Team Notebook

Nimroo

September 25, 2021

Contents

1 Combinatorics

1.1 dummy

```
#include<bits/stdc++.h>
using namespace std;
const int MAX=1e6+9;
int main()
{
 ios_base::sync_with_stdio(false),cin.tie(0);
 return 0;
}
```

2 DataStructures

2.1 Dynamic Convex Hull

```
#include<bits/stdc++.h>
typedef long long ll;
typedef long double ld;
using namespace std;
const ld is_query = -(1LL << 62);
struct Line {
 ld m, b;
 mutable std::function<const Line *()> succ;
 bool operator<(const Line &rhs) const {
  if (rhs.b != is_query) return m < rhs.m;
  const Line *s = succ();
  if (!s) return 0;
  ld x = rhs.m;
  return b - s->b < (s->m - m) * x;
 }
};
struct HullDynamic : public multiset<Line> { // dynamic
     upper hull + max value query
 bool bad(iterator y) {
  auto z = next(y);
  if (y == begin()) {
   if (z == end()) return 0;
   return y->m == z->m && y->b <= z->b;
  }
  auto x = prev(y);
  if (z == end()) return y->m == x->m && y->b <= x->b;
  return (x->b - y->b) * (z->m - y->m) >= (y->b - z->b) * (y
       ->m - x->m);
 }
 void insert_line(ld m, ld b) {
  auto y = insert({m, b});
  y->succ = [=] { return next(y) == end() ? 0 : &*next(y);
       };
  if (bad(y)) {
   erase(y);
   return;
  }
  while (next(y) != end() && bad(next(y))) erase(next(y));
  while (y != begin() && bad(prev(y))) erase(prev(y));

 }
 ld best(ld x) {
  auto l = *lower_bound((Line) {x, is_query});
  return l.m * x + l.b;
 }
};
int main()
{
 ios_base::sync_with_stdio(false),cin.tie(0);

 return 0;
}
```

2.2 Fenwik

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
typedef long double ld;
const int MAX=1e6+9;

int fen[MAX];
void upd(int x,int val)
{
 ++x;
 for (;x<MAX;x+=(x&-x)) fen[x]+=val;
}
int que(int x)
{
 ++x;
 int res=0;
 for (;x;x-=(x&-x))
  res+=fen[x];
 return res;
}
int main()
{
 ios_base::sync_with_stdio(false),cin.tie(0);
 upd(5,10);
 upd(7,9);
 cout<<que(8);
 return 0;
}
```

2.3 Heavy Light

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
typedef long double ld;
const int N = 2000*100 + 10;
const int L = 20;
int par[N][L], h[N], fath[N], st[N], en[N], sz[N];
vector<int> c[N]; //Adjacency List
int dsz(int s, int p) {
 sz[s] = 1;
 for(int xt = 0; xt < (int)c[s].size(); xt++) {
  int x = c[s][xt];
  if( x != p ) {
   sz[s] += dsz( x , s );
   if( sz[x] > sz[c[s][0]] )
    swap( c[s][0], c[s][xt] );
  }
 }
 return sz[s];
}
void dfs(int s, int p) {
 static int ind = 0;
 st[s] = ind++;
 for(int k = 1; k < L; k++)
  par[s][k] = par[par[s][k-1]][k-1];
 for(int xt = 0; xt < (int)c[s].size(); xt++) {
  int x = c[s][xt];
  if( x == p ) continue;
  fath[x] = x;
  if( xt == 0 ) fath[x] = fath[s];
  h[x] = h[s] + 1;
  par[x][0] = s;
  dfs(x, s);
 }
 en[s] = ind;
}
int n, q;
void upset(int u, int w, int qv) {
 int stL = max( st[w] , st[fath[u]] );
 set( stL, st[u] + 1 , qv , 0, n , 1 ); //l,r,val,s,e,id
 if( stL == st[w] ) return;
 upset( par[fath[u]][0] , w , qv );
```

```
}

int main()
{
 ios_base::sync_with_stdio(false),cin.tie(0);

 return 0;
}
```

2.4 Implicit Treap

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
typedef long double ld;
const int MAX=1e6+9;

typedef struct item * pitem;
struct item {
 int prior, value, cnt;
 bool rev;
 pitem l, r;
};

int cnt (pitem it) {
 return it ? it->cnt : 0;
}
void upd_cnt (pitem it) {
 if (it)
  it->cnt = cnt(it->l) + cnt(it->r) + 1;
}
void push (pitem it) {
 if (it && it->rev) {
  it->rev = false;
  swap (it->l, it->r);
  if (it->l) it->l->rev ^= true;
  if (it->r) it->r->rev ^= true;
 }
}
void merge (pitem & t, pitem l, pitem r) {
 push (l);
 push (r);
 if (!l || !r)
  t = l ? l : r;
 else if (l->prior > r->prior)
  merge (l->r, l->r, r), t = l;
 else
  merge (r->l, l, r->l), t = r;
 upd_cnt (t);
}

void split (pitem t, pitem & l, pitem & r, int key, int add
    = 0) {
 if (!t)
  return void( l = r = 0 );
 push (t);
 int cur_key = add + cnt(t->l);
 if (key <= cur_key)
  split (t->l, l, t->l, key, add), r = t;
 else
  split (t->r, t->r, r, key, add + 1 + cnt(t->l)), l = t;
 upd_cnt (t);
}
void reverse (pitem t, int l, int r) {
 pitem t1, t2, t3;
 split (t, t1, t2, l);
 split (t2, t2, t3, r-l+1);
 t2->rev ^= true;
 merge (t, t1, t2);
 merge (t, t, t3);
}
void output (pitem t) {
 if (!t) return;
 push (t);
 output (t->l);
 printf ("%d ", t->value);
 output (t->r);
}

int main()
{
 ios_base::sync_with_stdio(false),cin.tie(0);
 return 0;
}
```

2.5 Ordered Set

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp> // Common file
#include <ext/pb_ds/tree_policy.hpp> // Including
    tree_order_statistics_node_update
using namespace std;
using namespace __gnu_pbds;
typedef long long ll;
typedef long double ld;
const int MAX=1e6+9;
typedef tree<int,null_type,less<int>,rb_tree_tag,
    tree_order_statistics_node_update> ordered_set;

int main()
{
 ios_base::sync_with_stdio(false),cin.tie(0);
 ordered_set X;
 X.insert(1);
 X.insert(2);
 X.insert(4);
 X.insert(8);
 X.insert(16);

 cout<<*X.find_by_order(1)<<endl; // 2
 cout<<*X.find_by_order(2)<<endl; // 4
 cout<<*X.find_by_order(4)<<endl; // 16
 cout<<(end(X)==X.find_by_order(6))<<endl; // true

 cout<<X.order_of_key(-5)<<endl; // 0
 cout<<X.order_of_key(1)<<endl; // 0
 cout<<X.order_of_key(3)<<endl; // 2
 cout<<X.order_of_key(4)<<endl; // 2
 cout<<X.order_of_key(400)<<endl; // 5
 return 0;
}
```

