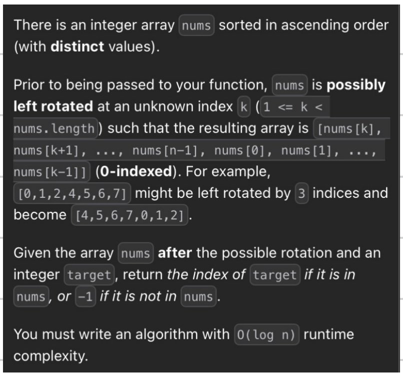
September 17, 2025

DOUBT SESSION

Ques 1. Search in Rotated Sorted Array

https://leetcode.com/problems/search-in-rotated-sorted-array/description/



There is an integer array `nums` sorted in ascending order (with **distinct** values).

Prior to being passed to your function, `nums` is **possibly left rotated** at an unknown index `k` (`1 <= k < nums.length`) such that the resulting array is `[nums[k], nums[k+1], ..., nums[n-1], nums[0], nums[1], ..., nums[k-1]]` (**0-indexed**). For example, `[0,1,2,4,5,6,7]` might be left rotated by `3` indices and become `[4,5,6,7,0,1,2]`.

Given the array `nums` **after** the possible rotation and an integer `target`, return *the index of* `target` *if it is in* `nums`*, or* `-1` *if it is not in* `nums`.

You must write an algorithm with `O(log n)` runtime complexity.

Example :

[1 2 3 4 5]

↓ Rotate

[2 3 4 5 1]

↓

0 1 2 3 4

[3 4 5 1 2]

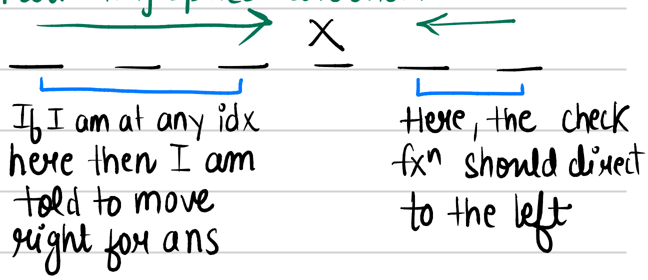
If target = 5 in the final rotated array then we will return the index of 5. [Ans = 2]

The Brute Force solution can be easily solved in linear time, O(N).

Approach for optimised solution using Binary Search :

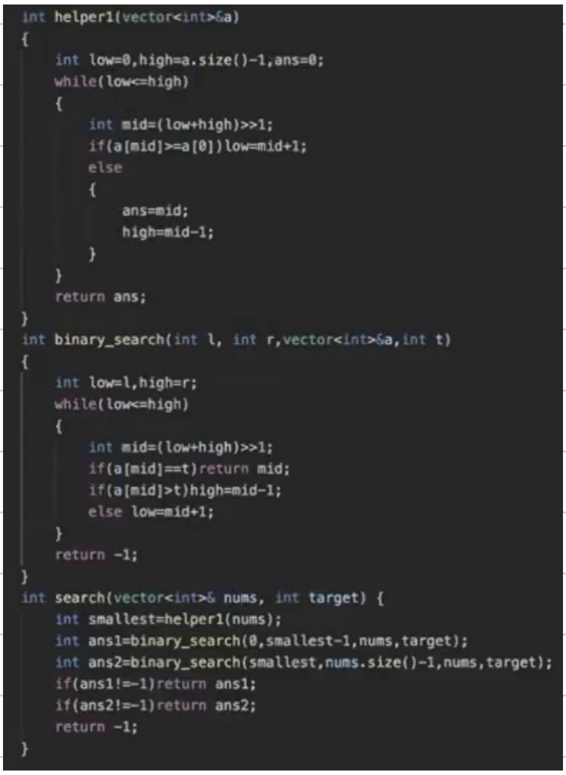
Given a X, we have to design a check function such that :

Partioning space around x.


