Standard Source code Library

mayf3

November 20, 2015

Contents

1 图论 2
1.1 图 . . . 2
1.1.1 Dominator Tree . . . 2
2 理论 6
2.1 数学 . . . 6
2.1.1 FFT fast . . . 6
2.1.2 最少需要平方数 . . . 7
2.1.3 分治 nnt . . . 8
2.1.4 x*x+r*y*y=p 的解 . . . 10
2.1.5 高斯消元 . . . 12
2.1.6 NNT mod 1000000007 . . . 14
2.1.7 NNT prime number . . . 17
2.1.8 多项式运算 (exp,ln,sqrt,inv,div) . . . 18

1 图论

1.1 图

1.1.1 Dominator Tree

```
#pragma comment(linker, "/STACK:102400000,102400000")
#include <cstdio>
#include <cstring>
#include <vector>
#include <stack>
#include <algorithm>

using namespace std;

const int N = 300000 + 1;

struct DominatorTree{
  int st;
  int n, m;
  int tot;
  int dfn[N], id[N], fa[N];
  int idom[N], semi[N];
  int f[N], best[N];
  vector<int> suc[N], pre[N];
  vector<int> dom[N];

  void dfs(int x){
    dfn[x] = ++tot;
    id[tot] = x;
    for(int i = 0; i < suc[x].size(); i++){
      int y = suc[x][i];
      if (dfn[y]) continue;
      fa[y] = x;
      dfs(y);
    }
  }

  int get(int x){
    if (x == f[x]) return x;
    int y = get(f[x]);
    if (dfn[semi[best[x]]] > dfn[semi[best[f[x]]]]) best[x] = best[f[x]];
    f[x] = y;
    return y;
  }

  void tarjan(){
    for(int i = tot; i > 1; i--){
```

```
    int y = id[i];
    int x = fa[y];
    for(int j = 0; j < pre[y].size(); j++){
      int z = pre[y][j];
      if (dfn[z] == 0) continue;
      get(z);
      if (dfn[semi[best[z]]] < dfn[semi[y]]) semi[y] = semi[best[z]];
    }
    dom[semi[y]].push_back(y);
    f[y] = x;
    for(int j = 0; j < dom[x].size(); j++){
      int z = dom[x][j];
      get(z);
      if (dfn[semi[best[z]]] < dfn[x]) idom[z] = best[z];
      else idom[z] = x;
    }
    dom[x].clear();
  }
  for(int i = 2; i < tot + 1; i++){
    int u = id[i];
    if (idom[u] != semi[u]) idom[u] = idom[idom[u]];
    dom[idom[u]].push_back(u);
  }
  idom[id[1]] = 0;
}

void init(int _n, vector<pair<int, int> > edges, int _st){
  n = _n, m = edges.size(), st = _st;
  for(int i = 1; i <= n; i++){
    f[i] = i;
    best[i] = i;
    suc[i].clear();
    pre[i].clear();
    dom[i].clear();
    dfn[i] = 0;
    id[i] = 0;
    semi[i] = i;
    idom[i] = i;
  }
  for(int i = 0; i < m; i++){
    int x, y;
    x = edges[i].first;
    y = edges[i].second;
    suc[x].push_back(y);
    pre[y].push_back(x);
  }
}
```

```
  void solve(){
    tot = 0;
    dfs(st);
    tarjan();
  }
}dt;

int n, m;
int tot, col;
vector<int> E[N];
bool ok[N];
stack<int> sta;

int dfn[N], low[N];
bool in_stack[N];

int co[N];
int id[N];
int iv[N];

void dfs(int u){
  dfn[u] = low[u] = tot++;
  in_stack[u] = true;
  sta.push(u);
  for(int i = 0; i < E[u].size(); i++){
    int v = E[u][i];
    if (dfn[v] == -1){
      dfs(v);
      low[u] = min(low[u], low[v]);
    }
    else if (in_stack[v]){
      low[u] = min(low[u], dfn[v]);
    }
  }
  if (low[u] == dfn[u]){
    vector<int> v;
    vector<pair<int, int> > edges;
    int cnt = 0;
    int st = -1;
    while(true){
      int x = sta.top(); sta.pop();
      v.push_back(x);
      co[x] = col;
      id[x] = ++cnt;
      iv[cnt] = x;
      in_stack[x] = false;
      if (x < n && st == -1) st = id[x];
      if (x == u) break;
```

```
    }
    col++;

    edges.clear();
    for(int i = 0; i < v.size(); i++){
      int x = v[i];
      for(int j = 0; j < E[x].size(); j++){
        int y = E[x][j];
        if (co[y] != col - 1) continue;
        edges.push_back(make_pair(id[x], id[y]));
      }
    }
    dt.init(cnt, edges, st);
    dt.solve();
    for(int i = 1; i <= cnt; i++){
      if (dt.dom[i].size() == 0) continue;
      int x = iv[i];
      if (x >= n) ok[x - n] = true;
    }

    edges.clear();
    for(int i = 0; i < v.size(); i++){
      int x = v[i];
      for(int j = 0; j < E[x].size(); j++){
        int y = E[x][j];
        if (co[y] != col - 1) continue;
        edges.push_back(make_pair(id[y], id[x]));
      }
    }
    dt.init(cnt, edges, st);
    dt.solve();
    for(int i = 1; i <= cnt; i++){
      if (dt.dom[i].size() == 0) continue;
      int x = iv[i];
      if (x >= n) ok[x - n] = true;
    }
  }
}

void init(){
  for(int i = 0; i < n + m; i++){
    E[i].clear();
  }
  for(int i = 0; i < m; i++){
    int x, y;
    scanf("%d%d", &x, &y);
    x--, y--;
    E[x].push_back(n + i);
```

```
    E[n + i].push_back(y);
  }
}

void solve(){
  for(int i = 0; i < n + m; i++){
    ok[i] = false;
    dfn[i] = low[i] = -1;
    in_stack[i] = false;
    co[i] = -1;
  }
  tot = 0;
  col = 0;
  for(int i = 0; i < n + m; i++){
    if (dfn[i] != -1) continue;
    dfs(i);
  }
  for(int i = 0; i < m; i++){
    if (ok[i]) putchar('1');
    else putchar('0');
  }
  puts("");
}

int main(){
  int T;
  scanf("%d", &T);
  for(int cas = 1; cas <= T; cas++){
    scanf("%d%d", &n, &m);
    init();
    solve();
  }
}
```

