Code Spectrum

1. Monotonic Function

```cpp
#include <bits/stdc++.h>
using namespace std;
int main() {
    int x;
    cin>>x;
    int f = pow(x,3)+2*pow(x,2)+3*x+5;
    cout<<f<<endl;
    return 0;
}
```

2. Seh Lenge Thoda

```cpp
#include <bits/stdc++.h>
using namespace std;

int main()
{
  int n;
    cin>>n;
    vector<pair<int,int>> arr(n);
    for(int i=0;i<n;i++) cin>>arr[i].first;
```

```
    for(int i=0;i<n;i++) cin>>arr[i].second;

    int sum=0;
    for(int i=0;i<n;i++)
    {
      sum+=(arr[i].first*arr[i].second);
    }
   cout<<sum<<endl;
}
```

3. Basant - The festival of roses

```
#include <bits/stdc++.h>
using namespace std;
int main() {
    int n,x;
    cin>>n>>x;
    vector<pair<int,string>> freshers(n);
    for(int i=0;i<n;i++){
        cin>>freshers[i].second>>freshers[i].first;
    }
    sort(freshers.begin(),freshers.end());
    reverse(freshers.begin(),freshers.end());
```

```
    cout<<freshers[x].second<<endl;
    return 0;
}
```

Alternate Approach:

```
#include <bits/stdc++.h>
using namespace std;
int main() {
    int n,x;
    cin>>n>>x;
    vector<pair<int,string>> freshers(n);
    for(int i=0;i<n;i++){
        cin>>freshers[i].second>>freshers[i].first;
    }
    sort(freshers.begin(),freshers.end());
    cout<<freshers[n-x].second<<endl;
    return 0;
}
```

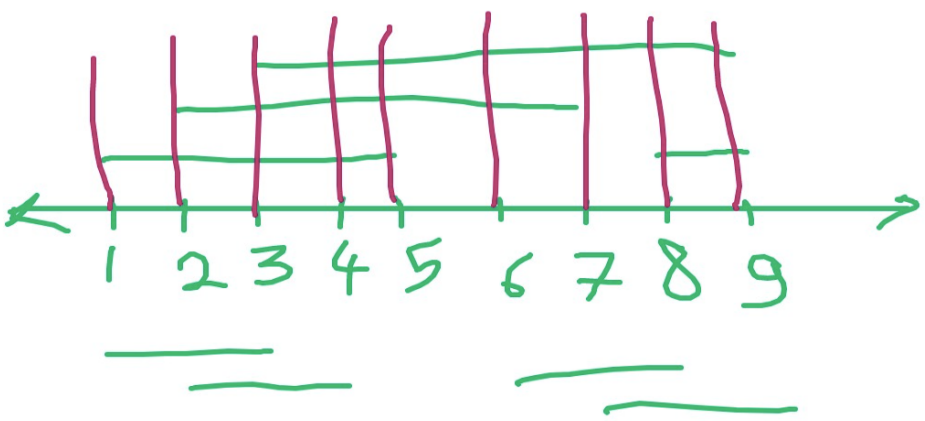
4. Despo Raju

```
#include <bits/stdc++.h>
using namespace std;
int main() {
    int q;
    cin>>q;
    while(q--){
        int n;
```

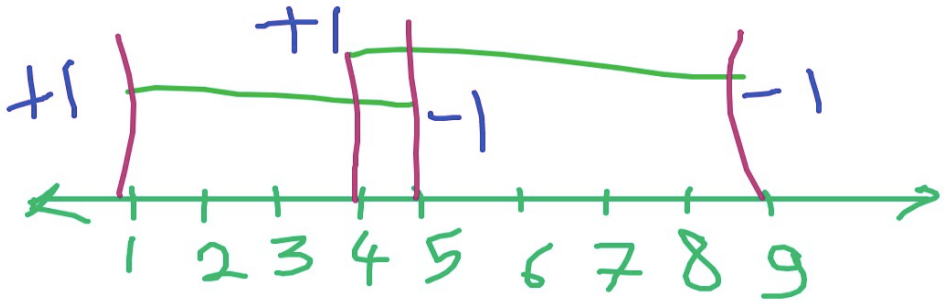
```
        cin>>n;
        string s;
        cin>>s;
        long long int swaps=0;
        int countOne=0;
        for(int i=0;i<n;i++){
            if(s[i]=='1'){
                countOne++;
            }
            else{
                swaps+=countOne;
            }
        }
        cout<<swaps<<endl;
    }
    return 0;
}
```

5. Lalit in Trouble

We need to find maximum number of overlapping classes at any instant.



We can consider the overlapping classes at the start and end-times. No need to consider all the points on number line.



First, take a variable for counting overlapping segments, cnt=0

When u find a start time of any class, just increase cnt by 1.
When u find a end time of any class, just decrease cnt by 1.

