mahjong

题目描述

ame 是一个可爱的女孩子，她喜欢打麻将。

ame 找来了一副特殊的麻将，为了防止你不知道麻将的相关规则，在此给出一些定义：

- 这副麻将共含有 m 种花色，有 n 种不同的数字，大小分别为 1 到 n。
- 一共有 $n \times m$ 种牌，每种牌为一个两两不同的二元组 (x, y)，表示这种牌数字为 x，花色为 y，每种牌都有 4 张。其中 $1 \leq x \leq n, 1 \leq y \leq m$ 。
- 一组面子可以为一组刻子或一组顺子。
- 定义一组顺子是三张花色相同、数字连续的牌。
- 定义一组刻子是三张花色和数字组合完全相同的牌。
- 定义一组对子是两张花色和数字组合完全相同的牌。

定义一个麻将牌可重集合 S 是胡的当且仅当：

- 大小为 $3k + 2$，其中 k 为给定的数值。
- 能够被划分为 k 组面子和一组对子。

游戏开始时，ame 从所有牌当中取出了一个大小为 $3k + 2$ 的麻将牌集合 S，她想知道对于所有不同的 S，其中能胡的有多少个。

定义两个麻将牌集合 S 和 T 不同当且仅当存在一种牌，在 S 中的出现次数和在 T 中的出现次数不同。

对于所有数据，$4 \leq n \leq 10^9$，$1 \leq m \leq 10^9$，$1 \leq k \leq 100$。

输入格式

输入共 1 行 3 个正整数 n, m, k。

输出格式

输出共 1 行 1 个整数，表示对 998244353 取模之后的答案。

数据范围

对于所有数据，$4 \leq n \leq 10^9$，$1 \leq m \leq 10^9$，$1 \leq k \leq 100$。

数据编号	$n \leq$	m	$k \leq$	时间限制
1	9	$= 1$	4	4s
2,3,4,5,6	100	$\leq 10^9$	100	4s
7,8,9,10,11	10^9	$= n$	100	4s
12	10^9	$\leq 10^9$	1	4s
13	10^9	$\leq 10^9$	2	4s
14	10^9	$\leq 10^9$	3	4s
15	10^9	$\leq 10^9$	4	4s
16,17,18,19,20	10^9	$\leq 10^9$	30	4s
21,22,23,24,25	10^9	$\leq 10^9$	100	4s

题解

算法一

暴力枚举麻将牌集合，然后贪心判断即可。

算法二

可以手玩方案数！

算法三

显然能够发现如果设 $f_{i,0/1}$ 表示合法麻将牌集合 S 的大小为 i，且这些牌的花色相同，其中包不包含一个对子的数量，则能用组合数 $O\left(k^3\right)$ 求出答案。

具体而言，设 $g_{i,j}$ 表示共有 i 种花色，集合大小为 j 的方案数。那么可以用 f 背包求出 g ，然后再用 g 组合数即可。

考虑如何求出 $f_{i,0/1}$，我们能够设一个 dp，然后从 1 到 n 依次考虑每个大小的牌加多少个，现在

需要判断的是加进去之后是否合法。

转化为一个这样的命题：给定一个集合，能否胡牌。有一个简单的 dp，设 $dp_{i,s_1,s_2,0/1}=[0,1]$ 表示当前考虑第 i 张牌，从 $i-2$ 开始的顺子有 s_1 个，从 $i-1$ 开始的顺子有 s_2 个，有没有对子的情况是否合法。转移显然。

