数位DP

数位dp是用到数字位数性质来计数的动态规划问题。一般形式比较固定：通常问一个范围内满足某种性质的数的计数问题。我们通过做差可以将问题转化为前n个数中满足条件的数的个数。要解决这个问题，一般步骤分为：①初始化；②计算每个数位对答案的贡献；③累加得到答案。

分析思路

这里强烈推荐yxc的数位dp分析法。将数字n分解为一棵树，然后求每一部分对答案的贡献。在初始化的过程，我们一般会求得前X位数关于数字X的答案。我们定义一个函数f(n)来解决前n个数中满足条件的数的个数。首先特判n是否为0，不为0就按位记录每个数码。我们在计算每位数的贡献时，在[0,num[i]-1]范围枚举数码，这样做是为了保证统计到的是小于n的答案，然后判一下最后一个数是否满足条件即可。

例题

数字计数（模板题）

题意：输出[L,R]范围内每个数码的出现次数。

我们设f(i,j)表示i位数中j数码出现次数，就有以下递推公式

$$\begin{cases} f(i,j)=0, i=0 \\ f(i,j)=10\times f(i-1,j)+10^{i-1}, i>0 \end{cases}$$

预处理这个出来之后就是常规的数位DP了，处理前导0另外减去10的幂次即可。

```
#include<bits/stdc++.h>
#define inf 0x3f3f3f3f
```

```
#define int long long
using namespace std;
const int N=30;

int pw[N],num[N],cnt[10],dp[N],tmp[10];

void init(){
    pw[0]=1;for(int i=1;i<N;i++) pw[i]=pw[i-1]*10;
    dp[0]=0;for(int i=1;i<N;i++) dp[i]=(dp[i-1]*10+pw[i-1]);
}

int solve(int n){
    int len=0,res=0,lst=n;
    while(n){
        num[++len]=n%10;
        n/=10;
    }
    for(int i=len;i>=1;i--){
        for(int j=0;j<=9;j++)   cnt[j]+=dp[i-1]*num[i];
        for(int j=0;j<num[i];j++)   cnt[j]+=pw[i-1];
        lst-=num[i]*pw[i-1];
        cnt[num[i]]+=lst+1;
        cnt[0]-=pw[i-1];
    }
    return res;
}

signed main(){
    ios::sync_with_stdio(0);
    cin.tie(0);cout.tie(0);
    init();
    int L,R;
    cin>>L>>R;
    solve(R);
    for(int i=0;i<=9;i++) tmp[i]=cnt[i];
    memset(cnt,0,sizeof(cnt));
    solve(L-1);
    for(int i=0;i<=9;i++) cout<<tmp[i]-cnt[i]<<" ";
    return 0;
}
```

数字游戏

题意：求[L,R]的不降数个数。定义不降数为从高位到低位的数字是不下降的。

```
//#pragma GCC optimize("Ofast", "inline", "-ffast-math")
//#pragma GCC target("avx,sse2,sse3,sse4,mmx")
#include<bits/stdc++.h>
#define inf 0x3f3f3f3f
#define int long long
using namespace std;
const int N=2e5+7;
const int mod=1e9+7;
```

```
//int read(){   int x=0,f=1;char ch=getchar();while(ch<'0'||ch>'9'){if(ch=='-')
f=f*-1;ch=getchar();}while(ch>='0'&&ch<='9'){x=x*10+ch-'0';ch=getchar();}return
x*f;}

int a,b,dp[N][40],num[40];

void init(){ //预处理dp[i][j]表示最高位是j，一共有i位的不降数个数
    for(int i=0;i<=9;i++) dp[1][i]=1;
    for(int i=2;i<35;i++){
        for(int j=0;j<=9;j++){
            for(int k=j;k<=9;k++)
                dp[i][j]+=dp[i-1][k];
        }
    }
}

int Solve(int n){
    int len=0,res=0,lst=0;
    if(n==0) return 1;
    while(n){
        num[++len]=n%10;
        n/=10;
    }
    for(int i=len;i>=1;i--){  //从高位向低位枚举
        for(int j=lst;j<num[i];j++) res+=dp[i][j]; //当前位要填比上一位大的数
        if(num[i]<lst) break;
        lst=num[i]; //表示上一位的数
        if(i==1) res++; //最低位满足的答案
    }
    return res;
}

signed main(){
    ios::sync_with_stdio(0);
    cin.tie(0);cout.tie(0);
    init();
    while(cin>>a>>b){
        cout<<Solve(b)-Solve(a-1)<<"\n";
    }
    return 0;
}
```