2 理论

2.1 数学

2.1.1 FFT fast

```
struct Comp{
    double re,im;
    Comp(){}
    Comp(double _re, double _im):re(_re),im(_im){}
    Comp operator + (const Comp &a)const{ return Comp(re+a.re,im+a.im); }
    Comp operator - (const Comp &a)const{ return Comp(re-a.re,im-a.im); }
   Comp operator * (const Comp &a)const{ return Comp(re*a.re-im*a.im,a.re*im+re*a.im); }
    Comp operator * (const double &a)const{ return Comp(re*a,im*a); }
```

```
    Comp operator / (const double &a)const{ return Comp(re/a,im/a); }
    void init(){re=im=0;}
};

void fft(Comp a[], int n, bool invert){
    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){
        double ang=2*PI/len*(invert?-1:1);
        Comp wlen(cos(ang),sin(ang));
        for(int i=0; i<n; i+=len){
            Comp w(1,0);
            for(int j=0; j<len/2; j++){
                Comp u=a[i+j],v=a[i+j+len/2]*w;
                a[i+j]=u+v; a[i+j+len/2]=u-v;
                w=w*wlen;
            }
        }
    }
    if(invert)for(int i=0; i<n; i++)a[i]=a[i]/n;
}
```

2.1.2 最少需要平方数

```
int Work(LL n)  {
    while(n % 4 == 0) n >>= 2;
    if(n % 8 == 7) return 4;
    LL i = 8, t = 9;
    while(t <= n)
    {
        while(n % t == 0) n /= t;
        i += 8;
        t += i;
    }
    if(n == 1) return 1;
    if(n % 2 == 0) n >>= 1;
    if(n % 4 == 3) return 3;
    LL k = 3;
    while(k * k <= n)
    {
        if(n % k == 0) return 3;
        k += 4;
    }
    return 2;
```

```
}
```

