```
/*
 * Edmond's algoirthm for Minimum Directed Spanning Tree
 * runs in O(VE)
 */


// default code for competitive programming
// c2251393 ver 3.141 {{{

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

// Defines
#define NAME(x) #x
#define SZ(c) (int)(c).size()
#define ALL(c) (c).begin(), (c).end()
#define FOR(it, c) for(__typeof((c).begin()) it = (c).begin(); it !=
(c).end(); it++)
#define REP(i, s, e) for(int i = (s); i <= (e); i++)
#define REPD(i, s, e) for(int i = (s); i >= (e); i--)
#define DEBUG 1
#define fst first
#define snd second

using namespace std;

// Typedefs
typedef double real;
typedef long long ll;
typedef pair<ll, int> pli;
typedef pair<int, int> pii;
typedef unsigned long long ull;

// Some common const.
const double EPS = -1e8;
const double Pi = acos(-1);

// Equal for double
bool inline equ(double a, double b)
{return fabs(a - b) < EPS;}

// }}}
// start ~~QAQ~~

const int MAXV = 10010;
const int MAXE = 10010;
const int INF  = 2147483647;

struct Edge
{
  int u, v, c;
  Edge(){}
  Edge(int x, int y, int z) :
    u(x), v(y), c(z){}
};

int V, E, root;
Edge edges[MAXE];

inline int newV()
{
  V++;
  return V;
}

inline void addEdge(int u, int v, int c)
{
  E++;
  edges[E] = Edge(u, v, c);
}

bool con[MAXV];
int mnInW[MAXV], prv[MAXV], cyc[MAXV], vis[MAXV];

inline int DMST()
{
  fill(con, con+V+1, 0);
  int r1 = 0, r2 = 0;
  while(1)
  {
    fill(mnInW, mnInW+V+1, INF);
    fill(prv, prv+V+1, -1);
    REP(i, 1, E)
    {
      int u = edges[i].u, v = edges[i].v, c = edges[i].c;
      if(u != v && v != root && c < mnInW[v])
        mnInW[v] = c, prv[v] = u;
    }
    fill(vis, vis+V+1, -1);
    fill(cyc, cyc+V+1, -1);

    r1 = 0;
    bool jf = 0;
    REP(i, 1, V)
    {
      if(con[i]) continue ;
      if(prv[i] == -1 && i != root) return -1;
      if(prv[i] > 0) r1 += mnInW[i];
      int s;
      for(s = i; s != -1 && vis[s] == -1; s = prv[s])
        vis[s] = i;
      if(s > 0 && vis[s] == i) // get a cycle
      {
        jf = 1;
        int v = s;
        do
        {
          cyc[v] = s, con[v] = 1;
          r2 += mnInW[v];
          v = prv[v];
        }while(v != s);
        con[s] = 0;
      }
    }
    if(!jf) break ;
    REP(i, 1, E)
    {
      int &u = edges[i].u;
      int &v = edges[i].v;
      if(cyc[v] > 0) edges[i].c -= mnInW[edges[i].v];
      if(cyc[u] > 0) edges[i].u = cyc[edges[i].u];
      if(cyc[v] > 0) edges[i].v = cyc[edges[i].v];
      if(u == v) edges[i--] = edges[E--];
    }
  }
  return r1+r2;
}

int main()
{
  ios_base::sync_with_stdio(0);
}
```

```
#include <math.h>
#include <stdio.h>
#define N 64
#define ll unsigned long long