不要62

题意：求[L,R]中有多少个带有62或者带4的数。

这题数据范围出的巨水，暴力也可以过，数位DP的话多记录前面一个数码，剩下的就是模板了。

```
/*  这里是暴力的代码
//#pragma GCC optimize("Ofast", "inline", "-ffast-math")
//#pragma GCC target("avx,sse2,sse3,sse4,mmx")
#include<bits/stdc++.h>
using namespace std;
const int N=1e7+7;
const int mod=1e9+7;
```

```
int l,r,flag=0,tmp=0,sum[N];

signed main(){
    sum[0]=0;
    for(int i=1;i<=1e7;i++){
        tmp=i,flag=0;
        while(tmp){
            if(tmp%10==4) flag=1;
            else if(tmp>=62&&(tmp-62)%100==0) flag=1;
            tmp/=10;
        }
        sum[i]=sum[i-1]+flag;
    }
    cin>>l>>r;
    while(l||r){
        cout<<(r-l+1)-(sum[r]-sum[l-1])<<"\n";
        cin>>l>>r;
    }
    return 0;
}*/

#include<bits/stdc++.h>
using namespace std;
const int N=10;

int L,R,dp[N][10],num[N];

void init(){
    for(int i=0;i<=9;i++) if(i!=4) dp[1][i]=1;
    for(int i=2;i<N;i++){
        for(int j=0;j<=9;j++){
            if(j==4) continue;
            for(int k=0;k<=9;k++){
                if(k==4||j==6&&k==2) continue;
                dp[i][j]+=dp[i-1][k];
            }
        }
    }
}

int solve(int n){
    int len=0,res=0,lst=0;
    if(!n) return 1;
    while(n){
        num[++len]=n%10;
        n/=10;
    }
    for(int i=len;i>=1;i--){
        for(int j=0;j<num[i];j++){
            if(j==4||lst==6&&j==2) continue;
            res+=dp[i][j];
        }
        if(num[i]==4||lst==6&&num[i]==2) break;
        lst=num[i];
        if(i==1) res++;
    }
    return res;
```

```
}

signed main(){
    init();
    while(cin>>L>>R,L||R){
        cout<<solve(R)-solve(L-1)<<"\n";
    }
}
```

Windy数

题意：求[L,R]有多少个满足相邻数位差的绝对值大于等于2的数字个数

```
//#pragma GCC optimize("Ofast", "inline", "-ffast-math")
//#pragma GCC target("avx,sse2,sse3,sse4,mmx")
#include<bits/stdc++.h>
#define inf 0x3f3f3f3f
#define int long long
using namespace std;
const int N=2e5+7;
const int mod=1e9+7;

//int read(){   int x=0,f=1;char ch=getchar();while(ch<'0'||ch>'9'){if(ch=='-')
f=f*-1;ch=getchar();}while(ch>='0'&&ch<='9'){x=x*10+ch-'0';ch=getchar();}return
x*f;}

int L,R,dp[40][15],num[40];

void init(){
    for(int i=0;i<=9;i++) dp[1][i]=1;
    for(int i=2;i<=35;i++){
        for(int j=0;j<=9;j++){
            for(int k=0;k<=j-2;k++) dp[i][j]+=dp[i-1][k];
            for(int k=j+2;k<=9;k++) dp[i][j]+=dp[i-1][k];
        }
    }
}

int Solve(int n){
    int len=0,res=0,lst=-2;
    if(!n) return 0;
    while(n){
        num[++len]=n%10;
        n/=10;
    }
    for(int i=len;i>=1;i--){
        if(i==len) for(int j=1;j<num[i];j++) res+=dp[i][j];
        else{
            for(int j=0;j<num[i];j++)
                if(abs(j-lst)>=2) res+=dp[i][j];
        }
        if(abs(num[i]-lst)>=2) lst=num[i];
        else break;
        if(i==1) res++;
    }
```

```
    for(int i=1;i<len;i++) for(int j=1;j<=9;j++) res+=dp[i][j]; //处理前导零的情况
    return res;
}

signed main(){
    ios::sync_with_stdio(0);
    cin.tie(0);cout.tie(0);
//  freopen("in.cpp","r",stdin);
//  freopen("out.cpp","w",stdout);
    init();
    cin>>L>>R;
    cout<<Solve(R)-Solve(L-1)<<"\n";
    return 0;
}
```

