数位DP

BY GY

简介&应用范围

- 求出在给定区间 [A,B] 内，符合条件 f(i) 的数 i 的个数。条件 f(i) 一般与数的大小无关，而与数的组成有关
- 数位，指十位、百位、千位……等
- 顾名思义，DP是数位上进行的，所以数的大小对复杂度影响不大
- 遇到给出两个很大的数确定范围，要求记录符合某一条件的数的个数时，大概率是数位DP
- 推荐使用记忆化搜索实现
- 存在背模板这种操作，但是建议理解原理

数位DP

- 从最高位向最低位搜索，达到最底层时得到方案数，返回并累加方案数，返回起点时得到最终答案
- 需要注意的是，题目一般都会有上限与下限，搜索时同时兼顾这两者会使程序十分复杂，一般转化为两次数位DP，一次[0,l-1],一次[0,r]。两次结果相减就得到了我们需要的结果

状态设计

- 最基本的：当前在哪一位（now），最高位限制（lim）
- 可能需要的：前导零标记（lead），前一位数（pre）
- 需要注意的是，前导零也可以使用特殊的数字表示
- 更多参数视题意而定，有时可能需要记录前几位数而非仅仅一位。

前导零标记（lead）

- 大部分题目不允许前导零的存在
- 所以我们需要一个标记来区分正常零与前导零
- 如果前一位为前导零当前位也是零，那么当前位也是前导零
- 如果前一位不是前导零而当前位是零，那么当前位不是前导零

最高位限制（lim）

- 记录高位是否是达到上限的状态
- 高位是否达到上限对当前位的选择范围有影响
- 例如范围是[0,666]
- 百位为6时，十位的范围为[0,6]
- 百位为[0,5]时,十位的范围为[0,9]

从洛谷上翻到的一个模板

```
ll dfs(int pos,int pre,int st,……,int lead,int limit)//记搜
{
    if(pos>len) return st;//剪枝
    if((dp[pos][pre][st]……[……]!=-1&&(!limit)&&(!lead))) return dp[pos][pre][st]……[……];//记录当前值
    ll ret=0;//暂时记录当前方案数
    int res=limit?a[len-pos+1]:9;//res当前位能取到的最大值
    for(int i=0;i<=res;i++)
    {
        //有前导0并且当前位也是前导0
        if((!i)&&lead) ret+=dfs(……,……,……,i==res&&limit);
        //有前导0但当前位不是前导0，当前位就是最高位
        else if(i&&lead) ret+=dfs(……,……,……,i==res&&limit);
        else if(根据题意而定的判断) ret+=dfs(……,……,……,i==res&&limit);
    }
    if(!limit&&!lead) dp[pos][pre][st]……[……]=ret;//当前状态方案数记录
    return ret;
}
ll part(ll x)//把数按位拆分
{
    len=0;
    while(x) a[++len]=x%10,x/=10;
    memset(dp,-1,sizeof dp);//初始化-1（因为有可能某些情况下的方案数是0）
    return dfs(……,……,……,……);//进入记搜
}
int main()
{
    scanf("%d",&T);
    while(T--)
    {
        scanf("%lld%lld",&l,&r);
        if(l) printf("%lld",part(r)-part(l-1));//[l,r](l!=0)
        else printf("%lld",part(r)-part(l));//从0开始要特判
    }
    return 0;
}
```

例1：P2657windy数

题目描述

windy定义了一种windy数。不含前导零且相邻两个数字之差至少为2的正整数被称为windy数。 windy想知道，

在A和B之间，包括A和B，总共有多少个windy数?

输入输出格式

输入格式：

包含两个整数，A B。

输出格式：

一个整数

Windy数

- 十分明显的数位DP
- 满足了数很大，有上下限，计数几个特点
- 状态设计需要记录当前在第几位，上一位是几，是否达到上限和是否有前导零
- 其中前导零可以用特殊数字代替，简化一维

Windy数

- 平平无奇的输入输出
- 放出来只是因为提醒一下要注意l--

```
int l,r;
scanf("%d%d",&l,&r);
l--;
int ansl=slove(l);
int ansr=slove(r);
printf("%d",ansr-ansl);
```

Windy数

- 将输入拆分
- 不要忘记数字0也是一位的
- 前导零使用特殊数字12代替
- 因为前导零对后续数字无影响
- 而12与[0,9]的差距均大于二
- 对后续数字也无影响
- 记忆化数组不要忘记初始化

```
int slove(int x){
    cntl=0;
    while(x){
        L[++cntl]=x%10;
        x/=10;
    }
    if(!cntl) cntl++;
    int ans=0;
    memset(f,-1,sizeof(f));
    for(int i=0;i<=L[cntl];i++)
        ans+=dfs(cntl-1,i==0?12:i,i==L[cntl]?1:0);
    return ans;
}
```

