Contents

1 Basic

1.1 .vimrc

```
syn on
se ai nu ru cul mouse=a
se cin et ts=2 sw=2 sts=2
so $VIMRUNTIME/mswin.vim
colo desert
se gfn=Monospace\ 14
```

2 flow

2.1 Dinic

```
#include <bits/stdc++.h>
using namespace std;
#define N 5010
#define M 60010
#define ll long long
#define inf 1ll<<62
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;
}
```

2.2 DMST

```
/*
 * 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);
}
```

2.3 generalWeightedGraphMaxmatching

```
#include <bits/stdc++.h>
using namespace std;
#define N 110
#define inf 0x3f3f3f3f
int G[ N ][ N ], ID[ N ];
int match[ N ], stk[ N ];
int vis[ N ], dis[ N ];
int n, m, k, top;
bool SPFA( int u ){
  stk[ top ++ ] = u;
  if( vis[ u ] ) return true;
  vis[ u ] = true;
  for( int i = 1 ; i <= k ; i ++ ){
    if( i != u && i != match[ u ] && !vis[ i ] ){
      int v = match[ i ];
      if( dis[ v ] < dis[ u ] + G[ u ][ i ] - G[ i ][ v
          ] ){
        dis[ v ] = dis[ u ] + G[ u ][ i ] - G[ i ][ v
            ];
        if( SPFA( v ) ) return true;
      }
    }
  }
  top --; vis[ u ] = false;
  return false;
}
int MaxWeightMatch() {
  for( int i = 1 ; i <= k ; i ++ ) ID[ i ] = i;
  for( int i = 1 ; i <= k ; i += 2 ) match[ i ] = i + 1
      , match[ i + 1 ] = i;
  for( int times = 0 , flag ; times < 3 ; ){
    memset( dis , 0 , sizeof( dis ) );
    memset( vis , 0 , sizeof( vis ) );
    top = 0; flag = 0;
    for( int i = 1 ; i <= k ; i ++ ){
      if( SPFA( ID[ i ] ) ){
        flag = 1;
        int t = match[ stk[ top - 1 ] ] , j = top - 2;
        while( stk[ j ] != stk[ top - 1 ] ){
          match[ t ] = stk[ j ];
          swap( t , match[ stk[ j ] ] );
          j --;
        }
```

```
        match[ t ] = stk[ j ]; match[ stk[ j ] ] = t;
        break;
      }
    }
    if( !flag ) times ++;
    if( !flag ) random_shuffle( ID + 1 , ID + k + 1 );
  }
  int ret = 0;
  for( int i = 1 ; i <= k ; i ++ )
    if( i < match[ i ] ) ret += G[ i ][ match[ i ] ];
  return ret;
}
int main(){
  int T; scanf("%d", &T);
  for ( int cs = 1 ; cs <= T ; cs ++ ){
    scanf( "%d%d%d" , &n , &m , &k );
    memset( G , 0x3f , sizeof( G ) );
    for( int i = 1 ; i <= n ; i ++ ) G[ i ][ i ] = 0;
    for( int i = 0 ; i < m ; i ++ ){
      int u, v, w;
            scanf( "%d%d%d" , &u , &v , &w );
      G[ u ][ v ] = G[ v ][ u ] = w;
    }
    printf( "Case %d: " , cs );
    if( k & 1 ){
      puts( "Impossible" );
      continue;
    }
    for( int tk = 1; tk <= n ; tk ++ )
      for( int i = 1 ; i <= n ; i ++ )
        for( int j = 1 ; j <= n ; j ++ )
          G[ i ][ j ] = min( G[ i ][ j ] , G[ i ][ tk ]
              + G[ tk ][ j ] );
    for( int i = 1 ; i <= k ; i ++ ){
      for( int j = 1 ; j <= k ; j ++ )
        G[ i ][ j ] = -G[ i ][ j ];
      G[ i ][ i ] = -inf;
    }
    printf( "%d\n" , -MaxWeightMatch() );
  }
}
```