数字游戏

题意：给定L,R,N，求[L,R]区间中有多少个数每位之和模N等于0。

多记录一维表示取模N的余数，并且记下前面k位的数字和。

```
#include<bits/stdc++.h>
using namespace std;
const int N=11,M=110;

int dp[N][10][M],L,R,P,num[N];

int mod(int x,int y){
    return (x%y+y)%y;
}

void init(){  //预处理dp[i][j][k]表示i位数最高位为j，取模为k的方案数
    memset(dp,0,sizeof(dp));
    for(int i=0;i<=9;i++) dp[1][i][i%P]=1;
    for(int i=2;i<N;i++){
        for(int j=0;j<=9;j++){
            for(int k=0;k<P;k++){
                for(int x=0;x<=9;x++){
                    dp[i][j][k]+=dp[i-1][x][mod(k-j,P)];
                }
            }
        }
    }
}

int solve(int n){
    int len=0,res=0,lst=0;
    if(!n) return 1;
    while(n){
        num[++len]=n%10;
        n/=10;
    }
    for(int i=len;i>=1;i--){
        for(int j=0;j<num[i];j++) res+=dp[i][j][mod(-lst,P)];
        lst+=num[i];
```

```
        if(i==1&&lst%P==0) res++;
    }
    return res;
}

signed main(){
    while(cin>>L>>R>>P){
        init();
        cout<<solve(R)-solve(L-1)<<"\n";
    }
}
```

恨7不成妻

题意：给定[L,R]，求区间内满足（如下条件的）数字的平方和。

1. 整数中某一位是 7；
2. 整数的每一位加起来的和是 7 的整数倍；
3. 这个整数是 7 的整数倍。

这道题性质比较多，难度也较大。我们可以用记忆化搜索的写法，来处理dp[i][j][k]表示前i为每一位加起来模7为j,整个数模7为k的数字的是否存在，记录个数和平方。

$s0$直接个数相加即可

$$s1=\sum_k(\overline{aA})=\sum_k(a\times 10^{p-1}+A)=(\sum_k 1)\times(a\times 10^{p-1})+\sum_k A$$
$$s2=\sum_k(\overline{iA})^2=\sum_k(i\times 10^{p-1}+A)^2$$

```
#include<bits/stdc++.h>
#define int long long
using namespace std;
typedef long long ll;
const ll mod=1e9+7;
int t,L,R,len,num[25],pw[30];

struct P{
    ll s0,s1,s2;
    P(){s0=-1,s1=s2=0;}
    P(ll s0,ll s1,ll s2):s0(s0),s1(s1),s2(s2){}
}vis[25][10][10];

P dfs(int stp,int x,int k,bool lim,bool zero){
    if(!stp){
        if(x&&k) return P(1,0,0);
        return P(0,0,0);
    }
    if(!lim&&!zero&&vis[stp][x][k].s0!=-1) return vis[stp][x][k];
    P ans(0,0,0);
    int j=lim?num[stp]:9;
    for(int i=0;i<=j;++i){
        if(i==7) continue;
        P nxt=dfs(stp-1,(x*10+i)%7,(k+i)%7,lim&&(i==j),zero&&(i==0));
        ans.s0=(ans.s0+nxt.s0)%mod;
        ans.s1=(ans.s1+nxt.s1+pw[stp]*i%mod*nxt.s0%mod)%mod;
```

```
        ans.s2=(ans.s2+nxt.s2+2ll*pw[stp]*i%mod*nxt.s1%mod)%mod;
        ans.s2=(ans.s2+(nxt.s0*pw[stp]%mod*pw[stp]%mod*i*i%mod))%mod;
    }
    if(!lim&&!zero) vis[stp][x][k]=ans;
    return ans;
}

ll solve(ll x){
    len=0;
    while(x){
        num[++len]=x%10;
        x/=10;
    }
    return dfs(len,0,0,1,1).s2;
}


signed main(){
    ios::sync_with_stdio(0);
    cin.tie(0);cout.tie(0);
    cin>>t;
    pw[1]=1;for(int i=2;i<30;++i) pw[i]=pw[i-1]*10%mod;
    while(t--){
        cin>>L>>R;
        ll ans=(solve(R)-solve(L-1)+mod)%mod;
        cout<<ans<<"\n";
    }
    return 0;
}
```

