Team notebook

September 29, 2017

Contents

1 articulationpoints

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

using namespace std;

vector<int> g[10000];

int  visit[10006];
int    low[10006];
int   dist[10006];
int parent[10006];
int    ap[10006];

int n,m,level;

//articulation points for a undirected graph
void dfs(int u){

    level++;
    visit[u]=1;
    dist[u]= low[u]=level;

    int child=0;
```

```
    for(int i=0;i < g[u].size();i++){
        int v =g[u][i];
        child++;

        if(visit[v]==0){

            parent[v]=u;
            dfs(v);
            low[u] = min(low[u], low[v]);

            if(parent[u]==-1 && child >1)
                 ap[u]=1;

            if(parent[u]!=-1 && low[v]>=dist[u])
                   ap[u]=1;

        }else{
             if(parent[u]!=v)
               low[u]= min(low[u],dist[v]);
        }
    }

}


int main(){


    int t,caseno=1;
    scanf("%d",&t);

    while(t--){
        scanf("%d %d",&n,&m);
        int u,v;

        for(int i=0;i<m;i++){
            scanf("%d %d",&u,&v);
            u--; v--;
            g[u].push_back(v);
            g[v].push_back(u);
        }

        int res;
        res = level=0;

        for(int i=0;i<n;i++){
            visit[i]=low[i]=ap[i]=dist[i]=0;
            parent[i]=-1;
        }

        for(int i=0;i<n;i++){
            if(!visit[i]){
                dfs(i);
            }
        }

        for(int i=0;i<n;i++){
            if(ap[i])
                res++;
        }

        printf("Case %d: %d\n",caseno++, res);
        for(int i=0;i<n;i++)
            g[i].clear();

    }
    return 0;
}
```

2 bellmanford

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

using namespace std;

const int INF = 2000000000;
typedef pair<int,int> edge;

vector<edge> g[200];

int a[200],d[200];

int n,m;

void relax(int u,int v,int w){
    if(d[u]!=INF && d[v] > d[u] + w ){
        d[v]=d[u]+w;
```

```
    }
}


bool bellmanFord(int s){

    for(int i=0;i<n;i++){
        d[i]=INF;
    }

    d[s]=0;

    for(int k=0;k<n;k++){
        for(int i=0;i<n;i++){
             for(int j=0;j<g[i].size();j++){
                int w = g[i][j].first; int v = g[i][j].second;
                relax(i,v,w);
            }
        }
    }

     for(int i=0;i<n;i++){
           for(int j=0;j<g[i].size();j++){
                int w = g[i][j].first; int v = g[i][j].second;
                if(d[i]!=INF && d[v] > d[i] + w )
                    return false;
            }
    }

    return true;
}
```

3 binaryexpo

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

using namespace std;

typedef unsigned long long ull;


int m=1000003;

//return (a^b)%m

ull eb (ull a, ull b){

    ull res=1;
    ull x= a%m;

    while(b>0){
        if(b%2)
            res= (res*x)%m;
        x = (x*x)%m;
        b/=2;
    }

    return res;
}

//return (a*b)%m
ll mul(ull a, ull b, ull mod) {
    ull  ret = 0;
    for (a %= mod, b %= mod; b != 0; b >>= 1, a <<= 1, a = a >= mod ? a -
         mod : a) {
        if (b&1) {
            ret += a;
            if (ret >= mod)  ret -= mod;
        }
    }
    return ret;
}

int main(){
        return 0;
}
```

4 binarysearch

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

using namespace std;

int a [100000];
```

```
int n;

int binarySearch(int l, int h, int t){

    while(l <= h){
        int m = (l+h)/2;
        if(a[ m] == t) return m;
        else if( t < a[m]) h = m-1;
        else l = m+1;
    }
    return -1;
}

//Finds the first element
//where the predicate is true
int upperBound(long long x){

    int l = 0, h = n-1,m;

    while(l< h){
         m = l + ( (h-l)/2 );
        if( a[m] > x)
            h = m;
        else l = m +1;
    }
    return l;
}

//Finds the last element (index)
//where the predicate is false
int lowerBound(long long x){

    int l=0, h = n-1,m;
    while(l<h){
        m = l + ( (h-l+1)/2 );
        if(a[m] >= x)
            h = m-1;
        else l = m;
    }
    return l;
}

//bisection method
double binarySearch(){
    double l=0, h=1,m;

    int t=100;
    while(t--){
        m = (l+h)/2;
        if( f(m) * f(l) > 0 )
            l = m;
        else
            h=m;
    }
    return m;
}

int main(){

    scanf("%d",&n);

    for(int i=0;i<n;i++)
        scanf("%d",a+i);

    int m;
    scanf("%d",&m);
    int aux;

    for(int i=0;i<m;i++){
        scanf("%d",&aux);
        printf("%d\n",binarySearch(0,n-1,aux));
    }
    return 0;
}
```