2.4 ISAP

```
#include <bits/stdc++.h>
#define SZ(c) ((int)(c).size())
using namespace std;
struct Maxflow {
  static const int MAXV = 20010;
  static const int INF  = 1000000;
  struct Edge {
    int v, c, r;
    Edge(int _v, int _c, int _r) : v(_v), c(_c), r(_r)
        {}
  };
  int s, t;
  vector<Edge> G[MAXV*2];
  int iter[MAXV*2], d[MAXV*2], gap[MAXV*2], tot;
  void flowinit(int x) {
    tot = x+2;
    s = x+1, t = x+2;
    for(int i = 0; i <= tot; i++) {
      G[i].clear();
      iter[i] = d[i] = gap[i] = 0;
    }
  }
  void addEdge(int u, int v, int c) {
    G[u].push_back(Edge(v, c, SZ(G[v]) ));
    G[v].push_back(Edge(u, 0, SZ(G[u]) - 1));
  }
  int dfs(int p, int flow) {
    if(p == t) return flow;
    for(int &i = iter[p]; i < SZ(G[p]); i++) {
      Edge &e = G[p][i];
      if(e.c > 0 && d[p] == d[e.v]+1) {
        int f = dfs(e.v, min(flow, e.c));
        if(f) {
          e.c -= f;
          G[e.v][e.r].c += f;
          return f;
        }
      }
    }
    if( (--gap[d[p]]) == 0) d[s] = tot;
    else {
      d[p]++;
      iter[p] = 0;
      ++gap[d[p]];
    }
    return 0;
  }

  int maxflow() {
    //puts("MF");
    int res = 0;
    gap[0] = tot;
    for(res = 0; d[s] < tot; res += dfs(s, INF));
    return res;
  }
} flow;

Maxflow::Edge e(1, 1, 1);
```

2.5 MinCostFlow

```
/*
  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;
```

3 Math

3.1 FFT

```
#include <bits/stdc++.h>
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 & 0
      xAAAAAAAAU ) >> 1 ) ;
  a = ( ( a & 0x33333333U ) << 2 ) | ( ( a & 0
      xCCCCCCCCU ) >> 2 ) ;
  a = ( ( a & 0x0F0F0F0FU ) << 4 ) | ( ( a & 0
      xF0F0F0F0U ) >> 4 ) ;
  a = ( ( a & 0x00FF00FFU ) << 8 ) | ( ( a & 0
      xFF00FF00U ) >> 8 ) ;
  a = ( ( a & 0x0000FFFFU ) << 16 ) | ( ( a & 0
      xFFFF0000U ) >> 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(){
  while ( work() ) ;
}
```

3.2 NTT

```
ll P=2013265921,root=31;
int MAXNUM=4194304;
// Remember coefficient are mod P
/*
p=a*2^n+1
n   2^n         p           a     root
5   32          97          3     5
6   64          193         3     5
7   128         257         2     3
8   256         257         1     3
9   512         7681        15    17
10  1024        12289       12    11
11  2048        12289       6     11
12  4096        12289       3     11
13  8192        40961       5     3
14  16384       65537       4     3
15  32768       65537       2     3
16  65536       65537       1     3
17  131072      786433      6     10
18  262144      786433      3     10 (605028353,
    2308, 3)
```

```
19  524288      5767169      11    3
20  1048576     7340033      7     3
21  2097152     23068673     11    3
22  4194304     104857601    25    3
23  8388608     167772161    20    3
24  16777216    167772161    10    3
25  33554432    167772161    5     3 (1107296257, 33,
    10)
26  67108864    469762049    7     3
27  134217728   2013265921   15    31
*/
ll bigmod(ll a,ll b){
  if(b==0)return 1;
  return (bigmod((a*a)%P,b/2)*(b%2?a:1ll))%P;
}
ll inv(ll a,ll b){
  if(a==1)return 1;
  return (((long long)(a-inv(b%a,a))*b+1)/a)%b;
}
std::vector<ll> ps(MAXNUM);
std::vector<ll> rev(MAXNUM);
struct poly{
  std::vector<ll> co;
  int n;//polynomial degree = n
  poly(int d){n=d;co.resize(n+1,0);}
  void trans2(int NN){
    int r=0,st,N;
    unsigned int a,b;
    while((1<<r)<(NN>>1))++r;
    for(N=2;N<=NN;N<<=1,--r){
      for(st=0;st<NN;st+=N){
        int i,ss=st+(N>>1);
        for(i=(N>>1)-1;i>=0;--i){
          a=co[st+i]; b=(ps[i<<r]*co[ss+i])%P;
          co[st+i]=a+b; if(co[st+i]>=P)co[st+i]-=P;
          co[ss+i]=a+P-b; if(co[ss+i]>=P)co[ss+i]-=P;
        }
      }
    }
  }
  void trans1(int NN){
    int r=0,st,N;
    unsigned int a,b;
    for(N=NN;N>1;N>>=1,++r){
      for(st=0;st<NN;st+=N){
        int i,ss=st+(N>>1);
        for(i=(N>>1)-1;i>=0;--i){
          a=co[st+i]; b=co[ss+i];
          co[st+i]=a+b; if(co[st+i]>=P)co[st+i]-=P;
          co[ss+i]=((a+P-b)*ps[i<<r])%P;
        }
      }
    }
  }
  poly operator*(const poly& _b)const{
    poly a=*this,b=_b;
    int k=n+b.n,i,N=1;
    while(N<=k)N*=2;
    a.co.resize(N,0); b.co.resize(N,0);
    int r=bigmod(root,(P-1)/N),Ni=inv(N,P);
    ps[0]=1;
    for(i=1;i<N;++i)ps[i]=(ps[i-1]*r)%P;
    a.trans1(N);b.trans1(N);
    for(i=0;i<N;++i)a.co[i]=((long long)a.co[i]*b.co[i
        ])%P;
    r=inv(r,P);
    for(i=1;i<N/2;++i)std::swap(ps[i],ps[N-i]);
    a.trans2(N);
    for(i=0;i<N;++i)a.co[i]=((long long)a.co[i]*Ni)%P;
    a.n=n+_b.n; return a;
  }
};
```