[SDOI2014] 数数（AC自动机+数位DP）

给$m \leq 100$个数集，$\sum_{i=1}^{m} |s_i| \leq 1500$，问$n \leq 10^{1201}$以内有多少个数不包含数字中的数作为子串。

```
#include<bits/stdc++.h>
#define int long long
using namespace std;
const int mod=1e9+7;
const int N=2007;

int n,ch[N][10],fail[N],sz,dp[N][N];
char S[N],T[N];
bool ed[N];

void insert(char s[]){  //AC自动机的插入
    int now=0,len=strlen(s);
    for(int i=0;i<len;++i){
        int p=s[i]-'0';
        if(!ch[now][p]) ch[now][p]=++sz;
        now=ch[now][p];
    }
    ed[now]|=1;
    return;
}
```

```
void build_fail(){ //建立Fail树
    queue<int>q;
    for(int i=0;i<10;++i) if(ch[0][i]) q.emplace(ch[0][i]);
    while(q.size()){
        int x=q.front();
        q.pop();
        for(int i=0;i<10;++i){
            int y=ch[x][i];
            if(!y) ch[x][i]=ch[fail[x]][i];
            else{
                fail[y]=ch[fail[x]][i];
                ed[y]|=ed[fail[y]];
                q.emplace(y);
            }
        }
    }
}
//dp[i][j]表示字符串前i位匹配到Trie树上第j个点的答案
inline int dfs(int stp,int pos,bool lim,bool zero){
    if(stp<0) return !ed[pos];
    if(ed[pos]) return 0;
    if(!lim&&!zero&&dp[stp][pos]!=-1) return dp[stp][pos];
    int res=0,j=(lim)?(S[stp]-'0'):9;
    for(int i=0;i<=j;++i){
        res=(res+dfs(stp-1,(zero&(i==0))?0:ch[pos][i],lim&(i==S[stp]-'0'),zero&
(i==0)))%mod;
    }
    if(!lim&&!zero) dp[stp][pos]=res;
    return res;
}

signed main(){
    scanf("%s%lld",S,&n);
    int Len=strlen(S);
    for(int i=1;i<=n;++i){
        scanf("%s",T);
        insert(T);
    }
    build_fail();
    reverse(S,S+Len); //翻转字符串方便操作
    memset(dp,-1,sizeof(dp));
    int ans=dfs(Len-1,0,1,1);
    ans=(ans-1+mod)%mod;
    cout<<ans<<"\n";
    return 0;
}
```