发现 $(s_1,s_2,0/1)$ 的总状态数量为 18，也就是说，用一个大小为 18 的数组就能包括当前麻将牌集合能表示出的所有状态。

那么我们设 $dp_{i,j,S}$ 为，当前加入到第 i 张牌，一共加入了 j 张牌，能表示状态的集合为 S 的方案数。转移显然。

不过 2^{18} 种方案显然是非常大的，我们能从初始状态预处理出能到达的所有状态，经打表一共只有 68 种。

68 种的预处理可以参考 ZJOI2019 麻将的 bfs 写法，接着套用上面那个 dp 就行了。

时间复杂度：$O(68nk)$。

算法四

经实践，当 k 固定时，能用 BM 算法暴力求出递推式，可以通过 $k=1,2,3,4$ 的部分分。

算法五

k 较小的情况下，发现有大量的数字都不会被加入。

那么我们只需考虑若干连续的数字组成的方案数，然后再合并即可。

考虑每一个合法的仅有一种花色且集合大小为 s 的牌，显然 $s=3n+2(n\in\mathbb{N})$，且能够将其分为若干段，这若干段每段的数字连续（可重复），但是不同段之间的数字不连续，显然对每段的牌内部也应是合法的。那么设 f_i 表示分出来的某一段长度为 i 的方案数，这个可以用**算法三**中的方法计算，然后再令 $g_{i,j}$ 表示同一种花色共有 i 段，集合大小为 j 的方案数，这个可以用求得的 f 背包得到。

注意这个 $g_{i,j}$ 不考虑具体的牌的大小，但是考虑不同段之间大小的相对顺序。

- 例如对于牌组 $\{2,3,4,6,7,8,9,9\}$ 在 $g_{i,j}$ 中与 $\{1,2,3,6,7,8,9,9\}$ 相同，但是与 $\{2,3,4,5,5,7,8,9\}$ 不相同。

求得 $g_{i,j}$ 后可以用组合数求出考虑具体牌的大小后的方案数，最后用求得的 $g_{i,j}$ 再组合数背包求出考虑多个花色的答案。

时间复杂度：$O\left(68\times 9\times k^3\right)$。

算法六

我们发现被卡常了，想一下怎么优化常数。

对于每个状态 S，当前合法的麻将牌集合大小模 3 的余数一定相等。

发现常数主要集中在求 f 的地方，然后发现对预处理的 68 种状态，满足某个状态的牌，一定满足数量大小模 3 的余数大小一定。于是可以用这个性质优化常数，再优化一下循环的上下界即可。

