Team: Zero^Infinity(DUET)

***** Format**

```
#include<bits/stdc++.h>
#define Fast ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
#define Freed freopen ("0in.txt","r",stdin);
#define Fout freopen ("0out.txt","w",stdout);
#define ll long long int
#define pb push_back
#define pi acos(-1.0)
#define inf 1e18
#define Mod 1000000007
#define limit 1000008
using namespace std;
void Please_AC(ll tt)
{
   ll i,j,n,m,k,q;
   cin >> n;
   ll d[n+5];
   return ;
}
int  main()
{
   Fast
//    Freed
//    Fout
   ll t,tt=1;
   cin >> tt;
   for(t=1; t<=tt; t++)
      Please_AC(t);
}
```

*****/PBDS or Order set/multiset/map any operation O(logn)**

```
///insert,erase,size,order_of_key,find_by_order
#include <ext/pb_ds/assoc_container.hpp>
using namespace __gnu_pbds;
using namespace std;
typedef tree<int, null_type, less<int>, rb_tree_tag,
          tree_order_statistics_node_update>
         new_data_type;
// Deleting 2 from the set if it exists
   if (o_set.find(2) != o_set.end())
      o_set.erase(o_set.find(2));
   // Finding the second smallest element in the set
   cout << *(o_set.find_by_order(1)) << endl;
   // number of elements strictly less than k=4
   cout << o_set.order_of_key(4) << endl;
```

*****Bigmod O(logp)**

```
ll bigmod(ll b,ll p)
{
   ll ans=1;
   while(p)
   {
      if(p&1)     ans = (ans*b)%Mod;
      b = (b*b)%Mod;
      p = p/2;
   }
   return ans;
}
```

*****///seive O(nloglogn)**

```
bool vis[limit];
vector<int>prime;
void seive()
{
   vis[0]=vis[1]=1;
   for(int i=4; i<limit; i+=2) vis[i] = 1;
   for(int i=3; i*i<limit; i+=2)
   {
      if(vis[i]) continue;
      for(int j=i*i; j<limit; j+=2*i)
      {
         vis[j] = 1;
      }
   }
   prime.pb(2);
   for(int j=3; j<limit; j+=2)
      if(vis[j]==0)
         prime.pb(j);
}
```

///prime factorization O(nlogn) (call seive())

```
   cin >> n ;
   vector <int> Div;
   for(i=0; prime[i]*prime[i]<=n; i++)
   {
      while(n%prime[i]==0)
      {
         Div.pb(prime[i]);
         n/=prime[i];
      }
   }
   if(n>1)
      Div.pb(n);
   for(i=0; i<Div.size(); i++) cout <<Div[i]<<" ";
```

*****Number of Divisor for 1 to n. O(n*logn)**

```
ll nod[limit];
void NOD(ll n)
{
   ll i,j;
   for(i=1; i<=n; i++)
   {
      for(j=i; j<=n; j+=i)
         nod[j]++;
   }
}
```

*****//Sum of Divisor for 1 to n. O(n*logn)**

```
ll sod[limit];
void SOD(ll n)
{
   ll i,j;
   for(i=1; i<=n; i++)
   {
      for(j=i; j<=n; j+=i)
         sod[j] += i;
   }
}
```

Team: Zero^Infinity(DUET)

*****Euler totient or Phi O(n*logn)**

```
bool vis[limit];
void seive()
{
   vis[0]=vis[1]=1;
   for(int i=4; i<limit; i+=2) vis[i] = 1;
   for(int i=3; i*i<limit; i+=2)
   {
      if(vis[i]) continue;
      for(int j=i*i; j<limit; j+=2*i)
      {
         vis[j] = 1;
      }
   }
}
```

*****///nPr O(r)**

```
ll nPr(ll n,ll r)
{
   ll ans=1;
   for(ll i=n,j=0; j<r; i--,j++)
      ans = (ans*i)%Mod;
   return ans;
}
```

///nCr O(n*r)

```
long long int ncr[2000][2000];
long long int nCr(long long int n, long long int r)
{
   if(n==r) return 1;
   if(r==1) return n;
   if(ncr[n][r]) return ncr[n][r];
   return ncr[n][r] = (nCr(n-1,r-1)+nCr(n-1,r))%Mod;
}
```

