Counting Team Selection 2019. (Problem 2 of Day 1) Pearl

By Jiarui Zheng, 190623,

Problem (in Chinese): https://loj.ac/problem/3120

[Labels]

Algorithmic: Counting, Concrete Mathematics, Combinatorics, Fourier Transforms, Recurrent Equations, Generation Functions

Source: Counting Team Selection, 2019

[Problem Description]

There are n random integers in $[1, D]$. A pair is two equal integers. A sequence is valid only when m pairs can be selected.

Find the number of valid sequences of the total D^n sequences.

$$D \leq 10^5, \quad n, m \leq 10^9$$

[Solution]

Credit to: Laofu

Derivation

Obviously,
  when $n \leq 2m$, there are no valid sequences.
  when $d \leq n - 2m + 1$, all sequences are valid.

Let (d_i) be how many times i appeared in the sequence $a_1, a_2, \cdots, a_n$.

$$\because \textit{there are m pairs}$$

$$\therefore \left(\sum_{i=1}^{D} d_i \bmod 2\right) \leq (n - 2m)$$

Consider finding $m = \sum_{i=1}^{D} d_i \bmod 2$

Let $F(x) = \sum_{i=0}^{\infty} [i \bmod 2 = 1] \frac{x^i}{i!}, \quad G(x) = \sum_{i=0}^{\infty} [i \bmod 2 = 0] \frac{x^i}{i!}$

Use generating functions to simplify it

$$ans = \sum_{k=0}^{n-2m} \binom{D}{k} n! \, [x^n] \, F^i(x) G^{D-k}(x)$$

$$1+x+\frac{x^2}{2!}+\frac{x^3}{3!}+\ldots=e^x$$
$$1-x+\frac{x^2}{2!}-\frac{x^3}{3!}+\ldots=e^{-x}$$
$$1+\frac{x^2}{2!}+\frac{x^4}{4!}+\frac{x^6}{6!}+\ldots=\frac{e^x+e^{-x}}{2}$$
$$x+\frac{x^3}{3!}+\frac{x^5}{5!}+\frac{x^7}{7!}+\ldots=\frac{e^x-e^{-x}}{2}$$

$$\therefore F(x)=\frac{e^x-e^{-x}}{2},\quad G(x)=\frac{e^x+e^{-x}}{2}$$

The complexity is $\mathcal{O}\left(D^3\right)$. We can optimize it to $\mathcal{O}\left(D^2\right)$.

We can get $72pts$ using this method.

Obviously, except $(-1)^{k-i}$, it has nothing to do with i and j. We only need $i+j$.

$$ans=\frac{D!}{2^D}\sum_{k=0}^{n-2m}\sum_{i+j=0}^{D}\frac{(2(i+j)-D)^n}{(i+j)!(D-i-j)!}\sum_{i=0}^{k}\frac{(i+j)!}{i!j!}\cdot\frac{(D-i-j)!}{(k-i)!(D-k-j)!}(-1)^{k-i}$$

Consider $\frac{(x+y)!}{x!y!}=\binom{x+y}{x}$,

The equation converts into walking from $(0,0)$ to (i,j), then from (i,j) to $(k,d-k)$. Let x_i be the value $(-1)^{k-i}$ of approach i.

Use generating functions to solve the walking problem, in the form of $\left[x^k\right](1+x)^D$.

$$\begin{aligned}ans&=\frac{D!}{2^D}\sum_{i=0}^{D}\frac{(2i-D)^n}{i!(D-i)!}\sum_{k=0}^{n-2m}\left[x^k\right](1+x)^i(1-x)^{D-i}\\&=\frac{1}{2^D}\sum_{i=0}^{D}(2i-D)^n\binom{D}{i}\sum_{k=0}^{n-2m}\left[x^k\right](1+x)^i(1-x)^{D-i}\end{aligned}$$

