牛客NOIP冲刺模拟赛

第2场毕克吕欣



■普及组

题目 1: 你好 a+b

题目 2: 最后一次

题目 3:选择颜色

题目 4: 合法括号序列

比赛地址: https://www.nowcoder.com/acm/contest/165#question



I你好 a+b

题目大意

输入2个整数a和b

保证输入的a和b在long long范围之内,即满足 -9223372036854775808 <= a, b <= 9223372036854775807

计算a+b的值,即这两个数字的和。

如果a+b在long long范围之内,即满足 -9223372036854775808 <= a + b <= 9223372036854775807

那么输出一行一个整数表示a+b的结果。

如果a+b不在long long范围之内,即越界了,那么输出"hello, %lld\n",包含引号。



I你好 a+b

做法

编程技巧题目

首先如果A和B一个是负数,一个是非负数,肯定不会越界。如果都是非负数,我们需要判断 A + B <= C 考虑到A+B会越界,可以改写为A <= C - B

同样的,如果都是负数,我们需要判断 A+B>= C 考虑到A+B会越界,可以改写为A>= C-B

其他的方法,比如long double,高精度也是可以的。 double应该是不行的。

I你好 a+b

做法

输出那个奇怪的字符串,主要就是考察转义

输出反斜杠是\\ 输出双引号是\" 输出百分号是%%

I最后一次

题目大意

牛牛最近学习了质数的概念。

质数指在大于1的自然数中,除了1和它本身以外不再有其他因数。

输入一个n,输出小于等于n最大的质数。



I最后一次

解法

主要就是一个暴力判断质数

对于一个合数n来说,一定有一个质因数<=sqrt(n)

所以可以O(sqrt(n))的时间判断一个数字n是不是质数

然后如果n不是,执行n -= 1,继续判断就可以了。



I选择颜色

题面

n个人排成一个环形,每个人要从c种颜色中选择一个。

牛牛希望相邻的人选择的颜色是不同的

问有多少种方案。

输出方案数对10007取模的结果。

人是有顺序的,环旋转同构算不同的方案。

I选择颜色

解法

print (pow(c - 1, n, p) + (c - 1) * pow(-1, n, p)) % p

结论可以找规律得到,证明就是DP+化简。

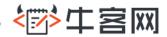
假设第一个人颜色固定,

设f[i]为前i个人颜色都决定好了,第i个人和第1个人颜色相同,的方案数设g[i]为前i个人颜色都决定好了,第i个人和第1个人颜色不同,的方案数

(注意g[i]实际上是c-1个状态,但是因为这c-1个状态数值一样,只需要记一个)

转移方程

- * f[i] = (c 1) * g[i 1] * g[i] = f[i - 1] + (c - 2) * g[i - 1]
- 化简之后得到 g[i] = (c 1) * g[i 2] + (c 2).* g[i 1] 解递推,可以得到上面的结论。



I合法括号序列

题面

键盘上有左括号(, 右括号), 和退格键-, 共三个键。

牛牛希望按键n次, 使得输入的字符串恰好一个合法的括号序列。

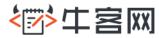
每按一次左括号(, 字符串末尾追加一个左括号(每按一次右括号), 字符串末尾追加一个右括号)

每按一次退格键-,会删掉字符串的最后一个字符,

特别地,如果字符串为空,牛牛也可以按退格,但是什么都不会发生。

输出方案数对p取模,注意p可能不是质数。

注:只要按键方法不同,就是不同的方案,即使得到的序列一样。



■解法

做法

这个题可以分为两部分,

- 一部分是讨论最后合法括号序列的长度2k。
- 二部分是有多少种输入他的方式。

对于第一部分,是基本的卡特兰数。

对于第二部分,注意到:

输入一个长度为2k的字符串,无论序列是什么,方案数是一定的。

也就是说方案数只和想输入的字符串长度有关,和具体是什么没关系

设f[i][j]为按键i次,输入了长度为j的序列。

f[i][j] = f[i - 1][max(j - 1, 0)] + 2 * f[i - 1][j + 1]

最后枚举k,统计答案即可。



I提高组

题目 1: 方差

题目 2: 分糖果

题目 3: 集合划分

比赛地址: https://www.nowcoder.com/acm/contest/173#question



I方差

题目大意

对一个长度为 m 的序列 $\{b_i\}$, 定义方差:

$$Var(b) = \frac{1}{m} \sum_{i=1}^{m} (b_i - \overline{b})^2$$

给定长度为 N 的序列 $\{a_i\}$, 对每个位置, 求出删掉这个位置后, 剩下元素的方差乘以 $(N-1)^2$ 的值

I方差

解法

考虑:

$$m \times \sum_{i=1}^{m} (b_i - \overline{b})^2 = m \times \sum_{i=1}^{m} (b_i^2 - 2b_i \overline{b} + \overline{b}^2) = m \times \left(\sum_{i=1}^{m} b_i^2 - 2\overline{b} \sum_{i=1}^{m} b_i + m\overline{b}^2\right)$$
$$= m \times \left(\sum_{i=1}^{m} b_i^2 - m\overline{b}^2\right) = m \sum_{i=1}^{m} b_i^2 - \left(\sum_{i=1}^{m} b_i\right)^2$$