///nCr O(1) when Mod is prime

```
ll fact[limit],factorialNumInverse[limit],
                    naturalNumInverse[limit];
//Function to precompute inverse of numbers
void InverseofNumber()
{
   naturalNumInverse[0] = naturalNumInverse[1] = 1;
   for (int i = 2; i <limit; i++)
      naturalNumInverse[i] =
     naturalNumInverse[Mod % i] * (Mod - Mod / i) % Mod;
}
```

*****Function to precompute inverse of factorials**

```
void InverseofFactorial()
{
   factorialNumInverse[0] = factorialNumInverse[1] = 1;
   for (int i = 2; i <limit; i++)
      factorialNumInverse[i] =
(naturalNumInverse[i] * factorialNumInverse[i - 1]) % Mod;
}
```

*****Factorial**

```
void factorial()
{
   fact[0] = 1;
   for (int i = 1; i <limit; i++) {
      fact[i] = (fact[i - 1] * i) % Mod;
   }
}
```

*****// Function to return nCr % p in O(1) time**

```
ll nCrr(ll n, ll r)
{
   ll ans = ((fact[n]
            * factorialNumInverse[r])% Mod
            * factorialNumInverse[n - r])% Mod;
   return ans;
}
int  main()
{
   InverseofNumber();
   InverseofFactorial();
   factorial();
   ll n,r;
   cin >> n >> r;
   cout <<nCr(n,r)<<endl;
   cout <<nCrr(n,r)<<endl;
   return 0;
}
```

*****///Sparse table. build O(n*log2(n)), query O(1).**

```
ll n,d[limit];
ll stbl[limit][23];
void Sparse_build()
{
   for(ll i=0; i<n; i++)
      stbl[i][0] = d[i];

   for(ll j=1; j<22; j++)
   {
      for(ll i=0; i+(1<<j)-1 < n; i++)
      {
         stbl[i][j] = min(stbl[i][j-1], stbl[i+(1<<(j-1))][j-1]);
      }
   }
   return ;
}
ll Query(ll l, ll r)
{
   ll len = log2(r-l+1);
   return min(stbl[l][len], stbl[r-(1<<len)+1][len]);
}
```

*****0-1 knapsack O(n*W). W=Total weight, p=profit, w=weight.**

```
ll n,dp[1002][limit],w[1005],p[1005];
ll rec(ll pos,ll W)
{
   if(pos==n)  return 0;
   if(dp[pos][W]!=-1)  return dp[pos][W];
   if(W>=w[pos])
      dp[pos][W] = p[pos]+rec(pos+1,
                              W-w[pos]);
   ll temp = rec(pos+1, W);
   if(dp[pos][W] < temp)
      dp[pos][W] = temp;
   return dp[pos][W];
}
```

Team: Zero^Infinity(DUET)

*****Bellmanford O(m*n)**

```
struct edge{
   ll u,v,c;
}z;
ll n, m, dis[limit];
vector<edge>E;
void bellmanford(ll tt)
{
   ll i,j;
   cin >> n >> m;
   for(i=0; i<=n; i++)
      dis[i] = 1e18;
   for(i=0; i<m; i++)
   {
      cin >> z.u >> z.v >> z.c;
      z.c = z.c;
      E.pb(z);
   }
   bool negcycle = 0;
   dis[1] = 0;
   for(i=0; i<=n; i++)
   {
      for(edge e: E)
      {
         if(dis[e.u]+e.c < dis[e.v]){
            dis[e.v] =  e.c + dis[e.u];
            if(i==n)  negcycle=1;
         }
      }
   }
   if(negcycle)
      cout <<-1<<endl;
   else
      cout <<dis[n]<<endl;

   return ;
}
```

