Lc qps stack, hashmap

27 May 2023 11:54

2020/2021 OA Story FAANG

- Form minimum number from given sequence
- Number of valid Subarrays (LC Hard)
- Group anagrams (Hashmap)

Form min number from given sequence

Auxiliary Given a pattern containing only I's and D's. **I** for increasing and **D** for decreasing. Device an algorithm to print the minimum number following that pattern. Digits from 1-9 and digits can't repeat.

Examples:

```
Input: D     Output: 21
Input: I     Output: 12
Input: DD    Output: 321
Input: II    Output: 123
```

I D
D I

1, 2, 3, 4, 5, 6, 7, 8, 9

"D" → 21 Decrease 2→1

"I" → 12 Increase 1→2

"DD" → 321

"II" → 123

no repetition

max length of numb → 9

max length of input → 8

Input.length → l

numberoutput.length → l+1

l+1 = 9

l = 8

max. l of output → 9

max.l of input → 8

→ 1-9 digits
→ no repition
→ Input+1 → l
output.l → l+1

→ constraint
1 - 9
no repeate
max L = 9

DD → 321

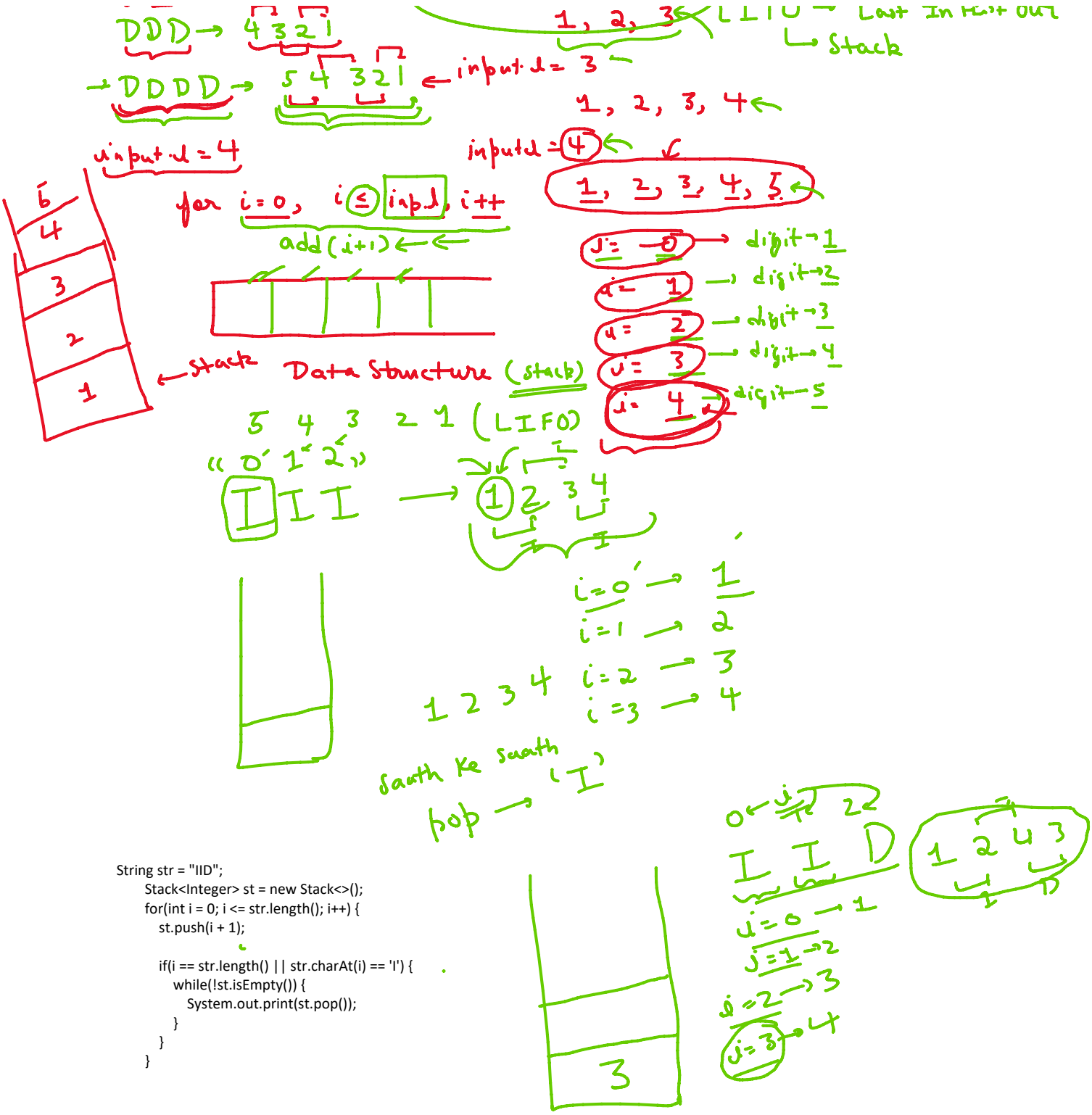
DDD → 4321

input.l = 2

1, 2, 3

input.l = 3

LIFO → Last In First Out
↳ Stack




```
String str = "IID";