X
If I am at any idx here then I am told to move right for ans
Here, the check fx^n should direct to the left

1. Find index of smallest element (i)
2. [i → n] : increasing array
3. [0 → i-1] : increasing

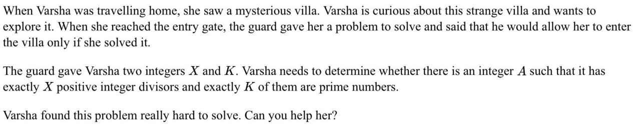
individual B·S



```
int helper1(vector<int>&a)
{
    int low=0,high=a.size()-1,ans=0;
    while(low<=high)
    {
        int mid=(low+high)>>1;
        if(a[mid]>=a[0])low=mid+1;
        else
        {
            ans=mid;
            high=mid-1;
        }
    }
    return ans;
}
int binary_search(int l, int r,vector<int>&a,int t)
{
    int low=l,high=r;
    while(low<=high)
    {
        int mid=(low+high)>>1;
        if(a[mid]==t)return mid;
        if(a[mid]>t)high=mid-1;
        else low=mid+1;
    }
    return -1;
}
int search(vector<int>& nums, int target) {
    int smallest=helper1(nums);
    int ans1=binary_search(0,smallest-1,nums,target);
    int ans2=binary_search(smallest,nums.size()-1,nums,target);
    if(ans1!=-1)return ans1;
    if(ans2!=-1)return ans2;
    return -1;
}
```

Ques 2. Strange Number

https://www.codechef.com/APRIL20B/problems/STRNO



When Varsha was travelling home, she saw a mysterious villa. Varsha is curious about this strange villa and wants to explore it. When she reached the entry gate, the guard gave her a problem to solve and said that he would allow her to enter the villa only if she solved it.

The guard gave Varsha two integers X and K. Varsha needs to determine whether there is an integer A such that it has exactly X positive integer divisors and exactly K of them are prime numbers.

Varsha found this problem really hard to solve. Can you help her?

Suppose we have a number, i.e,

$P_1^{\alpha_1} P_2^{\alpha_2} P_3^{\alpha_3} P_4^{\alpha_4} = A$

We know that,

$(\alpha_1+1)(\alpha_2+1)(\alpha_3+1)(\alpha_4+1) = X$

If we somehow make the above four different values and if K=4 then our answer will be yes otherwise No (like for K = 3, 5).

→ Suppose $x = 20$

So, how can we write 20 :

$2*10 = X$ — ①

$2*5*2 = X$ — ②

X can be written in 2 diff ways so can we say that in ①,

$X = 2*10$

$= (1+1)*(9+1)$ — ③

So if we have 2 prime members $P_1^{\alpha_1} P_2^{\alpha_2} = A$ and I have such values of α_1, α_2 such that $(\alpha_1+1)*(\alpha_2+1)=X$

Comparing this with ③ we can say that there will some prime number whose value will be 1, 9 which will be equal to some A

$A = P_1^1 P_2^9$

Now,

→ $X = 20$

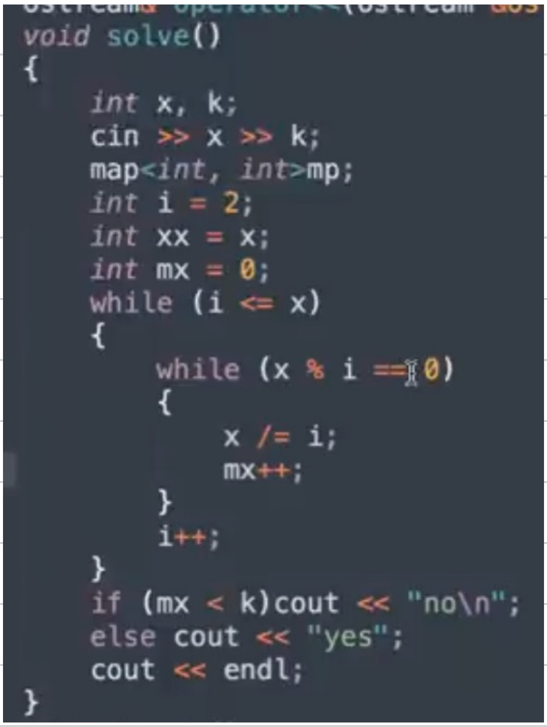
$= 2*2*5$ (Most Broken)

$= (1+1)*(1+1)*(4+1)$

$A = P_1^1 \; P_2^1 \; P_3^4$

If $x=20$, $k=3$, then answer = Yes.

→ If $K=2$, then we won't have to look for most broken form but simply can use $2*10$ or $4*5$.



```
void solve()
{
        int x, k;
        cin >> x >> k;
        map<int, int>mp;
        int i = 2;
        int xx = x;
        int mx = 0;
        while (i <= x)
        {
                while (x % i == 0)
                {
                        x /= i;
                        mx++;
                }
                i++;
        }
        if (mx < k)cout << "no\n";
        else cout << "yes";
        cout << endl;
}
```