2.6 Seg Lazy

```
#include<bits/stdc++.h>
using namespace std;
const int MAX=1e6+100;
struct node{
 int val,lazy;
}seg[MAX*4];
int n,q;
void merge(int id, int left, int right)
{
 seg[id].val=seg[left].val+seg[right].val;
}
void build(int s=0,int e=n,int id=0)
{
 seg[id].lazy=0;
 if (e-s==1)
 {
  seg[id].val=0;
  return ;
 }
 int mid=(s+e)>>1;
 build(s,mid,id*2+1),build(mid,e,id*2+2);
```

```
 merge(id,id*2+1,id*2+2);
}
void shift(int id)
{
 if (seg[id].lazy)
 {
  seg[id*2+1].val+=seg[id].lazy;
  seg[id*2+2].val+=seg[id].lazy;
  seg[id].lazy=0;
 }
}
void update(int l,int r,int val, int s=0,int e=n,int id=0)
{
 if (e<=l || r<=s) return ;
 if (l<=s && e<=r)
 {
  seg[id].val+=val;
  seg[id].lazy+=val;
  return ;
 }
 int mid=(s+e)>>1;
 shift(id);
 update(l,r,val,s,mid,id*2+1);
 update(l,r,val,mid,e,id*2+2);
 merge(id,id*2+1,id*2+2);
}
int main()
{
 ios_base::sync_with_stdio(false),cin.tie(0),cout.tie(0);
 return 0;
}
```

2.7 Seg Persistant

```
#include<bits/stdc++.h>
using namespace std;
typedef pair<pair<int,int>,int > ANS;
#define MX second
#define LE first.first
#define RI first.second

const int MAXN=1e5+9,LOG=22;

int root[MAXN], le[LOG * MAXN * 2], ri[LOG * MAXN * 2], sz,
     lleft[LOG * MAXN * 2], rright[LOG * MAXN * 2];
int maxi[LOG * MAXN * 2];
int n, q;
int h[MAXN], vec[MAXN];
pair<int, int> sec[MAXN];
int build(int b, int e){
 int id = sz++;
 if (e - b == 1) return
  id;
 int mid = (b + e)/ 2;
 le[id] = build(b, mid);
 ri[id] = build(mid, e);
 return id;
}
void merge(int id, int b, int e, int mid){
 maxi[id] = max(maxi[le[id]], maxi[ri[id]]);
 maxi[id] = max(maxi[id], rright[le[id]] + lleft[ri[id]]);
 lleft[id] = lleft[le[id]];
 if (lleft[id] == (mid - b))
  lleft[id] += lleft[ri[id]];
 rright[id] = rright[ri[id]];
 if (rright[id] == (e - mid)) rright[id] += rright[le[id]];
}
int modify(int id, int b, int e, int x){
 int nid = sz++;
 if (e - b == 1){
  lleft[nid] = rright[nid] = maxi[nid] = 1;
  return nid;
 }
 int mid =(b+e)/2;
 le[nid]=le[id];
 ri[nid]=ri[nid];
 if (x<mid)
  le[nid]=modify(le[nid],b,mid,x);
 else
  ri[nid] = modify(ri[nid], mid , e, x);
 merge(nid,b,e,mid);
 return nid;
}
ANS mg(ANS a, ANS b, int s1, int s2){
 ANS ret;
 ret.MX = max(a.MX, b.MX);
 ret.MX = max(ret.MX, a.RI + b.LE);
 ret.LE = a.LE;
 if (a.LE == s1) ret.LE += b.LE;
 ret.RI = b.RI;
 if (b.RI == s2) ret.RI += a.RI;
 return ret;
}
ANS get(int id, int b, int e, int l, int r){
 if (l <= b && e <= r)
  return {{maxi[id], lleft[id]}, rright[id]};
 if (r <= b || e <= l)
  return {{0, 0}, 0};
 int mid = (b + e)/ 2;
 return mg(get(le[id], b, mid, l, r), get(ri[id], mid, e, l,
       r
    ), min(mid - b, max(0, mid - l)), min(e - mid, max(0, r -
      mid)));
}


void init(){
 copy(h, h + n, vec);
 sort(vec, vec + n);
 reverse(vec, vec + n);
 for (int i = 0; i < n; i++)
  sec[i] = {h[i], i};
 sort(sec, sec + n);
 reverse(sec, sec + n);
 root[0] = build(0, n);
 for (int i = 0; i < n; i++)
  root[i + 1] = modify(root[i], 0, n, sec[i].second);
}
int main(){
 ios::sync_with_stdio(false);
 cin.tie(0);
 cin >> n;
 for (int i = 0; i < n; i++)
  init();
 cin >> q;
 while (q--){
  int l, r, w;
  cin >> l >> r >> w;
  l--;
  int b = 0, e = n, mid, ret = n;
  while (b <= e){
   mid = (b + e)/ 2;
   if (get(root[mid], 0, n, l, r).MX >= w){
    ret = mid;
    e = mid - 1;
   }
   else
    b = mid + 1;
  }
  cout << vec[ret - 1] << "\n";
 }
 return 0;
}
```

2.8 Treap

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

```
using namespace std;
typedef long long ll;
typedef long double ld;
const int MAX=1e6+9;
struct item {
 int key, prior;
 item * l, * r;
 item() { }
 item (int key, int prior) : key(key), prior(prior), l(NULL)
      , r(NULL) { }
};
typedef item * pitem;
void split (pitem t, int key, pitem & l, pitem & r) {
 if (!t)
  l = r = NULL;
 else if (key < t->key)
  split (t->l, key, l, t->l), r = t;
 else
  split (t->r, key, t->r, r), l = t;
}
void insert (pitem & t, pitem it) {
 if (!t)
  t = it;
 else if (it->prior > t->prior)
  split (t, it->key, it->l, it->r), t = it;
 else
  insert (it->key < t->key ? t->l : t->r, it);
}
void merge (pitem & t, pitem l, pitem r) {
 if (!l || !r)
  t = l ? l : r;
 else if (l->prior > r->prior)
  merge (l->r, l->r, r), t = l;
 else
  merge (r->l, l, r->l), t = r;
}
void erase (pitem & t, int key) {
 if (t->key == key)
  merge (t, t->l, t->r);
 else
  erase (key < t->key ? t->l : t->r, key);
}
pitem unite (pitem l, pitem r) {
 if (!l || !r) return l ? l : r;
 if (l->prior < r->prior) swap (l, r);
 pitem lt, rt;
 split (r, l->key, lt, rt);
 l->l = unite (l->l, lt);
 l->r = unite (l->r, rt);
 return l;
}
pitem root = NULL;
int main()
{
 ios_base::sync_with_stdio(false),cin.tie(0);
 item a = item(10,20);
 item b = item(10,20);
 insert(root, &a);
 insert(root, &b);
 return 0;
}
```