ll nb[ N ];
ll getint(){
ll x=0LLU; char c=getchar();
while(c<'0'||c>'9') c=getchar();
while(c>='0'&&c<='9') x*=10LLU,x+=(c-'0'),c=getchar();
return x;
}
ll n , ans , tmp;
void init(){
    n = getint(); ans = 1LLU;
    for( ll i = 0LLU ; i < n ; i ++ ){
        nb[ i ] = 0LLU;
        for( ll j = 0LLU ; j < n ; j ++ ){
            tmp = getint();
            if( tmp ) nb[ i ] |= ( 1LLU << j );
        }
    }
}
void B( ll r , ll p , ll x , ll cnt , ll res ){

    if( cnt + res < ans ) return;

    if( p == 0LLU && x == 0LLU ){
        if( cnt > ans ) ans = cnt;
        return;
    }

    ll y = p | x; y &= -y;
    ll q = p & ( ~nb[ int( log2( y ) ) ] );

    while( q ){
        ll i = int( log2( q & (-q) ) );
        B( r | ( 1LLU << i ) , p & nb[ i ] , x & nb[ i ]
        , cnt + 1LLU , __builtin_popcountll( p & nb[ i ] ) );
        q &= ~( 1LLU << i );
        p &= ~( 1LLU << i );
        x |= ( 1LLU << i );
    }
}
void process(){
    if( n < 64LLU ) B( 0LLU , ( 1LLU << n ) - 1LLU , 0LLU , 0LLU , n );
    else{
        ll b = 0LLU;
        for( ll i = 0LLU ; i < 64LLU ; i ++ )
            b |= ( 1LLU << i );
        B( 0LLU , b , 0LLU , 0LLU , n );
    }
    printf( "%llu\n" , ans );
}
int main(){
    ll t; t = getint(); while( t -- ){
        init(); process();
    }
}
```

```
const int MAXM = 1000010;
struct SAM
{
  int tot, root, lst, mom[MAXM], mx[MAXM];
  int acc[MAXM], nxt[MAXM][33];
  int newNode()
  {
    int res = ++tot;
    fill(nxt[res], nxt[res]+33, 0);
    mom[res] = mx[res] = acc[res] = 0;
    return res;
  }
  void init()
  {
    tot = 0;
    root = newNode();
    mom[root] = 0, mx[root] = 0;
    lst = root;
  }
  void push(int c)
  {
    int p = lst;
    int np = newNode();
    mx[np] = mx[p]+1;
    for(; p && nxt[p][c] == 0; p = mom[p])
      nxt[p][c] = np;
    if(p == 0) mom[np] = root;
    else
    {
      int q = nxt[p][c];
      if(mx[p]+1 == mx[q]) mom[np] = q;
      else
      {
        int nq = newNode();
        mx[nq] = mx[p]+1;
        for(int i = 0; i < 33; i++)
          nxt[nq][i] = nxt[q][i];
        mom[nq] = mom[q];
        mom[q] = nq;
        mom[np] = nq;
        for(; p && nxt[p][c] == q; p = mom[p])
          nxt[p][c] = nq;
      }
    }
    lst = np;
  }
  void print()
  {
    REP(i, 1, tot)
    {
```

```
      printf("node %d :\n", i);
      printf("mx %d, mom %d\n", mx[i], mom[i]);
      REP(j, 1, 26) if(nxt[i][j])
        printf("nxt %c %d\n", 'a'+j-1, nxt[i][j]);
      puts("----------------------------");
    }
  }
  void push(char *str)
  {
    for(int i = 0; str[i]; i++)
      push(str[i]-'a'+1);
  }
};
SAM sam;
```

