Stack, Queues, Priority Queues & Heaps Lecture 2

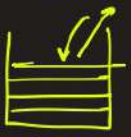
Thursday, 8 August 2024 6:01 AM

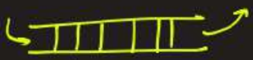
Queues

vector
pop_back()

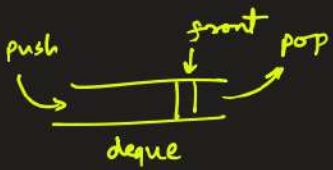
LIFO
Stack

FIFO
Queue





Queue implementation



C++

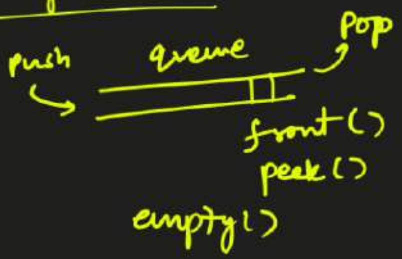
```
# include <queue>
        int
queue <T>   q ;

q.push (3);        O(1)
q.pop ();          O(1)
q.front() → int :  O(1)
q.size () ;        O(1)
q.empty() ;        O(1)
q.back();          O(1)
```

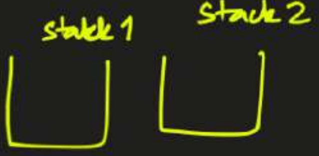
Python

```
from collections import deque
q = deque()
q.appendleft()     O(1)
q.pop()            O(1)
q[-1]   front      O(1)
q [0]   rear       O(1)
len(q)             O(1)
if (q)             O(1)
```

=x=

Implement queue using stacks.

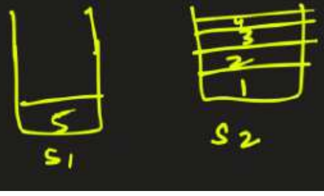




push()
pop()
top()
empty()

https://leetcode.com/problems/implement-queue-using-stacks/

① push slow



① $\underline{2}, \underline{3}, \underline{4}, 5$ $O(n)$ $O(n)$

$\underline{\text{push}()} \rightarrow$ $S1 \rightarrow S2$ push (S1) $S2 \rightarrow S1$ $\underline{O(n)}$

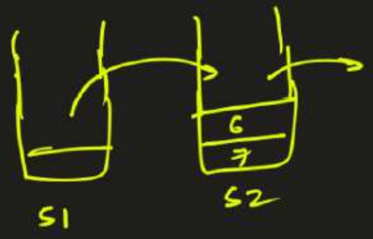
$\text{pop}() \rightarrow$ pop from S1 $O(1)$

