动态规划 1

数位DP

算法简介

一般用于处理形如区间 $[l,r]$ 中满足条件的数有几个之类问题。

算法思想

首先考虑将区间 $[l,r]$ 拆分成 $[0,r]-[0,l]$

考虑记忆化搜索，一般搜索函数为 $f(pos,s,eq,zero)$

其中 pos 表示当前搜索位置，s 表示前缀状态，eq 表示前缀是否与查询上界相等，$zero$ 表示是否有前导零。

算法练习

习题一

洛谷p2657

题意

询问区间 $[l,r]$ 中满足相邻数位之差至少为 2 的数的个数。

题解

搜索函数中 s 记录前一位状态即可。

```
int a[10],dp[10][10];
int dfs(int pos,int pre,bool eq,bool zero){
    if(!pos)return 1;
    if(!eq&&~dp[pos][pre])return dp[pos][pre];
    int ans=0,v=eq?a[pos]:9;
    _rep(i,0,v){
        if(!zero&&abs(i-pre)<2)
        continue;
        ans+=dfs(pos-1,i,eq&&i==a[pos],zero&&i==0);
    }
    if(!eq&&!zero)dp[pos][pre]=ans;
```

```
    return ans;
}
int solve(int n){
    int len=0;
    mem(dp,-1);
    while(n){
        a[++len]=n%10;
        n/=10;
    }
    return dfs(len,0,true,true);
}
int main()
{
    int a=read_int(),b=read_int();
    printf("%d",solve(b)-solve(a-1));
    return 0;
}
```

习题二

洛谷p2602

题意

询问区间 $[l,r]$ 所有整数中 $0\sim 9$ 分别出现的次数。

题解

依次计算 $0\sim 9$ 出现次数。每次把搜索函数中的 s 当成当前前缀的贡献即可。

```
const int MAXL=15;
LL a[MAXL],dp[MAXL][MAXL];
int goal;
LL dfs(int pos,int pre,bool eq,bool zero){
    if(!pos)return pre;
    if(!eq&&~dp[pos][pre])return dp[pos][pre];
    int v=eq?a[pos]:9;LL ans=0;
    _rep(i,0,v){
        if(zero&&i==0)
        ans+=dfs(pos-1,pre,eq&&i==a[pos],true);
        else
        ans+=dfs(pos-1,pre+(i==goal),eq&&i==a[pos],false);
    }
    if(!eq&&!zero)dp[pos][pre]=ans;
    return ans;
```

```
}
LL solve(LL n){
    int len=0;
    mem(dp,-1);
    while(n){
        a[++len]=n%10;
        n/=10;
    }
    return dfs(len,0,true,true);
}
int main()
{
    LL a=read_LL(),b=read_LL();
    _rep(i,0,9){
        goal=i;
        printf("%lld ",solve(b)-solve(a-1));
    }
    return 0;
}
```

习题三

洛谷p4127

题意

给出两个数 a,b 求出 $[a,b]$ 中各位数字之和能整除原数的数的个数。

题解

发现在各位数字之和不确定情况下难以判定最后结果是否被各位数字之和相当困难。

于是考虑枚举所有可能的各位数字之和 Mod 于是 dp 过程中维护一下前面数位之和以及前缀模 Mod 的结果。

于是每次可能状态共有 $18\ast(18\ast 9)^2$ 个，最坏计算次数 $18\ast(18\ast 9)^3\approx 8\times 10^7$

当然每次合法状态数不可能达到上界。另外可以考虑在 dp 过程中适当剪枝。

```
const int MAXL=20,MAXV=180;
LL a[MAXL],dp[MAXL][MAXV][MAXV];
int Mod;
LL dfs(int pos,int s,int mod,bool eq){
    if(s>Mod||s+pos*9<Mod)return 0;
    if(!pos)return mod==0;
    if(!eq&&~dp[pos][s][mod])return dp[pos][s][mod];
```

```
    int v=eq?a[pos]:9;LL ans=0;
    _rep(i,0,v)
    ans+=dfs(pos-1,s+i,(mod*10+i)%Mod,eq&&i==a[pos]);
    if(!eq)dp[pos][s][mod]=ans;
    return ans;
}
LL solve(LL n){
    int len=0;
    while(n){
        a[++len]=n%10;
        n/=10;
    }
    LL ans=0;
    _rep(i,1,len*9){
        mem(dp,-1);
        Mod=i;
        ans+=dfs(len,0,0,true);
    }
    return ans;
}
int main()
{
    LL a=read_LL(),b=read_LL();
    printf("%lld",solve(b)-solve(a-1));
    return 0;
}
```