Suppose that

$$F(i,D)=\sum_{k=0}^{n-2m}\left[x^k\right](1+x)^i(1-x)^{D-i}$$

$$\therefore ans=\frac{1}{2^D}\sum_{i=0}^{D}(2i-D)^n\binom{D}{i}F(i,D)$$

We need to find $F(0..n,D)$, consider the recurrent equation

$$F(0,0)=1$$

$$
\begin{aligned}
F(0,D) &= \sum_{k=0}^{n-2m} \left[x^k\right](1-x)^D \\
&= \sum_{k=0}^{n-2m} (-1)^k \binom{D}{k} \\
&= \sum_{k=0}^{n-2m} (-1)^k \left(\binom{D-1}{k}\right) + \binom{D-1}{k-1}) \\
&= \left(\sum_{k=0}^{n-2m} (-1)^k \binom{D-1}{k}\right) + \left(\sum_{k=0}^{n-2m-1} (-1)^k \binom{D-1}{k}\right) \\
&= (-1)^{n-2m} \binom{D-1}{n-2m}
\end{aligned}
$$

$$
\begin{aligned}
F(i,D) &= \sum_{k=0}^{n-2m} \left[x^k\right](1+x)^i(1-x)^{D-i} \\
&= \sum_{k=0}^{n-2m} \left[x^k\right](-(1-x)+2)(1+x)^{i-1}(1-x)^{D-i} \\
&= \sum_{k=0}^{n-2m} \left[x^k\right] - \left((1+x)^{i-1}(1-x)^{D-i+1}\right) + 2\left(2(1+x)^{i-1}(1-x)^{D-i}\right) \\
&= -F(i-1,D) + 2F(i-1,D-1)
\end{aligned}
$$

Considering the walking problem, we can find that we can start from any point $\in [(0,1),(0,2),\cdots,(0,D)]$ and take the value of $F(0,1..D)$. There are two moving ways: the first, walk down and times the value -1; the second, walk down-right and times the value 2.

Obviously, every step will increase x by 1, equivalent to choosing $\Delta y = D-j$ from the i steps, walking down-right.

$$
F(i,D) = \sum_{j=0}^{D} F(0,j)(-1)^{i+j-D} 2^{D-j} \binom{i}{D-j}
$$

Therefore

$$
\begin{aligned}
ans &= \frac{1}{2^D} \sum_{i=0}^{D} (2i-D)^n \binom{D}{i} F(i,D) \\
&= \frac{1}{2^D} \sum_{i=0}^{D} (2i-D)^n \binom{D}{i} \sum_{j=0}^{D} F(0,j)(-1)^{i+j-D} 2^{D-j} \binom{i}{D-j} \\
&= \frac{(-1)^D D!}{2^D} \sum_{i=0}^{D} \sum_{j=0}^{D} \frac{1}{(i+j-D)!} \cdot \frac{(2i-D)^n}{(D-i)!} \cdot \frac{2^{D-j} F(0,j)}{(D-j)!}
\end{aligned}
$$

Let

$$f(x) = \sum_{i=0}^{D} \frac{(2i-D)^n x^i}{(D-i)!}$$
$$g(x) = \sum_{i=0}^{D} \frac{2^{D-j} F(0,i) x^i}{(D-i)!}$$

Therefore

$$ans = \frac{1}{2^D} \sum_{k=D}^{2D} \frac{\left[x^k\right]\left(f(x) \cdot g(x)\right)}{(-1)^k (k-D)!}$$

Consider using Fourier Transforms.

Code

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

#define RG register
#define IL inline
#define ll long long
#define ld long double
#define ui unsigned int
#define ull unsigned long long
#define LL long long
//#define FILE
//#define DEBUG

namespace stdoier
{
template <typename T>
IL T max(RG const T &a, RG const T &b)
{
    if (a > b)
        return a;
    else
        return b;
}

template <typename T>
IL T min(RG const T &a, RG const T &b)
{
    if (a < b)
        return a;
    else
        return b;
}

template <typename T>
IL void cmin(RG T &a, RG const T &b)
{
    if (a > b)
        a = b;
}
```

```
template <typename T>
IL void cmax(RG T &a, RG const T &b)
{
    if (a < b)
        a = b;
}
} // namespace stdoier

using namespace stdoier;

namespace io
{
const int MaxBuff = 1 << 15;
const int MaxOut = 1 << 24;
char b[MaxBuff], *S = b, *T = b;

