Contest Information

Date :- 20th March (Tentative)
7:00 AM IST
1.5 hrs

Discussion — 8:30 AM IST
Passing marks — 60 %
Total 3 Questions
(weightage may vary)

Absolutely necessary to appear for the contest.

What if I fail ?
Appear for the reattempts.
Total 3 reattempts

Our endeavour should be to clear the live contest itself.
Reattempts is only for exceptional cases.

How to prepare ?
Solve assignment questions religiously. and keep PSP above 90%

Q-1. Given a string s of lowercase characters, return the count of pairs (i, j) such that s[i] == 'a' & s[j] == 'g' and i < j

Ex:- String s = "abegag"

0	1	2	3	4	5
a	b	e	g	a	g

0, 3
0, 5
4, 5

ans = 3.

Quiz 1

String s = "acgdgag"

0	1	2	3	4	5	6
a	c	g	d	g	a	g

a	0	0	0	5
g	2	4	6	6

ans = 4.

Quiz 2

String s = "bcaggaag"

0	1	2	3	4	5	6	7
b	c	a	g	g	a	a	g

a	2	2	2	5	6
g	3	4	7	7	7

ans = 5.

Brute Force Solution

```
        0 1 2 3 4 5 6 7 8
s = " a c b a g k a g g "
```

i

j

ans = ~~0~~ ~~1~~ ~~2~~ ~~3~~ ~~4~~ ~~5~~ ~~6~~ ~~7~~ 8

```
int    count_ag ( string s )  {
          int   ans = 0;
   for ( i = 0 ;  i < s.size() ;  i++ ) {
      |      if ( s[i] == 'a' ) {
      |      |     for ( j = i+1 ; j < s.size() ; j++ ) {
      |      |     |    if ( s[j] == 'g' ) {
      |      |     |    |      ans++;
      |      |     |    }
      |      |     }
      |      }
   }
               return ans;
```

T.C → $O(N^2)$

S.C → $O(1)$

Optimised Solution

```
          0   1   2   3   4   5   6   7   8
S = "     a   c   b   a   g   K   a   g   g  "
                                          i
Count_a = 0   1   1   1   2   2   2   3   3   3.
ans.    = 0   0   0   0   0  +2   2   2   5   8
```

```
int    Count_ag( String s ) {

        int   ans = 0;
        int   countA = 0;

    for ( i = 0;  i < N;  i++ ) {
            if ( s[i] == 'a' ) {
                    countA++;
            }

        else if ( s[i] == 'g' ) {

                ans += CountA;
        }
    }
            return   ans;
}
```

$T.C \rightarrow O(N)$

$S.C \rightarrow O(1)$

Introduction to Subarrays

A subarray is a contiguous part of an array. It is formed by selecting a range of elements from the array. It can have one or more elements.

0 1 2 3 4 5 6 7 8

array - {4, 1, 2, 3, -1, 6, 9, 8, 12}

① 2, 3, -1, 6 ✓

② 9 ✓

③ 4, 1, 2, 3, -1, 6, 9, 8, 12 ✓

④ 4, 12 ✗

⑤ 1, 2, 6 ✗

⑥ 3, 2, 1, 4 ✗ (Order of elements IS important).

Quiz 3

0 1 2 3 4 5 6 7

{2, 4, 1, 6, -3, 7, 8, 4}

a) {1, 6, 8} ✗

b) {1, 4} ✗

c) {6, 1, 4, 2} ✗

d) {7, 8, 4} ✓

Representing a subarray

Two ways:-

1. Start index & End index.
2. Start index & length of a subarray.

$$\begin{array}{ccccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ \{4, & 1, & 2, & 3, & -1, & 6, & 9, & 8, & 12\} \end{array}$$

Subarray :- {2, 3, -1, 6}.

Subarray → S.I → 2
E.I → 5

→ S.I → 2
length → 4.

S.I.

Quiz 4

$$\begin{array}{ccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ \{4, & 2, & 10, & 3, & 12, & -2, & 15\}. \end{array}$$

0, 0
0, 1
0, 2
0, 3
0, 4
0, 5
0, 6.

7.

