# 8.1 排列组合数

**8.1.1 概述**

**排列数:**

从n个不同元素，任取个元素按照一定的顺序排成一列，叫做从n个不同元素中取出m个元素的一个排列；从n个不同元素中取出个元素的所有排列的个数，叫做从n个不同元素中取出m个元素的排列数，用符号(或者是)表示。

**组合数:**

从n个不同元素中，任取 个元素组成一个集合，叫做从n个不同元素中取出m个元素的一个组合；从n个不同元素中取出个元素的所有组合的个数，叫做从n个不同元素中取出m个元素的组合数。用符号来表示,组合数也常用表示

**8.1.2各种排列**

**8.1.2.1不相邻的排列**

1-n这n个自然数中选k个，这k个中任何两个数都不相邻的组合有

**8.1.2.2错位排列**

我们把错位排列问题具体化，考虑这样一个问题：

n封不同的信，编号分别是1,2,3,4,5，现在要把这五封信放在编号1,2,3,4,5的信封中，要求信封的编号与信的编号不一样。问有多少种不同的放置方法？

假设我们考虑到第n个信封，初始时我们暂时把第n封信放在第n个信封中，然后考虑两种情况的递推：

前面n-1个信封全部装错；

前面n-1个信封有一个没有装错其余全部装错。

对于第一种情况，前面n-1个信封全部装错：因为前面n-1个已经全部装错了，所以第n封只需要与前面任一一个位置交换即可，总共有种情况。

对于第二种情况，前面n-1个信封有一个没有装错其余全部装错：考虑这种情况的目的在于，若n-1个信封中如果有一个没装错，那么我们把那个没装错的与n交换，即可得到一个全错位排列情况。

其他情况，我们不可能通过一次操作来把它变成一个长度为 的错排。

于是可得错位排列的递推式为。

错位排列数列的前几项为0,1,2,9,44,265。

**8.1.2.3圆排列**

n个人全部来围成一圈，所有的排列数记为。考虑其中已经排好的一圈，从不同位置断开，又变成不同的队列。 所以有

部分圆排列公式

* + 1. **三种求组合数的方法**

**8.1.3.1**

**通过预处理**

**代码**

1. #include<bits/stdc++.h>
2. **using** **namespace** std;
3. **const** **int** N=2005,mod=1e9+7;
4. **int** c[N][N];
5. **void** init(**int** n,**int** m)
6. {
7. **for**(**int** i=0;i<=n;i++)
8. {
9. **for**(**int** j=0;j<=i;j++)
10. {
11. **if**(j==0) c[i][j]=1;
12. **else** c[i][j]=(c[i-1][j-1]+c[i-1][j])%mod;
13. }
14. }
15. }
16. **int** main()
17. {
18. init(2000,2000);
19. **int** n,a,b;
20. cin>>n;
21. **for**(**int** i=1;i<=n;i++)
22. {
23. cin>>a>>b;
24. cout<<c[a][b]<<endl;
25. }
26. **return** 0;
27. }

**8.1.3.2**

**随取随用，可预处理逆元和阶乘**

**代码**

1. #include<bits/stdc++.h>
2. #define int long long
3. **using** **namespace** std;
4. **const** **int** N=1e5+5,mod=1e9+7;
5. **int** fac[N],infac[N];
6. **int** qpow(**int** a,**int** n,**int** mod)
7. {
8. **int** ans=1;
9. **while**(n)
10. {
11. **if**(n&1) ans=ans\*a%mod;
12. a=a\*a%mod;
13. n>>=1;
14. }
15. **return** ans;
16. }
17. **void** init()
18. {
19. fac[0]=1;
20. infac[0]=1;
21. **for**(**int** i=1;i<N;i++)
22. {
23. fac[i]=fac[i-1]\*i%mod;
24. infac[i]=infac[i-1]\*qpow(i,mod-2,mod)%mod;
25. }
26. }
27. **int** C(**int** a,**int** b)
28. {
29. **return** fac[a]\*infac[b]%mod\*infac[a-b]%mod;
30. }
31. **signed** main()
32. {
33. **int** n;
34. cin>>n;
35. init();
36. //for(int i=1;i<=100;i++) cout<<fac[i]<<" "<<infac[i]<<endl;
37. **for**(**int** i=1;i<=n;i++)
38. {
39. **int** a,b;
40. cin>>a>>b;
41. cout<<C(a,b)<<endl;
42. }
43. **return** 0;
44. }

**8.1.3.3**

**Lucas定理，当p为质数时，有**

**代码**

1. #include<bits/stdc++.h>
2. #define int long long
3. **using** **namespace** std;
4. **int** qpow(**int** a,**int** n,**int** mod)
5. {
6. **int** ans=1;
7. **while**(n)
8. {
9. **if**(n&1) ans=ans\*a%mod;
10. a=a\*a%mod;
11. n>>=1;
12. }
13. **return** ans;
14. }
15. **int** C(**int** a,**int** b,**int** p)
16. {
17. **int** res=1;
18. **for**(**int** i=1,j=a;i<=b;i++,j--)
19. {
20. res=res\*j%p\*qpow(i,p-2,p)%p;
21. }
22. **return** res;
23. }
24. **int** lucas(**int** a,**int** b,**int** p)
25. {
26. **if**(a<b) **return** 0;
27. **if**(a<p&&b<p) **return** C(a,b,p);
28. **else** **return** lucas(a%p,b%p,p)\*lucas(a/p,b/p,p)%p;
29. }
30. **signed** main()
31. {
32. **int** n;
33. cin>>n;
34. **for**(**int** i=1;i<=n;i++)
35. {
36. **int** a,b,p;
37. cin>>a>>b>>p;
38. cout<<lucas(a,b,p)<<endl;
39. }
40. **return** 0;
41. }