Pascal's Triangle

Pascal Triangle

```
1
2  1  1
3  1  2  1
4  1  3  3  1
5  1  4  6  4  1
6  1  5  10 10 5  1
```

(1) Given R & C, tell the element at that place

R = 5 C = 3

ans = 6

(2) Print any Nth row of pascal triangle

N = 5

1 4 6 4 1

(3) Given N, print the entire triangle

N = 6

Pascal Triangle

(1) Given R & C, find the element at that place

R = 5 C = 3

ans = 6

(2) Print any Nth row of pascal triangle

N = 5

1 4 6 4 1

(3) Given N, print the entire triangle

N = 6

```
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1
```

R - 1

C

C - 1

$$^{n}C_{r} = \frac{n!}{r! \times (n-r)!}$$

$$^{4}C_{2} = \frac{4 \times 3 \times 2 \times 1}{(2 \times 1)(2 \times 1)} = 6$$

$$^{7}C_{2} = \frac{7!}{2! \times (5)!} = \frac{7 \times 6 \times (5 \times 4 \times 3 \times 2 \times 1)}{2 \times 1 \times (5 \times 4 \times 3 \times 2 \times 1)}$$

$$= \frac{7 \times 6}{2 \times 1}$$

$$^{10}C_{3} = \frac{10 \times 9 \times 8}{3 \times 2 \times 1} = \frac{10}{1} \times \frac{9}{2} \times \frac{8}{3}$$

$$\frac{10 \times 9 \times 8}{3 \times 2 \times 1} = \frac{10}{1} \times \frac{9}{2} \times \frac{8}{3}$$

```
funNcR(n, r)
{
    res = 1

    for(i = 0 ; i < r ; i++)
    {
        res = res x (n - i)
        res = res / (i + 1) ;
    }

    return res ;
}
```

TC -> O(r)
SC -> O(1)

long long

4 1 3 3 1

5 1 4 6 4 1

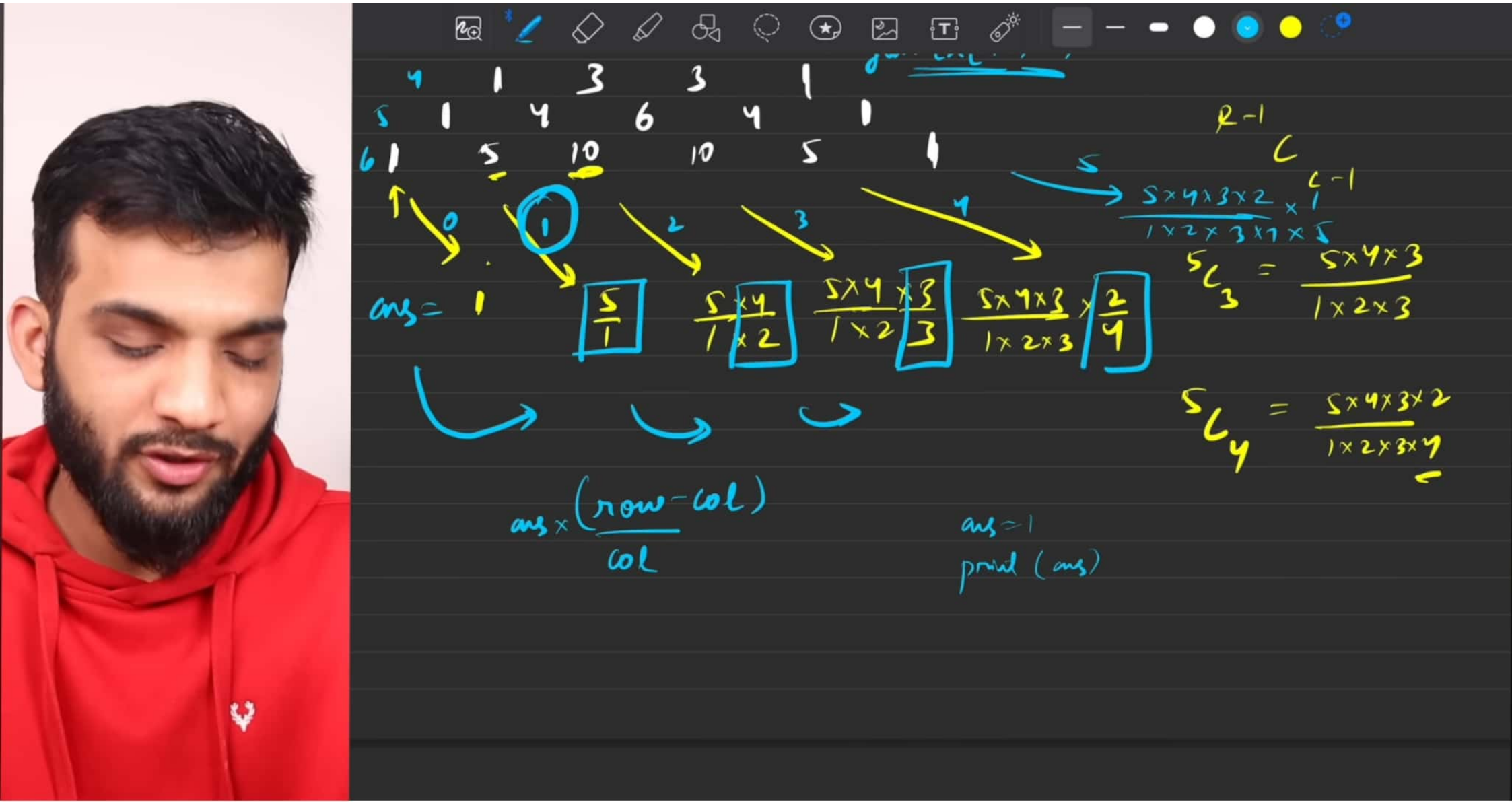
6 1 5 10 10 5 1

$^{n-1}C_{c-1}$

N^{th} row → N elements

```
for(c=1; c<=n; c++)
{
    print(funNCR(n-1, c-1));
}
```

TL → O(N × N)


4 1 3 3 1
5 1 4 6 4 1
6 1 5 10 10 5 1
0 1 2 3 4 5
ans = 1
5/1 5×4/1×2 5×4×3/1×2×3 5×4×3/1×2×3 × 2/4
ans × (row − col) / col
ans = 1
print (ans)
R − 1
C
5 × 4 × 3 × 2 / 1 × 2 × 3 × 4 × 5 × 1^(4−1)
5C3 = 5×4×3 / 1×2×3
5C4 = 5×4×3×2 / 1×2×3×4

0 1 2 3 4 5

ans = 1

$\frac{5}{1}$ $\frac{5\times4}{1\times2}$ $\frac{5\times4\times3}{1\times2\times3}$ $\frac{5\times4\times3}{1\times2\times3}\times\frac{2}{4}$ $\frac{5\times4\times3\times2}{1\times2\times3\times4}\times\frac{1}{5}$

$^{5}C_{3} = \frac{5\times4\times3}{1\times2\times3}$

$^{5}C_{4} = \frac{5\times4\times3\times2}{1\times2\times3\times4}$

$ans \times \frac{(row-col)}{col}$

```
ans = 1
print (ans)
for(i=1; i<n; i++)
{
    ans = ans x (n - i)
    ans = ans / (i)
print (ans);
}
```

```
        1
      1   2   1
    1   3   3   1
  1   4   6   4   1
1   5   10  10  5   1
```

$n-1 C_{r-1}$

ans = []

```
fun( row = 1 -> n)
{   temp list = []
    fun( col = 1 -> row)
    {
        temp list . add( ncr(n(row-1, col-1)) ;
    }
    ans . add ( temp list);
}
ret (ans)
```

```
        1
      1   1
    1   2   1
  1   3   3   1
1   4   6   4   1
1   5   10  10  5   1
```

$^{n-1}C_{i-1}$

ans = {}

$\leq O(n \times n \times n)$

$\leq O(n^2)$

```
fun( row = 1 -> n)
{   temp1 = {}
    fun( col = 1 -> row)
    {
        temp1.add( ncr(row-1, col-1) );
    }
    ans.add(temp1);
}
ret (ans)
```

Pascal Triangle

1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1

(1) Given R & C, tell the element at that place

R = 5 C = 3

ans = 6

$^{n-1}C_{C-1}$

(2) Print any Nth row of pascal triangle

N = 5

1 4 6 4 1

O(N)

(3) Given N, print the entire triangle

N = 6

O(N)

O(N × N) ≈ O(N²)

codestudio POWERED BY CODING NINJAS

Guided Paths Contests Interview Prep Practice Resources

Topics Problem Submissions Solution New

Current Submission

Report

Correct Answer
few secs ago

Test Cases 12/12

EXP: 40/40

Penalty 0%

C++

Did you find these test cases useful?