2.1.3 分治 nnt

```
#include <cstdio>
#include <vector>

using namespace std;

const int M = 15;
const int N = 1 << M;
const int MOD = 152076289;
const int ROOT = 106;

int qmod(int a, int n, int p){
    int ret = 1;
    while(n){
        if (n & 1) ret = 1LL * ret * a % p;
        a = 1LL * a * a % p;
        n >>= 1;
    }
    return ret;
}

class NNT {
    public:
        NNT(int n, int mod, int root);
        void forward(int a[]) {
            work(a, r);
        }
        void reverse(int a[]) {
            work(a, ir);
            for (int i = 0; i < n; ++i) a[i] = 1LL * a[i] * n_rev % mod;
        }
    private:
        int n, p, mod, n_rev;
        vector<int> rb;
        int r[20];
        int ir[20];
        void work(int a[], int* roots);
};

NNT::NNT(int n, int mod, int root) : n(n) , mod(mod), rb(n) , p(0) {
    n_rev = qmod(n, mod - 2, mod);
    while ((1 << p) < n) ++p;
    for(int i = 0; i < n; i++){
        int x = i, y = 0;
        for (int j = 0; j < p; ++j) {
```

```
            y = (y << 1) | (x & 1);
            x >>= 1;
        }
        rb[i] = y;
    }
    int inv = qmod(root, mod - 2, mod);
    r[p - 1] = qmod(root, (mod - 1) / (1 << p), mod);
    ir[p - 1] = qmod(inv, (mod - 1) / (1 << p), mod);
    for(int i = p - 2; i >= 0; i--){
        r[i] = 1LL * r[i + 1] * r[i + 1] % mod;
        ir[i] = 1LL * ir[i + 1] * ir[i + 1] % mod;
    }
}

void NNT::work(int a[], int* r) {
    for (int i = 0; i < n; ++i) if (rb[i] > i) swap(a[i], a[rb[i]]);
    for (int len = 2; len <= n; len <<= 1) {
        int root = *r++;
        for (int i = 0; i < n; i += len) {
            int w = 1;
            for (int j = 0; j < len / 2; ++j) {
                int u = a[i + j];
                int v = 1LL * a[i + j + len / 2] * 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 = 1LL * w * root % mod;
            }
        }
    }
}

NNT *nnt[M];

int n, m;
int fac[N], inv[N];
int a[N], b[N];
int dp[N], g[N];

void pre(){
    fac[0] = 1;
    for(int i = 1; i < N; i++) fac[i] = 1LL * fac[i - 1] * i % MOD;
    inv[N - 1] = qmod(fac[N - 1], MOD - 2, MOD);
    for(int i = N - 1; i > 0; i--) inv[i - 1] = 1LL * inv[i] * i % MOD;
    for(int i = 1; i < M; i++) nnt[i] = new NNT(1 << i, MOD, ROOT);
}

void init(){
    for(int i = 1; i <= n; i++){
```

```
        dp[i] = qmod(m + 1, i * (i - 1) / 2, MOD);
        g[i] = dp[i];
    }
}

void work(int l, int r){
    if (l == r) return;
    int mid = (l + r) / 2;
    work(l, mid);
    int p = 0, t = 1;
    while(t <= (r - l + 1)) p++, t <<= 1;
   for(int i = l; i <= mid; i++) a[i - l] = (1LL * dp[i] * inv[i - 1] % MOD + MOD) % MOD;
    for(int i = mid - l + 1; i < t; i++) a[i] = 0;
    for(int i = 0; i < t; i++) b[i] = (1LL * g[i] * inv[i] % MOD + MOD) % MOD;
    nnt[p]->forward(a);
    nnt[p]->forward(b);
    for(int i = 0; i < t; i++) a[i] = 1LL * a[i] * b[i] % MOD;
    nnt[p]->reverse(a);
    for(int i = mid + 1; i <= r; i++){
        dp[i] = (dp[i] - 1LL * a[i - l] * fac[i - 1]) % MOD;
    }
    work(mid + 1, r);
}

int main(){
    pre();
    int T;
    scanf("%d", &T);
    for(int cas = 1; cas <= T; cas++){
        scanf("%d%d", &n, &m);
        init();
        work(1, n);
        int ret = dp[n];
        ret = (ret - 1LL * qmod(n, n - 2, MOD) * qmod(m, n - 1, MOD)) % MOD;
        ret = (ret % MOD + MOD) % MOD;
        printf("Case #%d: %d\n", cas, ret);
    }
}
```