Windy数

- Abs为取绝对值
- 注意上限up
- 与高位是否达
- 到上限有关
- 注意前导零和最
- 高位限制标记

```
int abs(int x){
    return x>0?x:-x;
}
int dfs(int now,int pre,int lim){
    if(now==0) return 1;
    if(f[now][pre][lim]>-1) return f[now][pre][lim];
    int ans=0;
    int up=lim?L[now]:9;
    for(int i=0;i<=up;i++)
        if(abs(i-pre)>=2)
            ans+=dfs(now-1,((i==0)&&(pre==12))?12:i,((i==up)&&lim)?1:0);
    return f[now][pre][lim]=ans;
}
```

P2602数字计数

题目描述

给定两个正整数a和b，求在[a,b]中的所有整数中，每个数码(digit)各出现了多少次。

输入输出格式

输入格式：

输入文件中仅包含一行两个整数a、b，含义如上所述。

输出格式：

输出文件中包含一行10个整数，分别表示0-9在[a,b]中出现了多少次。

数字计数

- 熟悉的上下限，熟悉的巨大的数，熟悉的计数
- 没错就是你了，数位DP！
- 有一点不同，并非记录满足某个条件的数有多少个
- 而是记录每个数码出现了多少次
- 解决方法是多次DP，一次记录一种数码
- 同时数据范围和最终结果都是超过int类型的
- 不要忘记开long long

状态设计

- 需要记录的有：
- 当前在第几位
- 是否有前导零
- 高位是否达到上限
- 当前统计的数字是已经出现多少次

数字计数

- Slove函数几乎没变
- 注意要枚举每个数码
- 每次记忆化搜索前
- 不要忘记初始化
- 记忆化数组

```
void slove(ll x,int flag){
    cntl=0;
    while(x){
        L[++cntl]=x%10;
        x/=10;
    }
    if(!cntl) cntl++;
    for(int i=0;i<=9;i++){
        memset(f,-1,sizeof(f));
        if(!flag) ansl[i]=dfs(cntl,1,1,0,i);
        else ansr[i]=dfs(cntl,1,1,0,i);
    }
}
```

数字计数

- 注意返回的是出现的次数

```
ll dfs(int now,int lead,int lim,int num,int tag){
    if(now==0) return num;
    if(f[now][lead][lim][num]>-1) return f[now][lead][lim][num];
    int up=lim?L[now]:9;
    ll ans=0;
    for(int i=0;i<=up;i++){
        ans+=dfs(now-1,lead&&(i==0),lim&&(i==up),num+(tag==i&&(!lead||(lead&&i!=0))),tag);
    }
    return f[now][lead][lim][num]=ans;
}
```

P4317花神的数论题

题目描述

话说花神这天又来讲课了。课后照例有超级难的神题啦..... 我等蒟蒻又遭殃了。 花神的题目是这样的：设 $\mathrm{sum}(i)$ 表示 i 的二进制表示中 1 的个数。给出一个正整数 N ， 花神要问你 $\prod_{i=1}^{N}\mathrm{sum}(i)$ ， 也就是 $\mathrm{sum}(1)\sim\mathrm{sum}(N)$ 的乘积。

输入输出格式

输入格式：

一个正整数 N。

输出格式：

一个数，答案模 10000007 的值。

花神的数论题

- 题干说是数论题但其实和数论没关系
- 或许快速幂稍微沾点边？
- 因为统计的是二进制中1的个数
- 所以需要转化成二进制，对二进制的每一位进行数位DP
- 题干要求求出1~n每一个数中1出现的次数
- 具体做的时候可以反过来，枚举1出现的次数K，统计1~n中哪些数出现了K次1。

状态设计

- 需要记录的：
- 当前枚举到了第几位
- 高位是否达到上限
- 出现了几次1
- 当前目标结果是出现几次1

花神的数论题

- 拆分N时注意要转化成二进制
- 不要忘记初始化和取模
- Fast是快速幂
- i个1的情况出现了ans次
- 就相当于给结果乘上i^ans

```
scanf("%lld",&n);
while(n){
    L[++cntl]=n&1;
    n>>=1;
}
memset(f,-1,sizeof(f));
for(int i=1;i<=cntl;i++){
    ll ans=dfs(cntl,1,0,i);
    ansss=(ansss*fast(i,ans))%p;
}
printf("%lld",ansss);
```

花神的数论题

- 几乎没变的
- 记忆化搜索
- 只需要注意
- 二进制只有
- 0和1，上限
- 不再是9。
- 所以说数位DP几乎就是套模板

```
ll dfs(int now,int lim,int num,int aim){
    if(now==0) return num==aim;
    if(f[now][lim][num][aim]>-1) return f[now][lim][num][aim];
    int up=lim?L[now]:1;
    ll ans=0;
    for(int i=0;i<=up;i++){
        ans+=dfs(now-1,lim&&(i==up),num+i,aim);
    }
    return f[now][lim][num][aim]=ans;
}
```

课后练习

- P3413萌数
- P4127同类分布
- P4124手机号码
- 都不太难，就不再按按难度分类