#define getc() (S == T && (T = (S = b) + fread(b, 1, MaxBuff, stdin),
S == T) ? 0 : *S++)

template <class Type>
IL Type read()
{
    RG char ch;
    RG Type ans = 0;
    RG bool neg = 0;
    while (ch = getc(), (ch < '0' || ch > '9') && ch != '-')
        ;
    ch == '-' ? neg = 1 : ans = ch - '0';
    while (ch = getc(), '0' <= ch && ch <= '9')
        ans = ans * 10 + ch - '0';
    return neg ? -ans : ans;
}

IL int gets(RG char *s)
{
    RG char *iter = s;
    while (*iter = getc(), *iter == ' ' || *iter == '\n' || *iter ==
'\r')
        ;
    while (*++iter = getc(), *iter && *iter != ' ' && *iter != '\n' &&
*iter != '\r')
        ;
    *iter = 0;
    return iter - s;
}

char buff[MaxOut], *iter = buff;

template <class T>
IL void writeln(RG T x, RG char ch = '\n')
{
    static int stack[110];
    RG int O = 0;
    RG char *iter = io::iter;
    if (!x)
```

```
            *iter++ = '0';
        else
        {
            (x < 0) ? x = -x, *iter++ = '-' : 1;
            for (; x; x /= 10)
                stack[++O] = x % 10;
            for (; O; *iter++ = '0' + stack[O--])
                ;
        }
        *iter++ = ch, io::iter = iter;
    }

    template <class T>
    IL void write(RG T x)
    {
        static int stack[110];
        RG int O = 0;
        RG char *iter = io::iter;
        if (!x)
            *iter++ = '0';
        else
        {
            (x < 0) ? x = -x, *iter++ = '-' : 1;
            for (; x; x /= 10)
                stack[++O] = x % 10;
            for (; O; *iter++ = '0' + stack[O--])
                ;
        }
        io::iter = iter;
    }

    IL void puts(RG const char *s)
    {
        while (*s)
            *iter++ = *s++;
    }

    struct Output
    {
        ~Output() { fwrite(buff, 1, iter - buff, stdout), iter = buff; }
    } output_hlpr;
    } // namespace io

    namespace solve_the_problem
    {
    int (*in)() = io::read<int>;
    const int N = 800003, mod = 998244353, g = 3, gi = 332748118;

    int D, n, m, F[N], G[N];
    int rev[N], fac[N], inv[N], invfac[N], ans;

    inline int POW(int a, int b)
    {
        int res = 1;
        while (b)
```

```cpp
    {
        if (b & 1)
            res = (ll)res * a % mod;
        a = (ll)a * a % mod;
        b >>= 1;
    }
    return res;
}

inline void init(int n)
{
    fac[0] = 1;
    for (int i = 1; i <= n; i++)
        fac[i] = (ll)fac[i - 1] * i % mod;
    invfac[n] = POW(fac[n], mod - 2);
    for (int i = n; i; i--)
    {
        invfac[i - 1] = (ll)i * invfac[i] % mod;
        inv[i] = (ll)invfac[i] * fac[i - 1] % mod;
    }
}

inline int calrev(int n)
{
    int limit = 1, L = -1;
    while (limit <= n)
    {
        limit <<= 1;
        L++;
    }
    for (int i = 0; i < limit; i++)
        rev[i] = (rev[i >> 1] >> 1) | ((i & 1) << L);
    return limit;
}

inline void NTT(int *A, int limit, int type)
{
    for (int i = 0; i < limit; i++)
        if (i < rev[i])
            std ::swap(A[i], A[rev[i]]);
    for (int mid = 1; mid < limit; mid <<= 1)
    {
        int Wn = POW(type == 1 ? g : gi, (mod - 1) / (mid << 1));
        for (int j = 0; j < limit; j += (mid << 1))
        {
            int w = 1;
            for (int k = 0; k < mid; k++, w = (ll)w * Wn % mod)
            {
                int x = A[j + k], y = (ll)w * A[j + k + mid] % mod;
                A[j + k] = (x + y) % mod;
                A[j + k + mid] = (x - y + mod) % mod;
            }
        }
    }
    if (type == -1)
```