Previous Submissions

All languages

Status Test cases EXP Penalty Langua

Prev Next

Auto

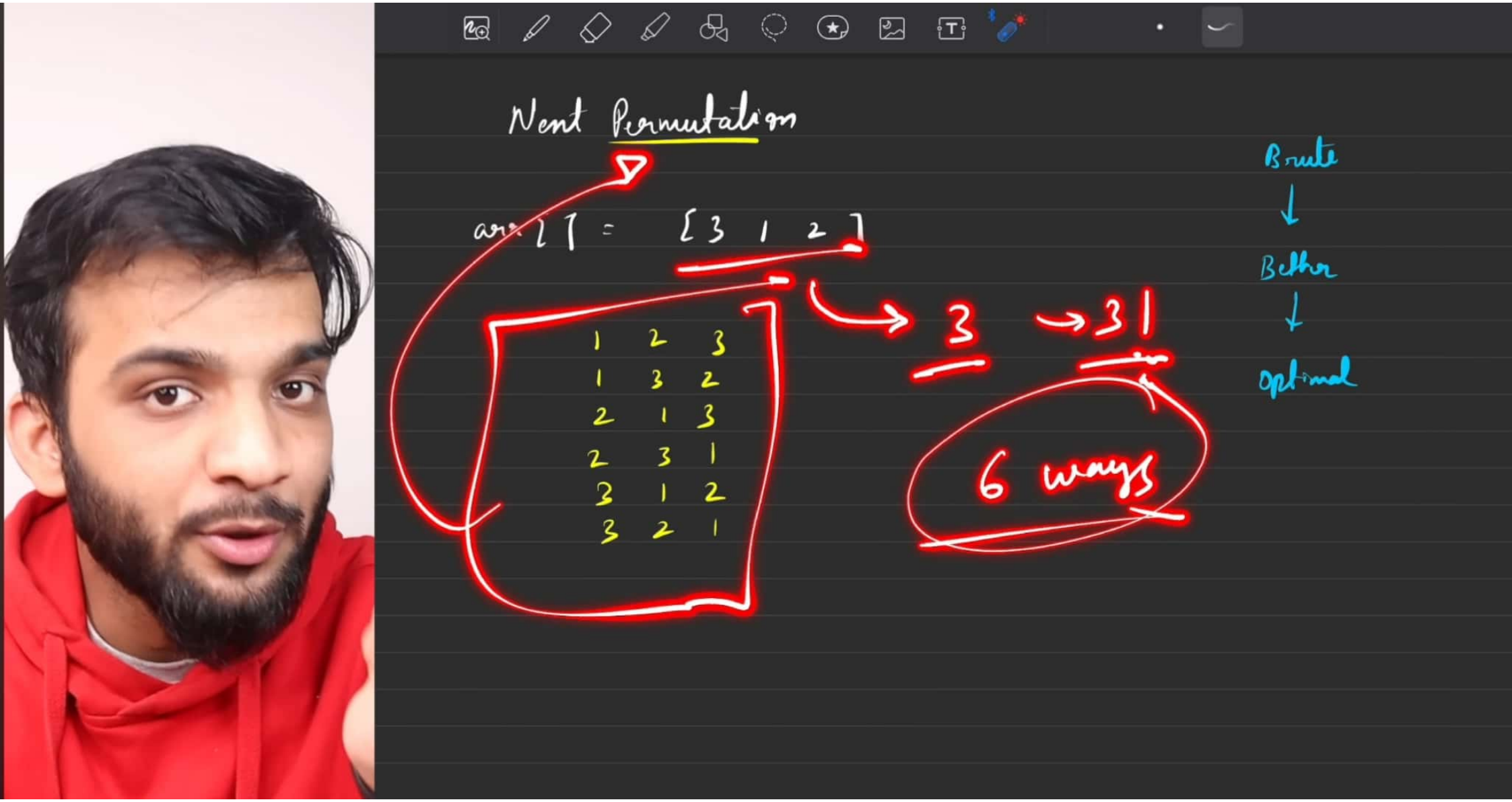
```cpp
vector<int> generateRow(int row) {
    long long ans = 1;
    vector<int> ansRow;
    ansRow.push_back(1);
    for(int col = 1;col<row;col++) {
        ans = ans * (row - col);
        ans = ans / (col);
        ansRow.push_back(ans);
    }
    return ansRow;
}
vector<vector<int>> pascalTriangle(int N) {
    vector<vector<int>> ans;
    for(int i = 1;i<=N;i++) {
        ans.push_back(ansRow);
    }
    return ans;
}
```

View hints

Last saved on 4:48:05 AM

Run

Next Permutation


Next Permutation
arr[] = [3 1 2]
1 2 3
1 3 2
2 1 3
2 3 1
3 1 2
3 2 1
3 → 31
6 ways
Brute
Better
Optimal

Next Permutation

arr[] = [3 1 2]

1 2 3
1 3 2
2 1 3
2 3 1
3 1 2
3 2 1

Brute

123 < 132 < 213

optimal

Next Permutation

arr[] = [3 1 2]

ans → [3, 2, 1]

Brute
↓
Better
↓
Optimal

1 2 3
1 3 2
2 1 3
2 3 1
3 1 2
3 2 1

Next Permutation

arr[] = [3 1 2]

Brute → Better → Optimal

ans → [3, 2, 1]

1 2 3
1 3 2
2 1 3
2 3 1
3 1 2
3 2 1

arr[] = [3 2 1]

ans → [1, 2, 3]

Next Permutation

{N! × N}

Brute XX

↓

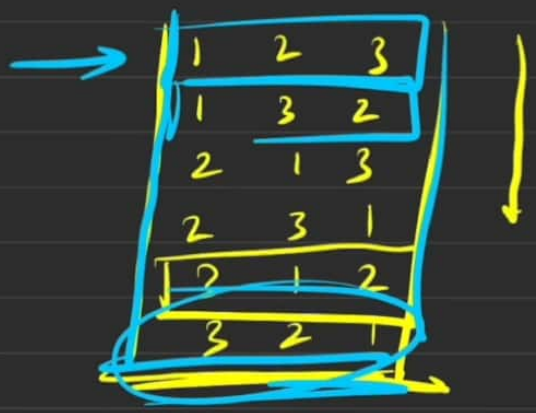
Better

↓

Optimal

arr[] = [3 1 2]

