

Tanish Mittal
IT-C
2000290130174

Day – 11 Binary Search

Problem Statement: Given two numbers N and M, find the Nth root of M.

The nth root of a number M is defined as a number X when raised to the power N equals M.

```
def find_nth_root(N, M):
```

```
    X = M ** (1/N)
```

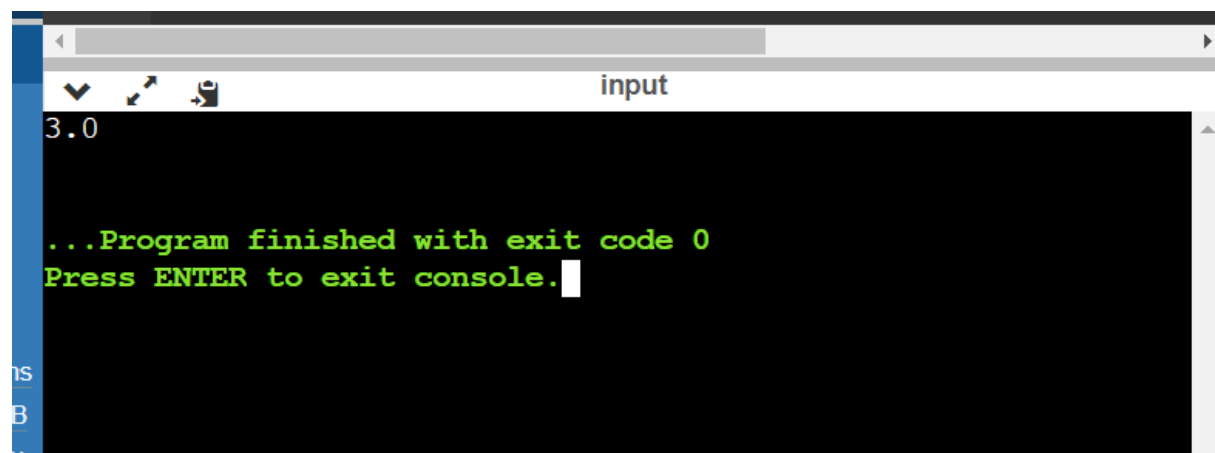
```
    return X
```

```
N = 3
```

```
M = 27
```

```
result = find_nth_root(N, M)
```

```
print(result)
```

A screenshot of a terminal window with a dark background. The title bar at the top says 'input'. The terminal shows the output '3.0' on the first line. The second line shows the message '...Program finished with exit code 0' in green text. The third line shows 'Press ENTER to exit console.' in green text, followed by a white cursor. On the left side of the terminal, there is a vertical blue bar with some text partially visible: 'ns', 'B', and 'v'.

Problem Statement: Given a row-wise sorted matrix of size r*c, where r is no. of rows and c is no. of columns, find the median in the given matrix.

```
def find_median(matrix):
```

```
    rows, cols = len(matrix), len(matrix[0])
```

```
    arr = []
```

```

for i in range(rows):
    for j in range(cols):
        arr.append(matrix[i][j])

arr.sort()

middle_idx = len(arr) // 2
median = arr[middle_idx]

return median

matrix = [
    [1, 4, 9],
    [2, 5, 6],
    [3, 8, 7]
]

median = find_median(matrix)
print("Median:", median)

```

The screenshot shows a code editor with a dark theme. The code from the previous block is visible. Below the editor, a console window is open, showing the output of the program. The output is 'Median: 5'. Below that, a green message indicates the program finished successfully with exit code 0 and prompts the user to press ENTER to exit the console.