```
    {
        int inv = POW(limit, mod - 2);
        for (int i = 0; i < limit; i++)
            A[i] = (ll)A[i] * inv % mod;
    }
}

void main()
{
    D = in(), n = in(), m = in();
    if (n < 2 * m)
    {
        io::puts("0");
        return;
    }
    if (D <= n - 2 * m)
    {
        io::writeln(POW(D, n));
        return;
    }
    init(D);
    for (int i = 0; i <= D; i++)
    {
        F[i] = (ll)POW((D - 2 * i + mod) % mod, n) * invfac[i] % mod;
        if (i & 1)
            F[i] = mod - F[i];
        G[i] = invfac[i];
    }
    int limit = calrev(D << 1);
    NTT(F, limit, 1);
    NTT(G, limit, 1);
    for (int i = 0; i < limit; i++)
        F[i] = (ll)F[i] * G[i] % mod;
    NTT(F, limit, -1);
    for (int i = 0; i < limit; i++)
        G[i] = 0;
    for (int i = D + 1; i < limit; i++)
        F[i] = 0;
    for (int i = 0; i <= D; i++)
    {
        F[i] = (ll)F[i] * fac[i] % mod * fac[D] % mod * POW(2, mod - 1
- i) % mod * invfac[D - i] % mod;
        G[D - i] = (i & 1) ? (mod - invfac[i]) : invfac[i];
    }
    NTT(F, limit, 1);
    NTT(G, limit, 1);
    for (int i = 0; i < limit; i++)
        F[i] = (ll)F[i] * G[i] % mod;
    NTT(F, limit, -1);
    for (int i = 0; i <= n - 2 * m; i++)
        ans = (ans + (ll)F[i + D] * invfac[i] % mod) % mod;
    io::writeln(ans);
}
} // namespace solve_the_problem
```

```
int main()
{
    solve_the_problem::main();

