组合数学

组合数

```
i64 qpow(i64 x, i64 p) {
    i64 ret = 1;
    while (p) {
        if (p & 1)ret = ret * x % mod;
        p >>= 1;
        x = x * x % mod;
    }
    return ret;
}

#define inv(x) qpow(x,mod-2)

std::vector<int> fact(1, 1);
std::vector<int> inv_fact(1, 1);

auto get_fact(int x, bool inv = 0) {
    while ((int)fact.size() < x + 1) {
        fact.push_back(1ll * fact.back() * fact.size() % mod);
        inv_fact.push_back(inv(fact.back()));
    }
    return (inv ? inv_fact[x] : fact[x]);
}

auto get_inv_fact(int x) { return get_fact(x, 1); }

i64 C(int n, int k) {
    if (k<0 || k>n)return 0;
    return 1ll * get_fact(n) * get_inv_fact(k) % mod * get_inv_fact(n - k) % mod;
}

i64 A(int n, int k) {
    return 1ll * get_fact(n) * get_inv_fact(n - k) % mod;
}
```

```
i64 F(int n) { return get_fact(n); }
```

递推法求组合数

```
int c[N][N];
for (int i = 0; i < N; i ++ )
    for (int j = 0; j <= i; j ++ )
        if (!j) c[i][j] = 1;
        else c[i][j] = (c[i - 1][j] + c[i - 1][j - 1]) % mod;
```

范德蒙德卷积

$$\sum_{i=0}^{k}\binom{n}{i}\binom{m}{k-i}=\binom{n+m}{k}$$

推论

推论 1 及证明

$$\sum_{i=-r}^{s}\binom{n}{r+i}\binom{m}{s-i}=\binom{n+m}{r+s}$$

证明与原公式证明相似。

推论 2 及证明

$$\sum_{i=1}^{n}\binom{n}{i}\binom{n}{i-1}=\binom{2n}{n-1}$$

根据基础的组合数学知识推导，有：

$$\sum_{i=1}^{n}\binom{n}{i}\binom{n}{i-1}=\sum_{i=0}^{n-1}\binom{n}{i+1}\binom{n}{i}=\sum_{i=0}^{n-1}\binom{n}{n-1-i}\binom{n}{i}=\binom{2n}{n-1}$$

推论 3 及证明

$$\sum_{i=0}^{n}\binom{n}{i}^2=\binom{2n}{n}$$

根据基础的组合数学知识推导，有：

$$\sum_{i=0}^{n}\binom{n}{i}^2=\sum_{i=0}^{n}\binom{n}{i}\binom{n}{n-i}=\binom{2n}{n}$$

推论 4 及证明

$$\sum_{i=0}^{m}\binom{n}{i}\binom{m}{i}=\binom{n+m}{m}$$

根据基础的组合数学知识推导，有：

$$\sum_{i=0}^{m}\binom{n}{i}\binom{m}{i}=\sum_{i=0}^{m}\binom{n}{i}\binom{m}{m-i}=\binom{n+m}{m}$$

其中 $\binom{n+m}{m}$ 是我们较为熟悉的网格图路径计数的方案数。所以我们可以考虑其组合意义的证明。

在一张网格图中，从 $(0,0)$ 走到 (n,m) 共走 $n+m$ 步。规定 $(0,0)$ 位于网格图左上角，其中向下走了 n 步，向右走了 m 步，方案数为 $\binom{n+m}{m}$。

换个视角，我们将 $n+m$ 步拆成两部分走，先走 n 步，再走 m 步，那么 n 步中若有 i 步向右，则 m 步中就有 $m-i$ 步向右，故得证。

lucas定理

Lucas 定理内容如下：对于质数 p，有

$$\binom{n}{m}\bmod p=\binom{\lfloor n/p\rfloor}{\lfloor m/p\rfloor}\cdot\binom{n\bmod p}{m\bmod p}\bmod p$$

观察上述表达式，可知 $n\bmod p$ 和 $m\bmod p$ 一定是小于 p 的数，可以直接求解，$\binom{\lfloor n/p\rfloor}{\lfloor m/p\rfloor}$ 可以继续用 Lucas 定理求解。这也就要求 p 的范围不能够太大，一般在 10^5 左右。边界条件：当 $m=0$ 的时候，返回 1。

时间复杂度为 $O(f(p)+g(n)\log n)$，其中 $f(n)$ 为预处理组合数的复杂度，$g(n)$ 为单次求组合数的复杂度。

```
long long Lucas(long long n, long lm, long long p) {
  if (m == 0) return 1;
  return (C(n % p, m % p, p) * Lucas(n / p, m / p, p)) % p;
}
```

```
def Lucas(n, m, p):
    if m == 0:
        return 1
    return (C(n % p, m % p, p) * Lucas(n // p, m // p, p)) % p
```

