Codebook

1. Fast Input-Output:

#define FIO                           \

    ios\_base::sync\_with\_stdio(false); \

    cin.tie(NULL);                    \

    cout.tie(NULL);

1. Extra:

typedef long long int ll;

#define all(v) v.begin(), v.end()

const int mod = 1e9 + 7;

1. Priority Queue:

priority\_queue<int> pr;     //max\_priority\_queue priority\_queue<int, vector<int>, greater<int>> pr; // min\_priority\_queue priority\_queue<Student, vector<Student>, cmp> pq; // custom Class

***Compare Class for priority queue:***

class Student { public: string name; int roll; int marks; Student(string name, int roll, int marks){ this->name = name; this->roll = roll; this->marks = marks; } };

class cmp

{

public:

    bool operator()(Student a, Student b)

    {

        if (a.marks == b.marks) return a.roll > b.roll;

        else    return a.marks < b.marks;

    }

};

1. ***Custom\_Hash for unordered\_map***:

struct custom\_hash {

    static uint64\_t splitmix64(uint64\_t x) {

        x += 0x9e3779b97f4a7c15;

        x = (x ^ (x >> 30)) \* 0xbf58476d1ce4e5b9;

        x = (x ^ (x >> 27)) \* 0x94d049bb133111eb;

        return x ^ (x >> 31);

    }

    size\_t operator()(uint64\_t x) const {

        static const uint64\_t FIXED\_RANDOM = chrono::steady\_clock::now().time\_since\_epoch().count();

        return splitmix64(x + FIXED\_RANDOM);

    }

};

5) Binary Search:

* ***closest from left: (1 indexed)***

ll x;   cin>>x; auto it=upper\_bound(all(v),x); cout<<it-v.begin()<<nl;

* ***closest from right: (1 indexed)***

ll x;   cin>>x; auto it=lower\_bound(all(v),x); cout<<it-v.begin()+1<<nl;

* ***Binary Search on Real Number***:

ld l=0; // good ld r=1e18; //bad

    for(int i=0;i<=100;i++)

    {

        ld mid=(l+r)/2;

        if(good(mid))   l=mid;

        else r=mid;

    }

    cout<<fixed<<setprecision(6)<<l<<nl;

6) Two Pointers:

***General Code:***

ll L=0;

for(ll R = 0; R < n; R++)

{

    add(v[R]); // this function will recalculate after updating right pointer

    while( !good() )

    {

        remove(v[L]); // this function will recalculate after updating left pointer

        L++;

    }

    ans+=R-L+1; // what is needed do here

}

***If we can add element and recalculate the function then we also can remove and recalculate the function using following technique:***

struct stack

{

    vector<ll>s,smin={LLONG\_MAX},smax={LLONG\_MIN};

    void push(ll x)

    {

        s.pb(x);

        smin.pb(::min(smin.back(),x));

        smax.pb(::max(smax.back(),x));

    }

    ll pop()

    {

        ll res=s.back();

        s.ppb();

        smin.ppb();

        smax.ppb();

        return res;

    }

    bool empty()

    {

        return s.empty();

    }

    ll min()

    {

        return smin.back();

    }

    ll max()

    {

        return smax.back();

    }

};

::stack s1,s2;

7) Bit Masking: (max array size 20)

for (int bits = 0; bits < (1 << n); bits++)

    {

        ll sum\_of\_subset = 0;

        for (int i = 0; i < n; i++)

        {

            if (bits & (1 << i)) sum\_of\_subset += v[i];

            else sum\_of\_subset -= v[i];

        }

    }

8) Next Permutation:

ll n;   cin >> n; string s;

    while (true){

        s += "47";

        sort(s.begin(), s.end());

        if (stoll(s) >= n)

        {

            cout << stoll(s) << nl;

            return 0;

        }

        while (next\_permutation(s.begin(), s.end()))

        {

            if (stoll(s) >= n)

            {

                cout << stoll(s) << nl;

                return 0;

            }

        }

    }

9) N er ith bit set ki na:

ll n,i; cin >> n >> i;

    if (n & (1 << i))   cout << "true" << nl;

    else    cout << "false" << nl;

10) Decimal to binary:

long long int x;

    cin >> x;

    bool flag = false;

    // for int it start from 30

    // for long long int it start from 62 then 1 should be 1LL to avoid overflow

    // or to avoid overflow we can write ((x >> i) & 1) equivalent of ( x & (1LL << i))

    for (int i = 62; i >= 0; i--)

    {

        if (((x >> i) & 1) != 0) // be careful about brackets

        {

            cout << "1";

            flag = true;

        }

        else if (flag)  cout << "0";