N

Quiz 5

$$\begin{array}{ccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ \{4, & 2, & 10, & 3, & 12, & -2, & 15\}. \end{array}$$

S.I.

1, 1
1, 2
1, 3
1, 4
1, 5
1, 6

6:

~~N-1~~

Total No. of Subarrays.

$$\begin{array}{ccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ \{4, & 2, & 10, & 3, & 12, & -2, & 15\}. \end{array}$$

Total Subarrays whose starting index

0 → N
1 → N-1
2 → N-2
3 → N-3
⋮
N-1 → N-(N-1) = 1

= Sum of first N natural numbers.

$$= \frac{N(N+1)}{2}$$

Q:- Given an array and start & end index of its subarray. Print it.

	0	1	2	3	4	5	6	
arr →	{4,	2,	10,	3,	12,	-2,	15}	.

s → 2

e → 5

```
void  printSubarray ( int arr[], int s, int e) {
        for ( i = s ; i <= e ; i++ ) {
            |     print ( arr[i]);
            }
}
```

T.C → O(N)

S.C → O(1)

Q:- Print all possible subarrays of the array.

Ex:- $a = \{\overset{0}{1}, \overset{1}{2}, \overset{2}{3}\}$.

Ans → {1} {2} {3}

{1, 2} {2, 3}

{1, 2, 3}

```
void printAllSubarrays(int arr[]) {
    // Generate all Subarrays.
    for (s = 0; s < n; s++) {
        for (e = s; e < n; e++) {
            for (i = s; i <= e; i++) {
                print(arr[i]);
            }
        }    Break line.
    }
}
```

s = 0, 1   e = 1

1

1 2

1, 2, 3

2.

T.C → $O(N^3)$.

S.C → $O(1)$

Q:- Given an array of N integers, return the length of smallest subarray which contains both maximum & minimum element of the array.

Quiz 6

0 1 2 3 4 5 6 7 8 9 10

$arr \rightarrow \{2, 2, 6, 4, 5, 1, 5, 2, \underbrace{6, 4, 1}\}$

ans = 3.

Another Example

0 1 2 3 4 5 6 7 8 9

$A[\,] - \{1, 2, 3, \underbrace{1, 3, 4, 6}, 4, 6, 3\}$

ans = 4.

Brute Force

→ Check all subarrays

→ $O(N^3)$. + $O(N)$

1, 1, 6

Optimisation

1. The answer subarray must have exactly one instance of minimum & one instance of maximum since we want the length to be minimum.

2. The minimum & maximum value must be present at the corners of the subarray.

3. So, we are looking for a subarray that either
   a) Starts with MAX & ends with MIN
   b) Starts with MIN & ends with MAX

```
        0   1   2   3   4   5   6   7   8   9   10
arr - { 2 , 2 , 6 , 4 , 5 , 1 , 5 , 2 , 6 , 4 , 1 }
                                                    i
```

smallest = 1

largest = 6.

ans = ~~4~~. 3.

last_max_found = 8.

last_min_found = 5

```
        0   1   2   3   4   5   6   7   8   9   10
arr - { 2 , 2 , 6 , 4 , 5 , 1 , 5 , 2 , 6 , 4 , 1 }
```

Code:

```
int    minSubarray ( int A[] ) {

    int   minValue  =  minOfArray ( arr);
    int   maxValue  =  maxOfArray ( arr);

  int   last_min_found  = -1 ;
  int   last_max_found  = -1 ;

    int     ans  =  INT_MAX;  / N

for ( i = 0 ;  i < N ;   i++ )  {

      if  ( A[i]  ==  minValue ) {
            last_min_found  =  i ;

        if ( last_max_found  != -1 ) {
           ans  =  min ( ans ,  i - last_max_found +1 ) ;
        }
        }
~~else~~ if (A[i] == maxValue) {
    lat_max_found = i;
    if (last_min_found != -1) {
        ans = min(ans, i - last_min
                  _found +1);
    }
}
}
        return ans;
```

T.C → O(N)

S.C → O(1)

Next Class.

1. Sliding window.
2. Contribution Technique.

```
        0   1   2   3   4   5   6   7   8   9
arr - { 1 , 6 , 6 , 4 , 5 , 1 , 5 , 2 , 6 , 4 , 1 }
```