*****DSU on Tree**

```
ll cnt[maxn];
bool big[maxn];
void add(ll v, ll p, ll x)
{
   cnt[ col[v] ] += x;
   for(auto u: g[v])
   {
      if(u != p && !big[u])
         add(u, v, x)
   }
}
void del(ll v, ll p, ll x)
{
   cnt[ col[v] ] -= x;
   for(auto u: g[v])
   {
      if(u != p && !big[u])
         add(u, v, x)
   }
}
void dfs(ll v, ll p, bool keep)
{
   ll mx = -1, bigChild = -1;
   for(auto u : g[v])
   {
      if(u != p && sz[u] > mx)
      {
         mx = sz[u];
         bigChild = u;
      }
   }
   for(auto u : g[v])
   {
      if(u != p && u != bigChild)
         dfs(u, v, 0);  // run a dfs on small childs and
                              clear them from cnt
   }
   if(bigChild != -1)
   {
      dfs(bigChild, v, 1);
      big[bigChild] = 1;  // bigChild marked as big and
                                not cleared from cnt
   }
   add(v, p, 1);         //added to the ans subtree of v
   if(bigChild != -1)
      big[bigChild] = 0;

   if(keep == 0)
      del(v, p, 1);         //delete subtree of v
}
```

*****///DSU O(n)**

```
int pr[limit];
int Find(int u)
{
   if(pr[u]==u) return u;
   return pr[u] = Find(pr[u]);
}
void dsu(int u,int v)
{
   pr[Find(v)] = Find(u);
}
```

*****Hashing O(n)**

```
void cumforwardhashing(string s,ll base,ll mod,ll A[])
{
   ll i,n=s.size();
   A[0] = s[0]-'a'+1;
   for(i=1; i<n; i++)
   {
      A[i] = ((A[i-1]*base)+s[i]-'a'+1)%mod;
   }
}
void cumbackwordhashing(string s,ll base,ll mod,ll A[])
{
   ll i,n=s.size();
   A[n-1] = s[n-1]-'a'+1;
   for(i=n-2; i>=0; i--)
   {
      A[i] = ((A[i+1]*base)+s[i]-'a'+1)%mod;
   }
}
```

Team: Zero^Infinity(DUET)

*****///kmp pi table build O(|p|)**

```
ll pi_tab[limit];
void build(string p)
{
    ll now = -1;   pi_tab[0] = -1;
    ll szp = p.size();

    {
        while( now!=-1 && p[now+1]!=p[i])
        {
            now = pi_tab[now];
        }
        if(p[now+1]==p[i])
            now++;;
        pi_tab[i] = now;
    }
}
```

///kmp O(|s|)

```
ll kmp(string s,string p)
{
    ll ans=0,now = -1, sz = s.size(),szp=p.size()-1;
    for(ll i=0; i<sz; i++)
    {
        while( now!=-1 && p[now+1]!=s[i])
        {
            now = pi_tab[now];
        }
        if(p[now+1]==s[i])
            now++;
        if(now==szp-1)  ans++;
    }
    return ans;
}

void Please_AC(ll tt)
{
    ll i,j,n,m;
    string s,p;
    cin >> s >> p;
    m = s.size();
    build(p);
    p = p+'#';
    cout <<"Case "<<tt<<": "<< kmp(s,p) <<endl;
    return ;
}
```

*****Structure mapping**

```
struct pos
{
    int cx,cy,nx,ny;
};
bool operator< (pos a,pos b )
{            ///there are lots of fact for this return
    if(a.cx!=b.cx) return a.cx<b.cx;
    else if(a.cy!=b.cy) return a.cy<b.cy;
    else if(a.nx!=b.nx) return a.nx<b.nx;
    else return a.ny<b.ny;
}
```

*****///LCA on a tree build O(n*logn)**

```
vector<ll> g[limit];
ll height[limit],st[limit][20];
void st_build(ll u,ll p)
{
    for(ll v: g[u])
    {
        if(v==p)  continue;
        height[v] = height[u] + 1;
        st[v][0] = u;
        for(ll j=1; j<20; j++)
        {
            st[v][j] = st[st[v][j-1]][j-1];
        }
        st_build(v,u);
    }
}
```

///LCA query O(logn)

```
ll LCA(ll u, ll v)
{
    if(height[u]>height[v])
        swap(u,v);
    ll dis = height[v]-height[u];
    /// make height u,v same
    for(ll j=19; j>=0; j--)
    {
        if( dis&(1<<j) )
            v = st[v][j];
    }
    if(u==v) return u;
    for(ll j=19; j>=0; j--)
    {
        if(st[u][j]!=st[v][j])
        {
            u = st[u][j];
            v = st[v][j];
        }
    }
    return st[u][0];
}
```