Recursion



1. Generate all sorted
2. Linear Search
3. Next Index

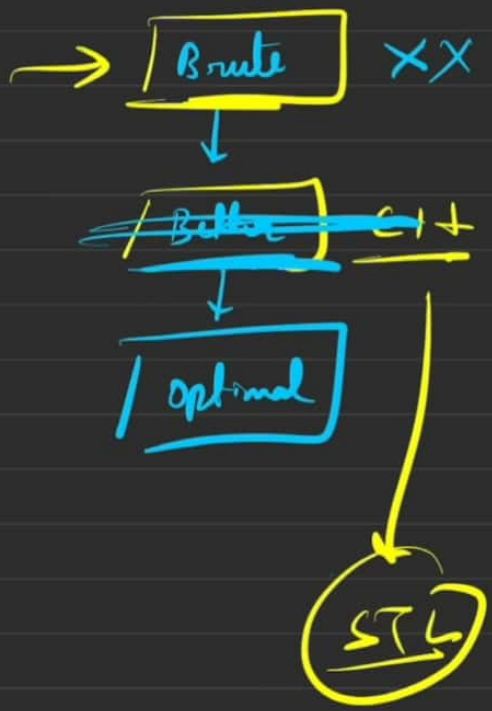
5 → 120

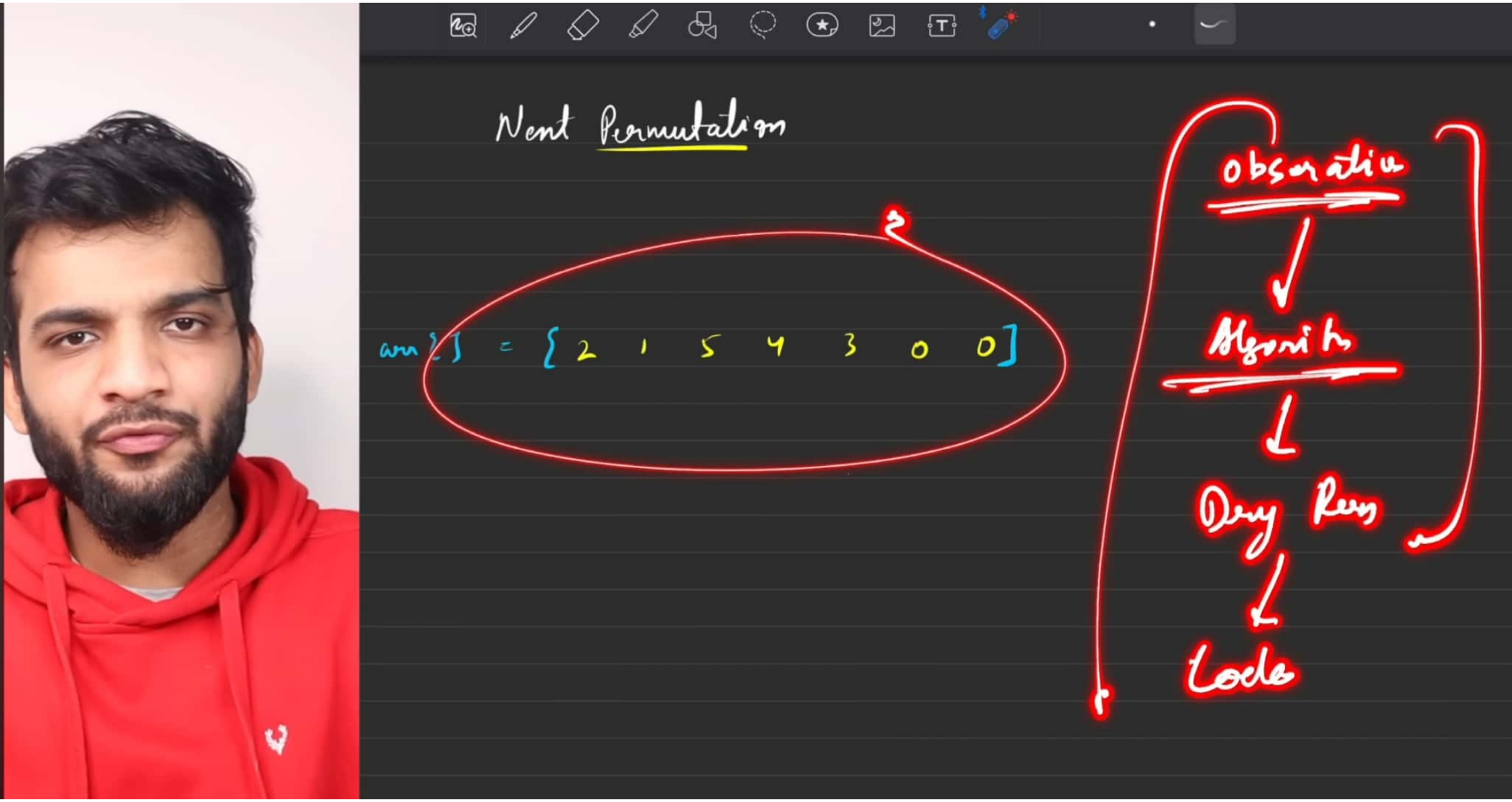
15 → 10^12

Next Permutation

arr[] = [3 1 2]

STL




Next Permutation
arr[] = [2 1 5 4 3 0 0]
Observation
Algorithm
Dry Run
Code

Next Permutation

raj

ram

rbn

arr[] = [2 1 5 4 3 0 0]

Next Permutation

1. longer prefix match
   ↳ a[i] < a[i+1]

i i+1

arr[] = { 2 | 1 | 5 4 3 0 0 }

2.

2

<1
>1

Next Permutation

1. longer prefix match
   ↳ a[i] < a[i+1]

2. find ≥ 1 , but the smallest one so that you stay close

arr[] = { 2 1 5 4 3 0 0 }

i i+1

2 3 0 0 1 4 5

5 4 1 0 0

3. Try to place rem in sorted array

arr[] = [2 1 | 5 4 3 0 0]

ind

ind = -1

for(i=n-2; i>=0; i--)
{
  if(a[i] < a[i+1])
  {
    ind = i;
    break;
  }
}

↳ a[i] < a[i+1]

2. find (>) , but the smallest one so that you stay close

3. Try to place num in sorted coreway

[1 2 3 4 5]

if (ind == -1)
reverse (arr)

[5 4 3 2 1]

Next Permutation

longer prefix match

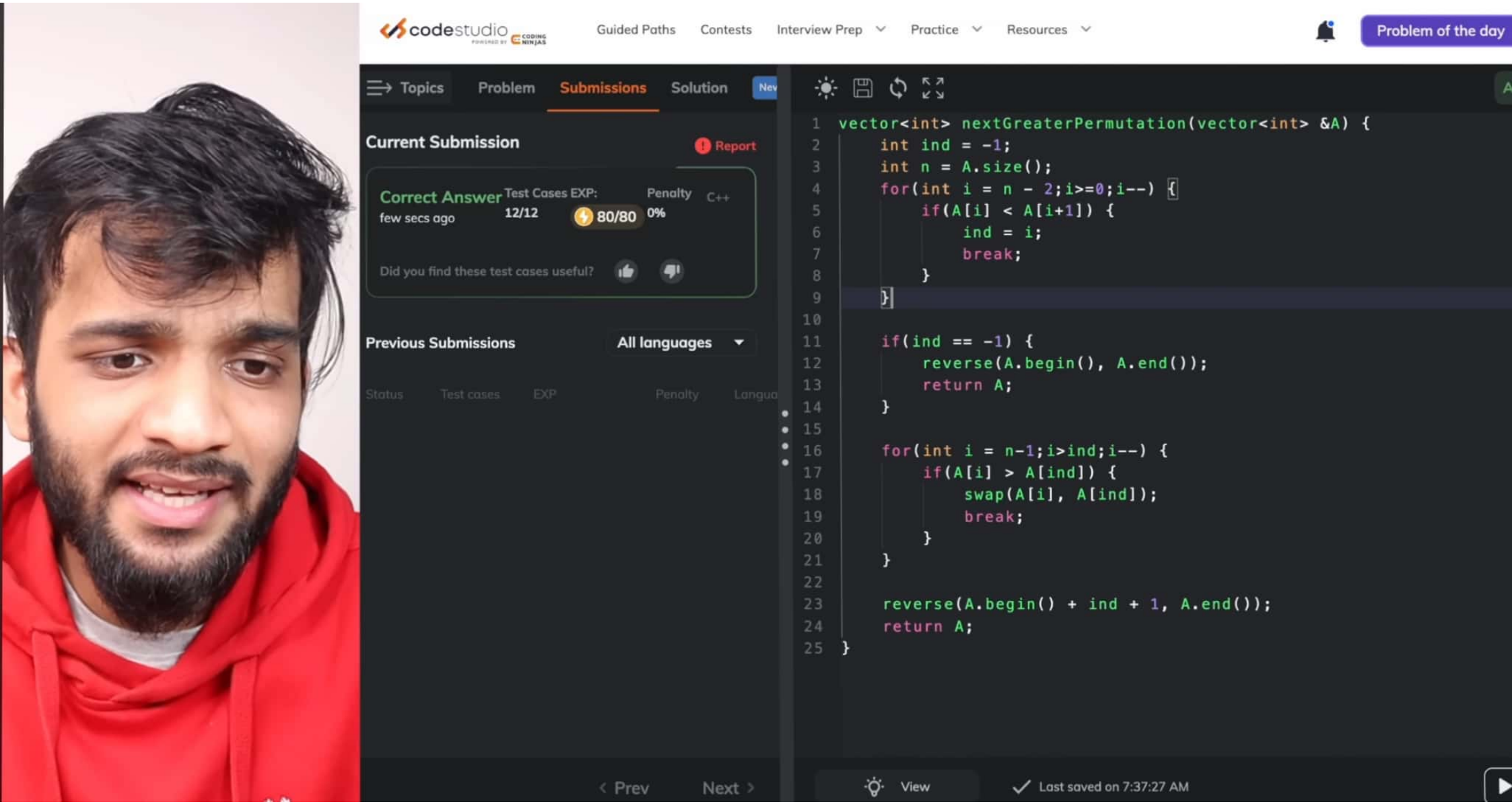
↳ a[i] < a[i+1]

arr[] = [2 1 | 5 4 3 0 0]

2. found (> N), but the

```
for(i=n-1 ; i>=ind ; i--)
    {
      if (arr[i] > arr[ind])
        {   swap(arr[i], arr[ind];
        }       break;
    }

rec(arr, ind+1, n-1)
```


codestudio
Guided Paths
Contests
Interview Prep
Practice
Resources
Problem of the day
Topics
Problem
Submissions
Solution
Current Submission
Report
Correct Answer
few secs ago
Test Cases EXP:
12/12
80/80
Penalty
0%
C++
Did you find these test cases useful?
Previous Submissions
All languages
Status
Test cases
EXP
Penalty
Prev
Next
View
Last saved on 7:37:27 AM
vector<int> nextGreaterPermutation(vector<int> &A) {
int ind = -1;
int n = A.size();
for(int i = n - 2;i>=0;i--) {
if(A[i] < A[i+1]) {
ind = i;
break;
}
}
if(ind == -1) {
reverse(A.begin(), A.end());
return A;
}
for(int i = n-1;i>ind;i--) {
if(A[i] > A[ind]) {
swap(A[i], A[ind]);
break;
}
}
reverse(A.begin() + ind + 1, A.end());
return A;
}