3 DpOptimizations

3.1 dummy

4 Geometry

4.1 Primitives

```
#include<bits/stdc++.h>
using namespace std;
typedef long double ld;
typedef complex<ld> pt;
typedef vector<pt> poly;
#define x real()
#define y imag()

typedef pair<pt, pt> line;
// +, -, * scalar well defined
const ld EPS = 1e-12;
const ld PI = acos(-1);
const int ON = 0, LEFT = 1, RIGHT = -1, BACK = -2, FRONT =
     2, IN = 3, OUT = -3;

inline bool Lss(ld a, ld b){ return a - b < -EPS; }
inline bool Grt(ld a, ld b){ return a - b > +EPS; }
inline bool Leq(ld a, ld b){ return a - b < +EPS; }
inline bool Geq(ld a, ld b){ return a - b > -EPS; }
inline bool Equ(ld a, ld b){ return abs(a-b) < EPS; }

bool byX(const pt &a, const pt &b)
{
 if (Equ(a.x, b.x)) return Lss(a.y, b.y);
 return Lss(a.x, b.x);
}
bool byY(const pt &a, const pt &b){
 if (Equ(a.y, b.y)) return Lss(a.x, b.x);
 return Lss(a.y, b.y);
}
struct cmpXY{ inline bool operator ()(const pt &a, const pt
     &b)const { return byX(a, b); } };
struct cmpYX{ inline bool operator ()(const pt &a, const pt
     &b)const { return byY(a, b); } };
bool operator < (const pt &a, const pt &b){ return byX(a, b)
     ; }

istream& operator >> (istream& in, pt p){ld valx,valy; in>>
     valx>>valy; p={valx,valy}; return in;}
ostream& operator << (ostream& out, pt p){out<<p.x<<' '<<p.y
     ; return out;}

ld dot(pt a, pt b){return (conj(a) * b).x;}
ld cross(pt a, pt b){return (conj(a) * b).y;}
ld disSQ(pt a, pt b){return norm(a - b);}
ld dis(pt a, pt b){return abs(a - b);}
ld angleX(pt a, pt b){return arg(b - a);}
ld slope(pt a, pt b){return tan(angleX(a,b));}
//polar(r,theta) -> cartesian
pt rotate(pt a, ld theta){return a * polar((ld)1, theta);}
pt rotatePiv(pt a, ld theta, pt piv){return (a - piv) *
     polar((ld)1, theta) + piv;}
ld angleABC(pt a, pt b, pt c){return abs(remainder(arg(a-b)
     - arg(c-b), 2.0 * PI));}
pt proj(pt p, pt v){return v * dot(p,v) / norm(v);}
pt projPtLine(pt a, line l){return proj(a - l.first,l.second
     -l.first)+l.first;}
ld disPtLine(pt p, line l){return dis(p-l.first, proj(p-l.
     first,l.second-l.first));}

int relpos(pt a, pt b, pt c) //c to a-b
{
 b = b-a, c= c-a;
 if (Grt(cross(b,c), 0)) return LEFT;
 if (Lss(cross(b,c), 0)) return RIGHT;
 if (Lss(dot(b,c), 0)) return BACK;
 if (Lss(dot(b,c), abs(b))) return FRONT;
 return ON;
}
int relpos(line l, pt b){return relpos(l.first, l.second, b)
     ;}

pair<pt,bool> intersection(line a, line b)
{
```

```
 ld c1 = cross(b.first - a.first, a.second - a.first);
 ld c2 = cross(b.second - a.first, a.second - a.first);
 if (Equ(c1,c2))
  return {{-1,-1},false};
 return {(c1 * b.second - c2 * b.first) / (c1 - c2), true};
}
bool intersect(line a, line b)
{
 pair<pt, bool> ret = intersection(a,b);
 if (!ret.second) return false;
 if (relpos(a, ret.first) == ON and relpos(b, ret.first) ==
      ON)
  return true;
 return false;
}
bool isconvex(poly &pl)
{
 int n = pl.size();
 bool neg = false, pos = false;
 for (int i=0;i<n;i++)
 {
  int rpos = relpos(pl[i], pl[(i+1)%n], pl[(i+2)%n]);
  if (rpos == LEFT) pos = true;
  if (rpos == RIGHT) neg = true;
 }
 return !(neg&pos);
}
int crossingN(poly &pl, pt a)
{
 int n = pl.size();
 pt b = a;
 for (pt p:pl)
  b.real(max(b.x,p.y));
 int cn = 0;
 for (int i=0;i<n;i++)
 {
  pt p = pl[i], q=pl[(i+1)%n];
  if (intersect({a,b},{p,q}) && (relpos({p,q},a)!= RIGHT ||
       relpos({p,q},b) != RIGHT))
   cn ++;
 }
 return cn;
}
int pointInPoly(poly &pl, pt p)
{
 int n = pl.size();
 for (int i=0;i<n;i++)
  if (relpos(pl[i], pl[(i+1)%n], p) == ON)
   return ON;
 return crossingN(pl,p)%2? IN : OUT;
}

poly getHull(poly &pl, bool lower)
{
 sort(pl.begin(), pl.end(), byX);
 poly res;
 int n = res.size();
 for (auto p : pl)
 {
  while (n >= 2 && relpos(res[n-2], res[n-1], p) == (lower?
       RIGHT : LEFT))
   res.pop_back(), n--;
  res.push_back(p), n++;
 }
 return res;
}

pair<pt, pt> nearestPair(poly &pl)
{
 int n = pl.size();
 sort(pl.begin(), pl.end(), byX);
 multiset<pt, cmpYX> s;
 ld rad = abs(pl[1] - pl[0]);
 pair<pt, pt> res = {pl[0], pl[1]};
 int l = 0, r = 0;
 for (int i=0;i<n;i++)
 {
  while (l<r && Geq(pl[i].x - pl[l].x, rad))
   s.erase(pl[l++]);
  while (r<l && Leq(pl[r].x, pl[i].x))
   s.insert(pl[r++]);
  for (auto it = s.lower_bound(pt(pl[i].x, pl[i].y-rad)); it
       != s.end(); it++)
  {
   if (Grt(it->y, pl[i].y+rad))
    break;
   ld cur = abs(pl[i] - (*it));
   if (Lss(cur, rad))
    rad = cur, res = {*it, pl[i]};
  }
 }
 return res;
}

typedef struct circle{
 pt c;
 ld r;
} cir;

//number of common tangent lines
int tangentCnt(cir c1, cir c2)
{
 ld d= abs(c1.c-c2.c);
 if (Grt(d, c1.r+c2.r)) return 4; //outside
 if (Equ(d, c1.r+c2.r)) return 3; //tangent outside
 if (Lss(d, c1.r+c2.r) && Grt(d, abs(c1.r-c2.r))) return 2;
      //interfere
 if (Equ(d, abs(c1.r-c2.r))) return 1; //tangent inside
 return 0;//inside
}

line intersection(line l, cir c)
{
 ld dis = disPtLine(c.c, l);
 ld d = sqrt(c.r*c.r - dis*dis);
 pt p = projPtLine(c.c, l);
 pt vec = (l.second-l.first)/abs(l.second - l.first);
 return {p + d * vec, p - d * vec};
}

/*
   0 = other is inside this, zero point
   1 = other is tangent inisde of this, one point
   2 = other is intersect with this, two point
   3 = other is tangent outside of this, one point
   4 = other is outside of this, zero point
 */
pair<int, vector<pt> > intersect(cir c, cir other) {
 ld r = c.r;
 pt o = c.c;
 vector<pt> v;
 ld sumr = other.r + r;
 ld rr = r - other.r;
 ld d = dis(o, other.c);
 ld a = (r*r - other.r*other.r + d*d)/(2*d);
 ld h = sqrt(r*r-a*a);
 pt p2 = a * (other.c - o) / d;
 if(Equ(sumr - d, 0)) {
  v.push_back(p2);
  return make_pair(3, v);
 }
 if(Equ(rr - d, 0)) {
  v.push_back(p2);
  return make_pair(1, v);
 }
 if(d <= rr)
  return make_pair(0, v);
 if(d >= sumr)
  return make_pair(4, v);
```

```
 pt p3(p2.x + h*(other.c.y - o.y)/d, p2.y - h*(other.c.x - o
      .x)/d);
 pt p4(p2.x - h*(other.c.y - o.y)/d, p2.y + h*(other.c.x - o
      .x)/d);
 v.push_back(p3);
 v.push_back(p4);
 return make_pair(2, v);
}
ld arcarea(ld l, ld r, ld R){//circle with radius(r)
     intersect with circle with radius (R) and distance
     between centers equal to (d)
 ld cosa = (l*l + r*r - R*R)/(2.0*r*l);
 ld a = acos(cosa);
 return r*r*(a - sin(2*a)/2);
}

int main()
{
 ios_base::sync_with_stdio(false),cin.tie(0),cout.tie(0);

 return 0;
}
```