由此易见,只需维护序列元素的和、平方和即可快速计算方差。

对一个序列,询问删掉每个元素之后剩下序列的元素和、平方和是容易完成的。

题目大意

N 个小朋友围成一圈, 要给他们分一些糖果。

第 i 个小朋友最少拿到 1 个、最多拿到 a[i] 个。

问有多少种方案使得任意两个相邻的小朋友拿到的糖果数不同。

部分分做法

考虑 N 和 a[i] 均不是很大的情形,此时我们可以DP

把环从某个位置断开,记 f[i][j][k] 表示考虑了前 i 个人,第 1 个人和第 i 个人分别了 j k 个糖果的方案数 暴力转移的复杂度是 $\Theta(a_i)$,如果加一个简单的前缀和优化的话就能优化到 $\Theta(1)$



容斥的想法

正难则反,我们考虑容斥

枚举哪些相邻的小朋友拿到的糖果数一样,整个环根据我们枚举的相等关系会被分成若干段

其中每段的糖果数一样,那么这一段的方案数必然为这段中的 a_i 最小值

同时,假设我们枚举了 k 个相等关系,它的容斥系数为 $(-1)^k$

我们考虑 DP 它的贡献,设 f_i 表示:

容斥到第 i 个元素, 前面划分了若干段, 所有不同的方案配上容斥系数的贡献之和

$$f_i = \sum_j f_j \times \min(a[j+1 ... i]) \times (-1)^{i-j-1}$$

容斥的想法

最后一个元素可以等于等于第一个元素,这个很难在我们目前的 DP 中体现出来

一个暴力的想法是把第一段的最小值记下来,然后和最后一段合并

但是我们可以优化: 我们把环适当地转一下, 使得 α[1] 为整个序列的最小值

此时把最后一段和第一段合并时地贡献可以直接计算。



容斥的想法

对于这个 DP, 暴力做的复杂度是 O(N^2)

优化思路有两种

使用线段树优化,复杂度为 O(N\log N),80 分

使用单调栈优化,复杂度为 O(N), 100 分



题意

visit_world 得到了 {1, 2, ..., n} 的 (2^n - 1) 个非空子集

他计划把这些集合分给小 S 和小 T, 每个子集恰好给其中一个人。

分配规则是这样的:

- 1. 若存在两个集合属于同一个人, 那么这两个集合的并也要属于那个人
- 2. 有 m 个集合是小 S 特别喜欢的, 你要把这些集合全部给小 S
- 3. 小 S 拿到了恰好 k 个集合。

请给出一种方案,或者说明无解。



M = 0 的做法

我们定义 lowbit(x) 为 x 的二进制下,最小的一个元素(换句话说,就是它对应集合里最小的元素)

我们用 x | y 表示 x 和 y 两个集合的并

断言: 我们把所有 lowbit 相同的元素都分给同一个人,这样构造的方案一定是合法的

证明很容易,注意到 lowbit(x | y) 恰好是 lowbit(x) 和 lowbit(y) 之一,若 lowbit(x) 和 lowbit(y) 均属于 A,那么 lowbit(x | y) 也属于 A

lowbit(x) = i 的元素恰好有 $2^{(N-i-1)}$ 个,我们把 K 二进制拆分,构造答案即可



M > 0 的做法

把上述做法推广一下,我们给每个元素定义一个优先级,使得不同元素优先级不同

对一个集合,我们定义它的"特征"为它之中优先级最高的元素编号

那么,我们把所有"特征"相同的集合分给同一个人,这样一定合法

另一方面,我们可以证明,每一种合法方案都可以按照这种方法构造

所以我们问题有两个:

- 确定每个元素的优先级,从而计算集合的特征
- 把特征相同的集合分给同一个人
- 一个观察是, 第二个问题其实是不需要考虑的, 因为有 K 的限制, 每个集合分给谁是确定好的



M > 0 的做法

使用状压DP, 从高到低确定每个元素的优先级

记 f[S] 表示集合 S 里的元素已经被分配了最高的若干优先级,是否可行

转移时枚举接下来的优先级最高的元素是哪个,使用一些简单的位运算技巧确定可行性

最后根据这个 DP 还原出一种方案即可。



Thanks