    Stack<Integer> st = new Stack<>();
    for(int i = 0; i <= str.length(); i++) {
      st.push(i + 1);

      if(i == str.length() || str.charAt(i) == 'I') {
        while(!st.isEmpty()) {
          System.out.print(st.pop());
        }
      }
    }
```

```
//str.charAt(i) == 'I'||i == str.length()
```

Number of Valid Subarrays

[1, 4, 2, 5, 3]

[1] [4] [2] [5] [3]

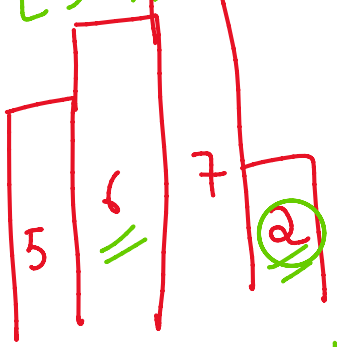
[1, 4, 2, 5, 3]

[1, 4] [1, 4, 2] [1, 4, 2, 5]

[2, 5] [2, 5, 3]

0 1 2 3 4

[1, 4, 2, 5, 3] ← Buildings



Tell me the index of next smaller building for each building

5

[1] [1,4] [1,4,2] [1,4,2,5] [1,4,2,5,3]

[5] [6] [7] [2] [4]

[5,6] [6,7] [2,5] [2,5,3]

[5,6,7] [5] [3]

~~Dont~~ include or stop at nse

nse

0 1 2 3 4 5

[1 2 3, 4, 5]

i - curr = 5

[1, 2, 3, 4, ...]

curr

no. of subarray = rindex − curr. index

[1]

[1, 2]

[1, 2, 3]

[1, 2, 3, 4]

[1, 2, 3, 4, 5]

arr.l − curr

5 − 0 = 5

5 − 1 = 4

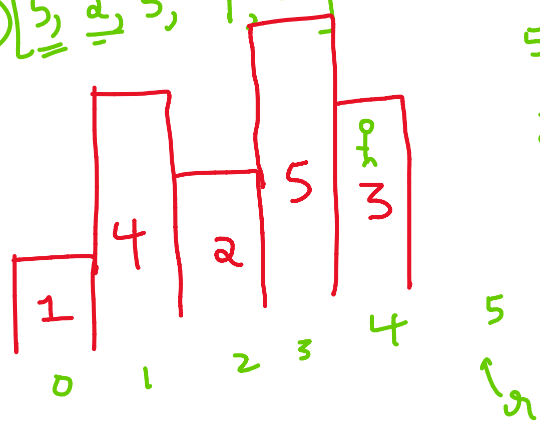
arr.l − curr

5 − 2

curr → 0 1 2 3 4 5

[1, 4, 2, 5, 3]

nse index [5, 2, 5, 4, 5]