4 Geometry

4.1 halfPlaneIntersection

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

#define N 100010
#define EPS 1e-8
#define SIDE 10000000

using namespace std;

struct PO {
  double x , y ;
} p[ N ] , o ;

struct LI {
  PO a , b ;
  double angle ;
  void in( double x1 , double y1 , double x2 , double
      y2 ) {
    a.x = x1 ; a.y = y1 ; b.x = x2 ; b.y = y2 ;
  }
} li[ N ] , deq[ N ] ;

int n , m , cnt ;

inline int dc( double x ) {
  if ( x > EPS ) return 1 ;
  else if ( x < -EPS ) return -1 ;
  return 0 ;
}

inline PO operator - ( PO a , PO b ) {
  PO c ;
  c.x = a.x - b.x ; c.y = a.y - b.y ;
  return c ;
}

inline double cross( PO a , PO b , PO c ) {
  return ( b.x - a.x ) * ( c.y - a.y ) - ( b.y - a.y )
      * ( c.x - a.x ) ;
}

inline bool cmp( const LI &a , const LI &b ) {
  if( dc( a.angle - b.angle ) == 0 ) return dc( cross(
      a.a , a.b , b.a ) ) < 0 ;
  return a.angle > b.angle ;
}

inline PO getpoint( LI &a , LI &b ) {
  double k1 = cross( a.a , b.b , b.a ) ;
  double k2 = cross( a.b , b.a , b.b ) ;
  PO tmp = a.b - a.a , ans ;
  ans.x = a.a.x + tmp.x * k1 / ( k1 + k2 ) ;
  ans.y = a.a.y + tmp.y * k1 / ( k1 + k2 ) ;
  return ans ;
}
inline void getcut() {
  sort( li + 1 , li + 1 + n , cmp ) ; m = 1 ;
  for ( int i = 2 ; i <= n ; i ++ )
    if ( dc( li[ i ].angle - li[ m ].angle ) != 0 )
      li[ ++ m ] = li[ i ] ;
  deq[ 1 ] = li[ 1 ] ; deq[ 2 ] = li[ 2 ] ;
  int bot = 1 , top = 2 ;
  for ( int i = 3 ; i <= m ; i ++ ) {
    while ( bot < top && dc( cross( li[ i ].a , li[ i
        ].b , getpoint( deq[ top ] , deq[ top - 1 ] ) )
        ) < 0 ) top -- ;
    while ( bot < top && dc( cross( li[ i ].a , li[ i
        ].b , getpoint( deq[ bot ] , deq[ bot + 1 ] ) )
        ) < 0 ) bot ++ ;
    deq[ ++ top ] = li[ i ] ;
  }
  while ( bot < top && dc( cross( deq[ bot ].a , deq[
      bot ].b , getpoint( deq[ top ] , deq[ top - 1 ] )
      ) ) < 0 ) top -- ;
  while ( bot < top && dc( cross( deq[ top ].a , deq[
      top ].b , getpoint( deq[ bot ] , deq[ bot + 1 ] )
      ) ) < 0 ) bot ++ ;
```