*****///Maximum ST. Kruskal's Algorithm O(E logE)**

```
struct E
{
    ll u,v,w;
};
bool operator<(E a,E b)
{
    return a.w>b.w;
}
ll p[limit];
ll Find(ll u)           // this part is for disjoint set union ,
initially p[x] = x;
{
    if(p[u]==u) return u;
    return p[u] = Find(p[u]);
}
```

Team: Zero^Infinity(DUET)

```
void MST()
{
   ll node,edge,i,j;
   E z;              // z is structure type variable
   vector <E> vec;
   cin >> node >> edge;
   ll u,v,w;
   for(i=0; i<edge; i++)
   {
      cin >> u >> v >> w;
      z.u = u;
      z.v = v;
      z.w = w;
      vec.pb(z);
   }
   sort(vec.begin(),vec.end());
   for(i=0; i<limit; i++)           // set DSU, parent of i is i
      p[i] = i;

   vector<pair<ll,ll> > g[limit];

   ll ans = 0;
   for(i=0; i<edge; i++)
   {
      z = vec[i];
      u = z.u;
      v = z.v;
      w = z.w;
      if(Find(u)!=Find(v))
      {
         ans += w;          // ans is the maximum cost
         p[u] = p[v];      // parent of u is v
         g[u].pb(mp(v,w));
         g[v].pb(mp(u,w));      // g Maximum spanning
                                  tree only
      }
   }
   cout << ans <<endl;
}
```

***** Merge sort O(n*logn)**

```
vector<int> v;
int D[limit];
void Mergesort(int l,int r)
{
   if(l==r) return ;
   int i,j,mid=(l+r)/2;
   Mergesort(l,mid);
   Mergesort(1+mid,r);
   v.clear();
   for(i=l,j=mid+1; ; )
   {
      if(i>mid && j>r) break;
      if(i>mid)
      {
         v.pb(D[j]);
         j++;
      }
      else if(j>r)
      {
         v.pb(D[i]);
         i++;
      }
      else if(D[i]<=D[j])
      {
         v.pb(D[i]);
         i++;
      }
      else
      {
         v.pb(D[j]);
         j++;
      }
   }
   for(i=l,j=0; i<=r; i++,j++)
   {
      D[i] = v[j];
   }
}
```

*****Quick Sort**

```
class Sort
{
public:
   int Partition(int A[],int left,int right)
   {
      int i=left,j;
      for(j=left; j<=right; j++)
      {
         if(A[j]<=A[right])          //pivot =A[right]
         {
            swap(A[i],A[j]);
            i++;
         }
      }
      return max(i-1,left);
   }
   void QuickSort(int A[],int left,int right)
   {
      if(left==right) return ;
      int mid = Partition(A,left,right);   //cout <<mid<<endl;
      QuickSort(A,left,max(mid-1,left));
      QuickSort(A,min(mid+1,right),right);
   }
};
```

Team: Zero^Infinity(DUET)
*****/// Strongly Connected Component(SCC) O(Node+Edge)**

```
vector<ll>g[limit],tg[limit];
bool vis[limit];
ll st[limit],ft[limit];
ll tme;

void dfs(ll u)
{
    st[u] = tme++;
    vis[u] = 1;
    for(ll v:g[u])
    {
        if(vis[v]==0)
        {
            dfs(v);
        }
    }
    ft[u] = tme++;
}

void dfs2(ll u)
{
    //cout <<u<<"->";
    vis[u] = 1;
    for(ll v:tg[u])
    {
        if(vis[v]==0)
        {
            dfs2(v);
        }
    }
}

void SCC(ll t)
{
    ll i,j;
    ll n,e,u,v;
    cin >> n >> e ;

    //clear
    for(i=0; i<n+5; i++)
    {
        g[i].clear();
        tg[i].clear();
        vis[i] = 0;
        st[i] = ft[i] = 0;
    }
    tme = 1;

    for(i=0; i<e; i++)
    {
        cin >> u >> v;
        g[u].pb(v);
        tg[v].pb(u);
    }
    for(i=1; i<=n; i++)
    {
        if(vis[i]==0)
        {
            dfs(i);
        }
    }
    vector<pair<ll,ll> > seq;

    for(i=1; i<=n; i++)
    {
        seq.pb(mp(ft[i],i));
    }
    sort(seq.begin(),seq.end());

    ll ct=0;
    memset(vis,0,sizeof(vis));
    for(i=n-1; i>=0; i--)
    {
        u = seq[i].second;
        if(vis[u]==0)
        {
            dfs2(u);
            ct++;
            cout <<endl;
        }
    }
    cout <<"Total number of SCC: "<<ct<<endl;
    return ;
}
```

