7.3 二项式定理和杨辉三角

|  |  |  |
| --- | --- | --- |
| 例7-2. 计算杨辉三角 | | |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | #include<bits/stdc++.h>  using namespace std;  int a[21][21];  int main(){  int n; cin >> n;  for(int i=1;i<=n;i++) a[i][1] = a[i][i] = 1; //赋初值  for(int i=1;i<=n;i++)  for(int j=2;j<i;j++)  a[i][j] = a[i-1][j]+a[i-1][j-1]; //递推求二项式系数  for(int i=1;i<=n;i++){  for(int j=1;j<=i;j++)  cout << a[i][j]<<" ";  cout << endl;  }  } |

下面的例题，用两种方法算二项式系数：递推公式、逆。

|  |
| --- |
| 例7-3. 计算系数 洛谷P1313 |

（1）用递推公式计算二项式系数（DFS写法）

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32 | #include<bits/stdc++.h>  using namespace std;  const int N = 1005;  #define mod 10007  int c[N][N]; //存二项式系数  int fastPow(int a, int n){ //标准快速幂  int ans = 1;  a %= mod; //防止下面的ans\*a越界  while(n) {  if(n & 1) ans = (ans\*a) % mod;  a = (a\*a) % mod;  n >>= 1;  }  return ans;  }  int dfs(int n,int m){ //用递推公式求二项式系数  if(!m) return c[n][m]=true;  if(m==1) return c[n][m]=n;  if(c[n][m]) return c[n][m]; //记忆化  if(n-m<m) m=n-m;  return c[n][m]=(dfs(n-1,m)+dfs(n-1,m-1))%mod;  }  int main(){  int a,b,k,n,m; cin >>a>>b>>k>>n>>m;  c[1][0] = c[1][1] = 1;  int ans = 1;  ans \*= (fastPow(a,n)\*fastPow(b,m))%mod;  ans \*= dfs(k,n)%mod;  ans %= mod;  cout << ans;  return 0;  } |

（2）用逆直接计算二项式公式

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | #include<bits/stdc++.h>  using namespace std;  #define mod 10007  int fac[10001]; //预计算阶乘  int inv[10001]; //预计算逆  int fastPow(int a, int n){} //和上一个代码完全一样  int C(int n,int m){ //算组合数，用到除法取模的逆  return (fac[n]\*inv[m]%mod\*inv[n-m]%mod)%mod;  }  int main(){  int a,b,n,m,k,ans; cin >>a>>b>>k>>n>>m;  fac[0] = 1;  for(int i=1;i<=n+m;i++){  fac[i] = (fac[i-1]\*i)%mod; //预计算阶乘，要取模  inv[i] = fastPow(fac[i],mod-2); //用费马小定理预计算逆  }  ans = (fastPow(a,n)%mod\*fastPow(b,m)%mod\*C(k,n)%mod)%mod;  cout << ans;  return 0;  } |

7.4 卢卡斯定理

|  |  |  |
| --- | --- | --- |
| 例7-5. 卢卡斯定理 洛谷P3807 | | |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33 | #include<bits/stdc++.h>  using namespace std;  const int N = 100010;  typedef long long ll;  ll fac[N]; //预计算阶乘，取模  ll fastPow(ll a, ll n, ll m){ //标准快速幂  ll ans = 1;  a %= m; //防止下面的ans\*a越界  while(n) {  if(n & 1) ans = (ans\*a) % m;  a = (a\*a) % m;  n >>= 1;  }  return ans;  }  ll inverse(ll a,int m){ return fastPow(fac[a],m-2,m); } //用费马小定理计算逆  ll C(ll n,ll r,int m){ //用逆计算C(n mod m, r mod m) mod m  if(r>n)return 0;  return ((fac[n] \* inverse(r,m))%m \* inverse(n-r,m)%m);  }  ll Lucas(ll n,ll r,int m){ //用递归计算C(n, r) mod m  if(r==0) return 1;  return C(n%m,r%m,m) \* Lucas(n/m,r/m,m)%m; //分两部分计算  }  int main(){  int T; cin >> T;  while(T--){  int a,b,m; cin >>a>>b>>m;  fac[0] = 1;  for(int i=1;i<=m;i++) fac[i]=(fac[i-1]\*i)%m; //预计算阶乘，取模  cout << Lucas(a+b,a,m) << endl;  }  } |