5 bipartitematching

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

using namespace std;

typedef pair<int,int> node;

int  blacks,whites;

typedef vector<int> VI;
```

```
typedef vector<VI> VVI;

bool FindMatch(int i, const VVI &w, VI &mc, VI &seen) {
     for (int j = 0; j < w[i].size(); j++) {
       if (!seen[ w[i][j] ]) {
         seen[ w[i][j] ] = true;
         if (mc[ w[i][j] ] < 0 || FindMatch(mc[ w[i][j] ], w, mc, seen)) {
           mc[w[i][j] ] = i;
           return true;
         }
       }
     }
     return false;
}

//return the maximun bipartite matching on a graph
//using whites as |A| and blacks as |B|
int BipartiteMatching(const VVI &w) {
     VI mc = VI(blacks, -1);
     int ct = 0;
     for (int i = 0; i < whites; i++) {
       VI seen(blacks);
       if (FindMatch(i, w,mc, seen)) ct++;
     }
     return ct;
}
```

6 bridge

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

using namespace std;

vector <int> g[10002];

int  visit[10002];
int   low[10002];
int  dist[10002];
int parent[10002];
int sets [10002];

int n,m,q,level;

int findset(int v) {
    if (sets[v] != -1 && sets[v] != v)
        return sets[v] = findset(sets[v]);
    return v;
}

void unionset(int x, int y) {
    int a = findset(x), b = findset(y);
    if (a<b) swap(a,b);
    sets[b] = a;
}

//finding bridge in an undirected graph
//using disjoints sets
void dfs(int u){

    visit[u]=1;
    level++;
    dist[u] = low[u] =level;

    for(int i=0;i<g[u].size();i++){
        int v =g[u][i];

        if(!visit[v]){
            parent[v]=u;
            dfs(v);
            low[u] = min(low[u],low[v]);
            if(low[v]>dist[u]){
                //there are a bridge betwen u and v
                unionset(u,v);
            }
        }else if(parent[u]!=v){
                low[u]= min(low[u],dist[v]);
        }
    }

}

int main(){


    while(true){

        scanf("%d %d %d",&n,&m,&q);
```

```
        if(n==0 && m==0 && q==0)
            break;

        int u,v;

        for(int i=0;i<m;i++){
            scanf("%d %d",&u,&v);
            u--; v--;
            g[u].push_back(v);
            g[v].push_back(u);
        }

        for(int i=0;i<n;i++){
            visit[i]= low[i]= dist[i]=0;
            parent[i]=-1;
            sets[i]=i;
        }

        level=0;

        for(int i=0;i<n;i++){
            if(!visit[i]){
                dfs(i);
            }
        }

        while(q--){
            scanf("%d %d",&u,&v);
            u--;v--;
            if(findset(u)==findset(v))
                printf("Y\n");
            else
                printf("N\n");
        }

        printf("-\n");
        for(int i=0;i<n;i++)
            g[i].clear();
    }
    return 0;
}
```

7 chul

```
long long cross(pair<int, int> A, pair<int, int> B, pair<int, int> C) {
       return (B.first - A.first)*(C.second - A.second)
            - (B.second - A.second)*(C.first - A.first);
}
// The hull is returned in param "hull"
void convex_hull(vector<pair<int, int> > pts, vector<pair<int, int> >&
    hull) {
       hull.clear(); sort(pts.begin(), pts.end());
       for (int i = 0; i < pts.size(); i++) {
              while (hull.size() >= 2 && cross(hull[hull.size()-2],
                  hull.back(), pts[i]) <= 0) {
                     hull.pop_back();
              }
              hull.push_back(pts[i]);
       }
       int s = hull.size();
       for (int i = pts.size()-2; i >= 0; i--) {
              while (hull.size() >= s+1 && cross(hull[hull.size()-2],
                  hull.back(), pts[i]) <= 0) {
                     hull.pop_back();
              }
              hull.push_back(pts[i]);
       }
}
```

