数位dp简单总结

几年前简单的做过几道数位dp的题目，基于单个数字，但是基于的模板是对于前导零与上界限制没有记忆化的一个板子，基本想法就是有限制就暴力，无限制就记忆化。

```
ll dp[20][state];
ll dfs(int pos,/*state变量*/,bool lead/*前导零*/,bool limit/*数位上界变量*/)//不是每个题都要判断前导零
{
    //递归边界，既然是按位枚举，最低位是0，那么pos==-1说明这个数我枚举完了
    if(pos==-1) return 1;/*这里一般返回1，表示你枚举的这个数是合法的，那么这里就需要你在枚举时必须每一位都要满足题目条件，也就是说当前枚举到pos位，一定要保证前面已经枚举的数位是合法的。不过具体题目不同或者写法不同的话不一定要返回1*/
    //第二个就是记忆化(在此前可能不同题目还能有一些剪枝)
    if(!limit && !lead && dp[pos][state]!=-1) return dp[pos][state];
    /*常规写法都是在没有限制的条件记忆化，这里与下面记录状态是对应，具体为什么是有条件的记忆化后面会讲*/
    int up=limit?a[pos]:9;//根据limit判断枚举的上界up;这个的例子前面用213讲过了
    ll ans=0;
    //开始计数
    for(int i=0;i<=up;i++)//枚举，然后把不同情况的个数加到ans就可以了
    {
        if() ...
        else if()...
        ans+=dfs(pos-1,/*状态转移*/,lead && i==0,limit && i==a[pos])
    }
    //计算完，记录状态
    if(!limit && !lead) dp[pos][state]=ans;
    return ans;
}
```

下面来换一种想法。

先看题：

H-Harmony Pairs

题意：给定n然后求出从0到n有多少个数对，满足第一个数小于第二个数，但是第一个数的各位之和要大于第二个数。

这类dp题与之前不同的是要统计的不是单个数字而是数对

题解$dp[i][j][2][2][2]$表示，当前到了第i位，前面搜素的数位之差为j第一个数最高位是否有限制，第二个数最高位是否有限制，第二个数是否大于了第一个数。

将限制也记忆化

```
LL DFS(int pos, int sum, bool lima, bool limb, bool limit) {
    if(pos < 0) {
        return sum > base;
    }
    if(DP[pos][sum][lima][limb][limit] != -1) {
        return DP[pos][sum][lima][limb][limit];
    }
    int up1 = lima ? str[pos] - '0' : 9;
    int up2 = limb ? str[pos] - '0' : 9;
    LL res = 0;
    for(int i = 0; i <= up1; ++ i) {
        for(int j = 0; j <= up2; ++ j) {
            if(!limit && i > j) {
                continue;
            }
            res += DFS(pos - 1, sum + i - j, lima && i == up1, limb && j ==
up2, limit || j > i);
            res %= mod;
        }
    }
    return DP[pos][sum][lima][limb][limit] = res;
}
```