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

#define N 100010                                // second approach: O(n log n)

char T[ N ];
int n;
int RA[ N ], tempRA[ N ];
int SA[ N ], tempSA[ N ];
int c[ N ];

void countingSort( int k ) {
      int i , sum , maxi = max( 300 , n ) ;
      memset( c , 0 , sizeof c ) ;
      for ( i = 0 ; i < n ; i ++ ) c[ ( i + k < n ) ? RA[i + k] : 0 ] ++ ;
      for ( i = sum = 0 ; i < maxi ; i ++ ) { int t = c[i] ; c[i] = sum ;
sum += t ; }
      for ( i = 0 ; i < n ; i ++ )
            tempSA[ c[ ( SA[ i ] + k < n ) ? RA[ SA[ i ] + k ] : 0 ] ++ ] =
SA[ i ] ;
      for ( i = 0 ; i < n ; i ++ ) SA[ i ] = tempSA[ i ] ;
}

void constructSA() {
      int r;
      for ( int i = 0 ; i < n ; i ++ ) RA[ i ] = T[ i ] - '.' ;
      for ( int i = 0 ; i < n ; i ++ ) SA[ i ] = i ;
      for ( int k = 1 ; k < n ; k <<= 1 ) {
            countingSort( k ) ; countingSort( 0 ) ;
            tempRA[ SA[ 0 ] ] = r = 0;
            for ( int i = 1 ; i < n ; i ++ )
                  tempRA[ SA[ i ] ] = ( RA[ SA[ i ] ] == RA[ SA[ i - 1 ] ]
&& RA[ SA[ i ] + k ] == RA[ SA[ i - 1 ] + k ] ) ? r : ++ r ;
            for ( int i = 0 ; i < n ; i ++ ) RA[ i ] = tempRA[ i ] ;
      }
}

int main() {
  n = (int)strlen( gets( T ) ) ;

  T[ n ++ ] = '.' ; // important bug fix!
  constructSA() ;
  return 0;
}
```

```
# include<stdio.h>
# include<string.h>
# include<string>
# include<cmath>
# include<vector>
# include<algorithm>
using namespace std;
typedef long long ll;
typedef unsigned int uint;

# define maxn 310010
# define nmaxn 141073
struct comp{
    double a , b ;
    comp( double a_ = 0.0 , double b_ = 0.0 ) : a( a_ ) , b( b_ ){ }
} null ;
comp operator+ ( const comp &a , const comp &b ) { return comp(a.a+b.a,a.b
+b.b); }
comp operator- ( const comp &a , const comp &b ) { return comp(a.a-b.a,a.b-
b.b); }
comp operator* ( const comp &a , const comp &b ) { return comp(a.a*b.a-
a.b*b.b,a.a*b.b+a.b*b.a); }

char s[ maxn ] ;
int n ;
comp A[ nmaxn ] , B[ nmaxn ] , C[ nmaxn ] ;
const double pi = acos( -1 ) ;
int L = 6 ;
ll base[ 10 ] , M = 1000000 ;

int get( comp *A ){
      if ( scanf( "%s" , s ) == EOF ) return 0 ;
      int a = 0 , p = 0 , l = 0 ;
      for ( register int i = strlen( s ) - 1 ; i >= 0 ; i -- ) {
            a += ( s[ i ] - '0' ) * base[ p ++ ] ;
            if( p == L ) A[ l ++ ] = comp( a , 0 ) , a = p = 0 ;
      }
      if ( a ) A[ l ++ ] = comp( a , 0 ) ;
      return l;
}

bool init( ){
      base[ 0 ] = 1 ;
      for ( register int i = 1 ; i <= L ; i ++ ) base[ i ] = base[ i - 1 ] *
10 ;
      int l = get( A ) + get( B );
      if ( l == 0 ) return false ;
      for ( n = 1 ; n < l ; n <<= 1 );
      //printf( "%d\n" , n ) ;
      return true ;
}

comp p[ 2 ][ nmaxn ];
int typ;

uint rev( uint a ){
      a = ( ( a & 0x55555555U ) << 1 ) | ( ( a & 0xAAAAAAAAU ) >> 1 ) ;
      a = ( ( a & 0x33333333U ) << 2 ) | ( ( a & 0xCCCCCCCCU ) >> 2 ) ;
      a = ( ( a & 0x0F0F0F0FU ) << 4 ) | ( ( a & 0xF0F0F0F0U ) >> 4 ) ;
      a = ( ( a & 0x00FF00FFU ) << 8 ) | ( ( a & 0xFF00FF00U ) >> 8 ) ;
      a = ( ( a & 0x0000FFFFU ) << 16 ) | ( ( a & 0xFFFF0000U ) >> 16 ) ;
      return a;
}