    }

11) flipping all bits of a number:

unsigned int x; cin >> x;

        x = ~x;

        cout << x << nl;

12) reversing bits of a number using bitset:

unsigned int n; cin >> n;

       bitset<32> x(n);

       bitset<32> rev(n);

       for (int i = 0; i < 32; i++)

       {

            rev[31 - i] = x[i];

       }

       unsigned int ans = rev.to\_ulong();

       cout << ans << nl;

17) Modular of power:

ll pow(ll a,ll b){

    ll result=1;

    while(b>0)

    {

        if(b&1) result=(result\*a)%mod;

        a=(a\*a)%mod;

        b/=2;

    }

    return result;

}

14) Divide & Conquer: (Merge Sort)

void merge(ll l,ll mid,ll r)

{

    ll n1=mid-l+1,n2=r-mid;

    vector<ll>a(n1),b(n2);

    for(ll i=0;i<n1;i++)    a[i]=v[l+i];

    for(ll i=0;i<n2;i++)    b[i]=v[mid+1+i];

//here you will have two sorted array (left & right part ). And here you can do your calculations.

    ll i=0,j=0,k=l;

    while(i<n1 && j<n2)

    {

        if(a[i]<b[j])

        {

            v[k]=a[i];

            k++,i++;

        }

        else

        {

            v[k]=b[j];

            k++,j++;

        }

    }

    while(i<n1)

    {

        v[k]=a[i];

        k++,i++;

    }

    while(j<n2)

    {

        v[k]=b[j];

        k++,j++;

    }

}

void mergesort(ll l,ll r)

{

    if(l<r)

    {

        ll mid=(l+r)/2;

        mergesort(l,mid);

        mergesort(mid+1,r);

        merge(l, mid, r);

    }

}

void solve()

{

    ll n;   cin>>n;

    v.resize(n);

    for (ll i=0;i<n;i++)  cin>>v[i];

    mergesort(0,n-1);

    for(ll i=0;i<n;i++)  cout<<v[i]<<" ";

    cout<<nl;

}

15. Compressing integer(range):

vector<pair<ll,ll>>v(n);

    set<ll>st;

    for(ll i=0;i<n;i++)

    {

        cin>>v[i].fi>>v[i].sec;

        st.insert(v[i].fi),st.insert(v[i].fi+1);

        st.insert(v[i].sec),st.insert(v[i].sec+1);

    }

    vector<ll>compress;

    for(ll x:st)    compress.pb(x);

    unordered\_map<ll,ll,custom\_hash>mp;

    for(ll i=0;i<compress.size();i++)   mp[compress[i]]=i;

    for(ll i=0;i<n;i++) v[i].fi=mp[v[i].fi],v[i].sec=mp[v[i].sec];

16. Modular formula’s:

* ab^c % m = ab^c % n % m [n=m-1]
* (a + b) % m = ( ( a % m ) + ( b % m ) ) % m
* (a \* b) % m = ( ( a % m ) \* ( b % m ) ) %m
* (a – b) % m = ( ( a % m ) – ( b % m ) + m ) % m
* (a / b) % m = ( ( a % m ) \* ( b-1 % m ) ) % m
* B-1 % m = power ( b , m-2)

17. **2D prefix sum**:

ll n,m,q; cin>>n>>m>>q;

    vector<vector<ll>>v(n,vector<ll>(m));

    for(ll i=0;i<n;i++)

        for(ll j=0;j<m;j++) cin>>v[i][j];

    vector<vector<ll>>prefix(n,vector<ll>(m));

    for(ll i=0;i<n;i++)

    {

        for(ll j=0;j<m;j++)

        {

            prefix[i][j]=v[i][j];

            if(i!=0)    prefix[i][j]+=prefix[i-1][j];

            if(j!=0)    prefix[i][j]+=prefix[i][j-1];

            if(i!=0 && j!=0)    prefix[i][j]-=prefix[i-1][j-1];

        }

    }

    while(q--)

    {

        ll a,b,c,d; cin>>a>>b>>c>>d;

        a--,b--,c--,d--;

        ll ans=prefix[c][d];

        if(a!=0 && b!=0)    ans+=prefix[a-1][b-1];

        if(a!=0)    ans-=prefix[a-1][d];

        if(b!=0)    ans-=prefix[c][b-1];

        cout<<ans<<nl;

    }

18. Pascal’s triangle:

// max possible size 66 //tried and tested

ll C[66][66];

void pascal\_triangle()

{

    C[0][0]=1;

    for(ll i=1;i<=66;i++)

    {

        C[i][0]=C[i][i]=1;

        for(ll j=1;j<i;j++)

        {

            C[i][j]=C[i-1][j-1]+C[i-1][j];

        }

    }

}