$O(3N)$

$= O(N) \; O(1)$

Kadane's Algorithm | Maximum Subarray Sum

Maximum Subarray Sum → contiguous part of the array

arr[] = [-2, -3, 4, -1, -2, 1, 5, -3]

ans = 7

[4, -1, 5] ← Subarray

Maximum Subarray Sum → contiguous part of the array

arr[] = [-2, -3, 4, -1, -2, 1, 5, -3]

ans = 7

7

Maximum Subarray Sum

Contiguous part of the array

arr[] = [-2, -3, 4, -1, -2, 1, 5, -3]

ans =

[0..0]
[0....1]
[0....2]
[0.....3]

```
for(i=0; i<n; i++)
{
    for(j=i; j<n; j++)
    {
    }
}
```

Brute

Better

Optimal

Maximum Subarray Sum

contiguous part of the array

arr[] = [-2, -3, 4, -1, -2, 1, 5, -3]

ans = 7

```
for(i=0; i<n; i++)
{
    for(j=i; j<n; j++)
    {
    }
}
```

[1...1]

[1...2]

[1...3]

Brute

Better

Optimal

Maximum Subarray Sum

arr[] = [-2, -3, 4, -1, -2, 1, 5, -3]

ans = 7

```
maxi = INT_MIN;
for(i=0; i<n; i++)
{
    for(j=i; j<n; j++)
    {
        sum=0
        for(k=i -> j)
            sum += arr[k]
        maxi = max(sum, maxi)
    }
}
```

TC — SC

- $O(N^3)$ Brute — $O(1)$
- Better
- Optimal

Optimal → carry sub arr

Maximum Subarray Sum

arr[] = [-2, -3, 4, -1, -2, 1, 5, -3]

ans = 7

```
maxi = INT_MIN;
for(i=0; i<n; i++)
{
    sum = 0
    for(j=i; j<n; j++)
    {
        sum += arr[j];
        maxi = max(sum, maxi);
    }
}
```

	TC	SC
•	$O(N^3)$ Brute	$O(1)$
	$O(N^2)$ Better	$O(1)$
	Optimal	

Optimal → carry sub arr

Maximum Subarray Sum

Contiguous part of the array

arr[] = [-2, -3, 4, -1, -2, 1, 5, -3]

0

ans = 7

mini = ~~INT_MIN~~ -2

Sum = ~~0~~ -2

[-2, -3]

[-3]

-5 < -3

	TL	SL
•	$O(N^3)$ Brute	O(1)
↓		
	$O(N^2)$ Better	O(1)

Optimal

[Kadane's Algorithm]

Maximum Subarray Sum

contiguous part of the array

arr[] = [-2, -3, 4, -1, -2, 1, 5, -3]
0

ans = 7

mini = ~~INT_MIN~~ -2

Sum = ~~0~~
= ~~-2~~
0

sum < 0

	TL	SL
•	$O(N^3)$ Brute	O(1)
	↓	
	$O(N^2)$ Better	O(1)

Optimal

[Kadane's Algorithm]

Maximum Subarray Sum

contiguous part of the array

arr[] = [-2, -3, 4, -1, -2, 1, 5, -3]
0 1

ans = 7

mani = 4

Sum = ~~0~~ ~~-3~~ 4
~~-2~~ 0
0

sum < 0

TL		SL
O(N³)	Brute	O(1)
O(N²)	Better	O(1)

Optimal

[Kadane's Algorithm]

Maximum Subarray Sum

contiguous part of the array

arr[] = [-2, -3, 4, -1, -2, 1, 5, -3]
index: 0 1

ans = 7

maxi = 4

Sum = ~~0~~ ~~-2~~ ~~-3~~ ~~4~~
~~-2~~ 0 3
0

sum < 0

+3

	TL	SL
•	$O(N^3)$ Brute	$O(1)$
↓		
	$O(N^2)$ Better	$O(1)$

Optimal

[Kadane's Algorithm]

Maximum Subarray Sum

contiguous part of the array

arr[] = [-2, -3, 4, -1, -2, 1, 5, -3]

0 1

ans = 7

maxi = 7

Sum = 0 -3 4
-2 0 3
0 1
2
7
4

sum < 0

	TL	SL
•	$O(N^3)$ Brute	O(1)
	↓	
	$O(N^2)$ Better	O(1)

Optimal

[Kadane's Algorithm]

codestudio POWERED BY CODING NINJAS

Guided Paths | Contests | Interview Prep | Practice | Resources | Problem of the day

Topics | Problem | Submissions | Solution | New Discuss

Maximum Subarray Sum

Contributed by
Achint Narang

Medium | 80/80 | Avg time to solve 35 mins | Success Rate 81 %

Share | 702 upvotes

Problem Statement Suggest Edit

You are given an array (ARR) of length N, consisting of integers. You have to find the sum of the subarray **(including empty subarray)** having maximum sum among all subarrays.
A subarray is a contiguous segment of an array. In other words, a subarray can be formed by removing 0 or more integers from the beginning, and 0 or more integers from the end of an array.

Note :

```
The sum of an empty subarray is 0.
```

Detailed explanation (Input/output format, Notes, Images)

Constraints :

```
1 <= N <= 10^6
-10^6 <= A[i] <= 10^6
```

‹ Prev Next ›

```cpp
#include <bits/stdc++.h>
long long maxSubarraySum(int arr[], int n)
{
    long long sum = 0, maxi = LONG_MIN;
    for(int i = 0;i<n;i++) {
        sum += arr[i];

        if(sum > maxi) {
            maxi = sum;
        }

        if(sum < 0) {
            sum = 0;
        }
    }
    return maxi;
}
```

View | Last saved on 9:39:18 PM

$\{-4 \quad -2 \quad -3 \quad \boxed{-1}$

$\{-1\}$ $\boxed{\sup = -1}$

mani = 4 7

Sum = 0 -3 4
-2 0 3
0 1
2
7
4

sum < 0

TL		SL
$O(N^3)$	Brute	$O(1)$
$O(N^2)$	Better	$O(1)$

Optimal

$O(N)$ [Kadane's Algorithm] $O(1)$

{ -4 -2 -3 [-1] }

Problem Maximum Subarray Sum → contiguous part of the array

arr[] = [-2, -3, 4, -1, -2, 1, 5, -3]

```
for (i=0; i<n; i++)
{
    sum = sum + arr[i];

    if (sum > maxi)
    {
        maxi = sum;
    }

    if (sum < 0)
    {
        sum = 0;
```

maxi = INT_MIN

sum = 0

Problem Maximum Subarray Sum → contiguous part of the array

arr[7] = [-2, -3, 4, -1, -2, 1, 5, -3]

```
for (i=0; i<n; i++)
  {
    sum = sum + arr[i];

    if (sum > maxi)
      {
        maxi = sum;
      }

    if (sum < 0)
      {
        sum = 0;
```

maxi = INT_MIN

sum = 0

```
for (i=0; i<n; i++)
  {  if (sum==0) start = i;
     sum = sum + arr[i];

     if (sum > maxi)
       {
           maxi = sum;
           ansStart = start, ansEnd = i;
       }

     if (sum < 0)
       {
           sum = 0;
       }

}
```

maxi = INT_MIN

sum = 0

ansStart = -1
ansEnd = -1

TC → O(N)
SC → O(1)

Sort an array of 0's 1's & 2's

Sort an array of 0's, 1's and 2's.

arr[] = | 0 | 1 | 2 | 0 | 1 | 2 | 1 | 2 | 0 | 0 | 0 | 1 |

0 / 1 / 2

TC (N log N) SC (N) Sort → Brute

Better

Optimal

Sort an array of 0's, 1's and 2's.

