# STACK :-

## Next Smaller — NSL NSR by indices.

```
stack<int> st
for(i:0 to n){
    while(notempty() &&
    arr[i] <= arr[st.top()])
    { st.pop() }
    if(st.empty())
        NSL[i] = -1
    else NSL[i] = st.top()
    st.push(i)
```

```
Stack<int> st
for(i=n-1 to 0)
    while(notempty() &&
    arr[i] <= arr[st.top()])
    { st.pop() }
    if(st.empty())
        NSR[i] = n
    else NSR[i] = st.top()
    st.push(i);
```

| 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 4 | 3 | 1 | 2 | 4 | 1 | 9 |

| 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 4 | 3 | 1 | 2 | 4 | 1 | 9 |

string index →

| st | NSL |
|---|---|
| {} | -1 |
| {0 1 | -1 -1 |
| {~~1~~ 2 | -1 -1 -1 |
| {2 3 | -1 -1 -1 2 |
| {2 3 4 | -1 -1 -1 2 3 |
| {~~2~~ ~~3~~ ~~4~~ 5 | -1 -1 -1 2 3 -1 |
| {5 6 | -1 -1 -1 2 3 -1 5 |

| st | NSR |
|---|---|
| {} | 7 |
| {~~6~~ 5 | 7 7 |
| {5 4 | 5 7 7 |
| {5 ~~4~~ 3 | 5 5 7 7 |
| {~~5~~ ~~3~~ 2 | 7 5 5 7 7 |
| {2 1 | 2 7 5 5 7 7 |
| {2 1 4 | 1 2 7 5 5 7 7 |

## Sum of Subarray Minimums

arr = [3 1 4]

subarrays : 3 , 1 , 4 [3 1] [1 4] [3 1 4]

minimums : 3 , 1 , 4 , 1 , 1 , 1

return sum : 11

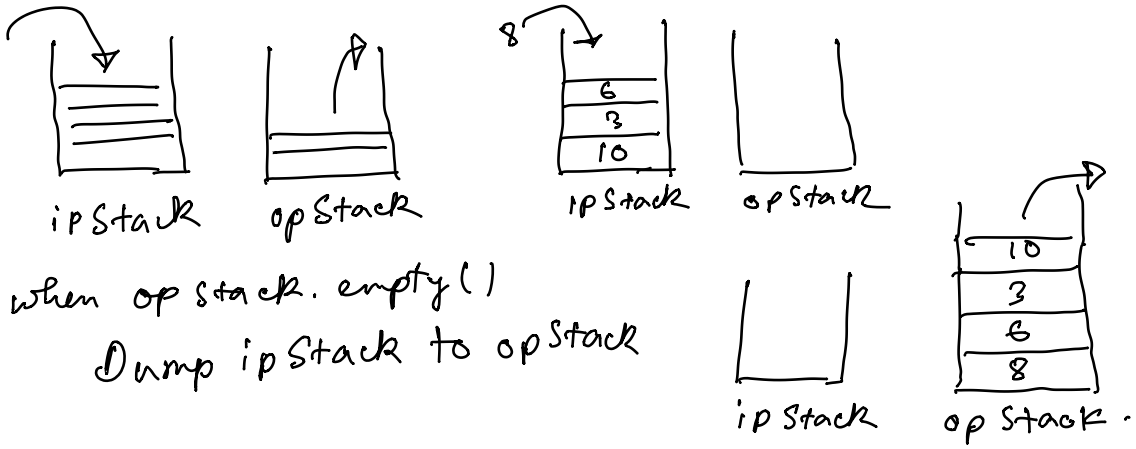
res += arr[i] x (i - nsl[i]) x (nsr[i] - i)

| 0 | 1 | 2 |
|---|---|---|
| 3 | 1 | 4 |
| -1 | -1 | 1 |
| 1 | 3 | 3 |

ans = 3 x (0 - (-1)) (1 - 0) + 1 (1 - (-1)) x (3 - 1) + 4 (2 - 1) (3 - 2)

= 3 + 1 x 2 x 2 + 4 = 11

## Implement Queue using Stack




when op stack. empty()
Dump ip Stack to op Stack

## Simplify Path :-

s: "/home // folder1 /.. /. / folder2" op: "/home/folder2".

```
stringstream ss(s);
string token;
while ( getline(ss, token, "/"))
{
    :
}
```

# Basic Calculator :-

Bracket + & -

"(a+b) - c + (d - e)"

```
for (int i=0 to n)
    if (isDigit(s[i])){
        num = num * 10 + s[i] - '0';}
    if (s[i] == '+'){
        result += num * sign
        num = 0
        sign = 1 }
    if (s[i] == '-'){
        result += num * sign
        num = 0
        sign = -1 }
    if (s[i] == '('){
        st.push (result)
        st.push (sign)
        result = 0 num = 0
        sign = 1 }
    if (s[i] == ')'){
        result += num * sign
        lastSign = st.top()
        st.pop.
        lastResult = st.top()
        st.pop.
        result *= lastSign
        result += lastResult
        num = 0 }

return result + num * sign;
```

i: 0 1 2 3 4 5 6 7

-(2 + 3) + 5

| | res | num | sign | st |
|---|---|---|---|---|
| | 0 | 0 | 1 | {} |
| i=0 | 0 | 0 | -1 | {} |
| i=1 | 0 | 0 | 1 | {0, -1} |
| i=2 | 0 | 2 | 1 | {0, -1} |
| i=3 | 2 | 0 | 1 | {0, -1} |
| i=4 | 2 | 3 | 1 | {0, -1} |
| i=5 | 2+3=5<br>5 x -1 = -5<br>-5 + 0 = -5 | 0 | 1 | {} |
| i=6 | -5 + (0x1) = -5 | 0 | 1 | {} |
| i=7 | -5 | 5 | 1 | {} |

result + num * sign

= -5 + 5x1 = 0

# Maximum Score from Removing Substrings

"xyaabbabaabbab"

Removing "ab" → 5 "ba" → 4

"xyaabbabaabbab" → Total 24

"xyaabbabaabbab" → Total 25

O/P = 25

Soln: in 2 pass first delete all "ab" then try for "ba" in second pass
or vice versa if "ba" price is >.

#⭐, #Revision, #Tricky business

$x_1 x_2 x_3 x_4$ →

aabb
abab
abba
bbaa
baab
baba

ab → x
ba → y

abbbaba

abbbaba  abbbaba

x + y + y

x + x + y  >

# always check ab first. check ba in 2nd pass

for baxba
no ab ; then it's fine
take 2 ba in 2nd pass.

another trick ; instead of using O(N) stack alter the string itself.

cdbabagh

cdbagh → cdbagh