8 dijkstra

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

using namespace std;

typedef pair<int,int> edge;

int dist[101];
vector< edge > g[101];

int n, m;
const int INF = 2000000000;

int dijkstra(int s,int t){
```

```
    priority_queue<edge, vector<edge>, greater<edge> > pq;

    for(int i=1;i<=n;i++) dist[i]=INF;

    pq.push(make_pair (0, s));
    dist[s] = 0;

    while (!pq.empty()){

        edge u = pq.top();
        pq.pop();

        if (u.second == t)
            return dist[u.second];

        int here = u.second;

        for (int i=0;i<g[here].size();i++){

            int v = g[here][i].second;
            int wv= g[here][i].first;

            if(dist[here] + wv < dist[v]){
                dist[v] = dist[here] + wv;
                pq.push(make_pair(dist[v],v));
            }
        }
  }

  return -1;
}

int main(){

    int t,caseno=1;

    scanf("%d",&t);

    while(t--){

        scanf("%d %d",&n,&m);

        while(m--){
            int u,v,w;
            scanf("%d %d %d",&u,&v,&w);
            g[u].push_back( make_pair(w,v));
            g[v].push_back( make_pair(w,u));

        }

        int res= dijkstra(1,n);

        if(res>=0) printf("Case %d: %d\n",caseno++,res);
        else printf("Case %d: Impossible\n",caseno++);

        for(int i=1;i<=n;i++)
            g[i].clear();
    }
    return 0;
}
```

9 dsu

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

using namespace std;

int parent[100000]; //valor representativo del set
int Rank [100000]; //maxima altura del set
int cant [100000]; //Cantidad de nodos por set
int n;             //Cantidad de sets

//Path Compression
int findset(int x){
    if(parent[x]!=x)
        parent[x] = findset(parent[x]);
    return parent[x];
}

//Union By rank
void unionset(int x, int y){
    int px= findset(x); int py= findset(y);
    if(px==py) return;
    if(Rank[px]< Rank[py]){ parent[px]=py; cant[py]+= cant[px];}
    if(Rank[py]< Rank[px]){ parent[py]=px; cant[px]+= cant[py]; }
```

```
    if(Rank[py]==Rank[px]){ parent[py]=px; Rank[px]++; cant[px]+=
        cant[py]; }
}

int main(){

    scanf("%d %d",&n);

    for(int i=0;i<n;i++){
        parent[i]=i; Rank[i] = cant[i]=1;
    }

    return 0;
}
```

10 fenwick

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

using namespace std;

int a[100000];

class Fenwick{

    public:
    vector <int> tree;
    int maxVal;

    Fenwick(int n) {
        maxVal=n;
        tree= vector<int>(maxVal+1,0);
    }

    int query(int idx){
        int sum =0;
        while(idx > 0){

            sum+= tree[idx];
            idx -= (idx & -idx);
        }
        return sum;
    }

    void update(int idx, int val){
        while(idx<=maxVal){
            tree[idx]+=val;
            idx+= (idx & -idx);
        }
    }

    void init(int a[],int n){
        for(int i=0;i<n;i++)
            update(i+1,a[i]);
    }
};


int main(){

    int t,caseno=1;

    scanf("%d",&t);

    while(t--){

        int n;
        cin >> n;

        for(int i=0;i<n;i++)
            cin >> a[i];

        Fenwick f(n);
    }
    return 0;
}
```

11 kosaraju

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

using namespace std;
```

```
int n,m;

vector<int> g[10000];
vector<int> t[10000];

vector<int> tps;
stack<int> s;

int visit[10000];

void dfs(int u){
    visit[u]=1;
    for(int i=0;i<g[u].size();i++){
        int v= g[u][i];
        if(!visit[v])
            dfs(v);
    }
    s.push(u);
}

void dfst(int u){
    tps.push_back(u);

    visit[u]=1;
    for(int i=0;i<t[u].size();i++){
        int v= t[u][i];
        if(!visit[v])
            dfst(v);
    }
}


void kosaraju(){

   for(int i=0;i<n;i++)
        visit[i]=0;

    for(int i=0;i<n;i++)
        if(!visit[i])
            dfs(i);

   for(int i=0;i<n;i++)
        visit[i]=0;
    tps = vector<int>();

    while(!s.empty()){
        int u = s.top();
        s.pop();

        if(!visit[u]){
            dfst(u);
        }

    }
}

int main(){

    int test,caseno=1;

    scanf("%d",&test);

    while(test--){

        scanf("%d %d",&n,&m);
        int u,v;

        for(int i=0;i<m;i++){
            scanf("%d %d",&u,&v);
            g[u-1].push_back(v-1);
            t[v-1].push_back(u-1);
        }

        kosaraju();


        for(int i=0;i<n;i++){
             g[i].clear();
             t[i].clear();
        }
        s = stack<int>();
        return 0;
}
```