Problem Statement: Given a sorted array of N integers, where every element except one appears exactly twice and one element appears only once. Search Single Element in a sorted array.

```

def find_single_element(arr):
    left, right = 0, len(arr) - 1

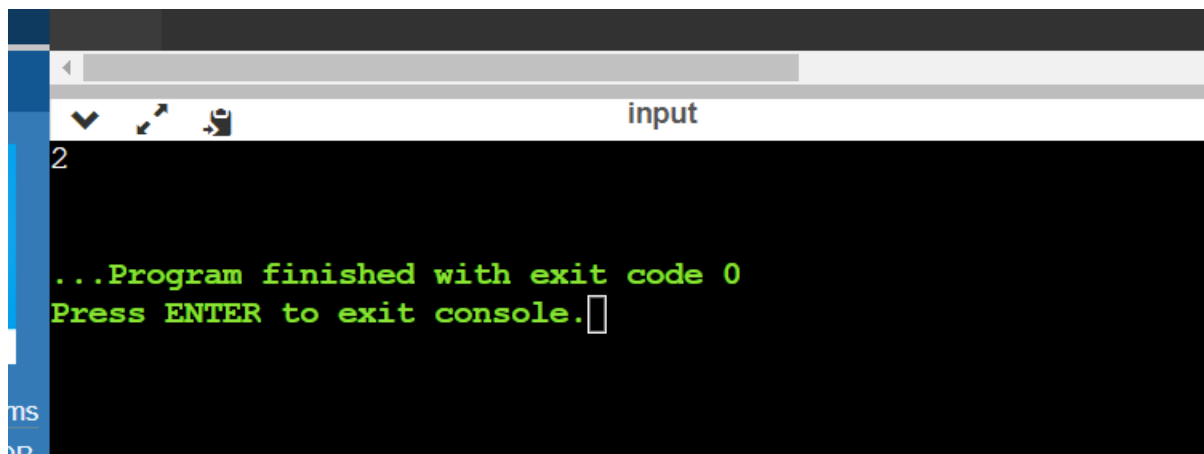
```

```

while left < right:
    mid = left + (right - left) // 2
    if mid % 2 == 1:
        mid -= 1

    if arr[mid] == arr[mid + 1]:
        left = mid + 2
    else:
        right = mid
return arr[left]
arr = [1, 1, 2, 3, 3, 4, 4, 8, 8]
print(find_single_element(arr))

```



```

input
2
...Program finished with exit code 0
Press ENTER to exit console.

```

Problem Statement: Given an integer array `arr` of size `N`, sorted in ascending order (with distinct values) and a target value `k`. Now the array is rotated at some pivot point unknown to you. Find the index at which `k` is present and if `k` is not present return `-1`.

```

def find_target_index(arr, k):
    low, high = 0, len(arr) - 1

    while low <= high:
        mid = (low + high) // 2

```

```

    if arr[mid] == k:
        return mid

    if arr[low] <= arr[mid]:
        if arr[low] <= k <= arr[mid]:
            high = mid - 1
        else:
            low = mid + 1
    else:
        if arr[mid] <= k <= arr[high]:
            low = mid + 1
        else:
            high = mid - 1

    return -1

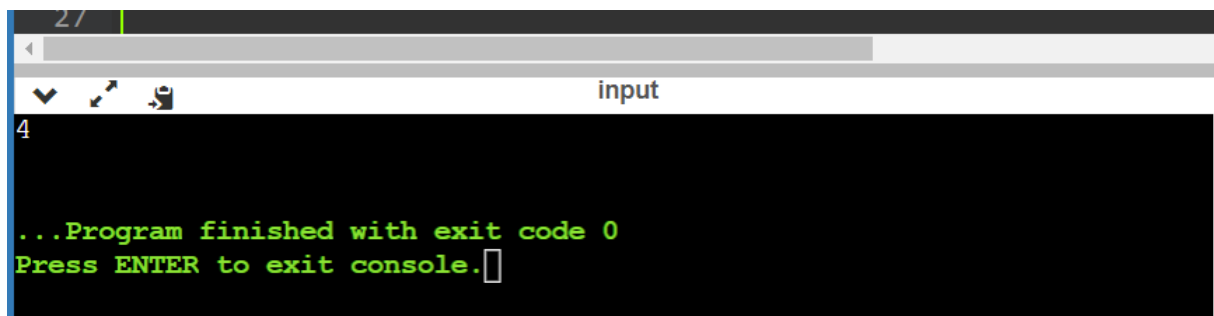
arr = [4, 5, 6, 7, 0, 1, 2, 3]
k = 0

```

```

index = find_target_index(arr, k)
print(index)

```



```

27
input
4
...Program finished with exit code 0
Press ENTER to exit console.