$\text{size}() \rightarrow$ size(S1) $O(1)$

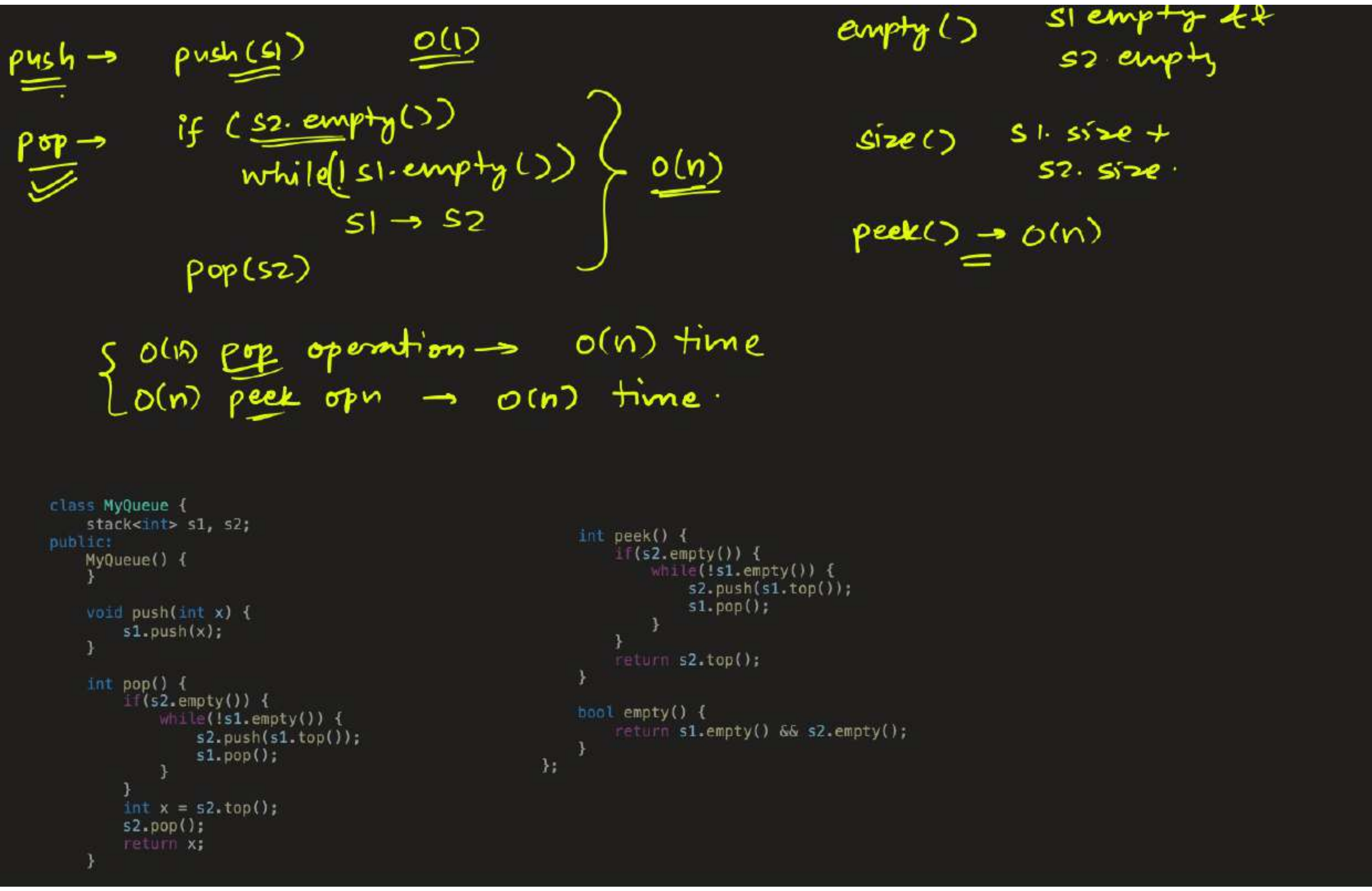
$\text{Empty}() \rightarrow$ S1.empty $O(1)$

n push operation $\rightarrow$ $\underline{O(n^2)}$ $\underline{\text{time}}$

②



push $\underline{1, 2, 3, 4, 5}$ ① pop   ② pop   6 push   7 push   ③ pop   ④ pop   ⑤ pop   ⑥ pop



push → push(s1) O(1)

pop → if (s2.empty())
    while(!s1.empty())
        s1 → s2
pop(s2)

} O(n)

empty() s1 empty &&
s2 empty

size() s1. size +
s2. size.

peek() → O(n)

{ O(n) pop operation → O(n) time
{ O(n) peek opn → O(n) time.

```
class MyQueue {
    stack<int> s1, s2;
public:
    MyQueue() {
    }

    void push(int x) {
        s1.push(x);
    }

    int pop() {
        if(s2.empty()) {
            while(!s1.empty()) {
                s2.push(s1.top());
                s1.pop();
            }
        }
        int x = s2.top();
        s2.pop();
        return x;
    }

    int peek() {
        if(s2.empty()) {
            while(!s1.empty()) {
                s2.push(s1.top());
                s1.pop();
            }
        }
        return s2.top();
    }

    bool empty() {
        return s1.empty() && s2.empty();
    }
};
```

