具体数学初级班第三周参考答案

517 老师

2020 年 5 月 11 日

0.1 A 题

经典的错排模版题

0.2 B 题

相当于将 N 个相同的球放到 K 个不同的盒子里，每个盒子可以为空，那么方案数等于 $\binom{N+K-1}{K-1}$

0.3 C 题

枚举那些位置放错了，然后这些位置再进行错排，因此需要预处理组合数以及错排

C 题

```
#include <bits/stdc++.h>
using namespace std;

long long c[30][30];

long long f[30];

void init() {
  f[0] = 1;
  f[1] = 0;
  f[2] = 1;
```

```
  for (int i = 3; i < 30; i++) {
    f[i] = (i - 1) * (f[i - 1] + f[i - 2]);
  }
  c[0][0] = 1;
  for (int i = 1; i < 30; i++) {
    c[i][0] = c[i][i] = 1;
    for (int j = 1; j < i; j++) {
      c[i][j] = c[i - 1][j] + c[i - 1][j - 1];
    }
  }
}

int main() {
  init();
  int t, n;
  while (scanf("%d", &n) == 1 && n) {
    long long ret = 0;
    for (int i = 0; i <= n / 2; i++) {
      ret += c[n][i] * f[i];
    }
    printf("%lld\n", ret);
  }
  return 0;
}
```

0.4 D 题

首先，假如不考虑回文串这个特点，应该想到这是一个多重集合排列相关模型的题

再考虑，回文串，那么我们只需要关心一半的字符串长度有多少不同的排列就行了，因为右半边是完整的复制左半边的

注意有除法，因此要使用逆元

回文串

```
#include <bits/stdc++.h>
```

```
using namespace std;

const int md = (int) 1e9 + 7;


char s[1010];
int fac[1010];

void init(){
  fac[0] = 1;
  for (int i = 1; i <= 1000; i++) {
    fac[i] = 1LL * fac[i - 1] * i % md;
  }
}

int pow_mod(int a, int b, int c) {
  int ret = 1;
  while (b) {
    if (b & 1){
      ret = 1LL * ret * a % md;
    }
    b >>= 1;
    a = 1LL * a * a % md;
  }
  return ret;
}

int inv(int x) {
  return pow_mod(x, md - 2, md);
}

int main() {
  init();
  int t;
  scanf("%d", &t);
  while (t--) {
```

```
    scanf("%s", s);
    int n = strlen(s);
    int cnt[26] = {0};
    for (int i = 0; i < n; i++) {
      cnt[s[i] - 'a']++;
    }
    long long ret = fac[n / 2];
    int odd = 0;
    for (int i = 0; i < 26; i++) {
      if (cnt[i] & 1) {
        odd++;
        cnt[i]--;
        cnt[i] /= 2;
      } else {
        cnt[i] /= 2;
      }
    }
    //超过一种字符有奇数个，无解
    if (odd > 1) {
      printf("0\n");
    } else {
      for (int i = 0; i < 26; i++) if (cnt[i] > 0){
        ret = ret * inv(fac[cnt[i]]) % md;
      }
      printf("%lld\n", ret);
    }
  }
  return 0;
}
```

0.5 E 题

XOOOOOOXOOOOOXOOOOOOXOOOO

假设 X 是点亮的灯，O 表示没有点亮，那么就会形成若干段连续的 O 被一些 X 分开了，我们可以发现对于不同段的 O 之间点亮灯的决策是独立的，他们之间不会互

相影响

你可以去某一段点亮一盏灯，再去其他段点灯

对于某一段连续的'O'，我们发现每次要么选择点亮最左边的，要么点亮最右边的，方案数一共是 2^{len-1}，len 表示连续的'O' 的个数

算答案的时候你可以先假设每一段连续的'O' 都是一样的，那么这个题就是一个多重集合的排列，最后再乘上每一段的方案数

点灯

```
#include <bits/stdc++.h>
using namespace std;
typedef long long lld;
const int inf = ~0u>>2;
const int mod = 1000000007 ;
const int maxn = 1010;
int pos[maxn];
lld Pow(lld a,lld b) {
    lld ans = 1;
    while(b) {
        if(b&1) ans = ans * a % mod;
        a = a * a % mod;
        b >>= 1;
    }
    return ans ;
}
lld fac[maxn],two[maxn];
vector<int> rec;
void solve(int n) {
     fac[0] = 1; two[0] = 1;
     for(int i = 1; i <= 1000; i++) fac[i] = fac[i-1] * i % mod;
     for(int i = 1; i <= 1000; i++) two[i] = two[i-1] * 2 % mod;
     int sum = 0;
     for(int i = 0; i < rec.size(); i++) sum += rec[i];
    // printf("sum=%d\n",sum);
     lld ans = fac[sum];
     for(int i = 0; i < rec.size(); i++) {
         ans = ans * Pow(fac[rec[i]],mod-2) % mod;
```

```
    }
    for(int i = 1; i < rec.size()-1;i++) {
     if(rec[i]>0) ans = ans * two[rec[i]-1] % mod;
    //  printf("ans=%d\n",rec[i]);
    }
    printf("%lld\n",ans);
}

int main() {
    int n , m;
    scanf("%d%d",&n,&m);
    for(int i = 0; i < m; i++) {
        scanf("%d",&pos[i]);
    }
    sort(pos,pos+m);
    rec.push_back(pos[0]-1);
    for(int i = 1; i < m ; i++) {
       rec.push_back(pos[i]-pos[i-1]-1);
    }
    rec.push_back(n-pos[m-1]);
    solve(n);
    return 0;
}
```