使用类似思想可以优化 9 倍常数。

时间复杂度：$O\left(68k^3\right)$。

参考代码

```
#include <algorithm>
#include <iostream>
#include <iomanip>
#include <cstring>
#include <cstdio>
#include <cmath>
#include <queue>
#include <ctime>
#include <map>
using namespace std;
#define mod 998244353
#define ll long long
int dp[2][2][305][70][305], n, m, k, K;
int g[305][305][305][2], h[305][305][2], f[305], inv[305];
int ans, nex, len[70];
struct data {
    int dp[3][3];
    data() {
        memset(dp, 0, sizeof(dp));
    }
    friend bool operator <(const data a, const data b) {
        for (int i = 0; i <= 2; i++) {
            for (int t = 0; t <= 2; t++) {
                if (a.dp[i][t] != b.dp[i][t]) {
                    return a.dp[i][t] < b.dp[i][t];
                }
            }
        }
        return 0;
    }
    friend bool operator !=(const data a, const data b) {
        for (int i = 0; i <= 2; i++) {
            for (int t = 0; t <= 2; t++) {
                if (a.dp[i][t] != b.dp[i][t]) {
                    return 1;
                }
            }
        }
        return 0;
    }
    void DP(data& c, int del) {
        for (int i = 0; i <= 2; i++) {
            for (int t = 0; t <= 2; t++) {
                if (dp[i][t]) {
                    if (del >= i + t) {
                        c.dp[t][(del - i - t) % 3] |= 1;
                    }
                }
            }
        }
    }
};
struct node {
    data is_pair[2];
    void clear() {
        is_pair[0] = data();
        is_pair[1] = data();
    }
    friend bool operator <(const node a, const node b) {
        if (a.is_pair[0] != b.is_pair[0]) {
            return a.is_pair[0] < b.is_pair[0];
        }
        if (a.is_pair[1] != b.is_pair[1]) {
            return a.is_pair[1] < b.is_pair[1];
        }
        return 0;
    }
    friend node operator +(node a, int b) {
        node c;
        if (b >= 2) {
            a.is_pair[0].DP(c.is_pair[1], b - 2);
        }
        a.is_pair[0].DP(c.is_pair[0], b);
        a.is_pair[1].DP(c.is_pair[1], b);
        return c;
    }
} st, sav[205];
map<node, int> mp;
int tot, c[200005][25];
```

```
void bfs() {
    queue<node> q;
    st.clear();
    st.is_pair[0].dp[0][0] = 1;
    q.push(st);
    mp[st] = ++tot;
    sav[tot] = st;
    len[tot] = 0;
    while (!q.empty()) {
        node x = q.front();
        q.pop();
        int now = mp[x];
        for (int i = 1; i <= 4; i++) {
            node nex = x + i;
            if (mp[nex]) {
                c[now][i] = mp[nex];
            } else {
                mp[nex] = ++tot;
                c[now][i] = tot;
                q.push(nex);
                sav[tot] = nex;
                len[tot] = (len[now] + i) % 3;
            }
        }
    }
}
int add(int x) {
    return x >= mod ? x - mod : x;
}
int ksm(int x, int y) {
    int ans = 1, t = x;
    while (y) {
        if (y& 1) {
            ans = 1ll * ans * t % mod;
        }
        t = 1ll * t * t % mod;
        y >>= 1;
    }
    return ans;
}
int C(int n, int m) {
    if (n < m || m < 0) {
        return 0;
    }
    return 1ll * f[n] * inv[n - m] % mod * inv[m] % mod;
}
int main() {
    freopen("mahjong.in", "r", stdin);
    freopen("mahjong.out", "w", stdout);
    bfs();
    dp[0][0][0][1][0] = g[0][0][0][0][0] = 1;
    scanf("%d%d%d", &n, &m, &k);
    K = 3 * k + 2;
    for (int s = 0; s <= k; s++) {
        nex ^= 1;
        for (int i = 0; i <= K; i++) {
            int qwq = min(4 * i, K);
            for (int j = 1; j <= tot; j++) {
                for (int t = i - i % 3 + len[j]; t <= qwq; t += 3) {
                    dp[nex][0][i][j][t] = 0;
                }
            }
            for (int j = 1; j <= tot; j++) {
                for (int t = i - i % 3 + (len[j] + 2) % 3; t <= qwq; t += 3) {
                    dp[nex][1][i][j][t] = 0;
                }
            }
        }
        for (int i = 0; i <= K; i++) {
            int qwq = min(4 * i, K);
            for (int t = 0; t <= qwq; t++) {
                g[s + 1][i][t][0] = mod - dp[nex ^ 1][0][i][1][t];
                g[s + 1][i][t][1] = mod - dp[nex ^ 1][1][i][1][t];