7.5 容斥原理

|  |  |  |
| --- | --- | --- |
| 例7-7. 硬币购物 洛谷P1450 | | |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32 | #include<cstdio>  const int N=100009;  long long dp[N];  int main(){  int c[4],d[4];  for(int i=0;i<4;i++) scanf("%d",&c[i]);  dp[0]=1;  for(int i=0;i<4;i++) //完全背包，预处理  for(int j=c[i];j<N;j++)  dp[j] += dp[j-c[i]];  int T; scanf("%d",&T);  while(T--){  for(int i=0;i<4;i++) scanf("%d",&d[i]);  int s; scanf("%d",&s);  long long ans = dp[s]; //容斥定理公式的第1项  for(int i=1;i<=15;i++){ //i：0001~1111，二进制数枚举集合  int now = s;  int tmp = i;  int ov = 0; //用ov判断奇偶  for(int j=0;tmp;j++){ //容斥  if(tmp&1)  ov^=1, now -= (d[j]+1)\*c[j];  tmp = tmp >>1; //tmp找i中的1  }  if(now<0) continue;  if(ov) ans -= dp[now]; //奇数，减去  else ans += dp[now]; //偶数，加上  }  printf("%lld\n",ans);  }  return 0;  } |

7.6.2 Stirling数

|  |
| --- |
| 例7-10. 小朋友的球 洛谷 P1655 |

。

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | N = 105  S = [[0] \* N for i in range(N)]  for i in range(1, N):  S[i][i] = S[i][1] = 1  for j in range(2, i): S[i][j] = S[i - 1][j - 1] + j \* S[i - 1][j]  while True: # 多组数据  try: n, k = map(int,input().split()); print(S[n][k])  except EOFError: break |

7.7.3 Pólya计数

|  |  |  |
| --- | --- | --- |
| 例7-12. Necklace of Beads poj 1286 | | |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | #include <cstdio>  #include <cmath>  #include <algorithm>  using namespace std;  #define ll long long  int main(){  ll n;  while(scanf("%lld", &n) && n != -1 ) {  if(n==0 ){ printf("0\n"); continue;}  ll ans = 0; //答案很大，用long long  for(ll i=0; i<n; i++) ans += (ll)pow(3,\_\_gcd(n,i)); //旋转。3指三种颜色  if(n%2) ans += n\*(ll)pow(3,n/2+1); //翻转：n是奇数  else{ //翻转：n是偶数  ans += n/2\*(ll)pow(3,n/2);  ans += n/2\*(ll)pow(3,n/2+1);  }  printf("%lld\n", ans/(n\*2));  }  return 0 ;  } |

7.8.1 普通型母函数

|  |
| --- |
| 例7-13. 整数划分 hdu 1028 |

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | #include<bits/stdc++.h>  using namespace std;  const int N = 200;  int dp[N+1][N+1]; //dp[n][m]: 将n划分成最大数不超过m的划分数  void part() { //预计算dp[n][m]，求出所有n的划分  for(int n=1; n<=N; n++)  for(int m=1; m<=N; m++){  if((n==1)||(m==1)) dp[n][m] = 1;  else if(n < m) dp[n][m] = dp[n][n];  else if(n == m) dp[n][n] = dp[n][n-1]+1;  else dp[n][m] = dp[n-m][m] + dp[n][m-1] ;  }  }  int main(){  int n;  part();  while(cin >> n) cout << dp[n][n] << endl;  return 0;  } |