4.2 Rotating Calipers

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

typedef long double ld;
typedef complex<ld> pt;
typedef vector<pt> poly;
#define x real()
#define y imag()
typedef pair<pt, pt> line;
// +, -, * scalar well defined
const ld EPS = 1e-12;
const ld PI = acos(-1);
const int ON = 0, LEFT = 1, RIGHT = -1, BACK = -2, FRONT =
     2, IN = 3, OUT = -3;

inline bool Lss(ld a, ld b){ return a - b < -EPS; }
inline bool Grt(ld a, ld b){ return a - b > +EPS; }
inline bool Leq(ld a, ld b){ return a - b < +EPS; }
inline bool Geq(ld a, ld b){ return a - b > -EPS; }
inline bool Equ(ld a, ld b){ return abs(a-b) < EPS; }

bool byX(const pt &a, const pt &b)
{
    if (Equ(a.x, b.x)) return Lss(a.y, b.y);
    return Lss(a.x, b.x);
}

ld dot(pt a, pt b){return (conj(a) * b).x;}
ld cross(pt a, pt b){return (conj(a) * b).y;}
int relpos(pt a, pt b, pt c) //c to a-b
{
    b = b-a, c= c-a;
    if (Grt(cross(b,c), 0)) return LEFT;
    if (Lss(cross(b,c), 0)) return RIGHT;
    if (Lss(dot(b,c), 0)) return BACK;
    if (Lss(dot(b,c), abs(b))) return FRONT;
    return ON;
}

//START:
vector<pair<pt, pt>> get_antipodals(poly &p)
{
 int n = p.size();
 sort(p.begin(), p.end(), byX);
 vector <pt> U, L;
 for (int i = 0; i < n; i++){
  while (U.size() > 1 && relpos(U[U.size()-2], U[U.size()
       -1], p[i]) != LEFT)
   U.pop_back();
  while (L.size() > 1 && relpos(L[L.size()-2], L[L.size()
       -1], p[i]) != RIGHT)
   L.pop_back();
  U.push_back(p[i]);
  L.push_back(p[i]);
 }
 vector <pair<pt, pt>> res;
 int i = 0, j = L.size()-1;
 while (i+1 < (int)U.size() || j > 0){
  res.push_back({U[i], L[j]});
  if (i+1 == (int)U.size())
   j--;
  else if (j == 0)
   i++;
  else if (cross(L[j]-L[j-1], U[i+1]-U[i]) >= 0) i++;
  else
   j--;
 }
 return res;
}
//END.

int main()
{
    ios_base::sync_with_stdio(false),cin.tie(0),cout.tie(0);
    return 0;
}
```

5 Graph

5.1 Hungarian

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

#define F first
#define S second
#define pii pair<int, int>
#define pb push_back

using namespace std;

typedef long long ll;
typedef long double ld;

const int N = 2002;
const int INF = 1e9;
int hn, weight[N][N]; //hn should contain number of vertices
      in each part. weight must be positive.
int x[N], y[N]; //initial value doesn't matter.

int hungarian() // maximum weighted perfect matching O(n^3)
{
 int n = hn;
 int p, q;
 vector<int> fx(n, -INF), fy(n, 0);
 fill(x, x + n, -1);
 fill(y, y + n, -1);

 for (int i = 0; i < n; ++i)
  for (int j = 0; j < n; ++j)
   fx[i] = max(fx[i], weight[i][j]);

 for (int i = 0; i < n; ) {
  vector<int> t(n, -1), s(n+1, i);
  for (p = 0, q = 1; p < q && x[i] < 0; ++p) {
   int k = s[p];
   for (int j = 0; j < n && x[i] < 0; ++j)
    if (fx[k] + fy[j] == weight[k][j] && t[j] < 0) {
     s[q++] = y[j], t[j] = k;
     if (y[j] < 0) // match found!
      for (int p = j; p >=0; j = p)
       y[j] = k = t[j], p = x[k], x[k] = j;
```

```
   }
  }
  if (x[i] < 0) {
   int d = INF;
   for (int k = 0; k < q; ++k)
    for (int j = 0; j < n; ++j)
     if (t[j] < 0) d = min(d, fx[s[k]] + fy[j] - weight[s[k
         ]][j]);
   for (int j = 0; j < n; ++j) fy[j] += (t[j] <0? 0: d);
   for (int k = 0; k < q; ++k) fx[s[k]] -= d;
  } else ++i;
 }
 int ret = 0;
 for (int i = 0; i < n; ++i) ret += weight[i][x[i]];
 return ret;
}

int main() {
 ios_base::sync_with_stdio(0); cin.tie(0);
 int n, e; cin >> n >> e;
 for (int i=0; i<e; i++)
 {
  int u, v; cin >> u >> v;
  --u; --v;
  cin >> weight[u][v];
 }
 hn = n;
 cout << hungarian() << '\n';
 return 0;
}
```

6 Numerical

6.1 Base Vector Z2

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

using namespace std;

const int maxL = 61;
typedef long long ll;

struct Base{
 ll a[maxL] = {};
 ll eliminate(ll x){
  for(int i=maxL-1; i>=0; --i) if(x >> i & 1) x ^= a[i];
  return x;
 }
 void add(ll x){
  x = eliminate(x);
  if(x == 0) return ;
  for(int i=maxL-1; i>=0; --i) if(x >> i & 1) {
   a[i] = x;
   return ;
  }
 }
 int size(){
  int cnt = 0;
  for(int i=0; i<maxL; ++i) if(a[i]) ++cnt;
  return cnt;
 }
 ll get_mx() {
  ll x = 0;
  for (int i=maxL-1; i>=0; i--) {
   if(x & (1LL << i)) continue ;
   else x ^= a[i];
  }
  return x;
 }
};
```

6.2 Chineese Reminder Theorem

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

#define F first
#define S second
#define lcm LLLCCM

using namespace std;

typedef long long ll;

long long GCD(long long a, long long b) { return (b == 0) ?
     a : GCD(b, a % b); }
inline long long LCM(long long a, long long b) { return a /
     GCD(a, b) * b; }
inline long long normalize(long long x, long long mod) { x
     %= mod; if (x < 0) x += mod; return x; }

struct GCD_type { long long x, y, d; };
GCD_type ex_GCD(long long a, long long b){
 if (b == 0) return {1, 0, a};
 GCD_type pom = ex_GCD(b, a % b);
 return {pom.y, pom.x - a / b * pom.y, pom.d};
}

const int N = 2;
long long r[N], n[N], ans, lcm;
// t: number of equations,
// r: reminder array, n: mod array
// returns {reminder, lcm}

pair <long long, long long> CRT(ll* r, ll *n, int t) {
 for(int i = 0; i < t; i++)
  normalize(r[i], n[i]);
 ans = r[0];
 lcm = n[0];

 for(int i = 1; i < t; i++){
  auto pom = ex_GCD(lcm, n[i]);
  ll x1 = pom.x;
  ll d = pom.d;
  if((r[i] - ans) % d != 0) {
   return {-1, -1}; //No Solution
  }
  ans = normalize(ans + x1 * (r[i] - ans) / d % (n[i] / d) *
        lcm, lcm * n[i] / d);
  lcm = LCM(lcm, n[i]); // you can save time by replacing
       above lcm * n[i] /d by lcm = lcm * n[i] / d
 }
 return {ans, lcm};
}
```