void FFT( comp *s , comp *bac , int n ){
      register int d = log2( n );
      for ( register int i = 0 ; i < n ; i ++ ) s[ rev( i ) >> ( 32 - d ) ]
= bac[ i ];
      for ( register int i = 1 ; i <= d ; i ++ ) {
            int step = 1 << i , v = step >> 1 , rstep = n / step ;
            for ( register int j = 0 ; j <= n - 1 ; j += step ) {
                  comp *t = p[ typ ];
                  for ( register int k = 0 ; k < v ; k ++ , t += rstep ) {
                        comp d = ( *t ) * s[ k + j + v ];
                        s[ k + j + v ] = s[ k + j ] - d ;
                        s[ k + j ] = s[ k + j ] + d ;
                  }
            }
      }
}

ll ans[ 4 * maxn ];

bool work( ){
      if ( !init( ) ) return false ;
      p[ 0 ][ 0 ] = comp( 1 , 0 ) , p[ 1 ][ 0 ] = comp( 1 , 0 );
      for ( register int i = 1 ; i < n ; i ++ ) {
            p[ 0 ][ i ] = comp( cos( 2 * i * pi / n ) , sin( 2 * i * pi /
n ) );
            p[ 1 ][ i ] = comp( cos( 2 * i * pi / n ) , -sin( 2 * i * pi / n
) );
      }
      typ = 0; FFT( C , A , n ) , FFT( A , B , n ) ;
      for ( register int i = 0 ; i < n ; i ++ ) A[ i ] = A[ i ] * C[ i ] ;
      typ = 1 ; FFT( C , A , n ) ;
      for ( register int i = 0 ; i < n ; i ++ )
            ans[ i ] = C[ i ].a / n + 0.1 , A[ i ] = null , B[ i ] = null ;
      for ( register int i = 0 ; i < n ; i ++ )
            if ( ans[ i ] >= M ) ans[ i + 1 ] += ans[ i ] / M , ans[ i ] %=
M ;
      while ( n > 1 && ans[ n - 1 ] <= 0 ) n -- ;
      printf( "%lld" , ans[ n - 1 ] ) ;
      for( register int i = n - 2 ; i >= 0 ; i -- ) printf( "%06lld" ,
ans[ i ] );
      puts( "" ) ;
      return true ;
}

int main(){
      //freopen( "input.txt" , "r" , stdin ) ;
      //freopen( "output.txt" , "w" , stdout ) ;
      //int cases ;
      //scanf( "%d" , &cases ) ;
      //while( cases-- ) work( ) ;
      while ( work() ) ;
      return 0;
}
```

```
/*
  A template for Min Cost Max Flow
  tested with TIOJ 1724
*/

#include <bits/stdc++.h>

using namespace std;

struct MinCostMaxFlow
{
  static const int MAXV = 20010;
  static const int INF  = 1000000000;
  struct Edge
  {
    int v, cap, w, rev;
    Edge(){}
    Edge(int t2, int t3, int t4, int t5)
    : v(t2), cap(t3), w(t4), rev(t5) {}
  };
  int V, s, t;
  vector<Edge> g[MAXV];
  void init(int n)
  {
    V = n+2;
    s = n+1, t = n+2;
    for(int i = 1; i <= V; i++) g[i].clear();
  }
  void addEdge(int a, int b, int cap, int w)
  {
    //printf("addEdge %d %d %d %d\n", a, b, cap, w);
    g[a].push_back(Edge(b, cap, w, (int) g[b].size()));
    g[b].push_back(Edge(a, 0, -w, ((int) g[a].size()) - 1));
  }
  int d[MAXV], id[MAXV], mom[MAXV];
  bool inqu[MAXV];
  int qu[2000000], ql, qr;//the size of qu should be much large than MAXV
  int mncmxf()
  {
    int mxf = 0, mnc = 0;
    while(1)
    {
      fill(d+1, d+1+V, -INF);
      fill(inqu+1, inqu+1+V, 0);
      fill(mom+1, mom+1+V, -1);
      mom[s] = s;
      d[s] = 0;
      ql = 1, qr = 0;
      qu[++qr] = s;
      inqu[s] = 1;
      while(ql <= qr)
      {
        int u = qu[ql++];
        inqu[u] = 0;
        for(int i = 0; i < (int) g[u].size(); i++)
        {
          Edge &e = g[u][i];
          int v = e.v;
          if(e.cap > 0 && d[v] < d[u]+e.w)// for min cost : d[v] > d[u]+e.w
          {
            d[v] = d[u]+e.w;
            mom[v] = u;
            id[v] = i;
            if(!inqu[v]) qu[++qr] = v, inqu[v] = 1;
```