https://leetcode.com/problems/first-unique-character-in-a-string/

a b c d b b e c a b a

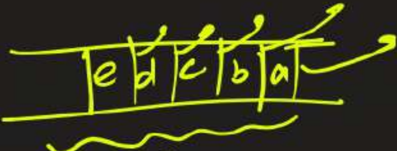
$\boxed{O(n) + O(Q)}$

unique element

e, 1
d, 1
c, 2
b, 2
a, 2

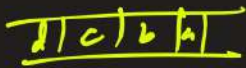
$O(n) + O(n)$

a a a a a a a a a a b c d

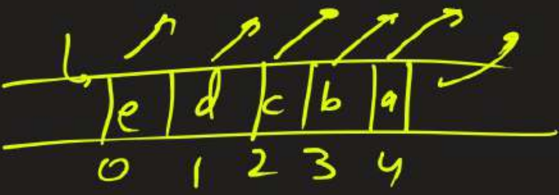


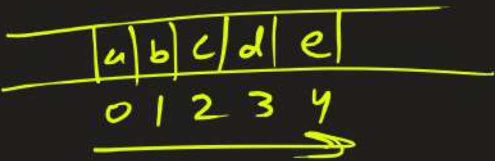
unique elements = 128

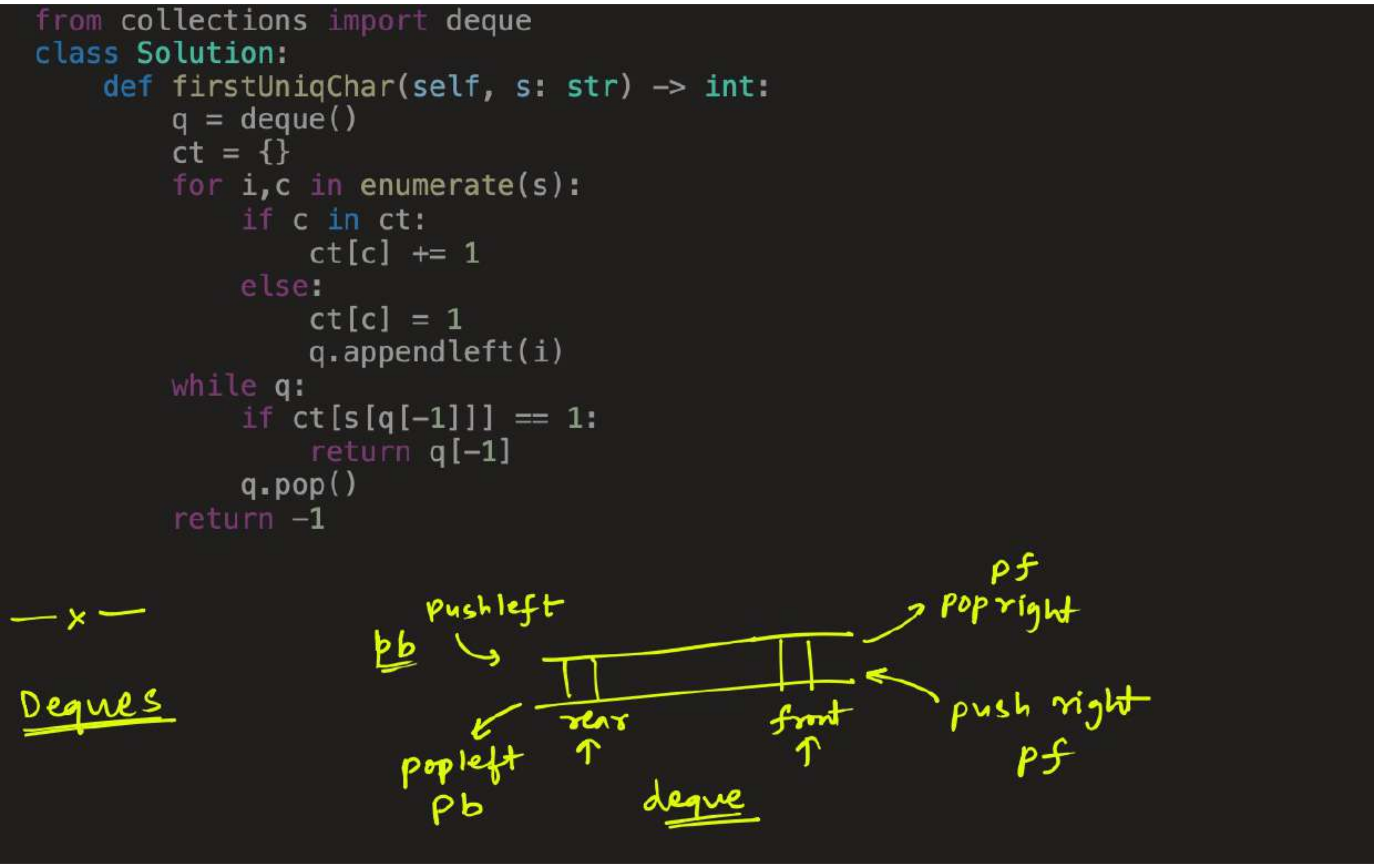
$10^5 + 128$



2×10^5



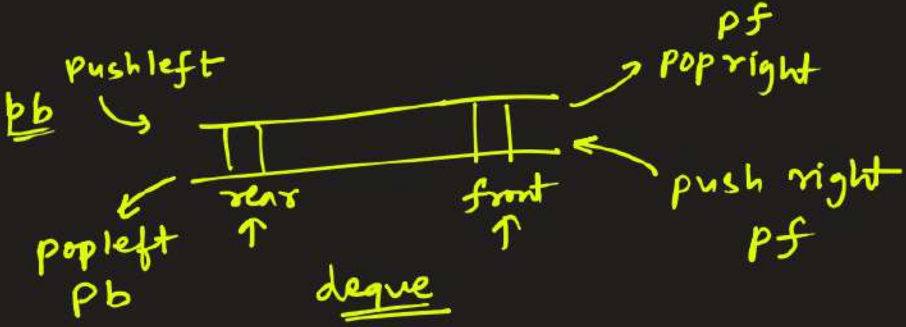




```python
from collections import deque
class Solution:
    def firstUniqChar(self, s: str) -> int:
        q = deque()
        ct = {}
        for i,c in enumerate(s):
            if c in ct:
                ct[c] += 1
            else:
                ct[c] = 1
                q.appendleft(i)
        while q:
