, but this time you will have to ignore 2 while calculating min, hence condition would

prev_min = min ← to store prev min

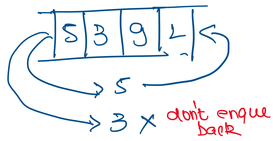
min = INT_MAX

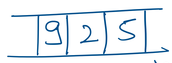
temp < min and temp > prev_min



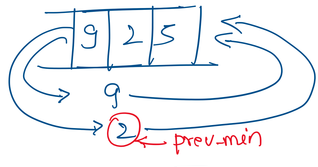


Hence now you got the 2nd min, but you need to put this 2nd min after first min. But before that you need to remove the 2nd min from its org position

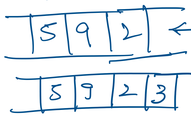




Now insert 2nd min after 1st min and Hence to do that keep dequing until you find 1st min i.e. prev_min



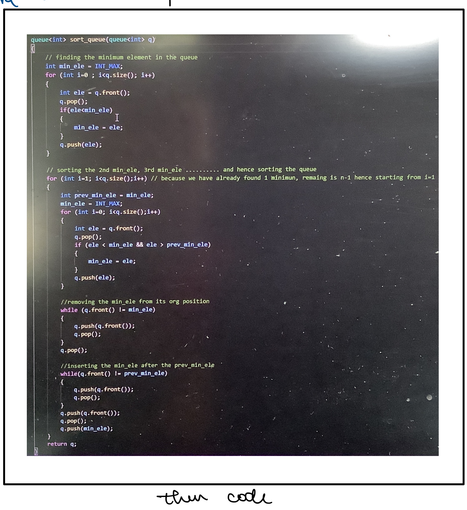
now enque 2nd min i.e. prev_min



Similarly continue, prev_min = min = 3

min = INT_MAX

think water first



```
queue<int> sort_queue(queue<int> q)
{
    // finding the minimum element in the queue
    int min_ele = INT_MAX;
    for (int i=0 ; i<q.size(); i++)
    {
        int ele = q.front();
        q.pop();
        if(ele<min_ele)
        {
            min_ele = ele;
        }
        q.push(ele);
    }

    // sorting the 2nd min_ele, 3rd min_ele ..... and hence sorting the queue
    for (int i=1; i<q.size();i++) // because we have already found 1 minimum, remaing is n-1 hence starting from i=1
    {
        int prev_min_ele = min_ele;
        min_ele = INT_MAX;
        for (int i=0; i<q.size();i++)
        {
            int ele = q.front();
            q.pop();
            if (ele < min_ele && ele > prev_min_ele)
            {
                min_ele = ele;
            }
            q.push(ele);
        }

        //removing the min_ele from its org position
        while (q.front() != min_ele)
        {
            q.push(q.front());
            q.pop();
        }
        q.pop();

        //inserting the min_ele after the prev_min_ele
        while(q.front() != prev_min_ele)
        {
            q.push(q.front());
            q.pop();
        }
        q.push(q.front());
        q.pop();
        q.push(min_ele);
    }
    return q;
}
```

then code