arr[] = 0 1 2 0 1 2 1 2 0 0 0 1

0 1 2

TC (N log N) SC (N) Sort → Brute

Better

Optimal

cnt 0 = 5

cnt 1 = 4

cnt 2 = 3

arr[] = | 0 | 1 | 2 | 0 | 1 | 2 | 1 | 2 | 0 | 0 | 0 | 1 |

cnt0 = 0 cnt1 = 1 cnt2 = 2

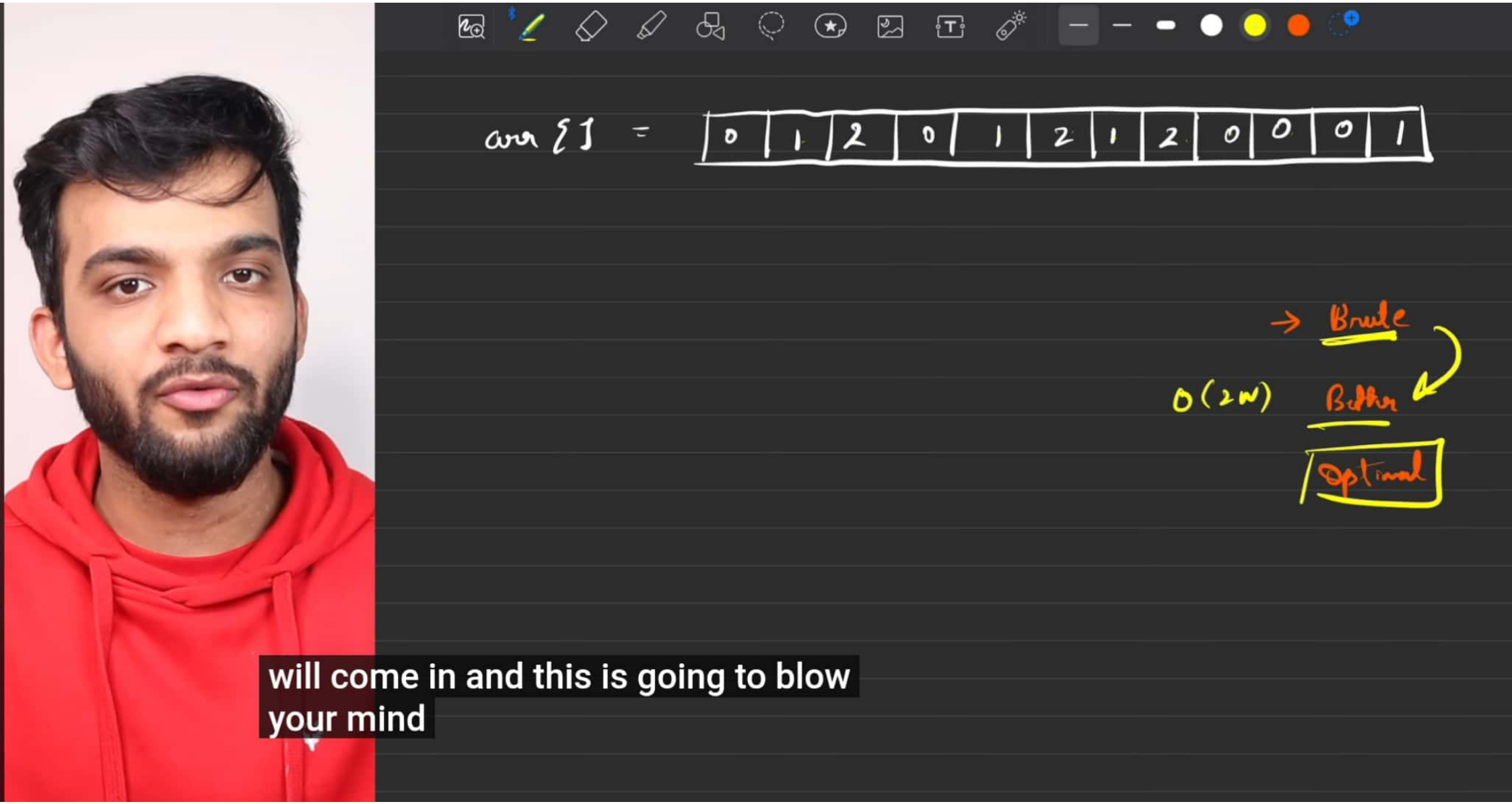
```
for(i=0; i<n; i++)
{
    if(a[i]==0) cnt0++;
    else if(a[i]==1) cnt1++
    else cnt2++;
}
```

→ O(N) → Brute → Better → Optimal

```
for(i=0; i<cnt0; i++) a[i]=0 →
for(i=cnt0; i<cnt0+cnt1; i++) a[i]=1 →
for(i=cnt0+cnt1; i<n; i++) a[i]=2 →
```

O(N)

TC → O(2N) SC → O(1)



will come in and this is going to blow your mind

Dutch National Flag Algorithm → Optimal

arr[] = | 0 | 1 | 1 | 0 | 1 | 2 | 1 | 2 | 0 | 0 | 0 |

low

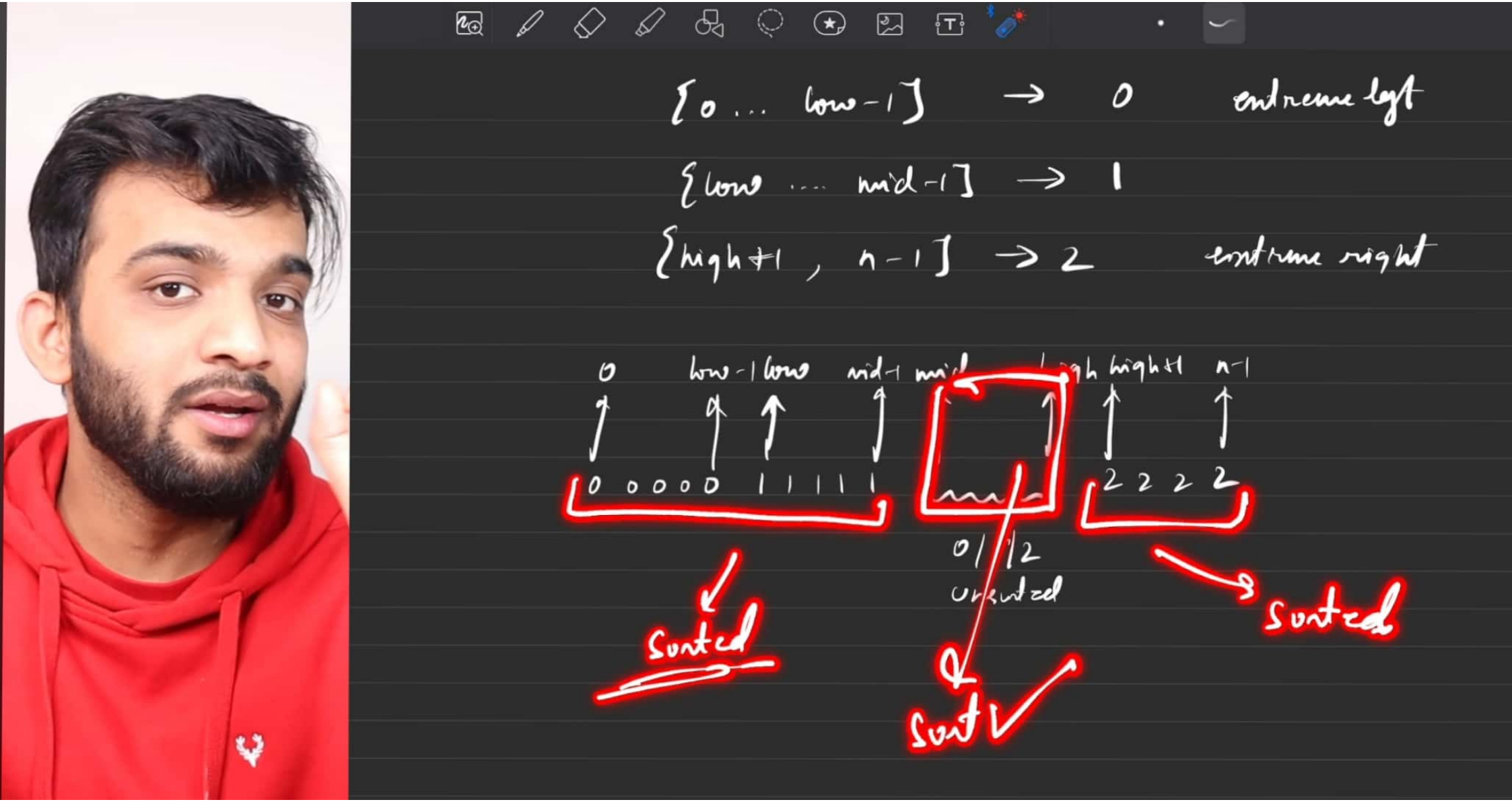
mid

high

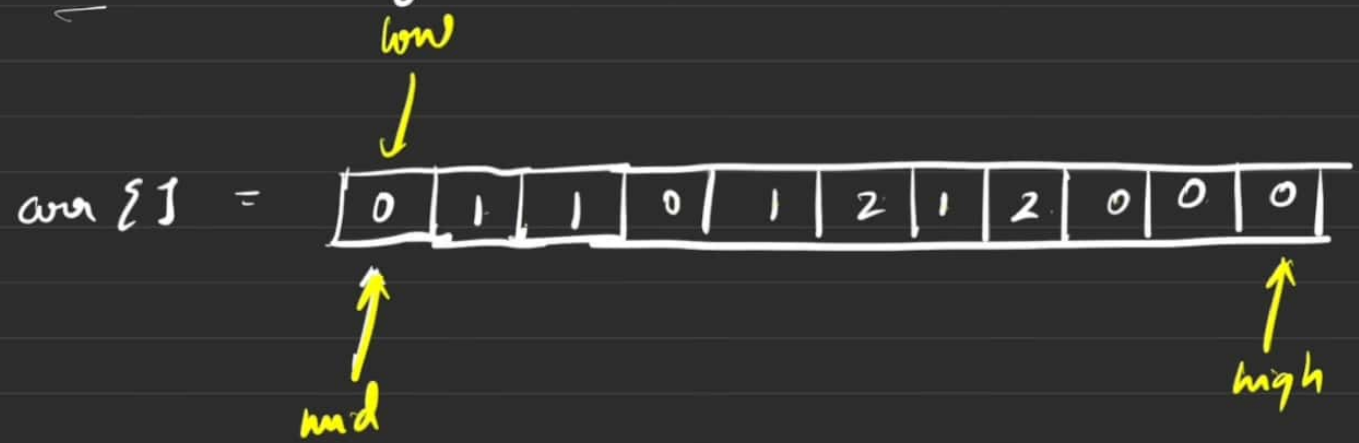
$[0 \ldots low-1] \rightarrow 0$ extreme left

