补充题解 - 《经典》- 第 10 章数学概念与方法

习题10-14 标准差 Standard Deviation, UVa10886

不难想到简单的暴力解法，考虑标准差的计算公式：

$$\frac{1}{n}\sum_{i=1}^{n}(x_i-m)^2=\frac{1}{n}\sum_{i=1}^{n}x_i^2-\frac{2m}{n}\sum_{i=1}^{n}x_i+m^2$$
$$=\frac{1}{n}\sum_{i=1}^{n}x_i^2-m^2 \quad \text{其中} \quad m=\frac{\sum_{i=1}^{n}x_i}{n}$$

但是这样时间效率并不是很高，即使AC，也是勉强通过。

思考一下有无有更好的办法，随机数生成器最容易出现重复问题。所以我们可以做个试验，使用hash判重(unordered_map)，就会发现在g = 0或者$g=2^{32}$之后就开始所有的g都一样。 g = 0之后的所有输出都是0，g = 2^{32}的所有输出都是2^{32}了。实际上回到题面看的也很容易发现将这两个数字代入之后，所有的seed就永远是固定的数字了，之后就不需要继续循环，直接计算结果并返回即可。

习题 10-21 二项式系数 Binomial coefficients, ACM/ICPC NWERC 2011, UVa1649

对于固定的k，$\binom{n}{k}$是相对于n单调递增的，不难想到使用对n使用二分来寻找所有等于m的$\binom{n}{k}$。

但是这里存在一个问题，计算$\binom{n}{k}$并且和二分查找中的mid比较时很容易溢出，有的同学考虑用浮点数，但是存在误差问题，并且计算速度较慢。不过可以考虑利用递推公式：$\binom{n}{k}=\frac{n-k+1}{n}\binom{n}{k-1}$，递推计算，每次先除以$\binom{n}{k-1}$和$n$的最大公约数，之后n一定能被$n-k+1$整除，这样一旦大于mid，直接返回结果即可。

但是即使这样仍然可能会乘法时溢出，怎么办呢，使用另外一个技巧：

$$a*b>2^{63}\leftrightarrow a>2^{63}/b$$

这样在可以在乘法之前就检测溢出，而且m一定是小于2^{63}的，如果发现即将溢出，就可以确定要计算的值一定是大于m的，可以直接返回比较结果。

习题 10-24 幂之和(Sum of Powers, UVa766)

$$(n+1)^{k+1} - n^{k+1} = \sum_{0 \le i \le k} \binom{k+1}{i} n^i$$

$$n^{k+1} - (n-1)^{k+1} = \sum_{0 \le i \le k} \binom{k+1}{i} (n-1)^i$$

$$\cdots$$

$$2^{k+1} - 1^{k+1} = \sum_{0 \le i \le k} \binom{k+1}{i} \cdot 1^i$$

令 $F_k = \sum_{i=1}^{n} i^k$, 对上以上公式求和可得:

$$(n+1)^{k+1} - 1 = \sum_{0 \le i \le k} \binom{k+1}{i} F_i$$

$$(k+1) \cdot F_k = (n+1)^{k+1} - 1 - \sum_{0 \le j < k} \binom{k+1}{j} F_j = \sum_{1 \le j \le k+1} \binom{k+1}{j} \cdot n^k - \sum_{0 \le i < k} \binom{k+1}{i}$$

这样就可以从i = 0到k从小到大一次性全部递推计算出来。

注意本题是要求有理数结果，所以可以使用有理数类来完成四则运算:

```
struct Rational {
  LL a, b;  // a/b
  Rational operator+(const Rational& r) {
    if (r.a == 0) return *this;
    LL na = a * r.b + b * r.a, nb = b * r.b;
    Rational ans = {na, nb};
    return ans.reduce();
  }
  Rational operator-(const Rational& r) {
    if (r.a == 0) return *this;
    Rational ans = {a * r.b - b * r.a, b * r.b};
    return ans.reduce();
  }
  Rational operator/(LL x) {
    assert(x);
    Rational ans = {a, b * x};
    return ans.reduce();
  }
  Rational operator*(LL x) {
    Rational ans = {a * x, b};
    return ans.reduce();
  }
  Rational& reduce() {
    LL g = gcd(a, b);
    a /= g, b /= g;
    return *this;
  }
};
```