12 lisnlogn

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

using namespace std;

const int inf = 2000000000;

vector<int> a,i,id,p;
int length,last;

int binarySearch(int t){
    int l = 0, h = i.size(),m;
    while(l< h){
        m = l+ ((h-l)/2);
        if( t <= i[m]) h = m;
        else l = m+1;
    }
    return l;
}

void dp(){

    i  = vector<int>(a.size(),inf);

    p = id = vector<int>(a.size(),-1);

    i[0]=a[0];id[0]=0;
    length = 1;last=0;

    for(int j=1;j<a.size();j++){

        int index = binarySearch(a[j]);
        i[index] = a[j];
        id[index] = j;

        p[j]= index? id[index-1] :-1;

        if(index >= length)
            length=index, last = j;
    }

    stack<int> s; int x = last;

    for(;p[x]>=0;x =p[x]) s.push(a[x]);
    s.push(a[x]);

    cout << length+1 << "\n-\n";
    while(!s.empty()) cout << s.top() << "\n",s.pop();
}

int main(){

    return 0;
}
```

13 maxflow

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

using namespace std;

typedef long long LL;

struct Edge {
  int from, to, cap, flow, index;
  Edge(int from, int to, int cap, int flow, int index) :
    from(from), to(to), cap(cap), flow(flow), index(index) {}
};

struct PushRelabel {
  int N;
  vector<vector<Edge> > G;
  vector<LL> excess;
  vector<int> dist, active, count;
  queue<int> Q;

  PushRelabel(int N) : N(N), G(N), excess(N), dist(N), active(N),
      count(2*N) {}

  void AddEdge(int from, int to, int cap) {
    G[from].push_back(Edge(from, to, cap, 0, G[to].size()));
    if (from == to) G[from].back().index++;
    G[to].push_back(Edge(to, from, 0, 0, G[from].size() - 1));
  }

  void Enqueue(int v) {
    if (!active[v] && excess[v] > 0) { active[v] = true; Q.push(v); }
```

```
}

void Push(Edge &e) {
  int amt = int(min(excess[e.from], LL(e.cap - e.flow)));
  if (dist[e.from] <= dist[e.to] || amt == 0) return;
  e.flow += amt;
  G[e.to][e.index].flow -= amt;
  excess[e.to] += amt;
  excess[e.from] -= amt;
  Enqueue(e.to);
}

void Gap(int k) {
  for (int v = 0; v < N; v++) {
    if (dist[v] < k) continue;
    count[dist[v]]--;
    dist[v] = max(dist[v], N+1);
    count[dist[v]]++;
    Enqueue(v);
  }
}

void Relabel(int v) {
  count[dist[v]]--;
  dist[v] = 2*N;
  for (int i = 0; i < G[v].size(); i++)
    if (G[v][i].cap - G[v][i].flow > 0)
  dist[v] = min(dist[v], dist[G[v][i].to] + 1);
  count[dist[v]]++;
  Enqueue(v);
}

void Discharge(int v) {
  for (int i = 0; excess[v] > 0 && i < G[v].size(); i++) Push(G[v][i]);
  if (excess[v] > 0) {
    if (count[dist[v]] == 1)
  Gap(dist[v]);
    else
  Relabel(v);
  }
}

LL GetMaxFlow(int s, int t) {
  count[0] = N-1;
  count[N] = 1;
    dist[s] = N;
    active[s] = active[t] = true;
    for (int i = 0; i < G[s].size(); i++) {
      excess[s] += G[s][i].cap;
      Push(G[s][i]);
    }

    while (!Q.empty()) {
      int v = Q.front();
      Q.pop();
      active[v] = false;
      Discharge(v);
    }

    LL totflow = 0;
    for (int i = 0; i < G[s].size(); i++) totflow += G[s][i].flow;
    return totflow;
  }
};


//maxflow using pushrelabel
int main(){

    int t,caseno=1;

    scanf("%d",&t);

    while(t--){
        int n;
        scanf("%d",&n);

        int s,t,c;
        scanf("%d %d %d",&s,&t,&c);
        PushRelabel r(n);

        while(c--){
            int u,v,b;
            scanf("%d %d %d",&u,&v,&b);
            r.AddEdge(u-1,v-1,b);
            r.AddEdge(v-1,u-1,b);
        }

        LL res = r.GetMaxFlow(s-1,t-1);
```

```
        printf("Case %d: %lld\n",caseno++,res);
    }

    return 0;
}
```