6.3 FFT

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

using namespace std;

const int LG = 20; // IF YOU WANT TO CONVOLVE TWO ARRAYS OF
     LENGTH N AND M CHOOSE LG IN SUCH A WAY THAT 2LG > n + m
const int MAX = 1 << LG;

#define M_PI acos(-1)

struct point{
 double real, imag;
 point(double _real = 0.0, double _imag = 0.0){
  real = _real;
  imag = _imag;
 }
};
point operator + (point a, point b){
 return point(a.real + b.real, a.imag + b.imag);
}
```

```
point operator - (point a, point b){
 return point(a.real - b.real, a.imag - b.imag);
}
point operator * (point a, point b){
 return point(a.real * b.real - a.imag * b.imag, a.real * b.
   imag + a.imag * b.real);
}
void fft(point *a, bool inv){
 for (int mask = 0; mask < MAX; mask++){
  int rev = 0;
  for (int i = 0; i < LG; i++)
   if ((1 << i) & mask)
    rev |= (1 << (LG - 1 - i));
  if (mask < rev)
   swap(a[mask], a[rev]);
 }
 for (int len = 2; len <= MAX; len *= 2){
  double ang = 2.0 * M_PI / len;
  if (inv)
   ang *= -1.0;
  point wn(cos(ang), sin(ang));
  for (int i = 0; i < MAX; i += len){
   point w(1.0, 0.0);
   for (int j = 0; j < len / 2; j++){
    point t1 = a[i + j] + w * a[i + j +
     len / 2];
    point t2 = a[i + j] - w * a[i + j +
     len / 2];
    a[i + j] = t1;
    a[i + j + len / 2] = t2;
    w = w * wn;
   }
  }
 }
 if (inv)
  for (int i = 0; i < MAX; i++){
   a[i].real /= MAX;
   a[i].imag /= MAX;
  }
}
```

6.4 Gaussian Elimination Xor

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

#define F first
#define S second
#define pii pair<int, int>
#define pb push_back

using namespace std;

typedef long long ll;
typedef long double ld;

const int maxN = 105;
typedef vector <int> vec;

bitset<maxN> matrix[maxN];
bitset<maxN> ans;

vec solve(int n, int m) {
 vec ptr;
 ptr.resize(n);

 int i = 0, j = 0;
 while(i < n and j < m) {
  int ind = -1;
  for(int row = i; row < n; row++)
   if(matrix[row][j])
    ind = row;
  if(ind == -1) {
   j++;
   continue ;
  }

  bitset<maxN> b;
  b = matrix[i];
  matrix[i] = matrix[ind];
  matrix[ind] = b;

  bool f = ans[i];
  ans[i] = ans[ind];
  ans[ind] = f;

  for(int row = i + 1; row < n; row++) {
   if(matrix[row][j]) {
    matrix[row] ^= matrix[i];
    ans[row] = ans[row] ^ ans[i];
   }
  }

  ptr[i] = j;
  i ++;
  j ++;


 }

 vec sol;

 if(i != n) {
  for (int row=i; row<n; row++)
   if(ans[row])
    return sol; //without answer;
 }
 sol.resize(m);

 for (int j=0; j<m; j++)
  sol[j] = 0;

 for (int row=i-1; row>=0; row--){
  int j = ptr[row];
  sol[j] = ans[row];
  for (int c=row-1; c>=0; c--)
   if(matrix[c][j]) ans[c] = ans[c] ^ sol[j];
 }
 return sol;
}
```

6.5 Gaussian Elimination

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

#define F first
#define S second
#define pii pair<int, int>
#define pb push_back

using namespace std;

typedef long long ll;
typedef long double ld;

const int N = 505, MOD = 1e9 + 7;
typedef vector <ll> vec;

ll pw(ll a, ll b) {
 if(!b)
  return 1;
 ll x = pw(a, b/2);
 return x * x % MOD * (b % 2 ? a : 1) % MOD;
}
ll inv(ll x) { return pw(x, MOD - 2); }

//matrix * x = ans
vec solve(vector<vec> matrix, vec ans) {
 int n = matrix.size(), m = matrix[0].size();
```

```
for (int i=0; i<n; i++)
 matrix[i].pb(ans[i]);

vector <int> ptr;
ptr.resize(n);

int i = 0, j =0;
while(i < n and j < m) {
 int ind = -1;
 for(int row = i; row < n; row++)
  if(matrix[row][j])
   ind = row;
 if(ind == -1) {
  j++;
  continue ;
 }

 matrix[i].swap(matrix[ind]);
 ll inverse = inv(matrix[i][j]);
 for(int row = i + 1; row < n; row++) {
  ll z = matrix[row][j] * inverse % MOD;
  for(int k = 0; k <= m; k++)
   matrix[row][k] = (matrix[row][k] % MOD - matrix[i][k]*z %
        MOD + MOD) % MOD;
 }

 ptr[i] = j;
 i ++;
 j ++;
}

vector <ll> sol;

if(i != n) {
 for (int row=i; row<n; row++)
  if(matrix[row][m] != 0)
   return sol; //without answer;
}
sol.resize(m);
for (int j=0; j<m; j++)
 sol[j] = 0;

for (int row=i-1; row>=0; row--){
 int j = ptr[row];
 sol[j] = matrix[row][m] * inv(matrix[row][j]) % MOD;
 for (int c=row-1; c>=0; c--)
  matrix[c][m] += (MOD - sol[j] * matrix[c][j] % MOD),
      matrix[c][m] %= MOD;
}
return sol;
```