$[low \ldots mid-1] \rightarrow 1$

$[high+1, n-1] \rightarrow 2$ extreme right


[0 ... low-1] → 0 extreme left
[low ... mid-1] → 1
[high+1, n-1] → 2 extreme right
0
low-1 low
mid-1 mid
high high+1
n-1
0 0 0 0 0 1 1 1 1 1
2 2 2 2
0/1/2
Unsorted
Sorted
Sorted
Sort

Dutch National Flag Algorithm → Optimal



low

mid

high

low

mid

high

mid

high

a[mid] == 0

a[mid] == 1

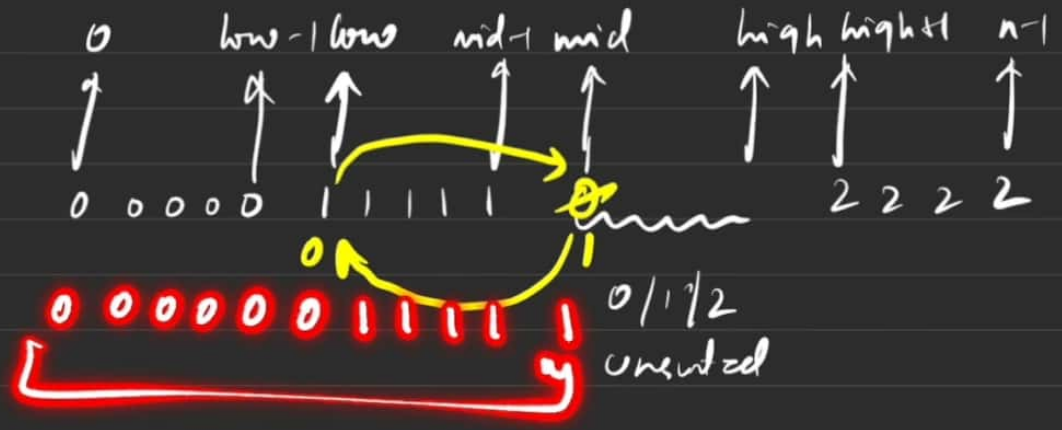
a[mid] == 2

[0 ... low-1] → 0 extreme left

[0 ... low-1] → 0 extreme left

[low ... mid-1] → 1

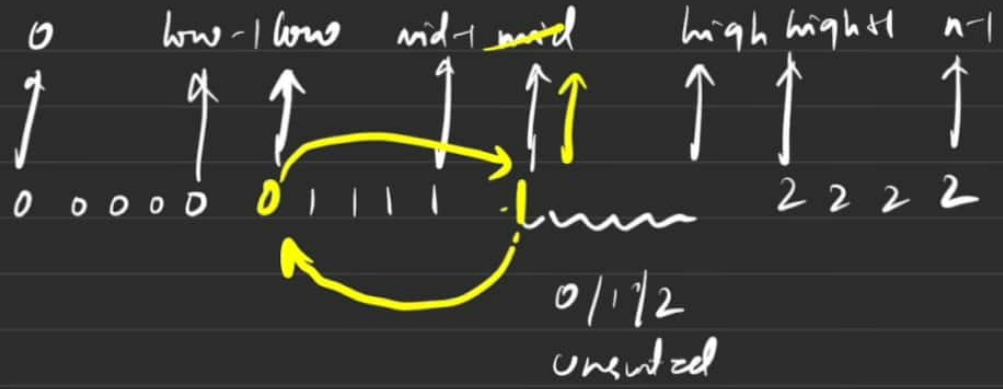
[high+1, n-1] → 2 extreme right



[0 ... low-1] → 0 extreme left

[low .... mid-1] → 1

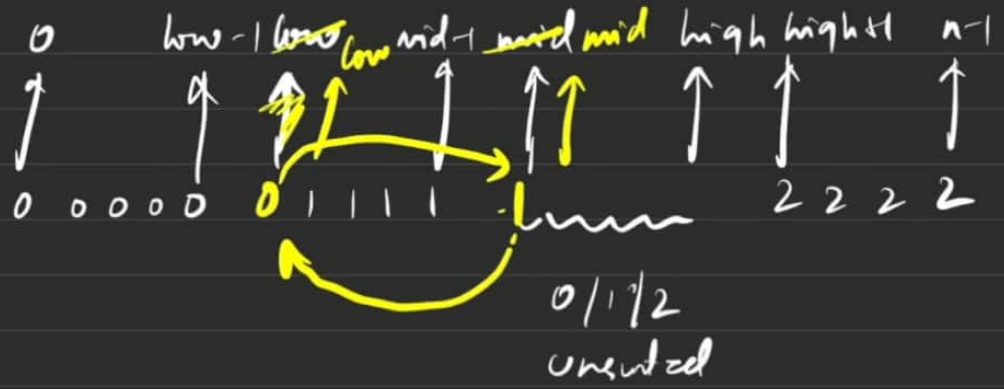
[high+1 , n-1] → 2 extreme right



$[0 \ldots low-1] \rightarrow 0$ extreme left

$[low \ldots mid-1] \rightarrow 1$

$[high+1, n-1] \rightarrow 2$ extreme right



arr[] = | 0 | 1 | 1 | 0 | 1 | 2 | 1 | 2 | 0 | 0 | 0 |

mid

high

low

mid

high

✓ a[mid] == 0 swap(a[low], a[mid])
low++, mid++

a[mid] == 1

a[mid] == 2

[0 ... low-1]

[low ... mid-1] → 1

[high+1, n-1] → 2 extreme right

0/1/2
unsorted

[0 ... low-1]