```

Problem Statement: Given two sorted arrays arr1 and arr2 of size m and n respectively, return the median of the two sorted arrays.

```

def findMedianSortedArrays(arr1, arr2):

```

```

merged = []

i, j = 0, 0

while i < len(arr1) and j < len(arr2):
    if arr1[i] <= arr2[j]:
        merged.append(arr1[i])
        i += 1
    else:
        merged.append(arr2[j])
        j += 1

while i < len(arr1):
    merged.append(arr1[i])
    i += 1

while j < len(arr2):
    merged.append(arr2[j])
    j += 1

length = len(merged)

if length % 2 == 0:
    mid1 = length // 2
    mid2 = mid1 - 1
    median = (merged[mid1] + merged[mid2]) / 2
else:
    mid = (length - 1) // 2

```

```
median = merged[mid]
```

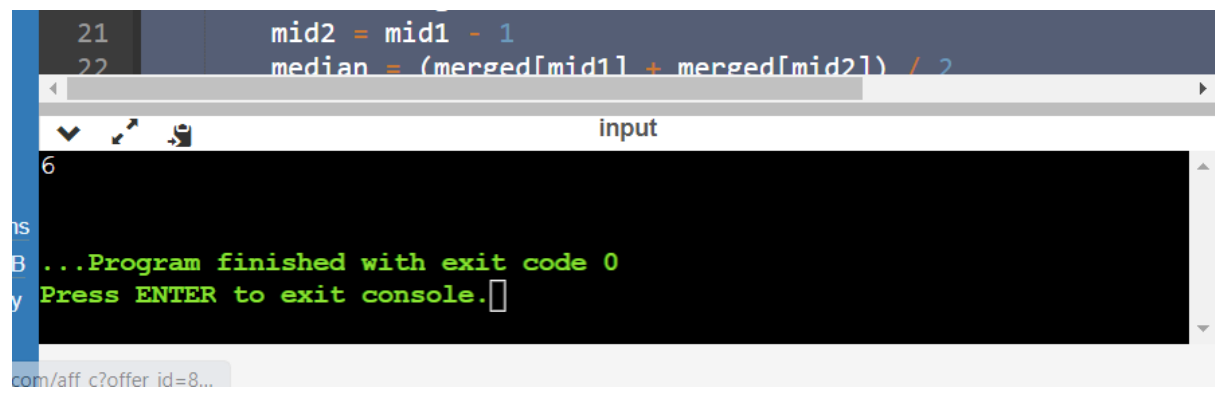
```
return median
```

```
arr1 = [1, 4, 7, 10, 12]
```

```
arr2 = [2, 3, 6, 15]
```

```
median = findMedianSortedArrays(arr1, arr2)
```

```
print(median)
```



The screenshot shows a code editor with two lines of Python code: `mid2 = mid1 - 1` and `median = (merged[mid1] + merged[mid2]) / 2`. Below the code, there is an 'input' field with the value '6'. The output area shows the number '6' and a message: '...Program finished with exit code 0 Press ENTER to exit console.'

Problem Statement: Given two sorted arrays of size m and n respectively, you are tasked with finding the element that would be at the k th position of the final sorted array.

```
def find_kth_element(array1, array2, k):
```

```
    m, n = len(array1), len(array2)
```

```
    i, j = 0, 0
```

```
    count = 0
```

```
    while i < m and j < n:
```

```
        if array1[i] <= array2[j]:
```

```
            current_element = array1[i]
```

```
            i += 1
```

else:

 current_element = array2[j]

 j += 1

count += 1

if count == k:

 return current_element

while i < m:

 count += 1

 if count == k:

 return array1[i]

 i += 1

while j < n:

 count += 1

 if count == k:

 return array2[j]

 j += 1

return "Error: k exceeds the total number of elements."

array1 = [2, 3, 6, 7, 9]

array2 = [1, 4, 8, 10]

k = 5

result = find_kth_element(array1, array2, k)

print(result)

```
36 print(result)
37
```

input

6

...Program finished with exit code 0
Press ENTER to exit console.