```
}

int main() {
 int n, m; cin >> n >> m;
 vector <vec> A;
 for (int i=0; i<n; i++)
 {
  vec B;
  for (int j=0; j<m; j++)
  {
   ll x; cin >> x;
   B.push_back(x);
  }
  A.push_back(B);
 }

 vec ans;
 for (int i=0; i<n; i++)
 {
  ll y; cin >> y;
  ans.pb(y);
 }

 vec sol = solve(A, ans);
 for (auto X : sol)
  cout << X << ' ';
 cout << endl;
}
```

6.6 General Linear Recursion

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

#define pb push_back

using namespace std;

const int maxL = 20; // IF YOU WANT TO CONVOLVE TWO ARRAYS
     OF LENGTH N AND M CHOOSE LG IN SUCH A WAY THAT 2LG > n
     + m
const int maxN = 1 << maxL, MOD = 998244353;

typedef long long ll;

#define M_PI acos(-1)

int root[maxL + 2] =
     {0,998244352,86583718,372528824,69212480,87557064,150535755,67475946,15032460,4097924,1762757,752127,299814,730033,227806,4205
```

```
int bpow(int a, int b){
 int ans = 1;
 while (b){
  if (b & 1)
   ans = 1LL * ans * a % MOD;
  b >>= 1;
  a = 1LL * a * a % MOD;
 }
 return ans;
}

void ntt(vector<int> &a, bool inv){
 int LG = 0, z = 1, MAX = a.size();
 while(z != MAX) z *= 2, LG ++;
 int ROOT = root[LG];

 for (int mask = 0; mask < MAX; mask++){
  int rev = 0;
  for (int i = 0; i < LG; i++)
   if ((1 << i) & mask)
    rev |= (1 << (LG - 1 - i));
  if (mask < rev)
   swap(a[mask], a[rev]);
 }
 for (int len = 2; len <= MAX; len *= 2){
  int wn = bpow(ROOT, MAX / len);
  if (inv)
   wn = bpow(wn, MOD - 2);
  for (int i = 0; i < MAX; i += len){
   int w = 1;
   for (int j = 0; j < len / 2; j++){
    int l = a[i + j];
    int r = 1LL * w * a[i + j + len / 2] %
     MOD;
    a[i + j] = (l + r);
    a[i + j + len / 2] = l - r + MOD;
    if (a[i + j] >= MOD)
     a[i + j] -= MOD;
    if (a[i + j + len / 2] >= MOD)
     a[i + j + len / 2] -= MOD;
    w = 1LL * w * wn % MOD;
   }
  }
 }
 if (inv){
  int x = bpow(MAX, MOD - 2);
  for (int i = 0; i < MAX; i++)
   a[i] = 1LL * a[i] * x % MOD;
```

```
}


int ans[maxN], bb[maxN];

//ans[i] = sum_j=1^i b_j * ans[i - j], ans[0] = 1;
void solve(int l, int r) {
 if(r - l == 1) return ;
 int mid = (l + r)/2;

 solve(l, mid);

 vector <int> a, b;
 for (int i=l; i<r; i++) {
  if(i < mid) a.pb(ans[i]);
  else a.pb(0);

  b.pb(bb[i-l+1]);
 }

 for (int i=l; i<r; i++) {
  a.pb(0);
  b.pb(0);
 }

 ntt(a, false);
 ntt(b, false);

 vector <int> c;
 c.resize(a.size());

 for (int i=0; i<2*r-2*l; i++)
  c[i] = 1LL * a[i] * b[i] % MOD;

 ntt(c, true);
 for (int i=0; i<r-mid; i++)
  ans[mid + i] += c[mid - l - 1 + i], ans[mid + i] %= MOD;

 solve(mid, r);
}

int main() {
 int n, m; cin >> n >> m;
 for (int i=1; i<=m; i++)
  cin >> bb[i];
 int k = 1;
 while(k < n) k = 2 * k;

 ans[0] = 1;
 solve(0, k);

 for (int i=0; i<n; i++)
  cout << ans[i] << ' ';
 cout << endl;
}
```

6.7 Miller Robin

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

//with probability (1/4) iter, we might make mistake in our
    guess.
//we have false positive here.
using u64 = uint64_t;
using u128 = __uint128_t;

using namespace std;

u64 binpower(u64 base, u64 e, u64 mod) {
 u64 result = 1;
 base %= mod;
 while (e) {
  if (e & 1)
   result = (u128)result * base % mod;
  base = (u128)base * base % mod;
  e >>= 1;
 }
 return result;
}

bool check_composite(u64 n, u64 a, u64 d, int s) {
 u64 x = binpower(a, d, n);
 if (x == 1 || x == n - 1)
  return false;
 for (int r = 1; r < s; r++) {
  x = (u128)x * x % n;
  if (x == n - 1)
   return false;
 }
 return true;
};

bool MillerRabin(u64 n, int iter=5) { // returns true if n
    is probably prime, else returns false.
 if (n < 4)
  return n == 2 || n == 3;

 int s = 0;
 u64 d = n - 1;
 while ((d & 1) == 0) {
  d >>= 1;
  s++;
 }

 for (int i = 0; i < iter; i++) {
  int a = 2 + rand() % (n - 3);
  if (check_composite(n, a, d, s))
   return false;
 }
 return true;
}
```

6.8 NTT

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

using namespace std;

const int MOD = 998244353;
const int LG = 16; // IF YOU WANT TO CONVOLVE TWO ARRAYS OF
    LENGTH N AND M CHOOSE LG IN SUCH A WAY THAT 2LG > n + m
const int MAX = (1 << LG);
const int ROOT = 44759; // ENSURE THAT ROOT2(LG - 1) = MOD -
    1
int bpow(int a, int b){
 int ans = 1;
 while (b){
  if (b & 1)
   ans = 1LL * ans * a % MOD;
  b >>= 1;
  a = 1LL * a * a % MOD;
 }
 return ans;
}
void ntt(int *a, bool inv){
 for (int mask = 0; mask < MAX; mask++){
  int rev = 0;
  for (int i = 0; i < LG; i++)
   if ((1 << i) & mask)
    rev |= (1 << (LG - 1 - i));
  if (mask < rev)
   swap(a[mask], a[rev]);
 }
 for (int len = 2; len <= MAX; len *= 2){
  int wn = bpow(ROOT, MAX / len);
  if (inv)
   wn = bpow(wn, MOD - 2);
  for (int i = 0; i < MAX; i += len){
   int w = 1;
```

```
   for (int j = 0; j < len / 2; j++){
    int l = a[i + j];
    int r = 1LL * w * a[i + j + len / 2] %
     MOD;
    a[i + j] = (l + r);
    a[i + j + len / 2] = l - r + MOD;
    if (a[i + j] >= MOD)
     a[i + j] -= MOD;
    if (a[i + j + len / 2] >= MOD)
     a[i + j + len / 2] -= MOD;
    w = 1LL * w * wn % MOD;
   }
  }
 }
 if (inv){
  int x = bpow(MAX, MOD - 2);
  for (int i = 0; i < MAX; i++)
   a[i] = 1LL * a[i] * x % MOD;
 }
}
```

6.9 Simplex

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

#define F first
#define S second
#define pb push_back
#define pii pair<int, int>

using namespace std;

typedef long double ld;
typedef vector <ld> vd;
typedef vector <int> vi;
const ld Eps = 1e-9;