习题四

洛谷p3413

题意

如果某个数字按 10 进位制表示的字串中含有长度至少为 2 的连续回文子串，则称该数为萌数。询问区间 $[l,r]$ 中萌数的个数。

数据范围 $1\le l\le r\lt 10^{1000}$输出结果对 10^9+7 取模。

题解

回文串信息难以维护，考虑计算不含回文字串的数字个数。

发现长度不小于 2 的回文串一定含长度为 2 或 3 的回文串。

于是一个数不含长度至少为 2 的回文串等价于一个数不含长度为 2 或 3 的回文串。

于是只需要维护一个数的前面两位，保证当前位不与前两位相同即可。

```
const int MAXL=1e3+5,Mod=1e9+7;
int a[MAXL],dp[MAXL][11][11];
int dfs(int pos,int pre1,int pre2,bool eq){
    if(!pos)return 1;
    if(!eq&&~dp[pos][pre1][pre2])return dp[pos][pre1][pre2];
    int ans=0,v=eq?a[pos]:9;
    _rep(i,0,v){
        if(i==pre1||i==pre2)continue;
        ans=(ans+dfs(pos-1,(i==0&&pre1==10)?10:i,pre1,eq&&i==a[pos]))%Mod;
    }
    if(!eq)dp[pos][pre1][pre2]=ans;
    return ans;
}
int solver(char *s){
    int len=strlen(s);
    _rep(i,1,len)
    a[i]=s[len-i]-'0';
    mem(dp,-1);
    return dfs(len,10,10,true);
}
char b[MAXL],c[MAXL];
int main()
{
    scanf("%s %s",b,c);
    LL ans=solver(c)-solver(b);
    int len_b=strlen(b),len_c=strlen(c),flag=1;
    _for(i,0,len_b){
        if(i+1<len_b&&b[i]==b[i+1]){
            flag=0;
            break;
        }
        if(i+2<len_b&&b[i]==b[i+2]){
            flag=0;
            break;
        }
    }
    ans+=flag;
    LL tot_b=0,tot_c=0;
    _for(i,0,len_b)tot_b=(tot_b*10+b[i]-'0')%Mod;
    _for(i,0,len_c)tot_c=(tot_c*10+c[i]-'0')%Mod;
    enter(((tot_c-tot_b+1-ans)%Mod+Mod)%Mod);
    return 0;
}
```

习题五

牛客暑期多校(第六场) H 题

题意

求 $0\le A\le B\le N$,满足 $S(A)\gt S(B)$ 的 (A,B) 个数，其中 $S(n)$ 代表 n 的数码和。

题解

搜索函数为 $f(pos,d,eq1,eq2)$，其中 d 表示 $S(B)-S(A)$，$eq1$ 表示 B 前缀是否与查询上界相等，$eq2$ 表示 A 前缀是否与 B 前缀相等。

接下来同时枚举 B,A 在 pos 位的数值即可，合法状态共 $O(18L^2)$ 个，每个状态递推时间复杂度 $O(100)$。

于是总时间复杂度 $O(1800L^2)$，其中 L 表示 N 的长度。需要注意搜索过程中需要对 $eq1,eq2$ 进行记忆化。

单个数搜索时不需要对 eq 进行记忆化是因为单个数搜索时从 $eq=true$ 的状态只能走到下一个 $eq=true$ 的状态。

而两个数搜索时，从 $eq1=true,eq2=false$ 的状态可以走到 10 个 $eq1=true,eq2=false$ 的状态。

所以存在某些 $eq1=true,eq2=false$ 的状态会被重复访问的可能，对 $eq1=false,eq2=true$ 的情况类似。

```
const int MAXL=105,MAXV=905,Mod=1e9+7;
int a[MAXL],dp[MAXL][MAXV<<1][2][2];
int dfs(int pos,int d,bool eq1,bool eq2){
    if(d>=MAXV+pos*9)return 0;
    if(!pos)return 1;
    if(~dp[pos][d][eq1][eq2])
    return dp[pos][d][eq1][eq2];
    int ans=0,v1=eq1?a[pos]:9,v2;
    _rep(i,0,v1){
        v2=eq2?i:9;
        _rep(j,0,v2)
        ans=(ans+dfs(pos-1,d+i-j,eq1&&i==v1,eq2&&j==v2))%Mod;
    }
    return dp[pos][d][eq1][eq2]=ans;
}
int solve(char *s){
    int len=strlen(s);
    mem(dp,-1);
    _rep(i,1,len)
    a[i]=s[len-i]-'0';
    return dfs(len,MAXV,true,true);
}
char buf[MAXL];
int main()
{
```

```
    scanf("%s",buf);
    enter(solve(buf));
    return 0;
}
```

习题六

题意

求满足 $0\le x\le A,0\le y\le B,x\oplus y=n,|x-y|\le m$ 的 (x,y) 个数。

题解

不难想到按二进制 dp 但 $|x-y|$ 在 dp 过程中可能上升，也可能下降，不方便处理。

故不妨假设初始时 $x=0,y=n$ 然后若 n 的第 i 位等于 1 且将 1 交给 x 则 $x-y$ 增加 2^{i+1} 这样可以保证 $x-y$ 在 dp 过程中单增。

$|x-y|\le m$ 等价于增量 $n-m\le d\le n+m$ 考虑计算出 $d\le n+m$ 的值减去 $d\le n-m-1$ 的值即可。

搜索函数为 $f(pos,eq1,eq2,eq3)$ 其中 $eq1$ 表示 x 前缀是否与 A 前缀相等 $eq2$ 表示 y 前缀是否与 B 前缀相等。

$eq3$ 表示 d 前缀是否与上界相等，然后枚举每一位可能，使其满足约束 $x\oplus y=n$ 直接转移即可。