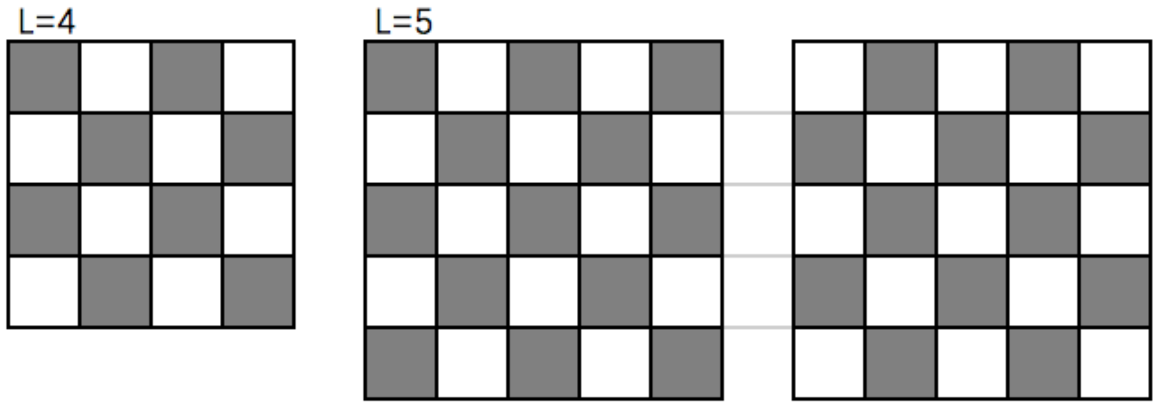
习题 10-25 因子(Factors, ACM/ICPC World Finals 2013, UVa1575)

对于一个整数k来说，考虑其素数分解$k = p_1^{e_1} \cdot_2^{e_2} \cdots \cdot p_m^{e_m}$。则$f(k) = \frac{(e_1+e_2+\cdots e_m)!}{e_1!e_2!\cdots e_m!}$。实际上与$p_1, p_2 \cdots p_m$无关。要求出最小的k, 那么就可以令$p_1, p_2 \cdots p_m$分别等于最小的素数，然后对$e_1, e_2 \cdots e_m$依次进行回溯，其中$e_i < 63$。计算f(k)时可能溢出，所以要提前算出所有可能的$e_i < 63$的素因子分解，使用《经典》一书中 例题 10- 3　选择 与 除法（ Choose and Divide, UVa10375）中介绍的方法来计算f(k)。

习题10-26 方形花园(Square Garden, UVa12520)

如果想得到最大周长，则显然各个涂色格子之间的公共边越少越好。根据L的奇偶性分情况讨论。

L为偶数时(下图以L=4为示例)，如果 $n \le \frac{L^2}{2}$，则可以做到每个涂色格子均无公共边，所求周长为4*n



如果$n > \frac{L^2}{2}$，则需要考虑涂在哪里损失最小，如上图所示，首先考虑涂在角上的2个白色格子，涂色之后周长不变。接着考虑涂在边上的白色格子，每吐一个格子周长减少2。如果还有未涂色的，就只能涂在不靠边的白格子内，每涂一个周长减少4。

L为奇数时，参考上图中的L=5，则要分两种情况考虑，细节逻辑请参考L为偶数的情况。

习题 10-27 互联(Interconnect, ACM/ICPC NEERC 2006, UVa1390)

仔细思考之后不难发现，当前图的状态只需要考虑连通分量的个数以及每个连通分量的大小。

假设当前已经有了k个连通分量，考虑当前每个连通分量的点的个数C_i，则数组S=C[0,1,...k]可以作为一个整体来考虑，状态转移时，所有可以连的边有$n \cdot (n-1)$个，而其中能让连通分量个数减少的有$\sum_{0 \le i,j \le k, i \ne j} C_i \cdot C_j$个。记D(S)为所求的让所有点连通的期望操作次数，S_{ij}为i,j所在的分量连通之后的行程的新的状态, p,e分别为通过一次操作让S长度减少1的概率以及数学期望。

则有:

$$e = \sum_{0 \le i,j \le k, i \ne j} \frac{1}{n(n-1)} C_i C_j (1 + D(S_{ij}))$$

$$p = \sum_{0 \le i,j \le k, i \ne j} \frac{1}{n(n-1)} C_i C_j$$

$$D(s) = e + (1 + D(s)) * (1 - p) \rightarrow D(s) = \frac{e}{p} + \frac{(1-p)}{p}$$

边界情况就是当len(S) = 1时，D(S)=0。

下面考虑最坏情况下的时间复杂度，有k个连通分量的状态点的个数就相当于把整数n切分成k个数字形成一个无序集合的方案个数，其实这个数字就是2类 Strirling Nubmer。对于这个数字的讨论已经超越《经典》的范围，有兴趣的读者可以参考《具体数学》的相关章节。但是在本题中，实际上只会遍历到极少一部分的状态空间，所以以上算法的速度还是比较快的。如果读者有更精确的推导证明，请联系笔者。