```
  cnt = 0 ;
  if ( bot == top ) return ;
  for ( int i = bot ; i < top ; i ++ ) p[ ++ cnt ] =
      getpoint( deq[ i ] , deq[ i + 1 ] ) ;
  if ( top - 1 > bot ) p[ ++ cnt ] = getpoint( deq[ bot
      ] , deq[ top ] ) ;
}

double px[ N ] , py[ N ] ;
void read( int rm ) {
  for( int i = 1 ; i <= n ; i ++ ) px[ i + n ] = px[ i
      ] , py[ i + n ] = py[ i ];
  for( int i = 1 ; i <= n ; i ++ ) {
    // half-plane from li[ i ].a -> li[ i ].b
    li[ i ].a.x = px[ i + rm + 1 ]; li[ i ].a.y = py[ i
        + rm + 1 ];
    li[ i ].b.x = px[ i ]; li[ i ].b.y = py[ i ];
    li[ i ].angle = atan2( li[ i ].b.y - li[ i ].a.y ,
        li[ i ].b.x - li[ i ].a.x ) ;
  }
}

inline double getarea( int rm ) {
  read( rm ) ; getcut( ) ;
  double res = 0.0 ;
  p[ cnt + 1 ] = p[ 1 ] ;
  for ( int i = 1 ; i <= cnt ; i ++ ) res += cross( o ,
      p[ i ] , p[ i + 1 ] ) ;
  if( res < 0.0 ) res *= -1.0;
  return res;
}

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

5 Graph

5.1 HeavyLightDecomp

```
#include <bits/stdc++.h>
using namespace std;
#define SZ(c) (int)(c).size()
#define ALL(c) (c).begin(), (c).end()
#define REP(i, s, e) for(int i = (s); i <= (e); i++)
#define REPD(i, s, e) for(int i = (s); i >= (e); i--)

typedef tuple< int , int > tii;

const int MAXN = 100010;
const int LOG  = 19;

struct HLD
{
  int n;
  vector<int> g[MAXN];
  int sz[MAXN], dep[MAXN];
  int ts, tid[MAXN], tdi[MAXN], tl[MAXN], tr[MAXN];
  //  ts : timestamp , useless after yutruli
  //  tid[ u ] : pos. of node u in the seq.
  //  tdi[ i ] : node at pos i of the seq.
  //  tl , tr[ u ] : subtree interval in the seq. of
      node u
  int mom[MAXN][LOG], head[MAXN];
  // head[ u ] : head of the chain contains u
  void dfssz(int u, int p)
  {
    dep[u] = dep[p] + 1;
    mom[u][0] = p;
    sz[u] = 1;
    head[u] = u;
    for(int& v:g[u]) if(v != p)
    {
      dep[v] = dep[u] + 1;
      dfssz(v, u);
      sz[u] += sz[v];
    }
  }
  void dfshl(int u)
  {
    //printf("dfshl %d\n", u);
    ts++;
    tid[u] = tl[u] = tr[u] = ts;
    tdi[tid[u]] = u;
    sort(ALL(g[u]), [&](int a, int b){return sz[a] > sz
        [b];});
    bool flag = 1;
    for(int& v:g[u]) if(v != mom[u][0])
    {
      if(flag) head[v] = head[u], flag = 0;
      dfshl(v);
      tr[u] = tr[v];
    }
  }
  inline int lca(int a, int b)
  {
    if(dep[a] > dep[b]) swap(a, b);
    //printf("lca %d %d\n", a, b);
    int diff = dep[b] - dep[a];
    REPD(k, LOG-1, 0) if(diff & (1<<k))
    {
      //printf("b %d\n", mom[b][k]);
      b = mom[b][k];
    }
    if(a == b) return a;
    REPD(k, LOG-1, 0) if(mom[a][k] != mom[b][k])
    {
      a = mom[a][k];
      b = mom[b][k];
    }
    return mom[a][0];
  }
  void init( int _n )
  {
    n = _n;
    REP( i , 1 , n ) g[ i ].clear();
  }
  void addEdge( int u , int v )
  {
    g[ u ].push_back( v );
    g[ v ].push_back( u );
  }
  void yutruli()
  {
    dfssz(1, 0);
    ts = 0;
    dfshl(1);
    REP(k, 1, LOG-1) REP(i, 1, n)
      mom[i][k] = mom[mom[i][k-1]][k-1];
  }
  vector< tii > getPath( int u , int v )
  {
    vector< tii > res;
    while( tid[ u ] < tid[ head[ v ] ] )
    {
      res.push_back( tii( tid[ head[ v ] ] , tid[ v ] )
          );
      v = mom[ head[ v ] ][ 0 ];
    }
    res.push_back( tii( tid[ u ] , tid[ v ] ) );
    reverse( ALL( res ) );
    return res;
    /*
     * res : list of intervals from u to v
     * u must be ancestor of v
     * usage :
     * vector< tii >& path = tree.getPath( u , v )
     * for( tii tp : path ) {
     *   int l , r;tie( l , r ) = tp;
     *   upd( l , r );
     *   uu = tree.tdi[ l ] , vv = tree.tdi[ r ];
     *   uu ~> vv is a heavy path on tree
     * }
     */
  }
} tree;
```