卡特兰数

1. $f[n]=f[0]*f[n-1]+f[1]*f[n-2]+...+f[n-1]*f[0](n\geq 2)$
2. $h[n]=h[n-1]*(4*n-2)/(n+1)$
3. $h[n]=C[2n,n]/(n+1)(n=0,1,2,...)$
4. $C[m,n]=C[m-1,n-1]+C[m-1,n]$
5. $h[n]=C[2n,n]-C[2n,n-1](n=0,1,2,...)$

```
int main()
{
    f[0]=f[1]=1;
    scanf("%d",&n);
    for(int i=2;i<=n;i++)
    {
        f[i]+=f[i-1]*(4*i-2)/(i+1);
    }
    printf("%lld",f[n]);
    return 0;
}
```

n个球 全部放入 m个盒

I：球互不相同，盒子互不相同。

每个球都有m种选择，根据乘法原理，答案是m^n

II：球互不相同，盒子互不相同，每个盒子至多装一个球。

$n > m$, 放不下，0 种可能
$n <= m$, $A(m, n)$

III：球互不相同，盒子互不相同，每个盒子至少装一个球。

容斥枚举空盒个数

$$\sum_{i=0}^{m} (-1)^i * C(m, i) * (m - i)^n$$

IV：球互不相同，盒子全部相同。

枚举几个盒子有球，第二类斯特林数，设f[k][b]为将k个互异元素分为b个不为空的集合

$$ans = \sum_{i=0}^{m} f[n][i]$$

V：球互不相同，盒子全部相同，每个盒子至多装一个球。

$n > m, 0$
$n <= m, 1$

VI：球互不相同，盒子全部相同，每个盒子至少装一个球。

正是第二类斯特林数的定义，答案是 $f[n][m]$

VII：球全部相同，盒子互不相同。

插板法， $C(n + m - 1, m - 1)$

VIII：球全部相同，盒子互不相同，每个盒子至多装一个球。

选n个盒子放球， $C(m, n)$

IX：球全部相同，盒子互不相同，每个盒子至少装一个球。

先把每个盒子放一个球，然后转化为第VII个问题，插板法
$C(n - 1, m - 1)$

X：球全部相同，盒子全部相同。

问题等价于将 $n+m$ 拆分为 m 个无序的正整数，根据 $Ferrers$ 图的理论，
等价于将 $n+m$ 拆分成若干个不超过 m 的正整数，直接生成函数做。

