Problems on Arrays

Tuesday, 23 August 2022 6:34 PM

Today's Content

1) What's a subsequence

2) Problem 1

3) Problem 2

4) Problem 3

5) Problem 4

Subsequence: Delete many, 1 or none elements from your array remaining elements are subsequence. Subsequence cannot be empty

arr[] : 3 7 1 -2 6 3

Eg1: 3 7 6

Eg2: 7 -2 6

Eg3: 3 7 6 -2 *

→ 3 7 = 6

Def: Take any random elements & delete them based on index

0 1 2 3 4 5 6

Eg: arr[]: 2 4 -3 7 6 8 9

{4, 8, -3, 2} → {2, 4, -3, 8}

{2, 4, 7, 6, 9}

Any problem involving subsequence : → If we can we enter & delete

possible cannot exist?

1.03) Given a string S, containing only lowercase characters. determine how many subsequences of "ag" exist. {a, b, c, ... z}

char array

Eg 3) abcgag : 0 1 2 3 4 5

Index

0 3

3 4 : ga *

0 5

ans = 3

a c g d g a g

0 1 2 3 4 5 6

0 2, 0 4, 0 6

5 6 : 1

3) b a c g g a a g

0 1 2 3 4 5 6 7

(1,3) (1,4) (1,7)

(5,7) (6,7)

Idea:

for any given g

Calculate {no. of 'a' on right side}

0 1 2 3 4 5 6 7

a g a d g g d g

: count = 4

: 2 g

for i=0

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

if (S[i] == 'a') {

for (j = i+1; j < N; j++)

if (S[j] == 'g') {

cnt

}

}

}

taking time

return ans; TC: $\frac{(N)(N-1)}{2}$ → $\frac{N^2}{2} - \frac{N}{2}$ ≈ O(N²)

3) d g a g a g f g

0 1 2 3 4 5 6 7 8

d g a g a g f g

ans = 3

0, 3, 4

0, 4, 6

0, 6, 8

3 2

0 1 2 3 4 5 6 7 8 9 10 11 12

b g c g a f a c g a g g f

5 → 4 3 3 3 2 1 0

ans = 0

ans += 2

ans += 3

ans = 5

2) Optimized

for ans = 0, for cg = 0

for (i = N-1; i >= 0; i--) {

if (S[i] == 'g') {

cg++;

}

else if (S[i] == 'a') {

ans = ans + cg

}

}

return ans, Iterations: N, TC: O(N)

2.5) Longest Max-Min

Given an array find the {length of longest subarray} which contains Both {Min and Max element of array} within its subarray

Note: Subarray : → Continuous part of an array is called subarray

→ A single element can also be a subarray

Eg1: 0 1 2 3 4 5 6 7 8

{1, 3, 2, 4, 3, 7, 5, 4, 6} Min 1, Max 7 → Index, len [3-8] 6, [3-6] 4

Eg2: 0 1 2 3 4 5 6 7 8 9

{3, 2, 6, 4, 5, 8, 5, 4, 3} Min 2, Max 8 → [2-5] 4, [7-9] 3

[2-5] [5-9]

min (4 , 5) = 4

Ans = {1, 3, 5, 1, 3, 4, 3}

observations

find subarray

1) In the subarray

→ we will only have 1 min & 1 max

→ Min & Max should be at corner of subarray

Idea: → Subarray will start from 0th or max

if (arr[i] = min)

Search for nearest max in right, j > i

if (arr[i] = max)

Search for nearest min in right, j > i

Pseudocode

1) Iterate & get min val & array → Min : O(N)

2) Iterate & get max val & array → Max : O(N)

if (Min == Max) return 1 VIMP

Int ans = INT_MAX

0 1 2 3 4 5 6 7 8

3 4 1 7 6 7 1 1 7

for i = 0; i < N; i++

if (arr[i] == Min) {

// find 1st max in right

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

if (arr[j] == Max) {

ans = min (ans, j - i)

break

}

}

if (arr[i] == Max) {

// find 1st min in right

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

if (arr[j] == Min) {

ans = min (ans, j - i)

break

}

}

}

Time Taken

Eg: 0 1 2 3

3 4 1 7 4

Min 1, Max 7

Idea for Optimized algo.