r Index − curr

5 − 0 = 5 +

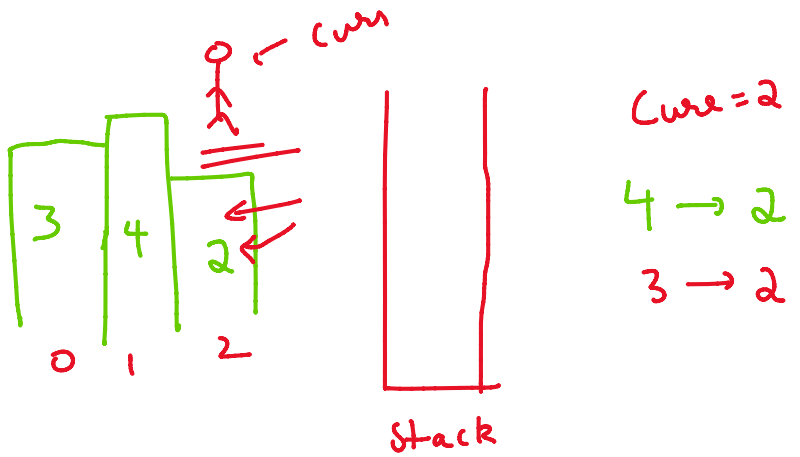
2 − 1 = 1 +

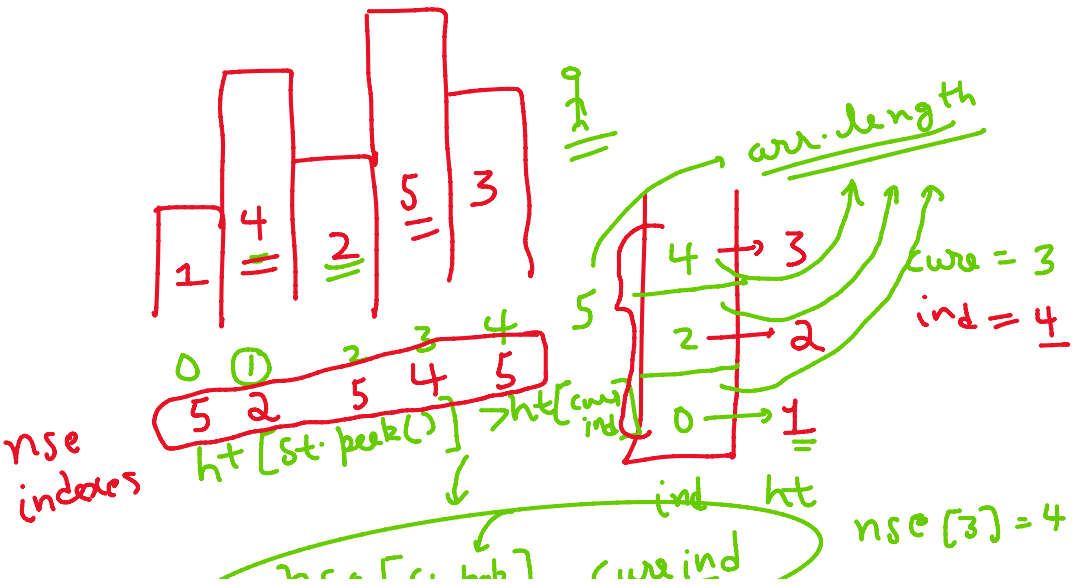
5 − 2 = 3 +

4 − 3 = 1 +

5 − 4 = 1

11


curr
3
4
2
0
1
2
stack
curr = 2
4 → 2
3 → 2


1
1
4
2
5
3
0
1
2
3
4
5
5
2
5
4
5
ht[st.peek()]
nse indexes
arr.length
4 → 3
2 → 2
0 → 1
5
curr = 3
ind = 4
ind
ht
curr ind
nse[3] = 4

increase

nse[st.pop] = curr ind

nse[3] = 4

nse[1] = 2

1, 4, 3 — curr

0 1 2 3

st.push(0)

for curr = 1 ; curr = arr.length-1 ; curr++) {

while(!st.isEmpty() and ht[st.peek()] > ht[curr]){

nse[st.pop] = curr

curr =

ht =

2 3

0 1

}

st.push(curr) ←

}

while(st is not empty){

nse[st.pop()] = arr.l

} curr

0 1 2 3 4

nse = [5, 2, 5, 4, 5] ← (arr.ind - curr)

ind

nse = [5, 2, 5, 4, 5]
int

5 - 0 +
2 - 1 +
5 - 2 +
4 - 3 +
5 - 4

nse → [0 1 2 3 4]

```
st.push(0)
for( curr= 1 to arr.l-1) {
    while ( !st.empty and ht[st.peek()] > ht[curr]){
        nse[st.pop()] = curr;
    }
    st.push(curr)
}
while ( !st.empty) {
    nse [st.pop] = arr.l
}
```

O(n)

```
Sum=0
for( i=0 to  nse.l-1){
    Sum += ( nse[i]-i)
}
ret Sum
```

O(n)