[low ... mid-1] → 1

[high+1, n-1] → 2 extreme right

low

mid

high

mid

high

a[mid] == 0 ✓ swap(a[low], a[mid])
low++, mid++

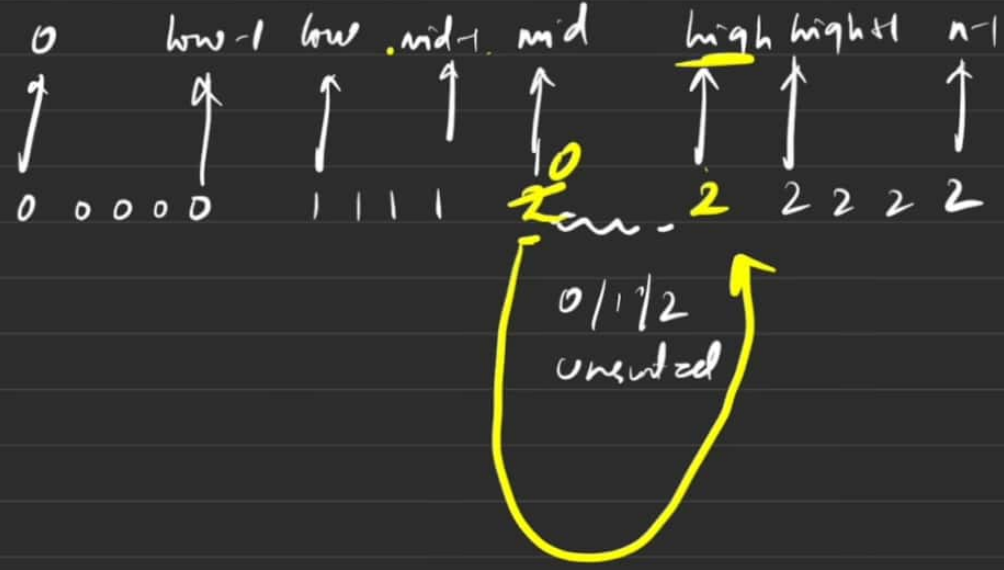
a[mid] == 1 mid++

a[mid] == 2

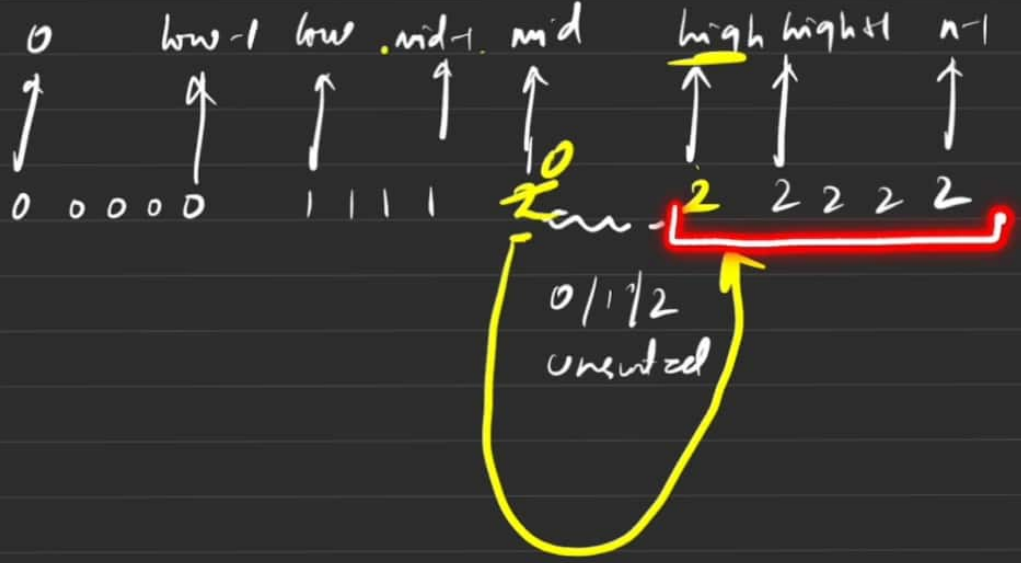
[0 ... low-1] → 0 extreme left

[high+1, n-1] → 2 extreme right

{high+1 , n-1} → 2 extreme right



{high+1 , n-1} → 2 extreme right



low

mid

high

mid

high

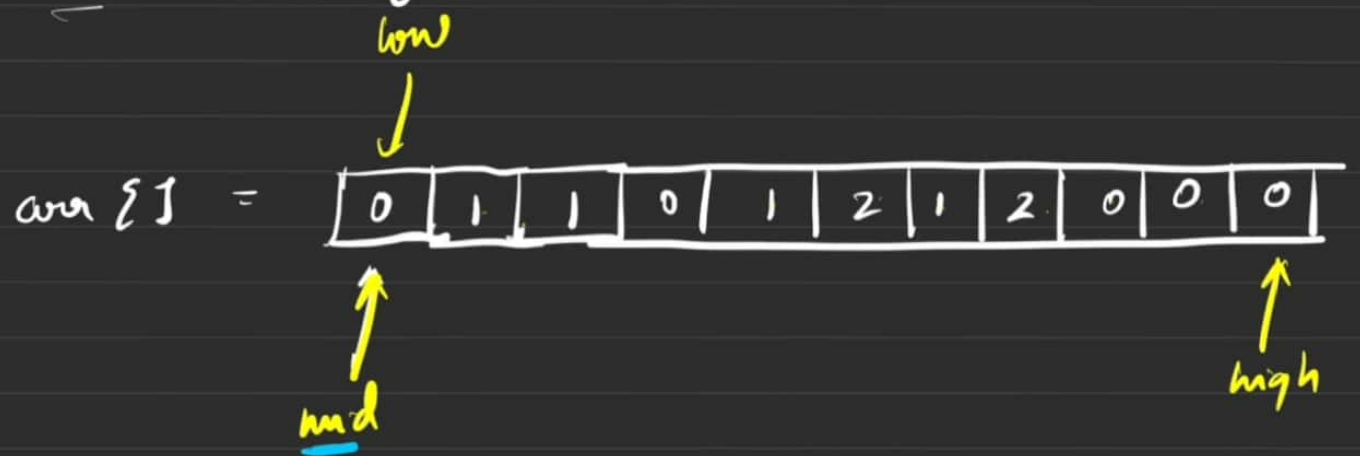
a[mid] == 0 → swap(a[low], a[mid]) low++, mid++

a[mid] == 1 → mid++

a[mid] == 2 → swap(a[mid], a[high]) high--;