。

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23 | //用母函数求整数划分（hdu 1028）  #include<bits/stdc++.h>  using namespace std;  const int N=200;  int c1[N+1], c2[N+1];  void part() {  int i, j, k;  for(i=0; i<=N; i++){ //初始化，第一部分(1 *+ x + x2 +* *...*)的系数，都是1  c1[i]=1; c2[i]=0;  }  for(k=2; k<=N; k++){ //从第2部分(1 + *x*2 + *x*4 + ...)开始展开  for(i=0; i<=N; i++) //k=2时，i循环第一部分(1*+x+x2+**...*)，j循环第二部分(1+*x*2+*x*4+...)  for(j=0; j+i <= N; j+=k)  c2[i+j] += c1[i];  for(i=0; i <= N; i++) { c1[i] = c2[i]; c2[i] = 0; } //更新本次展开的结果  }  }  int main(){  int n;  part();  while(cin >> n) cout << c1[n] << endl;  return 0;  } |

7.9.2 尼姆游戏

|  |
| --- |
| 例7-18. Being a Good Boy in Spring Festival hdu 1850 |

主要代码如下：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | int sum=0, ans=0; //sum是Nim-sum，ans是第一步可行方案数  for(int i=0; i<n; i++) sum ^= a[i]; //异或计算，求Nim-sum  if(sum==0) cout<<0<<endl; //开始局面是P-position，先手必败  else{ //开始局面是N-position，先手胜  for(int i=0; i<n; i++)  if((sum^a[i]) <= a[i]) //计算第一步所有可能方案  ans++;  cout<<ans<<endl;  } |

7.9.3 图游戏与Sprague-Grundy函数

hdu 1846为

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23 | #include<bits/stdc++.h>  using namespace std;  const int MAX = 1001;  int n, m, sg[MAX], s[MAX];  void getSG(){  memset(sg, 0, sizeof(sg));  for (int i=1; i<=n; i++){  memset(s, 0, sizeof(s));  for (int j=1; j<=m && i-j>=0; j++) s[sg[i-j]] = 1; //把i的后继点放到集合s中  for (int j=0; j<=n; j++) //计算sg[i]  if(!s[j]){ sg[i]=j; break;}  }  }  int main(){  int c; cin>>c;  while (c--){  cin>>n>>m;  getSG();  if (sg[n]) cout<<"first\n"; //sg != 0，先手胜  else cout<<"second\n"; //sg == 0，后手胜  }  return 0;  } |

|  |
| --- |
| 例7-19. Fibonacci again and again hdu 1848 |

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25 | #include<bits/stdc++.h>  using namespace std;  const int N = 1001;  int sg[N], s[N];  int fibo[15]={1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987};  void getSG(){ //计算每一堆的SG值  for(int i=0;i<=N;i++){  sg[i]=i;  memset(s, 0, sizeof(s));  for(int j=0; j<15 && fibo[j]<=i; j++){  s[sg[i-fibo[j]]] = 1;  for(int j=0; j<=i; j++)  if(!s[j]) { sg[i] = j; break;}  }  }  }  int main(){  getSG(); //预计算sg值  int n,m,p;  while(cin>>n>>m>>p && n+m+p){  if(sg[n]^sg[m]^sg[p]) cout << "Fibo" <<endl;  else cout << "Nacci"<<endl;  }  return 0;  } |

7.9.4 威佐夫游戏

|  |
| --- |
| 例7-20. 取石子游戏 洛谷 P2252 |

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | #include<bits/stdc++.h>  using namespace std;  int main(){  int n, m;  double gold = (1 + sqrt(5))/2; //黄金分割=1.618033988749894…精确到小数15位  while(cin >> n >> m){  int a = min(n, m), b = max(n, m);  double k = (double)(b – a);  int test = (int)(k\*gold); //乘以黄金分割数，然后取整  if(test == a) cout << 0 << endl; //先手败  else cout << 1 << endl; //先手胜  }  return 0;  } |