// ax <= b, max(cTx), x >= 0
// O(nm^2)

vd simplex(vector <vd> a, vd b, vd c) {
 int n = a.size(), m = a[0].size() + 1, r = n, s = m - 1;
 vector <vd> d(n + 2, vd(m + 1, 0)); vd x(m - 1);
 vi ix(n + m); iota(ix.begin(), ix.end(), 0);
 for(int i = 0; i < n; i ++) {
  for(int j = 0; j < m - 1; j ++) d[i][j] = -a[i][j];
  d[i][m - 1] = 1;
  d[i][m] = b[i];
  if(d[r][m] > d[i][m])
   r = i;
 }
 for(int j = 0; j < m - 1; j ++) d[n][j] = c[j];
 d[n + 1][m - 1] = -1;
 while(true) {
  if(r < n) {
   vd su;
   swap(ix[s], ix[r + m]); d[r][s] = 1 / d[r][s];
   for(int j = 0; j <=m; j ++) if(j != s) {
    d[r][j] *= -d[r][s]; if(d[r][j]) su.pb(j);
   }
   for(int i = 0; i <= n + 1; i ++) if(i != r) {
    for(int j = 0; j < su.size(); j ++)
     d[i][su[j]] += d[r][su[j]] * d[i][s];
    d[i][s] *= d[r][s];
   }
  }
  r = s = -1;
  for(int j = 0; j < m; j ++) if(s < 0 || ix[s] > ix[j])
   if(d[n + 1][j] > Eps || d[n + 1][j] > -Eps &&
     d[n][j] > Eps) s = j; if(s < 0) break;
  for(int i = 0; i < n; i ++) if(d[i][s] < -Eps) {
   if(r < 0) {
    r = i;
    continue;
   }
   double e = d[r][m] / d[r][s] - d[i][m] / d[i][s];
   if(e < -Eps || e < Eps && ix[r + m] > ix[i + m]) r = i;
  }
  if(r < 0)
  {return vd();} // Unbounded
 }
 if(d[n + 1][m] < -Eps) {return vd();}// No solution
 for(int i = m; i < n + m; i ++)
  if(ix[i] <m - 1) x[ix[i]] = d[i - m][m];
 return x;
}
```

6.10 Stirling Cycle

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

using namespace std;
typedef long long ll;

const int mod = 998244353;
const int root = 15311432;
const int root_1 = 469870224;
const int root_pw = 1 << 23;

const int N = 400004;
vector<int> v[N];

ll modInv(ll a, ll mod = mod){
 ll x0 = 0, x1 = 1, r0 = mod, r1 = a;
 while(r1){
  ll q = r0 / r1;
  x0 -= q * x1; swap(x0, x1);
  r0 -= q * r1; swap(r0, r1);
 }
 return x0 < 0 ? x0 + mod : x0;
}

void fft (vector<int> &a, bool inv) {
 int n = (int) a.size();
 for (int i=1, j=0; i<n; ++i) {
  int bit = n >> 1;
  for (; j>=bit; bit>>=1)
   j -= bit;
  j += bit;
  if (i < j)
   swap (a[i], a[j]);
 }
 for (int len=2; len<=n; len<<=1) {
  int wlen = inv ? root_1 : root;
  for (int i=len; i<root_pw; i<<=1)
   wlen = int (wlen * 1ll * wlen % mod);
  for (int i=0; i<n; i+=len) {
   int w = 1;
   for (int j=0; j<len/2; ++j) {
    int u = a[i+j], v = int (a[i+j+len/2] * 1ll * w % mod);
    a[i+j] = u+v < mod ? u+v : u+v-mod;
    a[i+j+len/2] = u-v >= 0 ? u-v : u-v+mod;
    w = int (w * 1ll * wlen % mod);
   }
  }
 }
 if(inv) {
  int nrev = modInv(n, mod);
  for (int i=0; i<n; ++i)
   a[i] = int (a[i] * 1ll * nrev % mod);
 }
}
void pro(const vector<int> &a, const vector<int> &b, vector<
    int> &res)
{
 vector<int> fa(a.begin(), a.end()), fb(b.begin(), b.end());
 int n = 1;
 while (n < (int) max(a.size(), b.size())) n <<= 1;
 n <<= 1;
```

```
 fa.resize (n), fb.resize (n);
 fft(fa, false), fft (fb, false);
 for (int i = 0; i < n; ++i)
  fa[i] = 1LL * fa[i] * fb[i] % mod;
 fft (fa, true);
 res = fa;
}
int S(int n, int r) {
 int nn = 1;
 while(nn < n) nn <<= 1;
 for(int i = 0; i < n; ++i) {
  v[i].push_back(i);
  v[i].push_back(1);
 }
 for(int i = n; i < nn; ++i) {
  v[i].push_back(1);
 }
 for(int j = nn; j > 1; j >>= 1){
  int hn = j >> 1;
  for(int i = 0; i < hn; ++i){
   pro(v[i], v[i + hn], v[i]);
  }
 }
 /*for (int k=0; k<=r; k++)
  cout << v[0][k] << ' '; cout << '\n';*/

 return v[0][r];
}
int fac[N], ifac[N], inv[N];
void prencr(){
 fac[0] = ifac[0] = inv[1] = 1;
 for(int i = 2; i < N; ++i)
  inv[i] = mod - 1LL * (mod / i) * inv[mod % i] % mod;
 for(int i = 1; i < N; ++i){fac[i] = 1LL * i * fac[i - 1] %
      mod;
  ifac[i] = 1LL * inv[i] * ifac[i - 1] % mod;
 }
}
int C(int n, int r){
 return (r >= 0 && n >= r) ? (1LL * fac[n] * ifac[n - r] %
      mod
   * ifac[r] % mod) : 0;
}
int main(){
 prencr();
 int n, k;
 cin >> n >> k;
 cout << S(n, k) << endl; //Also you have S(n, t) for all t.
}
```

6.11 Stirling Set

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

using namespace std;

const int MOD = 998244353;
const int LG = 16; // IF YOU WANT TO CONVOLVE TWO ARRAYS OF
     LENGTH N AND M CHOOSE LG IN SUCH A WAY THAT 2LG > n + m
const int MAX = (1 << LG);
const int ROOT = 44759; // ENSURE THAT ROOT2(LG - 1) = MOD -
      1
int bpow(int a, int b){
 int ans = 1;
 while (b){
  if (b & 1)
   ans = 1LL * ans * a % MOD;
  b >>= 1;
  a = 1LL * a * a % MOD;
 }
 return ans;
}
void ntt(int *a, bool inv){
 for (int mask = 0; mask < MAX; mask++){
  int rev = 0;
  for (int i = 0; i < LG; i++)
   if ((1 << i) & mask)
    rev |= (1 << (LG - 1 - i));
  if (mask < rev)
   swap(a[mask], a[rev]);
 }
 for (int len = 2; len <= MAX; len *= 2){
  int wn = bpow(ROOT, MAX / len);
  if (inv)
   wn = bpow(wn, MOD - 2);
  for (int i = 0; i < MAX; i += len){
   int w = 1;
   for (int j = 0; j < len / 2; j++){
    int l = a[i + j];
    int r = 1LL * w * a[i + j + len / 2] %
     MOD;
    a[i + j] = (l + r);
    a[i + j + len / 2] = l - r + MOD;
    if (a[i + j] >= MOD)
     a[i + j] -= MOD;
    if (a[i + j + len / 2] >= MOD)
     a[i + j + len / 2] -= MOD;
    w = 1LL * w * wn % MOD;
   }
  }
 }
 if (inv){
  int x = bpow(MAX, MOD - 2);
  for (int i = 0; i < MAX; i++)
   a[i] = 1LL * a[i] * x % MOD;
 }
}

int a[MAX], b[MAX], c[MAX];

int main() {
 int n; cin >> n;

 a[0] = 1;
 b[0] = 0;
 int inv_fact = 1;

 for (int i=1; i<=n; i++)
 {
  a[i] = 1LL * a[i - 1] * (MOD - 1) % MOD;
  a[i] = 1LL * a[i] * bpow(i, MOD - 2) % MOD;

  inv_fact = 1LL * inv_fact * bpow(i, MOD - 2) % MOD;
  b[i] = bpow(i, n);
  b[i] = 1LL * b[i] * inv_fact % MOD;
 }

 ntt(a, false);
 ntt(b, false);

 for (int i=0; i<MAX; i++)
  c[i] = 1LL * a[i] * b[i] % MOD;

 ntt(c, true);
 for (int j=0; j<n; j++)
  cout << c[j] << ' ';
 cout << endl;