14 mergesort

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

using namespace std;

long a[500000];

const int oo= INT_MAX;

long long res;

int mergeR(int l, int mid, int h){

    int n1 = mid- l+1;
    int n2 = h - mid;

    long *left = new long[n1 + 1];
    long *right= new long[n2 + 1];

    int i = 0;
    for(; i < n1; i++)
        left[i] = a[ l + i];

    int j = 0;
    for(; j < n2; j++)
        right[j] = a[mid + 1 + j];

    left [n1] = oo;
    right[n2] = oo;
    i = j = 0;

    for(int k = l; k <= h; k++ ){
        if(left[i] <= right[j]){
            a[k] = left[i++];
            res+= j;
        }
        else
            a[k] = right[j++];
    }
}

void mergeSort(int l, int h){

    if(l < h){
       int mid = (h + l) >> 1;
       mergeSort(l, mid);
       mergeSort(mid + 1, h);
       mergeR(l, mid, h);
    }
}

int main(){
    int n;
    while(scanf("%d",&n) && n){

        for(int i=0;i<n;i++)
            scanf("%ld",a+i);
        res=0;
        mergeSort(0,n-1);
        printf("%lld\n",res);
    }
    return 0;
}
```

15 mo

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

using namespace std;

 int n,q,BLOCK_SIZE;


inline bool mo_cmp(const pair< pair<int, int>, int> &x,
       const pair< pair<int, int>, int> &y)
{
    int block_x = x.first.first / BLOCK_SIZE;
```

```
    int block_y = y.first.first / BLOCK_SIZE;
    if(block_x != block_y)
        return block_x < block_y;
    return x.first.second < y.first.second;
}


struct Mo
{

    int MAX_VALUE = 100001;

    vector<long long> cnt;

    long long current_answer;

    public:

    Mo(){
        cnt = vector<long long>(MAX_VALUE, 0);
        current_answer = 0;
    }

    long long get_answer() const {

        return current_answer;
    }

    void add(int number){

        cnt[number]+=1;

        if(cnt[number]==1)
            current_answer++;
    }

    void remove(int number){

        cnt[number]-=1;

        if(cnt[number]==0)
            current_answer--;
    }
};
```

```
int main(){


    int t,caseno=1;
    scanf("%d",&t);

    while(t--){


        scanf("%d %d",&n,&q);

        BLOCK_SIZE= (int)(sqrt(n));

        vector<int> a(n);
        vector<long long> answers(q);

        vector < pair< pair<int, int>, int> > queries;
        queries.reserve(q);

        for(int i=0;i<n;i++)
            scanf("%d",&a[i]);

        for(int i=0;i<q;i++){
            int l,r;
            scanf("%d %d",&l,&r);
            queries.push_back(make_pair(make_pair(l-1,r-1),i));
        }

        sort(queries.begin(), queries.end(), mo_cmp);

        Mo mo;

        int ml=0, mr=-1;

        for(int i=0;i< q;i++){

            int l = queries[i].first.first;
            int r = queries[i].first.second;

            while(mr < r) {
                mr++;
                mo.add(a[mr]);
            }
```

```
            while(mr > r) {
                mo.remove(a[mr]);
                mr--;
            }

            while(ml < l) {
                mo.remove(a[ml]);
                ml++;
            }

            while(ml > l) {
                ml--;
                mo.add(a[ml]);
            }

            answers[queries[i].second] = mo.current_answer;
        }

        printf("Case %d:\n",caseno++);
        for(int i=0;i<q;i++)
            printf("%lld\n",answers[i]);

    }
    return 0;
}
```

