### (10) 组合数取模

C(n, m) % p的计算，无外乎以下几种，灵活运用即可。

1) n，m都不大，p是素数：预处理阶乘模，加上求逆元，复杂度O(logp)，如果预处理阶乘逆元，复杂度为O(1)。

2) n很大，m很小，p是素数：p固定则预处理1~m逆元，直接O(m)递推，P不固定则直接O(m)计算分子分母的模，然后O(log p)求逆元。

3) n，m都很小，p任意且查询情况较多：O(nm)杨辉三角打表。

4) n,m很大，p是小素数，使用lucas定理，然后利用1)的方法求解即可。

1. **int** fac[1000010];
2. LL c(LL n, LL m, LL p) {
3. **if** (n < m) **return** 0;
4. **if** (n == m) **return** 1;
5. **return** fac[n] \* qpow(fac[n] \* fac[n-m] % p, p - 2, p) % p;
6. }
7. //求出C(n, m) % p.
8. LL lucas(LL n, LL m, LL p) {
9. **if** (m == 0) **return** 1;
10. **return** c(n % p, m % p, p) \* lucas(n / p, m / p, p) % p;
11. }

5) n,m很大，p是合数 ，且p = \* \* \* … \*，不大。

求解C(n,m)对每一个的模，然后利用中国剩余定理合并即可。

求C(n,m)对的模转化为对n!阶乘的求解，考虑到分母的逆元不一定存在，那么我们只需要求得n! = \* s：

a可以使用勒让德定理logp(n)得出，那么关键是求s，n如果不是p的倍数，那么暴力算多余的部分；否则1~n中[kp+1, k(p+1)-1]这一段可以暴力求出一个，然后分成了n/p段，使用快速幂求解即可，剩下1到n中p的倍数，p，2p,3p,….除p之后转化成n / p的阶乘的子问题，递归求解即可。

n!关于p可以直接在幂上进行加减，之后分母中的s与互质，可以求解逆元。

1. #include <bits/stdc++.h>
2. **using** **namespace** std;
4. **typedef** **long** **long** LL;
5. LL sum, d[100][3], r[100], P;
6. **int** k;
8. LL qpow(LL x, LL n, LL p) {
9. LL res = 1;
10. **while** (n) {
11. **if** (n & 1) res = res \* x % p;
12. x = x \* x % p; n >>= 1;
13. }
14. **return** res;
15. }
16. //n = p^k \* a，计算a % s
17. LL fac(LL n, LL p, LL s) {
18. **if** (n == 0) **return** 1;
19. LL re = 1;
20. **for** (LL i = 2; i <= s; i++) **if** (i % p) re = re \* i % s;
21. re = qpow(re, n / s, s);
22. LL r = n % s;
23. **for** (LL i = 2; i <= r; i++) **if** (i % p) re = re \* i % s;
24. **return** re \* fac(n / p, p, s) % s;
25. }
26. //计算C(n, m) % s，s = p^c
27. LL C(LL n, LL m, LL p, LL s) {
28. LL e1 = 0, e2 = 0, e3 = 0;
29. LL r1 = fac(n, p, s), r2 = fac(m, p, s), r3 = fac(n - m, p, s);
30. **for** (LL i = n; i; e1 += i) i /= p;
31. **for** (LL i = n - m; i; e2 += i) i /= p;
32. **for** (LL i = m; i; e3 += i) i /= p;
33. **return** qpow(p, e1 - e2 - e3, s) \* r1 % s \* qpow(r2 \* r3 % s, s / p \* (p - 1), s) % s;
34. }
35. **void** exgcd(LL a, LL b, LL& d, LL& x, LL& y) {
36. **if** (b) {
37. exgcd(b, a % b, d, y, x);
38. y -= x \* (a / b);
39. } **else** d = a, x = 1, y = 0;
40. }
41. LL china() {
42. LL dd, y, x = 0;
43. **for** (**int** i = 0; i < k; i++) {
44. LL w = P / d[i][1];
45. exgcd(d[i][1], w, dd, dd, y);
46. x = (x + y \* w \* r[i]) % P;
47. }
48. **return** (x + P) % P;
49. }
50. LL solve(LL n, LL m) {
51. **for** (**int** i = 0; i < k; i++) r[i] = C(n, m, d[i][0], d[i][1]);
52. **return** china();
53. }
54. **void** f(LL x) {
55. LL u = sqrt(x + 0.5);
56. **for** (LL i = 2; i <= u && x != 1; i++) {
57. **if** (x % i == 0) {
58. d[k][0] = i; LL s = 1;
59. **while** (x % i == 0) x /= i, s \*= i;
60. d[k++][1] = s;
61. }
62. }
63. **if** (x != 1) d[k][0] = d[k++][1] = x;
64. }
65. **int** main() {
66. LL n, w[7];
67. **int** m;
68. scanf("%lld %lld %d", &P, &n, &m);
69. **for** (**int** i = 0; i < m; i++) scanf("%lld", w + i), sum += w[i];
70. **if** (sum > n) puts("Impossible");
71. **else** {
72. k = sum = 0;
73. f(P);
74. LL ans = 1;
75. **for** (**int** i = 0; i < m; i++) {
76. ans = ans \* solve(n, w[i]) % P;
77. n -= w[i];
78. }
79. printf("%lld\n", ans);
80. }
81. **return** 0;
82. }