三羊开泰

给定四个整数A,B,C,X，求三个非负整数组成的有序三元组(a,b,c)的个数，同时满足以下两个条件：

- $a \leq A, b \leq B, c \leq C$
- $a \text{ xor } b \leq X, b \text{ xor } c \leq X, c \text{ xor } a \leq X$

```
//#pragma GCC optimize("Ofast", "inline", "-ffast-math")
//#pragma GCC target("avx,sse2,sse3,sse4,mmx")
```

```cpp
#include<bits/stdc++.h>
#define inf 0x3f3f3f3f
#define int long long
using namespace std;
const int N=1e6+7;
const int mod=1e9+7;

//int read(){   int x=0,f=1;char ch=getchar();while(ch<'0'||ch>'9'){if(ch=='-')
f=f*-1;ch=getchar();}while(ch>='0'&&ch<='9'){x=x*10+ch-'0';ch=getchar();}return
x*f;}
//void write(int x){if(x>9) write(x/10);putchar(x%10+'0');}
int a,b,c,x,cnta,cntb,cntc,cntx;
int aa[N],bb[N],cc[N],xx[N];
int dp[100][2][2][2][2][2][2];
/*
对于二进制的每一位i（从高到低），考虑a,b,c能填0~num[i]之中的哪些数字
填上这些数字之后判断a^b,b^c,c^a是否大于x，合法则计算贡献。
*/
inline int dfs(int pos,int la,int lb,int lc,int lab,int lac,int lbc){
    if(pos==-1) return 1;
    int &ans=dp[pos][la][lb][lc][lab][lac][lbc];
    if(ans!=-1) return ans;
    ans=0;
    int upa=la?aa[pos]:1;
    int upb=lb?bb[pos]:1;
    int upc=lc?cc[pos]:1;
    int upx=xx[pos];
    for(int i=0;i<=upa;++i){
        for(int j=0;j<=upb;++j){
            for(int k=0;k<=upc;++k){
                if(lab&&((i^j)>xx[pos])) continue;
                if(lac&&((i^k)>xx[pos])) continue;
                if(lbc&&((j^k)>xx[pos])) continue;
                ans=(ans+dfs(pos-1,la&&(i==upa),lb&&(j==upb),lc&&(k==upc),lab&&
(i^j==xx[pos]),lac&&(i^k==xx[pos]),lbc&&(j^k==xx[pos]))%mod)%mod;
            }
        }
    }
    return ans;
}

void Solve(){
    cin>>a>>b>>c>>x;
    memset(aa,0,sizeof aa);
    memset(bb,0,sizeof bb);
    memset(cc,0,sizeof cc);
    memset(xx,0,sizeof xx);
    memset(dp,-1,sizeof dp);
    cnta=cntb=cntc=cntx=0;
    while(a) aa[cnta++]=a&1,a>>=1;
    while(b) bb[cntb++]=b&1,b>>=1;
    while(c) cc[cntc++]=c&1,c>>=1;
    while(x) xx[cntx++]=x&1,x>>=1;
    cout<<dfs(63,1,1,1,1,1,1)<<"\n";
}

signed main(){
    ios::sync_with_stdio(0);
```

```
    cin.tie(0);cout.tie(0);
//  freopen("in.cpp","r",stdin);
//  freopen("out.cpp","w",stdout);
    int T=1;
    cin>>T;
//  clock_t start,finish;
//  start=clock();
    while(T--){
        Solve();
    }
//  finish=clock();
//  cerr<<((double)finish-start)/CLOCKS_PER_SEC<<endl;
    return 0;
}
```

CF776G Sherlock and the Encrypted Data

q 组询问，以 16 进制的形式给出 L, RL,R，求出满足以下条件的整数 xx 的个数。

1. $L \leq x \leq R$。
2. $f(x) < x$。其中 f(x) 定义如下：

设 x 在 16 进制下的数位分别为 $\overline{d_k d_{k-1} d_{k-2} \cdots d_0}$，令 $y = \sum_{i=0}^{15} 2^i [i \in d]$，则 $f(x) = x \oplus y$。换句话说，$f(x)$ 等于 x 异或上2的所有在 x 的 16 进制下出现的数码"次幂之和"。

$1 \leq q \leq 10^4, 0 \leq L \leq R < 16^{15}$

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

int a[20],len;
int dp[20][2],high;

int dfs(int stp,bool lim,bool zero,int s){
    if(!stp) return s;
    if(!lim&&!zero&&dp[stp][s]!=-1) return dp[stp][s];
    int j=lim?a[stp]:15,ans=0;
    j=min(j,high);
    for(int i=0;i<=j;++i){
        if(((stp-1)<<2)==((high>>2)<<2)&&!(i>>(high%4)&1)) continue;
        int ns=s;
        if((zero&&i==0)==0) if(i==high) ns=1;
        ans+=dfs(stp-1,lim&&(i==a[stp]),zero&&(i==0),ns);
    }
    if(!lim&&!zero) dp[stp][s]=ans;
    return ans;
}

int solve(int x){
    len=0;
    while(x){
        a[++len]=x%16;
        x/=16;
```

```
    }
    int ans=0;
    for(high=4;high<min(16ll,len*4);high++){  //只需要存储sum最高位即可,0~3位不影响
        memset(dp,-1,sizeof(dp));
        ans+=dfs(len,1,1,0);
    }
    return ans;
}

signed main(){
    int t,l,r;
    scanf("%lld",&t);
    while(t--){
        scanf("%llx%llx",&l,&r);
        int ans=solve(r)-solve(l-1);
        printf("%lld\n",ans);
    }
    return 0;
}
```

2020ICPC上海 C-Sum of Log

给定$X,Y\leq 1e9$，求$\sum_{i=0}^{X}\sum_{j=[i=0]}^{Y}[i\&j=0]\lfloor log_2(i+j)+1\rfloor$