$$T(n,m)=T(n,m-1)+T(n-m,m)$$

$$[x^{n+m-m}]\prod_{i=1}^{m}\frac{1}{1-x^i}$$

XI：球全部相同，盒子全部相同，每个盒子至多装一个球。

同 V

XII：球全部相同，盒子全部相同，每个盒子至少装一个球。

$$[x^{n-m}]\prod_{i=1}^{m}\frac{1}{1-x^i}$$

code:

```cpp
#include <algorithm>
#include <cstdio>
int n, m, fac[400010], minv[400010];
int const mod = 998244353, g = 3, gi = (mod + 1) / g;
int C(int x, int y)
{
    if (x < 0 || y < 0 || x < y)
        return 0;
    else
        return 1ll * fac[x] * minv[y] % mod * minv[x - y] % mod;
}
int pow(int x, int y)
{
    int res = 1;
    while (y) {
        if (y & 1)
            res = 1ll * res * x % mod;
        x = 1ll * x * x % mod;
        y >>= 1;
    }
    return res;
}
struct NTT {
    int r[800010], lim;
    NTT()
        : r()
        , lim()
    {
    }
    void getr(int lm)
    {
        lim = lm;
        for (int i = 0; i < lim; i++)
            r[i] = (r[i >> 1] >> 1) | ((i & 1) * (lim >> 1));
    }
    void operator()(int* a, int type)
    {
        for (int i = 0; i < lim; i++)
            if (i < r[i])
                std::swap(a[i], a[r[i]]);
        for (int mid = 1; mid < lim; mid <<= 1) {
            int rt = pow(type == 1 ? g : gi, (mod - 1) / (mid << 1));
            for (int j = 0, r = mid << 1; j < lim; j += r) {
```

```
                int p = 1;
                for (int k = 0; k < mid; k++, p = 1ll * p * rt % mod) {
                    int x = a[j + k], y = 1ll * a[j + mid + k] * p % mod;
                    a[j + k] = (x + y) % mod, a[j + mid + k] = (x - y + mod) % mod;
                }
            }
        }
        if (type == -1)
            for (int i = 0, p = pow(lim, mod - 2); i < lim; i++)
                a[i] = 1ll * a[i] * p % mod;
    }
} ntt;
void inv(int const* a, int* ans, int n)
{
    static int tmp[800010];
    for (int i = 0; i < n << 1; i++)
        tmp[i] = ans[i] = 0;
    ans[0] = pow(a[0], mod - 2);
    for (int m = 2; m <= n; m <<= 1) {
        int lim = m << 1;
        ntt.getr(lim);
        for (int i = 0; i < m; i++)
            tmp[i] = a[i];
        ntt(tmp, 1), ntt(ans, 1);
        for (int i = 0; i < lim; i++)
            ans[i] = ans[i] * (2 - 1ll * ans[i] * tmp[i] % mod + mod) % mod, tmp[i] = 0;
        ntt(ans, -1);
        for (int i = m; i < lim; i++)
            ans[i] = 0;
    }
}
void inte(int const* a, int* ans, int n)
{
    for (int i = n - 1; i; i--)
        ans[i] = 1ll * a[i - 1] * pow(i, mod - 2) % mod;
    ans[0] = 0;
}
void der(int const* a, int* ans, int n)
{
    for (int i = 1; i < n; i++)
        ans[i - 1] = 1ll * i * a[i] % mod;
    ans[n - 1] = 0;
}
```

```
void ln(int const* a, int* ans, int n)
{
    static int b[800010];
    for (int i = 0; i < n << 1; i++)
        ans[i] = b[i] = 0;
    inv(a, ans, n);
    der(a, b, n);
    int lim = n << 1;
    ntt.getr(lim);
    ntt(b, 1), ntt(ans, 1);
    for (int i = 0; i < lim; i++)
        b[i] = 1ll * ans[i] * b[i] % mod, ans[i] = 0;
    ntt(b, -1);
    for (int i = n; i < lim; i++)
        b[i] = 0;
    inte(b, ans, n);
}
void exp(int const* a, int* ans, int n)
{
    static int f[800010];
    for (int i = 0; i < n << 1; i++)
        ans[i] = f[i] = 0;
    ans[0] = 1;
    for (int m = 2; m <= n; m <<= 1) {
        int lim = m << 1;
        ln(ans, f, m);
        f[0] = (a[0] + 1 - f[0] + mod) % mod;
        for (int i = 1; i < m; i++)
            f[i] = (a[i] - f[i] + mod) % mod;
        ntt.getr(lim);
        ntt(f, 1), ntt(ans, 1);
        for (int i = 0; i < lim; i++)
            ans[i] = 1ll * ans[i] * f[i] % mod, f[i] = 0;
        ntt(ans, -1);
        for (int i = m; i < lim; i++)
            ans[i] = 0;
    }
}
void solve1() { printf("%d\n", pow(m, n)); }
void solve2()
{
    if (m < n)
        puts("0");
```

```
    else
        printf("%lld\n", 1ll * fac[m] * minv[m - n] % mod);
}
void solve3()
{
    if (n < m)
        return puts("0"), void();
    int ans = 0;
    for (int i = 0; i <= m; i++)
        ans = (ans + 1ll * pow(mod - 1, i) * C(m, i) % mod * pow(m - i, n)) % mod;
    printf("%d\n", ans);
}
int s[800010];
void solve4()
{
    static int tmp[800010];
    for (int i = 0; i <= n; i++)
        tmp[i] = (i & 1 ? mod - 1ll : 1ll) * minv[i] % mod, s[i] = 1ll * pow(i, n) * minv[i] % r
    int lim = 1;
    for (lim = 1; lim <= n + n; lim <<= 1)
        ;
    ntt.getr(lim);
    ntt(tmp, 1), ntt(s, 1);
    for (int i = 0; i < lim; i++)
        s[i] = 1ll * s[i] * tmp[i] % mod;
    ntt(s, -1);
    for (int i = n + 1; i < lim; i++)
        s[i] = 0;
    int ans = 0;
    for (int i = 0; i <= m; i++)
        ans = (ans + s[i]) % mod;
    printf("%d\n", ans);
}
void solve5() { printf("%d\n", int(m >= n)); }
void solve6() { printf("%d\n", s[m]); }
void solve7() { printf("%d\n", C(n + m - 1, m - 1)); }
void solve8() { printf("%d\n", C(m, n)); }
void solve9() { printf("%d\n", C(n - 1, m - 1)); }
int ans[800010];
void solve10()
{
    static int tmp[800010];
    for (int i = 1; i <= m; i++)
```

```
        for (int j = 1; j * i <= n; j++)
            ans[i * j] = (ans[i * j] - 1ll * minv[j] * fac[j - 1] % mod + mod) % mod;
    int lim = 1;
    for (; lim <= n; lim <<= 1)
        ;

    exp(ans, tmp, lim);
    for (int i = 0; i < lim; i++)
        ans[i] = 0;
    inv(tmp, ans, lim);
    printf("%d\n", ans[n]);
}
void solve11() { printf("%d\n", int(m >= n)); }
void solve12()
{
    printf("%d\n", n - m >= 0 ? ans[n - m] : 0);
}
int main()
{
    scanf("%d%d", &n, &m);
    fac[0] = 1;
    for (int i = 1; i <= n + m; i++)
        fac[i] = 1ll * fac[i - 1] * i % mod;
    minv[n + m] = pow(fac[n + m], mod - 2);
    for (int i = n + m; i; i--)
        minv[i - 1] = 1ll * minv[i] * i % mod;
    solve1();
    solve2();
    solve3();
    solve4();
    solve5();
    solve6();
    solve7();
    solve8();
    solve9();
    solve10();
    solve11();
    solve12();
    return 0;
}
```