    return 0;
}
```

[Solution 2]

Credit to: Yanru Guan

Derivation

Obviously,
  when $n \le 2m$, there are no valid sequences.
  when $d \le n - 2m + 1$, all sequences are valid.

Let (d_i) be how many times i appeared in the sequence $a_1, a_2, \cdots, a_n$.

$$\because \textit{there are } m \textit{ pairs}$$

$$\therefore \left(\sum_{i=1}^{D} d_i \bmod 2 \right) \le (n - 2m)$$

Consider finding $m = \sum_{i=1}^{D} d_i \bmod 2$

Let $F(x) = \sum_{i=0}^{\infty} [i \bmod 2 = 1] \frac{x^i}{i!}, \quad G(x) = \sum_{i=0}^{\infty} [i \bmod 2 = 0] \frac{x^i}{i!}$

Use generating functions to simplify it

$$ans = \sum_{k=0}^{n-2m} \binom{D}{k} n! \, [x^n] \, F^i(x) G^{D-k}(x)$$

$$\begin{aligned} 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots &= e^x \\ 1 - x + \frac{x^2}{2!} - \frac{x^3}{3!} + \ldots &= e^{-x} \\ 1 + \frac{x^2}{2!} + \frac{x^4}{4!} + \frac{x^6}{6!} + \ldots &= \frac{e^x + e^{-x}}{2} \\ x + \frac{x^3}{3!} + \frac{x^5}{5!} + \frac{x^7}{7!} + \ldots &= \frac{e^x - e^{-x}}{2} \end{aligned}$$

$$\therefore F(x) = \frac{e^x - e^{-x}}{2}, \quad G(x) = \frac{e^x + e^{-x}}{2}$$

The complexity is $\mathcal{O}\left(D^3\right)$.

What we need is: $[x^n]$ (x^n's coefficient) and y (the number of integers that appear odd times).

Obviously, only when x has been chosen and y doesn't exceed $n - 2m$, we need to count the answer.

Thus

$$n!\sum_{k=0}^{n-2m}(\frac{e^x+e^{-x}}{2}+y\frac{e^x-e^{-x}}{2})^D[x^n][y^k]$$
$$=n!\frac{1}{2^D}\sum_{k=0}^{n-2m}(e^x(1+y)+e^{-x}(1-y))^D[x^n][y^k]$$

Therefore

$$=n!\frac{1}{2^D}\sum_{k=0}^{n-2m}\sum_{i=0}^{D}(e^x(1+y))^i(e^{-x}(1-y))^{D-i}[x^n][y^k]$$
$$=n!\frac{1}{2^D}\sum_{k=0}^{n-2m}\sum_{i=0}^{D}\binom{D}{i}e^{(2i-D)x}(1+y)^i(1-y)^{D-i}[x^n][y^k]$$
$$=\frac{1}{2^D}\sum_{i=0}^{D}\binom{D}{i}(2i-D)^n\sum_{k=0}^{n-2m}(1+y)^i(1-y)^{D-i}[y^k]$$

When i is a constant, we only need to consider $S=\sum_{k=0}^{n-2m}(1+y)^i(1-y)^{D-i}[y^k]$.

When $i=d$, we can find k.

When $i\neq D$, we can change $[y^k]$ into another form.

$$S=(1+y)^i(1-y)^{D-i}(1+y+y^2+\dots)[y^{n-2m}]$$

Therefore $S=\sum_{j=0}^{k}\binom{i}{j}\binom{d-i}{k-j}$

Let $d=D=1,k=n-2m$

$$\sum_{j=0}^{k}\binom{i}{j}\binom{d-i}{k-j}=\sum_{j=0}^{k}\frac{i!}{j!(i-j)!}\cdot\frac{(d-i)!}{(k-j)!(d-k-i+j)}$$

Consider using Fourier Transforms.

Don't forget $i=D$ and $\frac{1}{2^D}\binom{D}{i}(2i-D)^n$!

Code

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

#define RG register
#define IL inline
#define ll long long
#define ld long double
#define ui unsigned int
#define ull unsigned long long
#define LL long long
//#define FILE
//#define DEBUG

namespace stdoier
{
```

```
template <typename T>
IL T max(RG const T &a, RG const T &b)
{
    if (a > b)
        return a;
    else
        return b;
}

template <typename T>
IL T min(RG const T &a, RG const T &b)
{
    if (a < b)
        return a;
    else
        return b;
}

template <typename T>
IL void cmin(RG T &a, RG const T &b)
{
    if (a > b)
        a = b;
}

template <typename T>
IL void cmax(RG T &a, RG const T &b)
{
    if (a < b)
        a = b;
}
} // namespace stdoier

using namespace stdoier;

namespace io
{
const int MaxBuff = 1 << 15;
const int MaxOut = 1 << 24;
char b[MaxBuff], *S = b, *T = b;

#define getc() (S == T && (T = (S = b) + fread(b, 1, MaxBuff, stdin),
S == T) ? 0 : *S++)

template <class Type>
IL Type read()
{
    RG char ch;
    RG Type ans = 0;
    RG bool neg = 0;
    while (ch = getc(), (ch < '0' || ch > '9') && ch != '-')
        ;
    ch == '-' ? neg = 1 : ans = ch - '0';
    while (ch = getc(), '0' <= ch && ch <= '9')
        ans = ans * 10 + ch - '0';
```

```
    return neg ? -ans : ans;
}

IL int gets(RG char *s)
{
    RG char *iter = s;
    while (*iter = getc(), *iter == ' ' || *iter == '\n' || *iter ==
'\r')
        ;
    while (*++iter = getc(), *iter && *iter != ' ' && *iter != '\n' &&