2.1.4 x*x+r*y*y=p 的解

```
// (a^2+r*b^2)(c^2+r*d^2) = (ac-r*bd)^2 + r*(ad+bc)^2 = (ac + r*bd)^2 + r*(ad-bc)^2
// 所以x^2+r*y^2=n可以通过构造x^2+r*y^2=p来构成，但不是充要条件
// 例如x^2+3*y^2=2无解，但是x^2+3*y^2=2^2有解,局限性很大
LL mul(LL x, LL y, LL z){
    return (x * y - (LL)(x / (long double) z * y + 1e-3) * z + z) % z;
}
```

```
LL qmod(LL a, LL n, LL p){
    LL ret = 1;
    a %= p;
    while(n){
        if (n & 1) ret = mul(ret, a, p);
        a = mul(a, a, p);
        n >>= 1;
    }
    return ret;
}

bool getSqr(LL r, LL p, LL &r1, LL &r2) {
    if (p == 2){
        r1 = r2 = r;
        return true;
    }
    if (qmod(r, (p - 1) / 2, p) != 1) return false;
    LL S = 0, Q = p - 1;
    while (!(Q & 1)) {
        Q >>= 1;
        S++;
    }
    if (S == 1) {
        r1 = qmod(r, (p + 1) >> 2, p);
        r2 = p - r1;
        return true;
    }
    LL i, j, c, R, dt, t, M, z = 1;
    do {
        z = rand() % p;
    } while (qmod(z, (p - 1) >> 1, p) != p - 1);
    c = qmod(z, Q, p);
    R = qmod(r, (Q + 1) >> 1, p);
    t = qmod(r, Q, p);
    M = S;
    while (t != 1) {
        for (i = 1, dt = mul(t, t, p); dt != 1; i++) dt = mul(dt, dt, p);
        for (j = M - i - 1; j > 0; j--) c = mul(c, c, p);
        R = mul(R, c, p);
        c = mul(c, c, p);
        t = mul(t, c, p);
        M = i;
    }
    r1 = R;
    r2 = p - R;
    return true;
}
```

```
bool check(LL k, LL p, LL &a, LL &b){
    LL r1, r2, x;
    if (!getSqr(p - k % p, p, r1, r2)) return false;
    x = (r1 < r2) ? r1 : r2;
    bool ok = false;
    LL y = p;
    while(true){
        if (x == 0) break;
        if (x * x < p){
            if ((p - x * x) % k == 0){
                LL t1 = (p - x * x) / k;
                LL t2 = (LL)(sqrt((long double)t1) + 1E-6);
                for(int i = -3; i <= 3; i++){
                    if ((t2 + i) * (t2 + i) == t1){
                        a = x;
                        b = t2;
                        ok = true;
                        break;
                    }
                }
            }
        }
        y %= x;
        swap(x, y);
    }
    if (k == 1){
        if (a > b) swap(a, b);
    }
    return ok;
}
```

2.1.5 高斯消元

```
//在异或方程里，要求最小改变次数，那就从后往前面枚举，先枚举只有变量之后，前面的变量就确定了
void gauss(int n, int m, double p[M][M]){
    static double tmp[M][M];
    static double *b[M];
    rep(i, n){
        rep(j, m){
            tmp[i][j] = p[i][j];
        }
    }
    rep(i, n){
        b[i] = tmp[i];
    }
    rep(i, n){
        REP(j, i, n){
            if (sign(fabs(b[j][i]) - fabs(b[i][i])) > 0) swap(b[i], b[j]);
```

```
        }
        rep(j, n){
            if (i == j) continue;
            double rate = b[j][i] / b[i][i];
            rep(k, m) b[j][k] -= b[i][k] * rate;
        }
        double rate = b[i][i];
        rep(j, m) b[i][j] /= rate;
    }
    rep(i, n){
        rep(j, m){
            p[i][j] = b[i][j];
        }
    }
}

//整数答案不超过LL，可以用辗转相除法做高斯消元
void gauss(int n, int m, int p[M][M], int& ret){
    static int tmp[M][M];
    static int *b[M];
    for(int i = 0; i < n; i++){
        for(int j = 0; j < m; j++){
            tmp[i][j] = p[i][j];
        }
    }
    for(int i = 0; i < n; i++) b[i] = tmp[i];
    ret = 1;
    for(int i = 0; i < n; i++){
        for(int j = i; j < n; j++){
            if (abs(b[j][i]) > abs(b[i][i])){
                ret *= -1;
                swap(b[i], b[j]);
            }
        }
        if (b[i][i] == 0){
            ret = 0;
            return;
        }
        for(int j = i + 1; j < n; j++){
            if (b[j][i] == 0) continue;
            while(b[j][i]){
                if (abs(b[i][i]) > abs(b[j][i])){
                    ret *= -1;
                    swap(b[i], b[j]);
                }
                int rate = b[j][i] / b[i][i];
             for(int k = i; k < m; k++) b[j][k] = (b[j][k] - 1LL * b[i][k] * rate) % mod;
            }
```

```
        }
        ret = 1LL * ret * b[i][i] % mod;
    }
    for(int i = 0; i < n; i++){
        for(int j = 0; j < m; j++){
            p[i][j] = b[i][j];
        }
    }
}
```