*****Query Range**
*****Heavy Light Trick(HLT)**

```
const ll block = 350 ;
ll d[limit],cum[limit],PS[350][limit];

void solve(ll t)
{
    ll i,j,n,m,k,q;
    string s;
    cin >> n >> q;
    //block = sqrt(n);

    cum[0] =0;
    for(i=1; i<=n; i++)
    {
        cin >> d[i];
        cum[i] = cum[i-1]+d[i];
    }

    memset(PS, 0, sizeof(PS));
    for(k=1; k<block; k++)
    {
        for(ll pos = n-k; pos<=n+1; pos++)
            PS[k][pos] = 0;
        for(ll pos=n-k+1; pos>0; pos--)
        {
            PS[k][pos] = (cum[pos+k-1]-cum[pos-1]) - PS[k]
[pos+k];
            //cout <<k<<" "<<pos<<" "<<PS[k][pos]<<endl;
        }
    }
    for(i=0; i<q; i++)
    {
        ll l,r;
        cin >> l >> r >> k;
        ll ans = 0;
        if(k>=block)
        {
            ll sign = 1;
            for(j=l; j<r; j+=k)
            {
                ans += (cum[j+k-1]-cum[j-1])*sign;
                sign = sign*-1;
            }
        }
        else
        {
            ans = PS[k][l];   //cout <<ans<<" akhane\n";
            if(((r-l+1)/k)&1)
                ans += PS[k][r+1];
            else
                ans -= PS[k][r+1];
        }
        cout << ans <<endl;
    }
    return ;
}
```

*****MO’s Algorithm**

```
ll cnt[limit];
ll frequ[limit];
ll mxfrequ;
void add(ll x)
{
    cnt[frequ[x]]--;
    frequ[x]++;
    if(mxfrequ < frequ[x]) mxfrequ = frequ[x];
    cnt[frequ[x]]++;
}
void sub(ll x)
{
    cnt[frequ[x]]--;
    if(mxfrequ==frequ[x] && cnt[frequ[x]]==0) mxfrequ--;
    frequ[x]--;
    cnt[frequ[x]]++;
}
/* splite all the query in sqrt(n) block according to left side
    and in each block all data are sorted according to right
side
*/
///MO's algorithm O(n*sqrt(n))
ll block;
struct st
{
    ll l,r,idx;
    bool operator<(const st& a) const
    {
        if(l/block != a.l/block) return l/block < a.l/block;
        return r < a.r;
    }
};
///MO's
void MO(ll t)
{
    ll n,q,i;
    cin >> n >> q ;
    ll d[n+5]; d[0] = 0;
    vector<ll> out(q+5);

    for(i=1; i<=n; i++)
        cin >> d[i];

    ///MO's
    block = sqrt(n);
    st query[q+5];
    for(i=0; i<q; i++)
    {
        cin >> query[i].l >> query[i].r;
        query[i].idx = i;
    }
    sort(query,query+q);
    ///MO's

    ll left = 1,right = 0;
    for(i=0; i<q; i++)
    {         //cout <<i<<" "<<left<<" "<<right<<"
"<<query[i].l<<" "<<query[i].r<<endl;
        while(right < query[i].r)
        {
            right++;
            add(d[right]);
        }
        while( right > query[i].r)
        {
            sub(d[right]);
            right--;
        }
        while(left < query[i].l)
        {
            sub(d[left]);
            left++;
        }
        while(left > query[i].l)
        {
            left--;
            add(d[left]);
        }
        out[query[i].idx] = 1 + max(0, mxfrequ - (query[i].r -
query[i].l + 2 - mxfrequ));
        //cout <<left<<" "<<right<<" "<<mxfrequ<<"
"<<out[query[i].idx]<<endl;
    }
    for(i=0; i<q; i++)
        cout << out[i] <<endl;
    return ;
}
```