1) Iterate from R → L

→ keep updating your min & max values

0 1 2 3 4 5 6 7 8 9 10 11 12 13

arr[] 3 8 2 1 4 5 1 6 2 2 2 1 4 3

[1-3] [3-7] [7-8] [8-13]

1-3

Time Taken

Time min index in from max index

Min 1-7 Max 3-8

{3, 4} {4, 3}

1

Return ans

TC: N × N × N² ≈ O(N²)

Pseudocode Optimized algo

1) Iterate & get minV & maxV. ≈ O(N)

if (minV == maxV) return 1 (Edge Case)

Int ans = N; minInd = -1, maxInd = N

2) i = N-1; i >= 0; i--) ≈ O(N)

if (arr[i] == minV) {

minInd = i; // update

if (maxInd != -1)

ans = min (ans, maxInd - minInd - 1)

}

else if (arr[i] == maxV)

maxInd = i; // update

if (minInd != -1)

ans = min (ans, minInd - maxInd - 1)

}

}

return ans; TC: O(N) SC: O(1)

2.6) Max length consecutive 1s

→ Given a Binary String of length N. It is allowed to do atmost 2 swap

→ Find the length of the longest consecutive 1s that can be achieved

Eg1: 0 1 2 3 4 5 6 7 8 } Swap: 2-3 : 1 1 0 1 1 1 0 1 ans = 4

Str = 1 1 1 0 1 1 1 0 1 } Swap: 5-8 : 1 1 1 1 1 1 0 0 ans = 7

0 1 2 3 4 5 6

Str = 1 1 1 0 1 1 1 } Swap : 3-6 : 1 1 1 1 1 1 0 ans = 6

0-3 : 0 1 1 1 1 1

Str = 1 1 1 0 } ans = 3

0 1 2 3 4 5 6 7 8 9 10 11

Str = 0 1 1 0 1 1 1 0 1 1 1 } 2-8 : 0 1 1 0 0 1 1 1 1 1 1 1 0 : ans = 8

Eg2:

Str = 1 1 0 1 1 1 1 0 1 1 1

1 1 1 0 0 1 1 1 1 0 1 0 1

Swap 1 to 0

Swap 1 to 0

1 1 1 1 1 1 1 1 1 0 0 1

Idea: Max consecutive from 1 to 1

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

0 1 1 1 1 1 0 1 1 1 0 1 0 0 0 1

left[]: 0 1 2 3 4 0 1 2 3 4 5 6 0 0 1 2

right[]: 0 4 3 2 1 0 6 5 4 3 2 1 0 0 2 1

you need answer for find = 1

left[i-1], right[i+1] + 1

3 + 3 + 1 = 7

Edge Case:

1 1 1 1 1 0 1 1 1 0

3 3

3+3+1 = 7

Only possible when there can exist a zero 1

0 0 1 1 1 1 1 1 1 1 0

2 → 3 → 4 → 5

2 + 3 + 1 = Total 1's

Total 1's

We cannot space zero 1

left[i-1] right[i+1]

left[i] right[i]

Pseudocode:

Int left[N] = 0;

left[0] = arr[0]

for i = 1; i < N; i++

if (arr[i] == 0) {

left[i] = 0

}

else {

left[i] = left[i-1] + 1

}

}

TC: O(N)

Int right[N] = 0

right[N-1] = arr[N-1]

for i = N-2; i >= 0; i--) {

if (arr[i] == 0) {

right[i] = 0

}

else {

right[i] = right[i+1] + 1

}

}

TC: O(N)

Int cnt = 0; f = 0; i = 0; i < N; i++) {

take next in array

if (arr[i] == 0) {

cnt++

}

}

ans = 0;

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

if (cnt == 0) {

ans = max(ans, left[i] + right[i])

}

else {

if (cnt == 1 && arr[i] == 0) {

ans = max(ans, left[i-1] + right[i+1] + 1)

}

else {

ans = max(ans, left[i-1] + right[i])

}

}

}

return ans;

for loop out of bounds

Edge Case

i = 0 | i = N-1

Empirically fill the answers

Index

Edge Case

i = 0 | i = N-1

i = N-2

Total no. of 1's

I can span on index 2

Total 1's

ans = max(ans, left[i-1] + right[i+1] + 1)