            if ct[s[q[-1]]] == 1:
                return q[-1]
            q.pop()
        return -1
```

— x —

Deques



C++

```
# include <deque>
```

deque< int > dq;

dq. push_back(3);

dq. push_front(5);

dq. push_back(2);

dq. push_front(4);

dq. front () → $\bigcirc\!\!\!\!4$

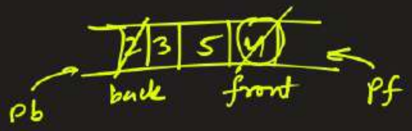
dq. back () → $\bigcirc\!\!\!\!2$

dq. pop_front(); → ~~4~~

dq. pop_back(); → ~~2~~

dq. size();

dq. empty ();



from collections import deque

dq = deque();

dq. appendleft (3);

dq. append (5)

dq. appendleft (2);

dq. append (4);

dq [-1] ← front

dq [0] ← back

dq. pop () ← Pf.

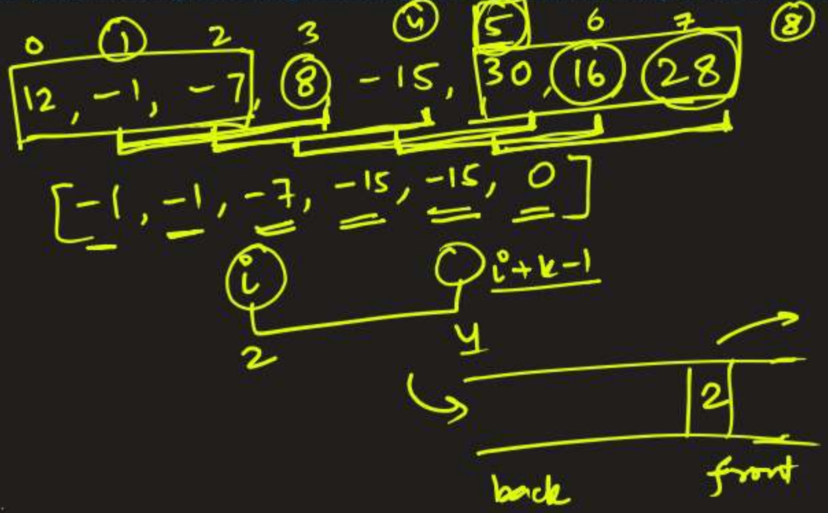
dq. popleft () ← Pb.

len(dq)

if (dq)

=x=

https://www.geeksforgeeks.org/problems/first-negative-integer-in-every-window-of-size-k3345/1