```
        }
      }
    }
    if(mom[t] == -1) break ;
    int df = INF;
    for(int u = t; u != s; u = mom[u])
      df = min(df, g[mom[u]][id[u]].cap);
    for(int u = t; u != s; u = mom[u])
    {
      Edge &e = g[mom[u]][id[u]];
      e.cap          -= df;
      g[e.v][e.rev].cap += df;
    }
    //printf("mxf %d mnc %d\n", mxf, mnc);
    mxf += df;
    mnc += df*d[t];
    //printf("mxf %d mnc %d\n", mxf, mnc);
  }
  return mnc;
 }
} flow;

const int MAXN = 110;
const int MAXM = 10010;

struct Edge
{
 int v, c, id;
 Edge(){}
 Edge(int t1, int t2, int t3) : v(t1), c(t2), id(t3) {}
};

int n, m;
vector<Edge> g[MAXN];
int mom[MAXN], in[MAXN], out[MAXN], stamp;
Edge toMom[MAXN];

void dfs(int u, int p)
{
 mom[u] = p;
 in[u] = ++stamp;
 for(Edge& e:g[u]) if(e.v != p)
 {
   toMom[e.v] = e;
   dfs(e.v, u);
 }
 out[u] = ++stamp;
}

int main()
{
 scanf("%d%d", &n, &m);
 flow.init(m+5);
 for(int i = 1; i <= n-1; i++)
 {
   flow.addEdge(flow.s, i, 1, 0);
   flow.addEdge(i, m+1, 1, 0);
   int u, v, c;
   scanf("%d%d%d", &u, &v, &c);
   g[u].push_back(Edge(v, c, i));
   g[v].push_back(Edge(u, c, i));
 }
 flow.addEdge(m+1, flow.t, n-1, 0);
 dfs(1, -1);
 for(int i = n; i <= m; i++)
 {
   flow.addEdge(i, flow.t, 1, 0);
   int u, v, c;
   scanf("%d%d%d", &u, &v, &c);
   while(!(in[u] <= in[v] && out[v] <= out[u]))
   {
     Edge &e = toMom[u];
     if(c < e.c)
       flow.addEdge(e.id, i, 1, e.c-c);
     u = mom[u];
   }
   while(v != u)
   {
     Edge &e = toMom[v];
     if(c < e.c)
       flow.addEdge(e.id, i, 1, e.c-c);
     v = mom[v];
   }
 }
 //printf("JIZZ\n");
 printf("%d\n", flow.mncmxf());
}
```

---