[0 ... low-1] → 0 extreme left

Dutch National Flag Algorithm → Optimal



low

mid

high

✓ a[mid] == 0 swap(a[low], a[mid])
low++, mid++

✓ a[mid] == 1 mid++

Dutch National Flag Algorithm → Optimal

low

arr[] = | 0 | 1 | 1 | 0 | 1 | 2 | 1 | 2 | 0 | 0 | 0 |

mid

high

low

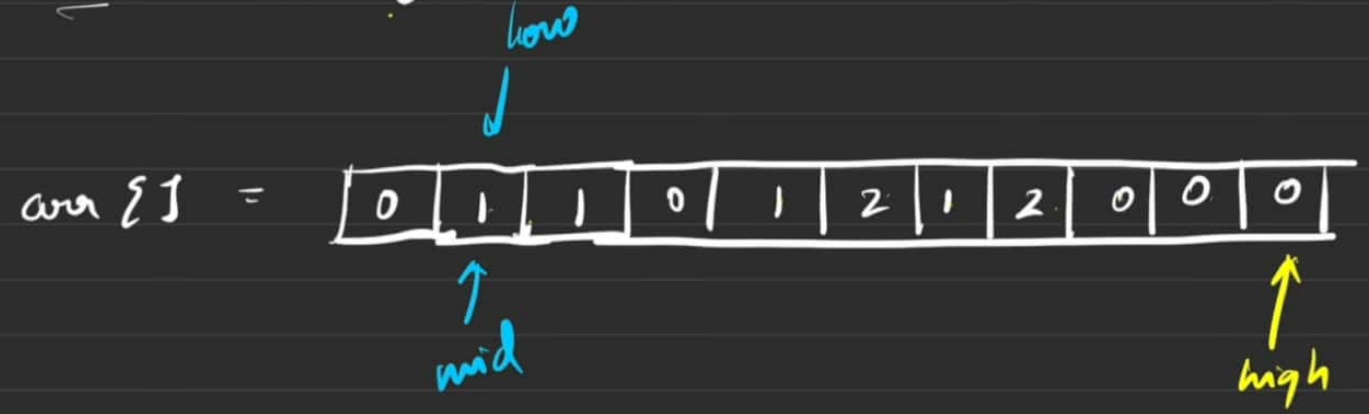
mid

high

✓ a[mid] == 0 → swap(a[low], a[mid]) ; low++ , mid++

✓ a[mid] == 1 → mid++

Dutch National Flag Algorithm → Optimal



low

mid

high

✓ a[mid] == 0 swap(a[low], a[mid])
low++, mid++

✓ a[mid] == 1 mid++

Dutch National Flag Algorithm → Optimal

low

arr[] = | 0 | 1 | 1 | 0 | 1 | 2 | 1 | 2 | 0 | 0 | 0 |

mid

high

low

mid

high

✓ a[mid] == 0 → swap(a[low], a[mid]); low++, mid++

✓ a[mid] == 1 → mid++

Dutch National Flag Algorithm → Optimal

low

arr[] = | 0 | 1 | 1 | 0 | 1 | 2 | 1 | 2 | 0 | 0 | 0 |

mid

high

low

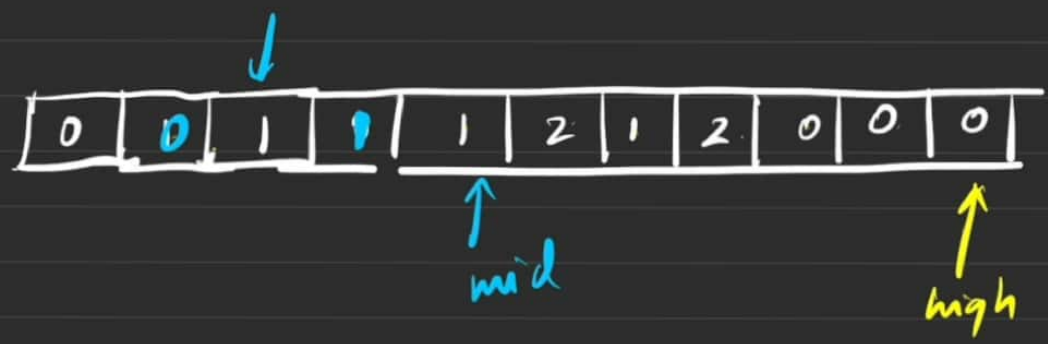
mid

high

✓ a[mid] == 0 → swap(a[low], a[mid]); low++, mid++

✓ a[mid] == 1 → mid++

Dutch National Flag Algorithm → Optimal



low

mid

high

✓ a[mid] == 0 → swap(a[low], a[mid]) low++, mid++

✓ a[mid] == 1 → mid++

Dutch National Flag Algorithm → Optimal

low

arr[] = | 0 | 0 | 1 | 1 | 1 | 2 | 1 | 2 | 0 | 0 | 0 |

mid

high

low

mid

high

✓ a[mid] == 0 → swap(a[low], a[mid]); low++, mid++

✓ a[mid] == 1 → mid++

Dutch National Flag Algorithm → Optimal

low

arr[] = | 0 | 0 | 1 | 1 | 1 | 2 | 1 | 2 | 0 | 0 | 2 |

mid

high

low

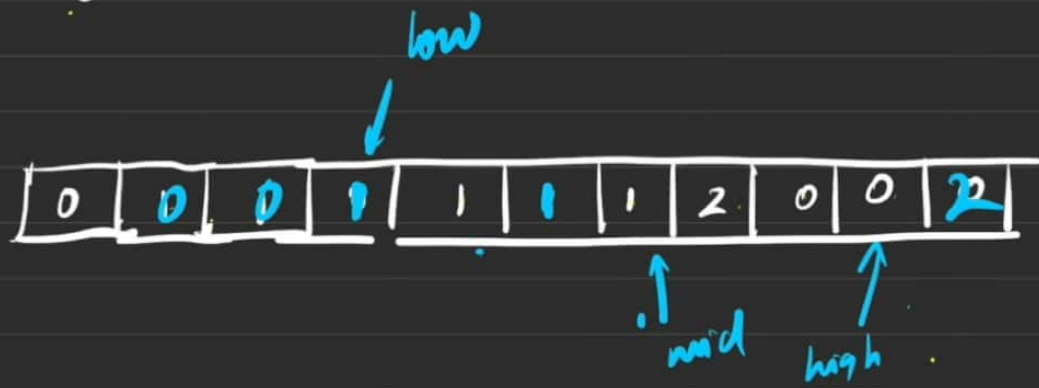
mid

high

✓ a[mid] == 0 → swap(a[low], a[mid]); low++, mid++

✓ a[mid] == 1 → mid++

Dutch National Flag Algorithm → Optimal



low

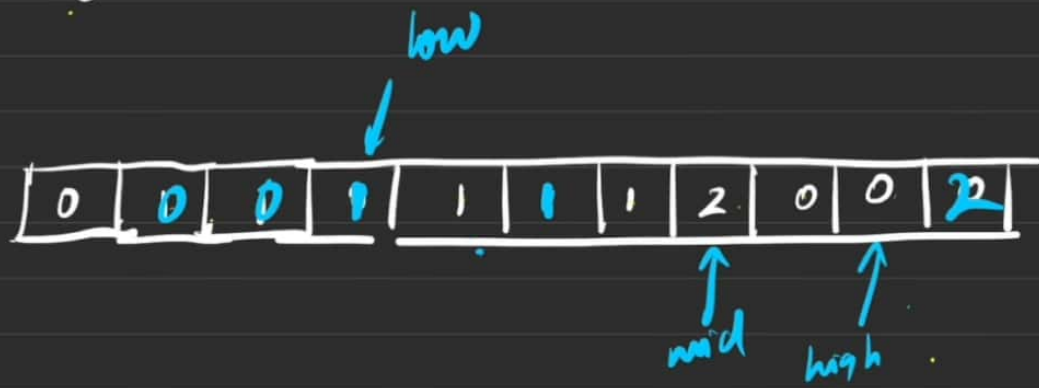
mid

high

✓ a[mid] == 0 → swap(a[low], a[mid]) ; low++, mid++

✓ a[mid] == 1 → mid++

Dutch National Flag Algorithm → Optimal



low

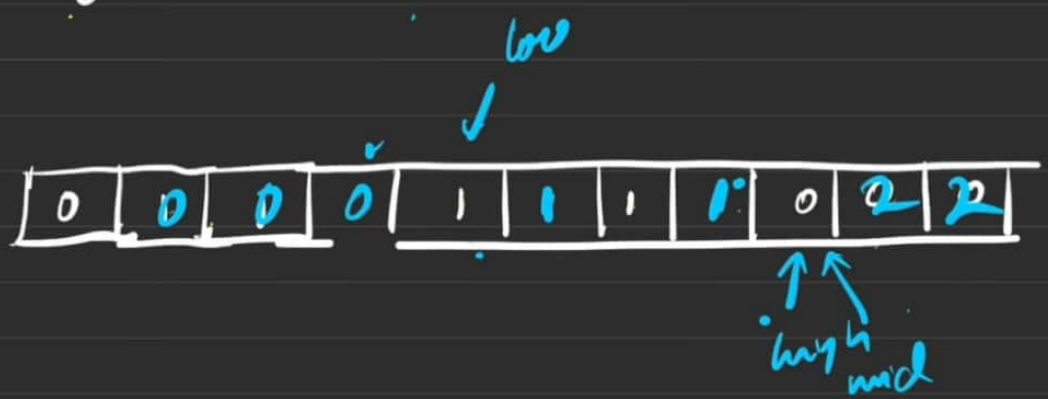
mid

high

✓ a[mid] == 0 → swap(a[low], a[mid]) low++, mid++

✓ a[mid] == 1 → mid++

Dutch National Flag Algorithm → optimal



low

mid

high

✓ a[mid] == 0 swap(a[low], a[mid])
low++, mid++

✓ a[mid] == 1 mid++

Dutch National Flag Algorithm → optimal

arr[] = 0 0 0 0 0 1 1 1 1 2 2

low, high, mid

low

mid

high

a[mid] == 0 → swap(a[low], a[mid]); low++, mid++

a[mid] == 1 → mid++

arr[] = [0 0 0 0 0 1 1 1 1 2 2]

high

mid

low

low

mid

high

a[mid] == 0 ✓ swap(a[low], a[mid])
low++, mid++

a[mid] == 1 ✓ mid++

a[mid] == 2 ✓ swap(a[mid], a[high])
high--;

codestudio POWERED BY CODING NINJAS

Guided Paths | Contests | Interview Prep | Practice | Resources | Problem of the day

Topics | Problem | Submissions | Solution | New

Sort An Array of 0s, 1s and 2s

Contributed by
Anup Kumar Singh

Easy | 0/40 | Avg time to solve 10 mins | Success Rate 90 %

Share | 3 upvotes

Problem Statement Suggest Edit

You have been given an array/list ARR consisting of 'N' elements. Each element in the array is either 0, 1 or 2. Now, your task is to sort this array/list in increasing order. For example, if ARR = [0, 2, 1, 1], then after sorting ARR must look like [0, 1, 1, 2].

Detailed explanation (Input/output format, Notes, Images)

Constraints:

```
1 <= N <= 5000
0 <= ARR[i] <= 2

Time limit: 1 sec
```

Prev | Next

Auto | C++ (g++ 5.4)

```cpp
#include <bits/stdc++.h>
void sortArray(vector<int>& arr, int n)
{
    int low = 0, mid = 0, high = n-1;
    while(mid <= high) {
        if(arr[mid] == 0) {
            swap(arr[low], arr[mid]);
            low++;
            mid++;
        }
        else if(arr[mid] == 1) {
            mid++;
        }
        else {
            swap(arr[mid], arr[high]);
            high--;
        }
    }
}
```

View hints | Last saved on 6:34:04 | Run | Submit

Console

Sample Test Case

5/5 test cases passsed

Test Case 1: Correct Ar
Test Case 2: Correct Ar
Test Case 3: Correct Ar
Test Case 4: Correct Ar
Test Case 5: Correct Ar

TL → O(N)

O(1)

SL →

TL ??

+1

mid +1

−1

high

N

Dutch National Flag Algorithm → Optimal

low

Best Time to Buy and Sell Stock

Best Time to Buy & Sell Stock

arr[] → [7, 1, 5, 3, 6, 4] n = 6

only one

Buy → 1

Sell → 6

6 − 1 = 5

Maximize

7 1 5 3 6 4

If you are selling on ith day
you buy on the minimum price from 1st → (i-1)

mini = a[0], profit = 0

for(i = 1; i < n; i++)
{

```
mini = a[0], profit = 0
for(i = 1; i < n; i++)
{
    cost = a[i] - mini;
    profit = max(profit, cost);

    mini = min(mini, a[i]);

}
```

codestudio POWERED BY CODING NINJAS

Problem | Submissions

Correct Answer Report

Submitted on Mar 11, 2022, 4:46:52 AM

Penalty	Score
0%	100%

Runtime	Language
196ms	C++ (g++ 5.4)

```cpp
int maximumProfit(vector<int> &prices){
    int mini = prices[0];
    int maxProfit = 0;
    int n = prices.size();
    for(int i = 0;i<n;i++) {
        int cost = prices[i] - mini;
        maxProfit = max(maxProfit, cost);
        mini = min(mini, prices[i]);
    }
    return maxProfit;
    // Write your code here.
}
```

Previous Next

Show Hint Last saved on 4:46:53 AM