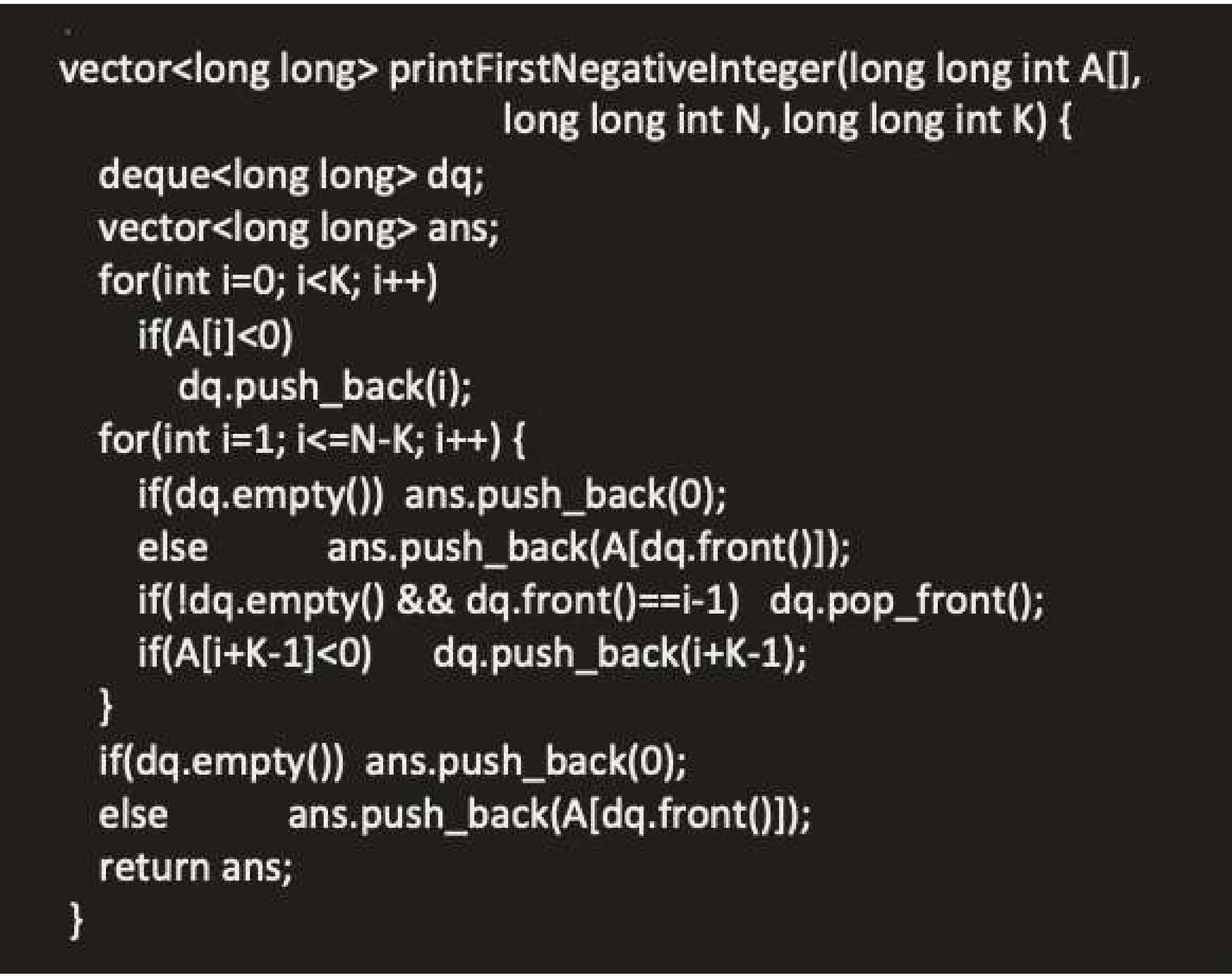


, k = 3 n = 8

deque (at most k)
indices of all negative numbers in current window

```
i → 0 to n-k
        if (dq.empty()) ans.pb(0);                 O(1)
        else. ans.pb(arr[dq.front()]);             O(1)
        if (dq.front() == i) dq.pop-front();       O(1)
        if (arr[i+k-1] < 0) dq.push-back(i+k-1);
                                                   O(1)
```

$T = O(n)$ $S = O(k)$



```
vector<long long> printFirstNegativeInteger(long long int A[],
                                  long long int N, long long int K) {
   deque<long long> dq;
   vector<long long> ans;
   for(int i=0; i<K; i++)
      if(A[i]<0)
         dq.push_back(i);
   for(int i=1; i<=N-K; i++) {
      if(dq.empty())  ans.push_back(0);
      else            ans.push_back(A[dq.front()]);
      if(!dq.empty() && dq.front()==i-1)   dq.pop_front();
      if(A[i+K-1]<0)      dq.push_back(i+K-1);
   }
   if(dq.empty())  ans.push_back(0);
   else            ans.push_back(A[dq.front()]);
   return ans;
}
```