*iter != '\r')
        ;
    *iter = 0;
    return iter - s;
}

char buff[MaxOut], *iter = buff;

template <class T>
IL void writeln(RG T x, RG char ch = '\n')
{
    static int stack[110];
    RG int O = 0;
    RG char *iter = io::iter;
    if (!x)
        *iter++ = '0';
    else
    {
        (x < 0) ? x = -x, *iter++ = '-' : 1;
        for (; x; x /= 10)
            stack[++O] = x % 10;
        for (; O; *iter++ = '0' + stack[O--])
            ;
    }
    *iter++ = ch, io::iter = iter;
}

template <class T>
IL void write(RG T x)
{
    static int stack[110];
    RG int O = 0;
    RG char *iter = io::iter;
    if (!x)
        *iter++ = '0';
    else
    {
        (x < 0) ? x = -x, *iter++ = '-' : 1;
        for (; x; x /= 10)
            stack[++O] = x % 10;
        for (; O; *iter++ = '0' + stack[O--])
            ;
    }
    io::iter = iter;
}
```

```
IL void puts(RG const char *s)
{
    while (*s)
        *iter++ = *s++;
}

struct Output
{
    ~Output() { fwrite(buff, 1, iter - buff, stdout), iter = buff; }
} output_hlpr;
} // namespace io

namespace solve_the_problem
{
int (*in)() = io::read<int>;
const int maxn = (5e5) + 10;
const ll mod = 998244353;

int n, m, D, N, d, k;
ll jc[maxn], iv[maxn], ivjc[maxn];
ll a[maxn], b[maxn], fa[maxn], fb[maxn];
ll rev[maxn], ans, tmp;

ll ksm(ll x, ll y)
{
    if (x < 0)
        x += mod;
    ll res = 1;
    while (y)
    {
        if (y & 1)
            res = res * x % mod;
        x = x * x % mod;
        y /= 2;
    }
    return res;
}

ll C(int x, int y)
{
    return jc[x] * ivjc[y] % mod * ivjc[x - y] % mod;
}

void init(int n)
{
    N = 1;
    int lg = 0;
    while (N < n)
        N *= 2, lg++;
    for (int i = 1; i <= N; i++)
        for (int j = 1, ii = i; j <= lg; j++, ii /= 2)
            rev[i] = rev[i] * 2 + (ii % 2);
}
```

```
void fft(ll *a, ll *out, int flag)
{
    static ll tmp[maxn];
    for (int i = 0; i < N; i++)
        tmp[rev[i]] = a[i];
    for (int step = 1; step < N; step *= 2)
    {
        ll wn = ksm(3, (mod - 1) / (step * 2));
        if (flag == -1)
            wn = ksm(wn, mod - 2);
        for (int i = 0; i < N; i += step * 2)
        {
            ll w = 1;
            for (int k = i; k < i + step; k++)
            {
                ll u = tmp[k], v = tmp[k + step] * w % mod;
                tmp[k] = (u + v) % mod;
                tmp[k + step] = (u - v + mod) % mod;
                w = w * wn % mod;
            }
        }
    }
    for (int i = 0; i < N; i++)
        out[i] = tmp[i];
    if (flag == -1)
    {
        ll t = ksm(N, mod - 2);
        for (int i = 0; i < N; i++)
            out[i] = out[i] * t % mod;
    }
}

void main()
{
    jc[0] = iv[0] = ivjc[0] = 1;
    jc[1] = iv[1] = ivjc[1] = 1;
    for (int i = 2; i < maxn / 5; i++)
    {
        jc[i] = jc[i - 1] * i % mod;
        iv[i] = (mod - mod / i) * iv[mod % i] % mod;
        ivjc[i] = ivjc[i - 1] * iv[i] % mod;
    }
    D = in(), n = in(), m = in();
    d = D - 1;
    k = min(D, n - 2 * m);
    for (int j = 0; j <= k; j++)
    {
        a[j] = ivjc[j] * ivjc[k - j] % mod;
        if ((k - j) & 1)
            a[j] = mod - a[j];
    }
    for (int j = 0; j <= d - k; j++)
        b[j] = ivjc[j] * ivjc[d - k - j] % mod;
    init(d * 2);
    fft(a, fa, 1);
```

```
    fft(b, fb, 1);
    for (int i = 0; i < N; i++)
        fa[i] = fa[i] * fb[i] % mod;
    fft(fa, a, -1);
    for (int i = 0; i <= d; i++)
        a[i] = a[i] * jc[i] % mod * jc[d - i] % mod;
    for (int i = 0; i <= d; i++)
        ans = (ans + a[i] * C(D, i) % mod * ksm(2 * i - D, n) % mod) %
mod;
    for (int i = 0; i <= k; i++)
        tmp = (tmp + C(D, i)) % mod;
    ans = (ans + tmp * ksm(D, n) % mod) % mod;
    io::writeln(ans * ksm(ksm(2, mod - 2), D) % mod);
}
} // namespace solve_the_problem

int main()
{
    solve_the_problem::main();

    return 0;
}
```