2.1.6 NNT mod 1000000007

```
#include <cstdio>
#include <vector>
#include <complex>
#include <cmath>
#include <cstring>

using namespace std;

typedef long long LL;

const int N = 1 << 17;
const int M = 3;
const int MO = 1000000007;

const int MOD[] = {998244353, 995622913, 786433};
const int ROOT[] = {3, 5, 10};
const LL M1 = 397550359381069386LL;
const LL M2 = 596324591238590904LL;
const LL MM = 993874950619660289LL;

LL mul(LL x,LL y,LL z){
    return (x * y - (LL)(x / (long double) z * y + 1e-3) * z + z) % z;
}

LL china(int x1, int x2){
    return (mul(M1, x1, MM) + mul(M2, x2, MM)) % MM;
}

int qmod(int a, int n, int p){
  int ret = 1;
  while(n){
    if (n & 1) ret = 1LL * ret * a % p;
    a = 1LL * a * a % p;
    n >>= 1;
  }
  return ret;
```

```
}

class NNT {
    public:
        NNT(int n, int mod, int root);
        void forward(int a[]) {
            work(a, r);
        }
        void reverse(int a[]) {
            work(a, ir);
            for (int i = 0; i < n; ++i) a[i] = 1LL * a[i] * n_rev % mod;
        }
    private:
        int n, p, mod, n_rev;
        vector<int> rb;
        int r[20];
        int ir[20];
        void work(int a[], int* roots);
};

NNT::NNT(int n, int mod, int root) : n(n) , mod(mod), rb(n) , p(0) {
    n_rev = qmod(n, mod - 2, mod);
    while ((1 << p) < n) ++p;
    for(int i = 0; i < n; i++){
        int x = i, y = 0;
        for (int j = 0; j < p; ++j) {
            y = (y << 1) | (x & 1);
            x >>= 1;
        }
        rb[i] = y;
    }
    int inv = qmod(root, mod - 2, mod);
    r[p - 1] = qmod(root, (mod - 1) / (1 << p), mod);
    ir[p - 1] = qmod(inv, (mod - 1) / (1 << p), mod);
    for(int i = p - 2; i >= 0; i--){
        r[i] = 1LL * r[i + 1] * r[i + 1] % mod;
        ir[i] = 1LL * ir[i + 1] * ir[i + 1] % mod;
    }
}

void NNT::work(int a[], int* r) {
    for (int i = 0; i < n; ++i) if (rb[i] > i) swap(a[i], a[rb[i]]);
    for (int len = 2; len <= n; len <<= 1) {
        int root = *r++;
        for (int i = 0; i < n; i += len) {
            int w = 1;
            for (int j = 0; j < len / 2; ++j) {
                int u = a[i + j];
```

```
                int v = 1LL * a[i + j + len / 2] * 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 = 1LL * w * root % mod;
            }
        }
    }
}

int n, k;
int cnt[5];
int ans[3][N];
int inv[N], fac[N];
int iv;
int a[3][N], b[N];
int top;
NNT *nnt[3];

void pre(){
  iv = qmod(1LL * MOD[0] * MOD[1] % MOD[2], MOD[2] - 2, MOD[2]);
  fac[0] = 1;
  for(int i = 1; i < N; i++){
    fac[i] = 1LL * fac[i - 1] * i % MO;
  }
  inv[N - 1] = qmod(fac[N - 1], MO - 2, MO);
  for(int i = N - 1; i > 0; i--){
    inv[i - 1] = 1LL * inv[i] * i % MO;
  }
}

int merge(int a, int b, int c){
  int ret;
  long long m1 = china(a, b);
  int m2 = c;
  int z = 1LL * ((m2 - m1) % MOD[2]) * iv % MOD[2];
  z = (z % MOD[2] + MOD[2]) % MOD[2];
  ret = (1LL * z * MOD[0] % MO * MOD[1] + m1) % MO;
  return (ret % MO + MO) % MO;
}

int work(){
  for(int i = 0; i < M; i++){
    int up = (cnt[0] > n) ? n : cnt[0];
    fill(a[i], a[i] + top, 0);
    memcpy(a[i], inv, (up + 1) * 4);
  }
  for(int i = 1; i < 5; i++){
    for(int j = 0; j < M; j++){
```

```
      nnt[j]->forward(a[j]);
      int up = (cnt[i] > n) ? n : cnt[i];
      fill(b, b + top, 0);
      memcpy(b, inv, (up + 1) * 4);
      nnt[j]->forward(b);
      for(int k = 0; k < top; k++) a[j][k] = 1LL * a[j][k] * b[k] % MOD[j];
      nnt[j]->reverse(a[j]);
    }
    for(int k = 0; k <= n; k++){
      int tmp = merge(a[0][k], a[1][k], a[2][k]);
      for(int j = 0; j < M; j++) ::a[j][k] = tmp;
    }
    for(int j = 0; j < M; j++){
      fill(a[j] + n + 1, a[j] + top, 0);
    }
  }
  int tmp = merge(a[0][n], a[1][n], a[2][n]);
  return 1LL * tmp * fac[n] % MO;
}

int main(){
  pre();
  int T;
  scanf("%d", &T);
  for(int cas = 1; cas <= T; cas++){
    scanf("%d", &n);
    for(int i = 0; i < 5; i++) scanf("%d", &cnt[i]);
    top = 1;
    while(top <= n * 2) top <<= 1;
    for(int i = 0; i < M; i++) nnt[i] = new NNT(top, MOD[i], ROOT[i]);
    int x = work();
    if (cnt[0]){
      cnt[0]--;
      n--;
      int y = work();
      x = (x - y) % MO;
    }
    x = (x % MO + MO) % MO;
    printf("Case #%d: %d\n", cas, x);
  }
}
```