And when calculating cnt, also store the maximum number of overlapping segments as the answer.

```cpp
#include <bits/stdc++.h>
using namespace std;
int main() {
    int n;
    cin>>n;
    int s,f;
    vector<pair<int,int> > vec;
    for(int i=0; i<n; i++)
    {
        cin>>s>>f;
        vec.push_back({s,+1});
        vec.push_back({f,-1});
    }
    sort(vec.begin(), vec.end());
    int cnt=0; // number of overlapping classes/intervals
    int ans=0; // max. no. of overlapping classes/intervals
    int sz=vec.size();

    for(int i=0; i<sz; i++)
    {
        cnt=cnt+vec[i].second;
        ans=max(ans,cnt);
    }
    cout << ans-1 <<'\n';
    return 0;
}
```

6. Raju Bhai and Group Photo

Naive Idea (Brute Force Approach):

Go through all possible subarrays and if sum is divisble by x, update ans as maximum size of subarray.

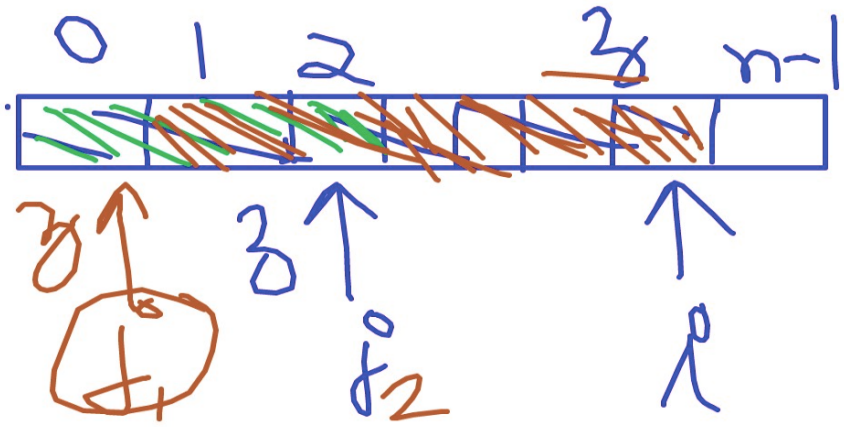
```cpp
#include <bits/stdc++.h>
using namespace std;
int main() {
    int n,x;
    cin>>x>>n;

    vector<int> vec(n);
    for(int i=0; i<n; i++)
    {
        cin>>vec[i];
    }

    int ans=0;

    for(int l=0; l<n; l++)
    {
        for(int r=l; r<n; r++)
        {
```

```
            long long sum=0;
            for(int i=l; i<=r; i++)
            {
                sum+=vec[i];
            }
            if(sum % x == 0)
            {
                ans=max(ans, r-l+1);
            }
        }
    }

    cout<<ans;
    return 0;
}
```

Time Complexity: $O(N^3)$

In general, you can perform 10^7 - 10^8 operations in 1 second.
Here, in worst case, N = 10^5 => $N^3 = 10^{15}$. **So, it gives TLE.**

A fast or optimised approach:
For every prefix, store the prefix sum mod x.
This prefix sum will always lie between [0, x-1].



1. If for any prefix ending at index i, if there exists any prefix ending at j (j<i) and pref[i] == prefix[j], then the subarray from [j+1, i] should be divisible by x.

2. Suppose there are 2 or more than 2 such possible values of j, satisfying the above criteria, then I need to consider only the smallest value of j.

```cpp
#include <bits/stdc++.h>

using namespace std;


int main() {
    int n,x;
    cin>>x>>n;

    vector<int> vec(n);
    for(int i=0; i<n; i++)
```

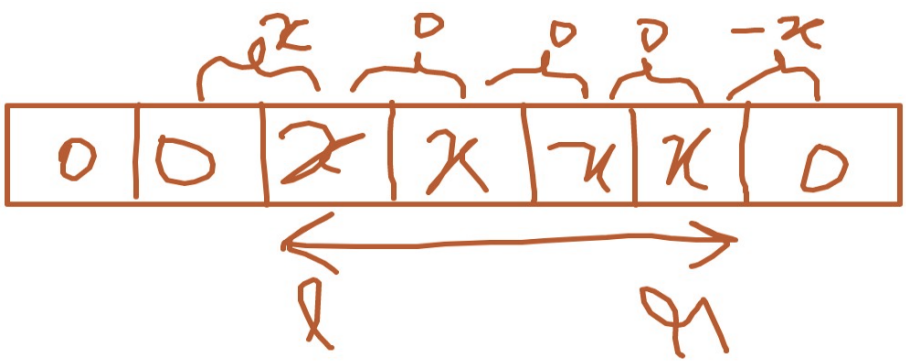
```
    {
        cin>>vec[i];
    }

    vector<int> low(x);
    // low[y] = The smallest value of j,
    // such that sum of prefix ending at j mod x = y

    for(int i=0; i<x; i++)
    {
     low[i]=-1; // -1 indicates no such prefix is available
    }

    int sum=0;
    int ans=0;
    for(int i=0; i<n; i++)
    {
      //        (a+b)%x = ((a%x) + (b%x))%x
        sum=((sum%x)+(vec[i]%x))%x;
        if(low[sum]!=-1)
        {
            int j=low[sum];
            ans=max(ans, i-j);
        }
        else
        {
            low[sum]=i;
        }
        if(sum%x==0)
```

```
        {
            ans=max(ans, i+1);
        }
    }

    cout<<ans;

    return 0;
}
```