```
#include <iostream>
#include <stdio.h>
#include <string.h>
#define N 5010
#define M 60010
#define ll long long
#define inf 1ll<<62
using namespace std;
ll to[ M ] , next[ M ] , head[ M ];
ll cnt , ceng[ M ] , que[ M ] , w[ M ];
ll n , m , start , end;
void add( ll a , ll b , ll flow ){
    to[ cnt ] = b , next[ cnt ] = head[ a ] , w[ cnt ] = flow , head[ a ] =
cnt ++;
    to[ cnt ] = a , next[ cnt ] = head[ b ] , w[ cnt ] = flow , head[ b ] =
cnt ++;
}
void read(){
    memset(head,-1,sizeof head);
    //memset(next,-1,sizeof next);
    scanf( "%lld%lld" , &n , &m );
    ll a , b , flow;
    for( ll i = 1 ; i <= m ; i ++ ){
        scanf( "%lld%lld%lld" , &a , &b , &flow );
        add( a , b , flow );
    }
    end = n ,start = 1;
}
bool bfs(){
    memset( ceng , -1 , sizeof(ceng) );
    ll h = 1 , t = 2;
    ceng[ start ] = 0;
    que[ 1 ] = start;
    while( h < t ){
        ll sta = que[ h ++ ];
        for( ll i = head[ sta ] ; ~i ; i = next[ i ] )
            if( w[ i ] > 0 && ceng[ to[ i ] ] < 0 ){
                ceng[ to[ i ] ] = ceng[ sta ] + 1;
                que[ t ++ ] = to[ i ];
            }
    }
    return ceng[ end ] != -1;
}
ll find( ll x , ll low ){
    ll tmp = 0 , result = 0;
    if( x == end ) return low;
    for( ll i = head[ x ] ; ~i && result < low ; i = next[ i ] )
        if( w[ i ] > 0 && ceng[ to[ i ] ] == ceng[ x ] + 1 ){
            tmp = find( to[ i ] , min( w[ i ] , low - result ) );
            w[ i ] -= tmp;
            w[ i^1 ] += tmp;
            result += tmp;
        }
    if( !result ) ceng[ x ] = -1;
    return result;
}
ll dinic(){
    ll ans = 0 , tmp;
    while( bfs() ) ans += find( start , inf );
    return ans;
}
int main(){
    read();
    cout << dinic() << endl;
}
```

---

```
#include <time.h>
#include <stdio.h>
#include <stdlib.h>
#define inf 1023456789
int getint(){
    int x=0,tmp=1; char c=getchar();
    while( (c<'0'||c>'9')&&c!='-' ) c=getchar();
    if( c == '-' ) c=getchar() , tmp=-1;
    while(c>='0'&&c<='9') x*=10,x+=(c-'0'),c=getchar();
    return x*tmp;
}
int max( int x , int y ){ return x>y?x:y; }
struct Treap{
    int lsum , rsum , sum , maxsum;
    int sz , num , val , pri , tag;
    bool tagn; Treap *l , *r;
    Treap( int _val ){
        lsum = rsum = sum = maxsum = val = _val; sz = 1;
        pri = rand(); l = r = NULL; tag = 0; tagn = false;
    }
};
void push( Treap * a ){
    if( a->tagn ){
        a->val = a->num;
        if( a->l ){
            a->l->sum = a->num * a->l->sz;
            if( a->num >= 0 )
                a->l->lsum = a->l->rsum = a->l->maxsum = a->l->sum;
            else a->l->lsum = a->l->rsum = a->l->maxsum = a->num;
            a->l->tagn = true , a->l->num = a->num;
        }
        if( a->r ){
            a->r->sum = a->num * a->r->sz;
            if( a->num >= 0 )
                a->r->lsum = a->r->rsum = a->r->maxsum = a->r->sum;
            else a->r->lsum = a->r->rsum = a->r->maxsum = a->num;
            a->r->tagn = true , a->r->num = a->num;
        }
        a->tagn = false;
    }
    if( a->tag ){
        Treap *swp = a->l; a->l = a->r; a->r = swp;
        int swp2;
        if( a->l ){
            a->l->tag ^= 1;
            swp2 = a->l->lsum; a->l->lsum = a->l->rsum; a->l->rsum = swp2;
        }
        if( a->r ){
            a->r->tag ^= 1;
            swp2 = a->r->lsum; a->r->lsum = a->r->rsum; a->r->rsum = swp2;
        }
        a->tag = 0;
    }
}
int Sum( Treap * a ){ return a ? a->sum : 0; }
int Size( Treap * a ){ return a ? a->sz : 0; }
int lSum( Treap * a ){ return a ? a->lsum : 0; }
int rSum( Treap * a ){ return a ? a->rsum : 0; }
int maxSum( Treap * a ){ return a ? a->maxsum : -inf; }
void pull( Treap * a ){
    a->sum = Sum( a->l ) + Sum( a->r ) + a->val;
    a->lsum = Sum( a->l ) + a->val + max( 0 , lSum( a->r ) );
    if( a->l ) a->lsum = max( lSum( a->l ) , a->lsum );
    a->rsum = Sum( a->r ) + a->val + max( 0 , rSum( a->l ) );
    if( a->r ) a->rsum = max( rSum( a->r ) , a->rsum );
    a->maxsum = max( 0 , rSum( a->l ) ) + a->val + max( 0 , lSum( a->r ) );
    a->maxsum = max( a->maxsum , max( maxSum( a->l ) , maxSum( a->r ) ) );
    a->sz = Size( a->l ) + Size( a->r ) + 1;
}
Treap* merge( Treap *a , Treap *b ){
    if( !a || !b ) return a ? a : b;
    if( a->pri > b->pri ){
        push( a );
        a->r = merge( a->r , b );
        pull( a );
        return a;
    }else{
        push( b );
        b->l = merge( a , b->l );
        pull( b );
        return b;
    }
}
void split( Treap *t , int k , Treap*&a , Treap*&b ){
    if( !t ){ a = b = NULL; return; }
    push( t );
    if( Size( t->l ) + 1 <= k ){
        a = t;
```