16 pollard$_r ho$

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

using namespace std;

const int rounds=10;

typedef unsigned long long ull;
typedef long long ll;

ll gcd(ll a, ll b) {
    if(b==0) return a;
    else return gcd(b,a%b);
}
```

```
ll mul(ull a, ull b, ull mod) {
    ull  ret = 0;
    for (a %= mod, b %= mod; b != 0; b >>= 1, a <<= 1, a = a >= mod ? a -
        mod : a) {
        if (b&1) {
            ret += a;
            if (ret >= mod)  ret -= mod;
        }
    }
    return ret;
}


ll powm(ll a,ll b,ll c){
    ll ans = 1;
    while (b > 0) {
        if (b & 1)
            ans = mul(ans, a, c);
        a = mul(a, a, c);
        b >>= 1;
    }
    return ans;
}

bool witness(ll a,ll n){

    ll u=n-1;
    ll t=0;

    if(n==a) return true;

    while(u%2==0){ t++; u>>=1; }

    ll next = powm(a,u,n);
    if(next==1 )return false;
    ll last;
    for(int i=0;i<t;i++){
        last = next;
        next = mul(last, last,n);
        if(next==1){
            return last != n-1;
        }
    }
    return next !=1;
}
```

```
bool miller_rabin(ull n, int it= rounds){
    if (n <= 1) return false;
    if (n == 2) return true;
    if (n % 2 == 0) return false;
    for (int i = 0; i < it; ++i) {
        ll a = rand() % (n - 1) + 1;
        if (witness(a, n)) {
            return false;
        }
    }
    return true;
}

ll pollard_rho(ll n){
    ll x, y, i = 1, k = 2, d;
    x = y = rand() % n;
    while (1) {
        ++i;
        x = mul(x, x, n);
        x += 2;
        if (x >= n) x -= n;
        if (x == y) return 1;
        d = gcd(abs(x - y), n);
        if (d != 1) return d;
        if (i == k) {
            y = x;
            k *= 2;
        }
    }
    return 1;
}

vector<ll> factorize(ll n) {
    vector<ll> ans;
    if (n == 1)
    return ans;
    if (miller_rabin(n)) {
        ans.push_back(n);
    } else {
        ull d = 1ull;
        while (d == 1)
            d = pollard_rho(n);
        vector<ll> dd = factorize(d);
        ans = factorize(n / d);
        for (int i = 0; i < dd.size(); ++i)
            ans.push_back(dd[i]);
    }
    return ans;
}
```

17 primesieve

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

using namespace std;

const int MAX = 10000000;
const int LMT =   10000;

int m[(MAX>>6)+1];

vector<int> primes;

#define isComp(n) (m[n>>6]&(1<<((n>>1)&31)))
#define setComp(n) m[n>>6]|=(1<<((n>>1)&31))

void sieve() {
    for (int i = 3; i <= LMT; i += 2)
        if (!isComp(i))
            for (int j = i*i; j <= MAX; j += i+i)
                setComp(j);

    primes.push_back(2);
    for (int i=3; i <= MAX; i += 2)
        if (!isComp(i))
            primes.push_back(i);
}

bool isPrime(int n) {
    if(n==2) return true;
    if (n < 2 || n % 2 == 0) return false;
    return ! isComp(n);
}
```

18 segmenttree

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

using namespace std;


class SegmentTree {

public:

   vector<int> a, tree;

   SegmentTree(vector<int> &_a) : a(_a) {
      int n = a.size();
      tree =vector<int> (4 * n, 1);
      build(1, 0, n - 1);
   }

   void build(int node, int lo, int hi) {

      if(lo == hi) {
         tree[node] = a[lo];
         return;
      }
      int mid = (lo+hi)/2;

      build(2*node, lo, mid);
      build(2*node + 1, mid + 1, hi);

      tree[node] = tree[2*node] * tree[2*node+1];
   }

   int query(int i, int j) {
      return _query(1, 0, a.size() - 1, i, j);
   }

   int _query(int node, int lo, int hi, int i, int j) {

      if(hi < i || lo > j)
         return 1;

      if(lo >= i && hi <= j)
         return tree[node];

      int mid = (lo + hi) / 2;

      return _query(2*node, lo, mid, i, j) * _query(2*node+1, mid + 1,
          hi, i, j);
   }

   void update(int ind, int val) {
      _update(1, ind, val, 0, a.size() - 1);
   }

   void _update(int node, int ind, int val, int lo, int hi) {

      if(ind < lo || ind > hi)
         return;

      if(lo == hi) {
         tree[node] = val;
         return;
      }

      int mid = (lo + hi) / 2;