2.1.7 NNT prime number

2281701377=17 227+1 是一个挺好的数，平方刚好不会爆 long long
1004535809=479 221+1 加起来刚好不会爆 int 也不错
下面是刚刚打出来的表格（g 是mod(r 2k+1)的原根）
r 2^k+1 r k g

```
3  1  1  2
5  1  2  2
17  1  4  3
97  3  5  5
193  3  6  5
257  1  8  3
7681  15  9  17
12289  3  12  11
40961  5  13  3
65537  1  16  3
786433  3  18  10
5767169  11  19  3
7340033  7  20  3
23068673  11  21  3
104857601  25  22  3
167772161  5  25  3
469762049  7  26  3
1004535809  479  21  3
2013265921  15  27  31
2281701377  17  27  3
3221225473  3  30  5
75161927681  35  31  3
77309411329  9  33  7
206158430209  3  36  22
2061584302081  15  37  7
2748779069441  5  39  3
6597069766657  3  41  5
39582418599937  9  42  5
79164837199873  9  43  5
263882790666241  15  44  7
1231453023109121  35  45  3
1337006139375617  19  46  3
3799912185593857  27  47  5
4222124650659841  15  48  19
7881299347898369  7  50  6
31525197391593473  7  52  3
180143985094819841  5  55  6
1945555039024054273  27  56  5
4179340454199820289  29  57  3
```