Time Complexity: O(n) + O(x) = O(max(n,x))

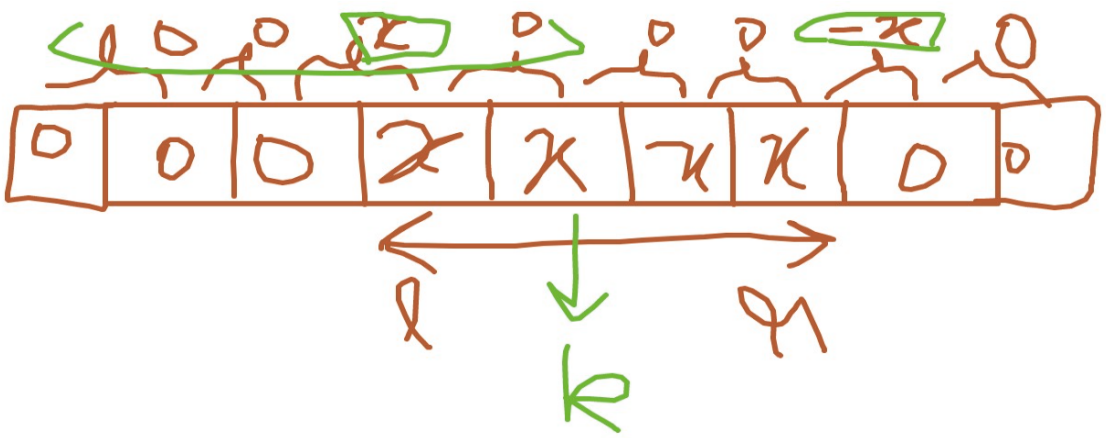
Follow Up: What if $x<=10^9$?
The problem is you can't create such a large array.
(Use map)

7. The Endgame



If we build a **difference array** diff, (an array containing difference of consecutive elements),
If we need to increase all elements of the given array in range [L, R], then in difference array, we need to make only 2 changes:
1. Increase diff[L] by x
2. Decrease diff[R+1] by x

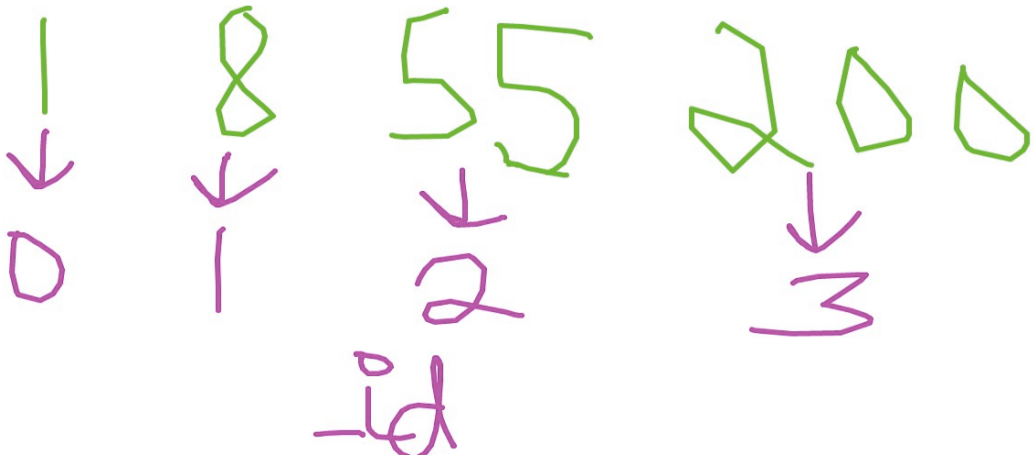
Now, each of the p operations can be performed in O(1) .
For all p operations, time complexity is: O(p)



To find arr[k], just find the prefix sum of the difference array upto k^{th} index.

Now, the problem is, we can't create a difference array of size 10^9 (Because, you can create integer arrays of size 10^6-10^7)

- Use map and give an id to all the unique elements
(Since, there are <= 10^3 distinct elements)



- See Setter's code in Editorial

Follow Up:
Suppose, there was no such constraint on distinct values of l and r. Then, how would you approach ?
- You use a map and use lower_bound()
See from here:
1. lower_bound() in vector:
https://www.geeksforgeeks.org/upper_bound-and-lower_bound-for-vector-in-cpp-stl/
2. lower_bound() in map:

https://www.geeksforgeeks.org/map-lower_bound-function-in-c-stl/

- See tester's code in Editorial