```
        split( t->r , k - Size( t->l ) - 1 , a->r , b );
        pull( a );
    }else{
        b = t;
        split( t->l , k , a , b->l );
        pull( b );
    }
}
void show( Treap *t ){
    if( t->l ) show( t->l );
    printf( " %d" , t->val );
    if( t->r ) show( t->r );
}
void Delete( Treap *t ){
    if( t->l ) Delete( t->l );
    if( t->r ) Delete( t->r );
    delete t;
}
char c[ 20 ];
int n , m;
void solve(){
    Treap *t = NULL , *tl = NULL , *tr = NULL;
    n = getint(); m = getint();
    for( int i = 0 ; i < n ; i ++ )
        t = merge( t , new Treap( getint() ) );
    while( m -- ){
        scanf( "%s" , c );
        if( c[ 0 ] == 'I' ){
            int p , k;
            p = getint(); k = getint();
            split( t , p , tl , tr );
            t = NULL;
            while( k -- )
                t = merge( t , new Treap( getint() ) );
            t = merge( t , tr );
            t = merge( tl , t );
        }else if( c[ 0 ] == 'D' ){
            int p , k;
            p = getint(); k = getint();
            split( t , p - 1 , tl , t );
            split( t , k , t , tr );
            Delete( t );
            t = merge( tl , tr );
        }else if( c[ 0 ] == 'R' ){
            int p , k;
            p = getint(); k = getint();
            split( t , p - 1 , tl , t );
            split( t , k , t , tr );
            t->tag ^= 1;
            int swp = t->lsum; t->lsum = t->rsum; t->rsum = swp;
            t = merge( t , tr );
            t = merge( tl , t );
        }else if( c[ 0 ] == 'G' ){
            int p , k;
            p = getint(); k = getint();
            split( t , p - 1 , tl , t );
            split( t , k , t , tr );
            printf( "%d\n" , Sum( t ) );
            t = merge( t , tr );
            t = merge( tl , t );
        }else if( c[ 2 ] == 'K' ){
            int p , k;
            p = getint(); k = getint();
            split( t , p - 1 , tl , t );
            split( t , k , t , tr );
            t->tagn = true; t->num = getint();
            t->sum = t->num * t->sz;
            if( t->num >= 0 )
                t->lsum = t->rsum = t->maxsum = t->sum;
            else t->lsum = t->rsum = t->maxsum = t->num;
            t = merge( t , tr );
            t = merge( tl , t );
        }else printf( "%d\n" , maxSum( t ) );

    }
}
int main(){
    srand( time( 0 ) );
    solve();
}
```