2.1.8 多项式运算 (exp,ln,sqrt,inv,div)

```
// BZOJ 3625
#include <algorithm>
#include <cstdio>

using std::swap;
using std::fill;
```

```
using std::copy;
using std::reverse_copy;
using std::reverse;

typedef int value_t;
typedef long long calc_t;
const int MaxN = 1 << 19;
const value_t mod_base = 119, mod_exp = 23;
const value_t mod_v = (mod_base << mod_exp) + 1;
const value_t primitive_root = 3;
int epsilon_num;
value_t eps[MaxN], inv_eps[MaxN], inv2;
value_t inv[MaxN];

value_t dec(value_t x, value_t v) { x -= v; return x < 0 ? x + mod_v : x; }
value_t inc(value_t x, value_t v) { x += v; return x >= mod_v ? x - mod_v : x; }
value_t pow(value_t x, value_t p) {
  value_t v = 1;
  for(; p; p >>= 1, x = (calc_t)x * x % mod_v)
    if(p & 1) v = (calc_t)x * v % mod_v;
  return v;
}

void init_eps(int num) {
  epsilon_num = num;
  value_t base = pow(primitive_root, (mod_v - 1) / num);
  value_t inv_base = pow(base, mod_v - 2);
  eps[0] = inv_eps[0] = 1;
  for(int i = 1; i != num; ++i) {
    eps[i] = (calc_t)eps[i - 1] * base % mod_v;
    inv_eps[i] = (calc_t)inv_eps[i - 1] * inv_base % mod_v;
  }
}

void transform(int n, value_t *x, value_t *w = eps) {
  for(int i = 0, j = 0; i != n; ++i) {
    if(i > j) swap(x[i], x[j]);
    for(int l = n >> 1; (j ^= l) < l; l >>= 1);
  }
  for(int i = 2; i <= n; i <<= 1) {
    int m = i >> 1, t = epsilon_num / i;
    for(int j = 0; j < n; j += i) {
      for(int p = 0, q = 0; p != m; ++p, q += t) {
        value_t z = (calc_t)x[j + m + p] * w[q] % mod_v;
        x[j + m + p] = dec(x[j + p], z);
        x[j + p] = inc(x[j + p], z);
      }
    }
```

```
  }
}

void inverse_transform(int n, value_t *x) {
  transform(n, x, inv_eps);
  value_t inv = pow(n, mod_v - 2);
  for(int i = 0; i != n; ++i) x[i] = (calc_t)x[i] * inv % mod_v;
}

void polynomial_inverse(int n, value_t *A, value_t *B) {
  static value_t T[MaxN];
  if(n == 1) {
    B[0] = pow(A[0], mod_v - 2);
    return;
  }

  int half = (n + 1) >> 1;
  polynomial_inverse(half, A, B);

  int p = 1;
  for(; p < n << 1; p <<= 1);

  fill(B + half, B + p, 0);
  transform(p, B);

  copy(A, A + n, T);
  fill(T + n, T + p, 0);
  transform(p, T);

 for(int i = 0; i != p; ++i) B[i] = (calc_t)B[i] * dec(2, (calc_t)T[i] * B[i] % mod_v) % mod_v;
  inverse_transform(p, B);
}

void polynomial_sqrt(int n, value_t *A, value_t *B) {
  static value_t T[MaxN];
  if(n == 1) {
    B[0] = 1; // sqrt A[0], here is 1
    return;
  }

  int p = 1;
  for(; p < n << 1; p <<= 1);

  int half = (n + 1) >> 1;
  polynomial_sqrt(half, A, B);
  fill(B + half, B + n, 0);
  polynomial_inverse(n, B, T);
  fill(T + n, T + p, 0);
```

```
  transform(p, T);

  fill(B + half, B + p, 0);
  transform(p >> 1, B);
  for(int i = 0; i != p >> 1; ++i) B[i] = (calc_t)B[i] * B[i] % mod_v;
  inverse_transform(p >> 1, B);
  for(int i = 0; i != n; ++i) B[i] = (calc_t)inc(A[i], B[i]) * inv2 % mod_v;
  transform(p, B);
  for(int i = 0; i != p; ++i) B[i] = (calc_t)B[i] * T[i] % mod_v;
