具体数学初级班第四周参考答案

517 老师

2020 年 5 月 20 日

0.1 A 题

卡特兰数可以理解为从 $(0,0)$ 走到 (n,n) 不超过对角线的方案数，现在假设已经输入了 x 个左括号，y 个右括号

那么就相当于从 (x,y) 要走到 (n,n) 的方案数，如果把 (x,y) 看成 $(0,0)$, 那么就相当于 $(0,0)$ 走到 $(n-x,n-y)$ 不超过对角线的方案数

注意：n 表示总括号长度的一半

设 a 表示剩下的左括号，b 表示剩下的右括号, 则 $a=n-x, b=n-y$

所以答案为总方案数减去非法的方案数，$\binom{n-x+n-y}{n-x}-\binom{n-x+n-y}{n-x-1}=\binom{a+b}{a}-\binom{a+b}{a-1}$

卡特兰数变形

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

const int md = 1000000007;

const int N = 1000010;

int fac[N];

int pow_mod(int a, int b, int md) {
  int ret = 1;
  while (b) {
    if (b & 1) {
      ret = 1LL * ret * a % md;
```

```
    }
    a = 1LL * a * a % md;
    b >>= 1;
  }
  return ret;
}

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

int binom(int n, int m) {
  if (m < 0) {
    return 0;
  }
  return 1LL * fac[n] * inv(fac[m]) % md * inv(fac[n - m]) % md;
}

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

char s[N];
int main() {
  init();
  int n;
  while (scanf("%d", &n) == 1) {
    scanf("%s", s);
    if (n % 2 == 1) {
      puts("0");
      continue;
    }
    int len = strlen(s);
```

```
    int x = 0, y = 0;
    bool flag = true;
    for (int i = 0; i < len; i++) {
      if (s[i] == '(' ) {
        x++;
      } else {
        y++;
      }
      if (x < y) {
        flag = false;
      }
    }
    if(flag == false) {
      puts("0");
      continue;
    }

    int a = n / 2 - x, b = n / 2 - y;
    int ret = binom(a+b, a) - binom(a+b,a-1) ;
    ret = (ret % md + md) % md;
    printf("%d\n", ret);
  }
  return 0;
}
```

0.2 B 题

最高的楼肯定是可以看到的，那么左边能看到 $F-1$, 右边能看到 $B-1$

然后我们观察楼的形态，一定是一幢能看见的楼后面跟着若干比自己矮的楼，这些楼其实可以任意排列，反正都不会被看见，然后接着是一幢比自己高的楼

这样就可以理解为有若干组楼，每组楼能看见最高的那个，假设一组楼有 x 幢，那么总方案就是 $(x-1)!$, 这其实就是圆排列

想到这里可以联想第一类斯特林数，将 $n-1$ 幢不同的楼分成 $F+B-2$ 个圆排列

最后再通过组合数枚举哪些放左边

斯特林数的应用

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

const int md = 1000000007;

const int N = 2010;

int s[N][N], c[N][N];

void init() {
  s[0][0] = s[1][1] = 1;
  for (int i = 2; i < N; i++) {
    for (int j = 1; j <= i; j++) {
      s[i][j] = 1LL * (i - 1) * s[i - 1][j] % md + s[i - 1][j - 1];
      s[i][j] %= md;
    }
  }

  c[0][0] = 1;
  for (int i = 1; i < N; 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];
      c[i][j] %= md;
    }
  }
}

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

```
    int ret;
    if (f + b - 2 < n)
      ret = 1LL * s[n - 1][f + b - 2] * c[f + b - 2][f - 1] % md;
    else ret = 0; //注意 f+b-2不合法的情况
    printf("%d\n", ret);
  }
  return 0;
}
```

0.3 C 题

如果是 m 个相同的小求放到 n 个不同的盒子里，盒子可以为空，那么方案数就是隔板法的变形 $\binom{n+m-1}{m}$

这道题相当于就是枚举具体放了几个小球，因此

$ans = \binom{n-1}{0} + \binom{n}{1} + \binom{n+1}{2} + \binom{n+2}{3} + ... + \binom{n+m-1}{m}$

根据 $\binom{i}{j} = \binom{i-1}{j-1} + \binom{i-1}{j}$ 可以得到

$ans = \binom{n+m}{m}$

数据范围较大，模数是个质数，可以使用卢卡斯定理

卢卡斯定理

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


int md;
int fac[100010];

void init () {
  fac[0] = 1;
  for (int i = 1; i <= md; 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 % c;
    }
    a = 1LL * a * a % c;
    b >>= 1;
  }
  return ret;
}

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

int binom(int n, int m) {
  if (n < m) return 0;
  return 1LL * fac[n] * inv(fac[m]) % md * inv(fac[n - m]) % md;
}

int lucas(int n, int m) {
  if (m == 0) return 1;
  return 1LL * lucas(n / md, m / md) * binom(n % md, m % md) % md;
}

int main() {
  int t;
  scanf("%d", &t);
  while (t--) {
    int n, m;
    scanf("%d%d%d", &n, &m, &md);
    init();
    printf("%d\n", lucas(n + m, n));
  }
  return 0;
}
```

0.4 D 题

行和列是可以分开考虑的，我们枚举走了 x 步走到终点，相当于从第一行走 x 步走到第 n 行

从第一列走 x 步走到第 m 列

这两个方案数乘起来就是从 $(1,1)$ 走 x 步走到 (n,m) 的步数

那么从 1 到 n 走 x 步的走法数量相当于将 $(n-1)$ 个相同的小球放到 x 个不同的盒子里，盒子不为空

想法题

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

const int md = 1000000007;

int fac[100010];

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

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

```
int binom(int n, int m) {
  return 1LL * fac[n] * inv(fac[m]) % md * inv(fac[n - m]) % md;
}

int main() {
  fac[0] = 1;
  for (int i = 1; i <= 100000; i++) {
    fac[i] = 1LL * fac[i - 1] * i % md;
  }
  int n, m;
  while (scanf("%d%d", &n, &m) == 2) {
    int ret = 0;
    for (int x = 1; x <= min(n - 1, m - 1); x++) {
      ret += 1LL * binom(n - 2, x - 1) * binom(m - 2, x - 1) % md;
      ret %= md;
    }
    printf("%d\n", ret);
  }
  return 0;
}
```