数位DP/状压dp

数位dp

数位dp是一种计数用的dp，一般就是要统计一个区间[l,r]内满足一些条件数的个数。

举个例子

不要62

统计一个大范围内不含有6,2连号或者4的数字有多少个(大范围一般都是几e9或者2^很多次爆了int)

基本模板

```
//#pragma GCC oplmize(2)
#include <bits/stdc++.h>
#define ll long long
#define sc(x) scanf("%lld",&x)
#define scs(x) scanf("%s",x)
#define pr(x) printf("%lld\n",x)
#define prs(x) printf("%s\n",x)
using namespace std;
const int maxn=1e3+5;
const int mod=998244353;
const double pi=acos(-1.0);
const double eps = 1e-8;
ll dp[15][15],l,r;  //dp[i][j]表示搜到第i位，上一位数是j的情况下的方案总数
int a[15];
ll dfs(int pos,int pre,bool zero,bool limit) //pos当前位置,pre前一位数,zero判断前面是否全是0,limit判断能否与前面的位匹配上，例如123456，枚举到1234时limit就是true
{
    if(!pos) return 1;
    if(!limit && !zero && dp[pos][pre]!=-1) return dp[pos][pre];
    int up=limit?a[pos]:9;   //当前位最大数字
    ll ans=0;
    for(int i=0;i<=up;i++){   //从零枚举到最大数字
        if(i==2 && pre==6) continue;   //不符合题意，跳过
        if(i==4) continue;
        ans+=dfs(pos-1,i,false,limit && i==up);
    }
    if(!limit && !zero) dp[pos][pre]=ans;    //没最高位限制，且没有前导零就记录
    return ans;
}
ll solve(ll x)
{
    int pos=0;
    while(x){
        a[++pos]=x%10;
        x/=10;
    }
    memset(dp,-1,sizeof(dp));
    return dfs(pos,-2,true,true);
}
int main()
{
```

```
    while(~scanf("%lld %lld",&l,&r) && l+r){
        pr(solve(r)-solve(l-1));
    }
    return 0;
}
```

状压dp

- 状压DP，顾名思义，就是使用状态压缩的动态规划。
- 什么是状态压缩呢，下面会举个例子

位运算

一般基础的状压就是将一行的状态压成一个数，这个数的二进制形式反映了这一行的情况。由于使用二进制数来保存被压缩的状态，所以要用到神奇的二进制位运算操作，将一个十进制数转成二进制进行位运算操作再转回十进制数。

位运算包括

- 按位与&（有0为0，其实就是且）
- 按位或|（有1为1，其实就是或）
- 按位取反~（注意负数补码的符号，最前面的第一位是1）
- 异或^（相同为0，不同为1）
- 左移<<
- 右移>>

例题

给你一个n*n的格子的棋盘，每个格子里面有一个非负数。
从中取出若干个数，使得任意的两个数所在的格子没有公共边，就是说所取的数所在的2个格子不能相邻，并且取出的数的和最大。

```
3
75 15 21
75 15 28
34 70 5
/*
比如说第一行的所有可行状态(1表示取，0表示不取)
编号    二进制
1     0 0 0
2     0 0 1
3     0 1 0
4     1 0 0
5     1 0 1
将表中的状态看作二进制表示,只需将每种状态转化为相应的十进制数，即可只用一个数字，就能表示某一种状态
编号    二进制   十进制
1     0 0 0    0
2     0 0 1    1
3     0 1 0    2
4     1 0 0    4
5     1 0 1    5
这种降低表示状态所需维数的方法就叫状态压缩
*/
```

基本模板

```
//#pragma GCC oplmize(2)
#include <bits/stdc++.h>
#define ll long long
#define sc(x) scanf("%lld",&x)
#define scs(x) scanf("%s",x)
#define pr(x) printf("%lld\n",x)
#define prs(x) printf("%s\n",x)
using namespace std;
const int maxn=1e3+5;
const int mod=1e9;
const double pi=acos(-1.0);
const double eps = 1e-8;
int a[21][21];
int dp[21][(1<<17)+5];   //前i行有前j种可能状态时的方案
int f[(1<<17)+5];   //预处理出合法状态
int solve(int i,int val){
    int ans=1,s=0;
    while(val){
        if(val & 1) s+=a[i][ans];
        val/=2;
        ans++;
    }
    return s;
}
int main()
{
    int n;
    while(cin>>n){
        memset(f,0,sizeof(f));
        memset(dp,0,sizeof(dp));
        for(int i=1;i<=n;i++){
            for(int j=1;j<=n;j++){
                cin>>a[i][j];
            }
        }
        int ans=0;
        for(int i=0;i<(1<<n);i++){
            if((i & (i<<1))==0) f[++ans]=i;   //合法的就存好
        }
        for(int i=1;i<=n;i++){    //枚举行
            for(int j=1;j<=ans;j++){   //枚举每行合法的所有方案
                int val=solve(i,f[j]);   //当前状态上的方格数
                for(int k=1;k<=ans;k++){   //枚举上一行合法的所有方案
                    if((f[j]&f[k])==0)  dp[i][j]=max(dp[i][j],dp[i-1][k]+val);
//如果不存在共边,就说明可以取这个数，更新
                }
            }
        }
        int s=0;
        for(int i=0;i<=ans;i++) s=max(s,dp[n][i]);   //最后取出前m行最大的状态
        cout<<s<<endl;
    }
    return 0;
}
```