  inverse_transform(p, B);
}

void polynomial_logarithm(int n, value_t *A, value_t *B) {
  static value_t T[MaxN];
  int p = 1;
  for(; p < n << 1; p <<= 1);
  polynomial_inverse(n, A, T);
  fill(T + n, T + p, 0);
  transform(p, T);

  // derivative
  copy(A, A + n, B);
  for(int i = 0; i < n - 1; ++i) B[i] = (calc_t)B[i + 1] * (i + 1) % mod_v;
  fill(B + n - 1, B + p, 0);
  transform(p, B);

  for(int i = 0; i != p; ++i) B[i] = (calc_t)B[i] * T[i] % mod_v;
  inverse_transform(p, B);

  // integral
  for(int i = n - 1; i; --i) B[i] = (calc_t)B[i - 1] * inv[i] % mod_v;
  B[0] = 0;
}

void polynomial_exponent(int n, value_t *A, value_t *B) {
  static value_t T[MaxN];
  if(n == 1) {
    B[0] = 1;
    return;
  }

  int p = 1;
  for(; p < n << 1; p <<= 1);

  int half = (n + 1) >> 1;
  polynomial_exponent(half, A, B);
  fill(B + half, B + p, 0);
```

```
  polynomial_logarithm(n, B, T);
  for(int i = 0; i != n; ++i) T[i] = dec(A[i], T[i]);
  T[0] = inc(T[0], 1);
  transform(p, T);
  transform(p, B);
  for(int i = 0; i != p; ++i) B[i] = (calc_t)B[i] * T[i] % mod_v;
  inverse_transform(p, B);
}

void polynomial_division(int n, int m, value_t *A, value_t *B, value_t *D, value_t *R) {
  static value_t A0[MaxN], B0[MaxN];

  int p = 1, t = n - m + 1;
  while(p < t << 1) p <<= 1;

  fill(A0, A0 + p, 0);
  reverse_copy(B, B + m, A0);
  polynomial_inverse(t, A0, B0);
  fill(B0 + t, B0 + p, 0);
  transform(p, B0);

  reverse_copy(A, A + n, A0);
  fill(A0 + t, A0 + p, 0);
  transform(p, A0);

  for(int i = 0; i != p; ++i) A0[i] = (calc_t)A0[i] * B0[i] % mod_v;
  inverse_transform(p, A0);
  reverse(A0, A0 + t);
  copy(A0, A0 + t, D);

  for(p = 1; p < n; p <<= 1);
  fill(A0 + t, A0 + p, 0);
  transform(p, A0);
  copy(B, B + m, B0);
  fill(B0 + m, B0 + p, 0);
  transform(p, B0);
  for(int i = 0; i != p; ++i) A0[i] = (calc_t)A0[i] * B0[i] % mod_v;
  inverse_transform(p, A0);
  for(int i = 0; i != m; ++i) R[i] = (A[i] - A0[i]) % mod_v;
  fill(R + m, R + p, 0);
}

value_t tmp[MaxN];
value_t A[MaxN], B[MaxN], C[MaxN], T[MaxN];

int main() {
  int n, m;
  std::scanf("%d %d", &n, &m);
```

```
  int min_v = ~0u >> 1;
  for(int i = 0; i != n; ++i) {
    std::scanf("%d", tmp + i);
    if(min_v > tmp[i]) min_v = tmp[i];
  }

  inv2 = mod_v - mod_v / 2;

  int p = 1;
  for(; p < (m + min_v + 1) << 1; p <<= 1);
  init_eps(p);

  A[0] = 1;
  for(int i = 0; i != n; ++i) {
    int x = tmp[i];
    T[x - min_v] = 2;
    A[x] = mod_v - 4;
  }

  polynomial_inverse(m + min_v + 1, T, C);
  polynomial_sqrt(m + min_v + 1, A, B);
  B[0] = dec(1, B[0]);
  for(int i = 1; i <= m + min_v; ++i) B[i] = mod_v - B[i];
  for(int i = 0; i <= m; ++i) B[i] = B[i + min_v];
  fill(B + m + 1, B + p, 0);
  fill(C + m + 1, C + p, 0);
  transform(p, B);
  transform(p, C);
  for(int i = 0; i != p; ++i) B[i] = (calc_t)B[i] * C[i] % mod_v;
  inverse_transform(p, B);
  for(int i = 1; i <= m; ++i) std::printf("%d\n", B[i]);
  return 0;
}
```