6) n,m不超过64位的数，p是不超过1e9的任意数。

1. #include <bits/stdc++.h>
2. **using** **namespace** std;
4. **typedef** **long** **long** LL;
5. **const** **int** N = 1000010;
6. **bool** vis[N];
7. vector<**int**> p;
8. LL fac[N];
9. **void** seive(**int** n) {
10. **int** up = sqrt(n + 0.5);
11. **for** (**int** i = 2; i <= up; i++) **if** (!vis[i]) {
12. **for** (**int** j = i \* i; j <= n; j += i) vis[j] = 1;
13. }
14. **for** (**int** i = 2; i <= n; i++) **if** (!vis[i]) p.push\_back(i);
15. p.push\_back(n);
16. }
17. LL qpow(LL x, LL n, LL mod) {
18. **if** (x == 1) **return** 1 % mod;
19. LL res = 1;
20. **while** (n) {
21. **if** (n & 1) res = res \* x % mod;
22. x = x \* x % mod;
23. n >>= 1;
24. }
25. **return** res;
26. }
27. **void** exgcd(LL a, LL b, LL& d, LL& x, LL& y) {
28. **if** (b) { exgcd(b, a % b, d, y, x);  y -= x \* (a / b); }
29. **else** d = a, x = 1, y = 0;
30. }
31. LL inv(LL a, LL n) {
32. LL d, x, y;
33. exgcd(a, n, d, x, y);
34. **return** d == 1 ? (x + n) % n : -1;
35. }
36. **void** modfac(LL n, LL p, LL pc, LL& t, LL& u) {
37. t = 1; u = 0;
38. **while** (n) {
39. t = t \* qpow(fac[pc - 1], n / pc, pc) % pc;
40. t = t \* fac[n % pc] % pc;
41. u += n / p; n /= p;
42. }
43. }
44. LL c(LL n, LL m, LL mod) {
45. LL tmp = mod, ans = 0;
46. **for** (**int** i = 0; p[i] <= tmp; i++) {
47. **if** (tmp % p[i] == 0) {
48. LL pc = 1;
49. **while** (tmp % p[i] == 0) pc \*= p[i], tmp /= p[i];
50. fac[0] = fac[1] = 1;
51. **for** (**int** j = 2; j < pc; j++)
52. j % p[i] ? fac[j] = fac[j-1] \* j % pc : fac[j] = fac[j-1];
53. LL t, u;
54. modfac(n, p[i], pc, t, u);
55. LL tmpans = t, tot = u;
56. modfac(m, p[i], pc, t, u);
57. tmpans = tmpans \* inv(t, pc) % pc;
58. tot -= u;
59. modfac(n - m, p[i], pc, t, u);
60. tmpans = tmpans \* inv(t, pc) % pc;
61. tot -= u;
62. tmpans \*= qpow(p[i], tot, pc);
63. LL k = mod / pc;
64. ans = (ans + tmpans \* k % mod \* inv(k, pc) % mod) % mod;
65. }
66. }
67. **return** ans;
68. }
69. **int** main() {
70. seive(1000000);
71. LL n, m, k;
72. scanf("%I64d %I64d %I64d", &n, &m, &k);
73. printf("%I64d\n", c(n, m, k));
74. **return** 0;
75. }