      _update(2*node, ind, val, lo, mid);
      _update(2*node + 1, ind, val, mid + 1, hi);

      tree[node] = tree[2*node] * tree[2*node+1];
   }
};


int main(){

    int n,k,d1,d2;
    char c;

    while(cin >> n >> k){
        vector<int> a;
        int ai;

        for(int i=0;i<n;i++){
            cin >> ai;
            if(ai>0)a.push_back(1);
            if(ai<0)a.push_back(-1);
            if(ai==0)a.push_back(0);
```

```
        }

        SegmentTree st(a);

        string s="";

        while(k--){
            cin >> c >> d1 >> d2;
            if(c=='C'){
               if(d2>0)d2=1;
               if(d2<0)d2=-1;
               if(d2==0)d2=0;

               st.update(d1-1,d2);
               a[d1-1]=d2;
            }
            else{
                int v = st.query(d1-1,d2-1);
                if( v > 0) s+='+';
                else if(v < 0) s+='-';
                else s+='0';
            }
        }
        cout << s << "\n";
    }
    return 0;
}
```

19 segmenttreelazy

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

using namespace std;

class SegmentTree{

public:

    vector<int> a;
    vector<long> tree;
    vector<long> state;
    int n;

    SegmentTree(int m) : n(m){
        int h = 1 + ceil(log2(n));
        a.resize(n);
        tree.resize( 1 << h );
        state.resize( 1 << h);
    }

    void init(){ build(1,0,n-1);}

    void build(int node,int l, int h){

        state[node] = 0;

        if(l==h){ tree[node]= a[l]; return; }

        int lf = node*2; int rt = lf+1; int mid = (l+h)/2;

        build(lf,l,mid); build(rt,mid+1,h);

        tree[node] = tree[lf] + tree[rt];
    }

    long query(int i,int j){ return _query(1 , 0 , n-1 , i , j ); }

    long _query(int node, int l, int h, int i, int j){

        if( h < i || l > j) return 0;

         propagate(node,l,h);

         if(l >= i && h <= j) return tree[node];

        int lf = node*2; int rt = lf+1; int mid = (l+h)/2;

        return _query(lf, l, mid, i, j) + _query(rt, mid+1,h,i,j);
    }

    void update(int i, int j, int v){ _update(1,0,n-1,i,j,v); }

    void _update(int node, int l, int h, int i, int j, int v){

        propagate(node,l,h);
```

```
        if( h < i || l > j) return;

        if(l==h){ tree[node]+=v; state[node]=0; return; }

        int lf = node*2; int rt = lf+1; int mid = (l+h)/2;

        if(l >= i && h <= j){
            tree[node] += (h-l+1)*v;
            state[lf] += v;
            state[rt] += v;
            return;
        }

        _update(lf, l, mid, i, j,v); _update(rt, mid+1,h,i,j,v);
        tree[node] = tree[lf] + tree[rt];
    }

    void propagate(int node, int l, int h){

        if(state[node]==0) return;

        tree[node]+= (h-l+1)*state[node];

        if(l!=h){
            int lf = node*2; int rt = lf +1;
            state[lf]+= state[node]; state[rt]+=state[node];
        }
        state[node]=0;
    }
};

int main(){
    int t,caseno=1;
    scanf("%d",&t);

    int n,q,qi,h,l,v;

    while(t--){
        scanf("%d %d",&n,&q);

        SegmentTree st(n);
        for(int i=0;i<n;i++)
            st.a[i]=0;
        st.init();

        printf("Case %d:\n",caseno++);

        while(q--){
            scanf("%d",&qi);
            if(qi){
                scanf("%d %d",&l,&h);
                printf("%ld\n",st.query(l,h));
            }else{
                scanf("%d %d %d",&l,&h,&v);
                st.update(l,h,v);
            }
        }
    }

    return 0;
}
```

20 util

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

using namespace std;

//On bits on bitmask
long bits(long i)
{
     i = i - ((i >> 1) & 0x55555555);
     i = (i & 0x33333333) + ((i >> 2) & 0x33333333);
     return (((i + (i >> 4)) & 0x0F0F0F0F) * 0x01010101) >> 24;
}

void allStringToLower(string a){
    transform(a.begin(), a.end(), a.begin(), ::tolower);
}

int main(){

}
```