```
#include<bits/stdc++.h>
#define pii pair<int,int>
#define mk make_pair
#define F first
#define S second
#define int long long
using namespace std;
const int mod=1e9+7;
const int N=70;

int X,Y,num1[N],num2[N],len;
int dp[N][2][2][2];

int dfs(int stp,int zero,int lim1,int lim2){
    if(!stp) return 1;
    if(dp[stp][zero][lim1][lim2]) return dp[stp][zero][lim1][lim2];
    int res=0,j=(lim1)?num1[stp]:1,k=(lim2)?num2[stp]:1;
    for(int a=0;a<=j;++a){
        for(int b=0;b<=k;++b){
            if(a&b) continue;
            int num=1;
            if(zero&&(a||b)) num=stp;
            res+=dfs(stp-1,zero&&!a&&!b,lim1&&(a==num1[stp]),lim2&&
(b==num2[stp]))*num%mod;
            res%=mod;
        }
    }
    return dp[stp][zero][lim1][lim2]=res;
}

int solve(){
```

```
    memset(num1,0,sizeof(num1));
    memset(num2,0,sizeof(num2));
    memset(dp,0,sizeof(dp));
    len=0;
    while(X||Y){
        ++len;
        num1[len]=X&1;
        num2[len]=Y&1;
        X>>=1;Y>>=1;
    }
    return (dfs(len,1,1,1)-1+mod)%mod;
}

signed main(){
    ios::sync_with_stdio(0);
    cin.tie(0);cout.tie(0);
    int T=1;
    cin>>T;
    while(T--){
        cin>>X>>Y;
        cout<<solve()<<"\n";
    }
    return 0;
}
```

2022 CCPC广州 M-XOR Sum

给定$n,m,k(0\le n\le 10^{15},0\le m\le 10^{12},1\le k\le 18)$，求一个长度为$k$的数组$a$，$a_i$为$[0,m]$的整数，满足$\sum_{i=1}^{k}\sum_{j=1}^{i-1}a_i\oplus a_j=n$的方案数，对1e9+7取模。

```
#include<bits/stdc++.h>
typedef long long ll;
using namespace std;
const int N=65;
const ll mod=1e9+7;

int k,len=0;
bool num[N];
ll n,m,C[N][N];
map<ll,int>dp[N][20];

int dfs(int stp,int d,ll lf){ //假设有d个数达到上界
    if(!stp) return (lf==0);
    if(((1ll<<stp)-1)*(k/2)*(k-k/2)<lf) return 0; //剩下的数没办法贡献到lf
    if(dp[stp][d].count(lf)) return dp[stp][d][lf];
    ll res=0;
    if(num[stp]){
        for(int i=0;i<=d;++i){ //如果保持i个数是1并且达到上界
            for(int j=0;j<=k-d;++j){ //再从剩下的数中选j个数为1
                ll tmp=(1ll<<(stp-1))*(i+j)*(k-(i+j));
                if(lf-tmp<0) continue;
                res+=C[d][i]*C[k-d][j]%mod*dfs(stp-1,i,lf-tmp)%mod;
                res%=mod;
            }
        }
```

```
    }else{
        for(int j=0;j<=k-d;++j){ //在没达到上界的数中选j个1
            ll tmp=(1ll<<(stp-1))*j*(k-j);
            if(lf-tmp<0) continue;
            res+=C[k-d][j]*dfs(stp-1,d,lf-tmp)%mod;
            res%=mod;
        }
    }
    return dp[stp][d][lf]=res;
}

signed main(){
    C[0][0]=1;
    for(int i=1;i<=40;++i){
        C[i][0]=C[i][i]=1;
        for(int j=1;j<i;++j){
            C[i][j]=C[i-1][j-1]+C[i-1][j];
        }
    }
    scanf("%lld%lld%d",&n,&m,&k);
    len=0;
    while(m){
        num[++len]=m%2;
        m>>=1;
    }
    printf("%d\n",dfs(len,k,n));
    return 0;
}
```

参考文献

https://www.bilibili.com/video/BV1yT4y1u7jW

https://www.luogu.com.cn/blog/writeSTL/solution-p4999

https://blog.csdn.net/hzf0701/article/details/116717851