            }
        }
        for (int i = 0; i < K; i++) {
            int qwq = min(4 * i, K - 1);
            for (int j = 1; j <= tot; j++) {
                for (int t = i - i % 3 + len[j]; t <= qwq; t += 3) {
                    if (!dp[nex ^ 1][0][i][j][t]) {
                        continue;
                    }
```

```
                dp[nex^1][0][i+1][c[j][1]][t+1] = add(dp[nex^1][0][i+1][c[j][1]][t+1]
                + dp[nex^1][0][i][j][t]);
                dp[nex^1][0][i+1][c[j][2]][t+2] = add(dp[nex^1][0][i+1][c[j][2]][t+2]
                + dp[nex^1][0][i][j][t]);
                dp[nex^1][0][i+1][c[j][3]][t+3] = add(dp[nex^1][0][i+1][c[j][3]][t+3]
                + dp[nex^1][0][i][j][t]);
                dp[nex^1][0][i+1][c[j][4]][t+4] = add(dp[nex^1][0][i+1][c[j][4]][t+4]
                + dp[nex^1][0][i][j][t]);
            }
        }
    }
    for (int i = 0; i < K; i++) {
        int qwq = min(4 * i, K - 1);
        for (int j = 1; j <= tot; j++) {
            for (int t = i - i % 3 + (len[j] + 2) % 3; t <= qwq; t += 3) {
                if (!dp[nex ^ 1][1][i][j][t]) {
                    continue;
                }
                dp[nex^1][1][i+1][c[j][1]][t+1] = add(dp[nex^1][1][i+1][c[j][1]][t+1]
                + dp[nex^1][1][i][j][t]);
                dp[nex^1][1][i+1][c[j][2]][t+2] = add(dp[nex^1][1][i+1][c[j][2]][t+2]
                + dp[nex^1][1][i][j][t]);
                dp[nex^1][1][i+1][c[j][3]][t+3] = add(dp[nex^1][1][i+1][c[j][3]][t+3]
                + dp[nex^1][1][i][j][t]);
                dp[nex^1][1][i+1][c[j][4]][t+4] = add(dp[nex^1][1][i+1][c[j][4]][t+4]
                + dp[nex^1][1][i][j][t]);
            }
        }
    }
    for (int j = 1; j <= tot; j++) {
        if (sav[j].is_pair[0].dp[0][0]) {
            for (int i = 0; i <= K; i++) {
                int qwq = min(4 * i, K);
                for (int t = i - i % 3; t <= qwq; t += 3) {
                    dp[nex][0][i][1][t]=add(dp[nex][0][i][1][t]+dp[nex^1][0][i][j][t]);
                }
                for (int t = i - i % 3 + 2; t <= qwq; t += 3) {
                    dp[nex][1][i][1][t]=add(dp[nex][1][i][1][t]+dp[nex^1][1][i][j][t]);
                }
            }
        }
        if (sav[j].is_pair[1].dp[0][0]) {
            for (int i = 0; i <= K; i++) {
                int qwq = min(4 * i, K);
                for (int t = i - i % 3 + 2; t <= qwq; t += 3) {
                    dp[nex][1][i][1][t]=add(dp[nex][1][i][1][t]+dp[nex^1][0][i][j][t]);
                }
            }
        }
    }
    for (int i = 0; i <= K; i++) {
        int qwq = min(4 * i, K);
        for (int t = 0; t <= qwq; t++) {
            g[s+1][i][t][0]=dp[nex][0][i][1][t]=add(g[s+1][i][t][0]+dp[nex][0][i][1][t]);
            g[s+1][i][t][1]=dp[nex][1][i][1][t]=add(g[s+1][i][t][1]+dp[nex][1][i][1][t]);
        }
    }
}
f[0] = 1;
for (int i = 1; i <= K; i++) {
    f[i] = 1ll * f[i - 1] * i % mod;
}
inv[K] = ksm(f[K], mod - 2);
for (int i = K - 1; i >= 0; i--) {
    inv[i] = 1ll * inv[i + 1] * (i + 1) % mod;
}
for (int i = 0; i <= K; i++) {
    int now = n + 1 - i;
    if (now < 0) {
        break;
    }
    for (int s = 1,w = now,res = now - 1; s <= k + 1; s++, w = 1ll * w * res % mod, res--) {
        for (int t = 0; t <= K; t++) {
            for (int q = 0; q <= 1; q++) {
                h[1][t][q] = (h[1][t][q] + 1ll * g[s][i][t][q] * w % mod * inv[s]) % mod;
            }
        }
    }
}
h[1][0][0] = 0;
for (int s = 1; s <= k; s++) {
    for (int i = 0; i <= K; i++) {
```

```
            for (int q = 0; q <= 1; q++) {
                for (int j = 0; i + j <= K; j++) {
                    h[s+1][i+j][q]=(h[s+1][i+j][q]+1ll*h[s][i][q]*h[1][j][0])%mod;
                }
            }
            for (int j = 0; i + j <= K; j++) {
                h[s + 1][i + j][1] = (h[s + 1][i + j][1] + 1ll * h[s][i][0] * h[1][j][1]) % mod;
            }
        }
    }
    for (int s = 1, now = m, res = m - 1; s <= k + 1; s++, now = 1ll * now * res % mod, res--) {
        ans = (ans + 1ll * h[s][K][1] * now % mod * inv[s]) % mod;
    }
    printf("%d", ans);
}
```