}
```

7 String

7.1 Aho Corrasick

```
int nxt[N][C];
int f[N], q[N], vcnt;
vector<int> adj[N];
```

```
int add(string s)
{
 int cur = 0;
 for(auto ch : s)
 {
  ch -= 'a';
  if(!nxt[cur][ch]) nxt[cur][ch] = ++vcnt;
  cur = nxt[cur][ch];
 }
 return cur;
}

void aho()
{
 int hi = 0, lo = 0;
 for(int i = 0; i < C; i++) if(nxt[0][i]) q[hi++] = nxt[0][i
     ];
 while(hi != lo)
 {
  int x = q[lo++];
  adj[f[x]].pb(x);
  for(int i = 0; i < C; i++)
  {
   if(nxt[x][i])
   {
    q[hi++] = nxt[x][i];
    f[nxt[x][i]] = nxt[f[x]][i];
   }
   else nxt[x][i] = nxt[f[x]][i];
  }
 }
}
```

7.2 Palindromic

```
int n, last, sz;
char s[N];
int len[N], link[N], cnt[N];
map<short, int> to[N];
void init() {
 n = 0; last = 0;
 for(int i = 0; i < N; i++) to[i].clear();
 s[n++] = -1;
 link[0] = 1;
 len[1] = -1;
 sz = 2;
}
int get_link(int v) {
 while(s[n - len[v] - 2] != s[n - 1]) v = link[v];
 return v;
}
void add_letter(int c) {
 s[n++] = c;
 last = get_link(last);
 if(!to[last][c]) {
  len [sz] = len[last] + 2;
  link[sz] = to[get_link(link[last])][c];
  to[last][c] = sz++;
 }
 last = to[last][c];
 cnt[last] = cnt[link[last]] + 1;
}
```

7.3 Suffix Array

```
string s;
int rank[LOG][N], n, lg;
pair<pair<int, int>, int> sec[N];
int sa[N];
int lc[N];

int lcp(int a, int b)
{
 int _a = a;
 for(int w = lg - 1; ~w && max(a, b) < n; w--)
  if(max(a, b) + (1 << w) <= n && rank[w][a] == rank[w][b])
   a += 1 << w, b += 1 << w;
 return a - _a;
}

int cnt[N];
pair<pii, int> gec[N];
void srt()
{
 memset(cnt, 0, sizeof cnt);
 for(int i = 0; i < n; i++) cnt[sec[i].F.S+1]++;
 for(int i = 1; i < N; i++) cnt[i] += cnt[i - 1];
 for(int i = 0; i < n; i++) gec[--cnt[sec[i].F.S+1]] = sec[i
     ];
 memset(cnt, 0, sizeof cnt);
 for(int i = 0; i < n; i++) cnt[gec[i].F.F+1]++;
 for(int i = 1; i < N; i++) cnt[i] += cnt[i - 1];
 for(int i = n - 1; ~i; i--) sec[--cnt[gec[i].F.F+1]] = gec[
     i];
}

void build()
{
 n = s.size();
 {
  int cur = 1; lg = 0;
  while(cur < n)
  {
   lg++;
   cur <<= 1;
  }
  lg++;
 }

 for(int i = 0; i < n; i++) rank[0][i] = s[i];
 for(int w = 1; w < lg; w++)
 {
  for(int i = 0; i < n; i++)
   if(i + (1 << w - 1) >= n)
    sec[i] = {{rank[w-1][i], -1}, i};
   else
    sec[i] = {{rank[w-1][i], rank[w-1][i+(1<<w-1)]}, i};
  srt();
  rank[w][sec[0].S] = 0;
  for(int i = 1; i < n; i++)
   if(sec[i].F == sec[i - 1].F)
    rank[w][sec[i].S] = rank[w][sec[i-1].S];
   else
    rank[w][sec[i].S] = i;
 }

 for(int i = 0; i < n; i++)
  sa[rank[lg-1][i]] = i;
 for(int i = 0; i + 1 < n; i++)
  lc[i] = lcp(sa[i], sa[i + 1]);
}
```

7.4 Suffix Automata

```
const int maxn = 2 e5 + 42; // Maximum amount of states
map < char , int > to [ maxn ]; // Transitions
int link [ maxn ]; // Suffix links
int len [ maxn ]; // Lengthes of largest strings in states
int last = 0; // State corresponding to the whole string
int sz = 1; // Current amount of states
void add_letter ( char c ) { // Adding character to the end
 int p = last ; // State of string s
 last = sz ++; // Create state for string sc
 len [ last ] = len [ p ] + 1;
 for (; to [ p ][ c ] == 0; p = link [ p ]) // (1)
  to [ p ][ c ] = last ; // Jumps which add new suffixes
```

```
 if ( to [ p ][ c ] == last ) { // This is the first
     occurrence of
  c if we are here
   link [ last ] = 0;
  return ;
 }
 int q = to [ p ][ c ];
 if ( len [ q ] == len [ p ] + 1) {
  link [ last ] = q ;
  return ;
 }
 // We split off cl from q here
 int cl = sz ++;
 to [ cl ] = to [ q ]; // (2)
 link [ cl ] = link [ q ];
 len [ cl ] = len [ p ] + 1;
 link [ last ] = link [ q ] = cl ;
 for (; to [ p ][ c ] == q ; p = link [ p ]) // (3)
  to [ p ][ c ] = cl ; // Redirect transitions where needed
}
```

7.5 Suffix Tree

```
#define fpos adla
const int inf = 1e9;
const int maxn = 1e4; //maxn = number of states of suffix
    tree
char s[maxn];
map<int, int> to[maxn]; //edges of tree
int len[maxn], fpos[maxn], link[maxn];
//len[i] is the length of the inner edge of v
//fpos[i] is start position of inner edge in string s
int node, pos;
int sz = 1, n = 0;

int make_node(int _pos, int _len) {
 fpos[sz] = _pos;
 len [sz] = _len;
 return sz++;
}
void go_edge() {
 while(pos > len[to[node][s[n - pos]]]) {
  node = to[node][s[n - pos]];
  pos -= len[node];
 }
}
void add_letter(int c) {
 s[n++] = c;
 pos++;
 int last = 0;
 while(pos > 0) {
  go_edge();
  int edge = s[n - pos];
  int &v = to[node][edge];
  int t = s[fpos[v] + pos - 1];
  if(v == 0) {
   v = make_node(n - pos, inf);
   link[last] = node;
   last = 0;
  } else if(t == c) {
   link[last] = node;
   return;
  } else {
   int u = make_node(fpos[v], pos - 1);
   to[u][c] = make_node(n - 1, inf);
   to[u][t] = v;
   fpos[v] += pos - 1;
   len [v] -= pos - 1;
   v = u;
   link[last] = u;
   last = u;
  }
  if(node == 0)
   pos--;
  else
   node = link[node];
 }
}
```

8 Tree

8.1 dummy