Team: Zero^Infinity(DUET)

*****///Sparse table. build O(n*log2(n)), query O(1).**

```
ll n,d[limit];
ll stbl[limit][23];
void Sparse_build()
{
    for(ll i=0; i<n; i++)
        stbl[i][0] = d[i];

    for(ll j=1; j<22; j++)
    {
        for(ll i=0; i+(1<<j)-1 < n; i++)
        {
            stbl[i][j] = min(stbl[i][j-1], stbl[i+(1<<(j-1))][j-1]);
        }
    }
    return ;
}
ll Query(ll l, ll r)
{
    ll len = log2(r-l+1);
    return min(stbl[l][len], stbl[r-(1<<len)+1][len]);
}
```

*****///Segment tree() O(nlogn)**

```
void build(ll at, ll L,ll R)
{
    sum[at] = 0;
    if(L==R){
        sum[at] = d[L];
        return;
    }   ll mid=(L+R)/2;
    build(at*2,L,mid);
    build(at*2+1,mid+1,R);
    sum[at] = sum[at*2]^sum[at*2+1];
}
///segupdate O(logn)
void update(ll at,ll L,ll R,ll pos,ll x)
{
    if(L==R){
        sum[at] = sum[at]^x;
        d[pos] = x;
        return ;
    }
    ll mid = (L+R)/2;
    if(pos<=mid)
        update(at*2,L,mid,pos,x);
    else
        update(at*2+1,mid+1,R,pos,x);
    sum[at] = sum[at*2]^sum[at*2+1];
}
///segquery O(logn) from l to r
ll Query(ll at,ll L,ll R,ll l,ll r)
{
    if(r<L || R<l)  return 0LL;
    if(l<=L && R<=r)   return sum[at];
    ll mid = (L+R)/2;
    ll x = Query(at*2,L,mid,l,r);
    ll y = Query(at*2+1,mid+1,R,l,r);
    return x^y;
}
void solve(ll t)
{
    ll i,j,n,q,m,k;
    cin >> n >> q;
    for(i=1; i<=n; i++)
        cin >> d[i];
    build(1,1,n);
    for(i=0; i<q; i++)
    {
        ll type,pos,key,from,to;
        cin >> type;
        if(type==1)
        {
            cin >> pos >> key;
            update(1,1,n,pos,key);
        }
        else
        {
            cin >> from >> to;
            cout <<Query(1,1,n,from,to)<<endl;
        }
    }
    return ;
}
```

*****///knapsack with meet in the middle O(n*2^n/2) ;**

```
ll D[limit],W;
void middle(ll pos,ll n,ll sum,vector<ll> &A)
{
    if(pos>n)
    {
        if(sum<=W) A.pb(sum);
        return ;
    }
    middle(pos+1,n,sum,A);
    middle(pos+1,n,sum+D[pos],A);
}

void solve(int t)
{
    ll n,i,j;
    scanf("%lld %lld",&n,&W);
    for(ll i=1; i<=n; i++)
    {
        scanf("%lld",&D[i]);
    }
    sort(D,D+n+1);
    vector<ll>A,B;
    A.pb(0);
    middle(1,n/2,0,A);
    middle(n/2+1,n,0,B);
    sort(A.begin(),A.end());
    ll mx=0,sz=A.size();
    for(i=B.size()-1; i>=0; i--)
    {
        n = (W-B[i]);
        ll l=0,h=sz-1,m;
        while(l<=h)
        {
            m = (l+h)/2;
```

```
             if(n>=A[m])
            {
                if((B[i]+A[m])<=W && mx<(B[i]+A[m]))
                    mx = B[i]+A[m];
                l = m+1;
            }
            else
            {
                h = m-1;
            }
        }
    }
    printf("%lld\n",mx);
    return ;
}
///Naeem Closed
```