5.2 MaxClique

```
#include <bits/stdc++.h>
using namespace std;
#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 , 0
        LLU , 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();
    }
}
```

6 String

6.1 PalTree

```
const int MAXN = 200010;

struct PalT{
    struct Node{
        int nxt[ 33 ] , len , fail;
        ll cnt;
    };
    int tot , lst;
    Node nd[ MAXN * 2 ];
    char* s;
    int newNode( int l , int _fail ){
        int res = ++tot;
        memset( nd[ res ].nxt , 0 , sizeof nd[ res ].
            nxt );
        nd[ res ].len = l;
        nd[ res ].cnt = 0;
        nd[ res ].fail = _fail;
        return res;
    }
    void push( int p ){
        int np = lst;
        int c = s[ p ] - 'a';
        while( p - nd[ np ].len - 1 < 0
            || s[ p ] != s[ p - nd[ np ].len - 1 ]
        )
            np = nd[ np ].fail;

        if( nd[ np ].nxt[ c ] ){
            nd[ nd[ np ].nxt[ c ] ].cnt++;
            lst = nd[ np ].nxt[ c ];
            return ;
        }

        int nq = newNode( nd[ np ].len + 2 , 0 );
        nd[ nq ].cnt++;
        nd[ np ].nxt[ c ] = nq;
        lst = nq;
        if( nd[ nq ].len == 1 ){
            nd[ nq ].fail = 2;
            return ;
        }

        int tf = nd[ np ].fail;
        while( p - nd[ tf ].len - 1 < 0
            || s[ p ] != s[ p - nd[ tf ].len - 1 ]
        )
            tf = nd[ tf ].fail;

        nd[ nq ].fail = nd[ tf ].nxt[ c ];
        return ;
    }
    void init( char* _s ){
        s = _s;
        tot = 0;
        newNode( -1 , 1 );
        newNode( 0 , 1 );
        lst = 2;
        for( int i = 0 ; s[ i ] ; i++ )
            push( i );
    }
    void yutruli(){
#define REPD(i, s, e) for(int i = (s); i >= (e); i--)
        REPD( i , tot , 1 )
            nd[ nd[ i ].fail ].cnt += nd[ i ].cnt;
        nd[ 1 ].cnt = nd[ 2 ].cnt = 0ll;
    }
} pA;

int main(){
    pA.init( sa );
}
```

6.2 SuffixArray

```
#include <bits/stdc++.h>
using namespace std;
#define N 100010
char T[ N ];
int n , RA[ N ], tempRA[ N ] , SA[ N ], tempSA[ N ] , 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;
}
```

6.3 SuffixAutomata

```
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;
```

7 Data Structrue

7.1 Treap

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