数位动态规划

数位动态规划是一类DP问题，该类问题的表现形式为给定区间$[l,r]$，求出满足条件的数的个数。下面以14届蓝桥杯--奇怪的数和洛谷--P2657_windy数为例，来讲解数位DP的思路。

1. windy数

不含有前导零且相邻数字之差至少为2的数称为windy数。解决该问题的最朴素的想法是通过枚举，假如有一个n位的数字，该数字中的每两位的绝对值大于等于2称这样的数是windy数。特别地，仅有一位数的数也是windy数。假设以最高位为根节点的话，先枚举最高位，在给定最高位的情况下枚举$n-1$位，在给定$n-1$位的情况下枚举$n-2$位，以此类推下去直到$n=0$。可见这个问题可以使用递归来解决，假设定义函数$dfs(n,j)$为第n位，且该位为j的windy数个数。这个过程可以通过递归来解决，以下是示例程序。

```
int dfs(int i, int p) // i 是当前的位数, p是当前位置数
{
    if (i == 1)
        return 1;
    int res = 0;
    int limit1 = 0; // lmit1即是下一位置的数
    int limit2 = 9;
    while (limit1 <= limit2) // 枚举i位置的数
    {
        if (abs(limit1 - p) >= 2)      // 满足windy数的条件
            res += dfs(i - 1, limit1); // 在该条件下继续向下深搜
        ++limit1;
    }
    return res;
}
```

这个过程实际存在大量的重复计算，可以通过$dp[i][j]$来存储计算得到的值。示例程序如下。

```
int dfs(int i, int p) // i 是当前的位数，p是当前位置数
{
    if (i == 1)
    {
        dp[i][p] = 1;
        return 1;
    }
    if (dp[i][p] != -1)
        return dp[i][p];
    int res = 0;
    int limit1 = 0; // lmit1即是下一位置的数
    int limit2 = 9;
    while (limit1 <= limit2) // 枚举i位置的数
    {
        if (abs(limit1 - p) >= 2)      // 满足windy数的条件
            res += dfs(i - 1, limit1); // 在该条件下继续向下深搜
        ++limit1;
    }
    dp[i][p] = res;
    return res;
}
```

这里$dp[i][j]$表示i位上为j的windy数的个数。通过这种记忆化深度优先搜索，有效降低了问题的计算次数。下面给出该问题的动态规划解。

```cpp
void init()
{
    // 将所有的windy数都求出来存放在数组dp里面
    for (int i = 0; i <= 9; ++i)
        dp[1][i] = 1;
    for (int i = 2; i <= 10; ++i)
    {
        for (int j = 0; j <= 9; ++j)
        {
            for (int k = 0; k <= 9; ++k)
            {
                if (abs(j - k) >= 2)
                    dp[i][j] += dp[i - 1][k];
            }
        }
    }
}
LL work(int x) // 求出<=x的windy数
{
    // 这里先将windy数拆分存放在数组num里面
    int len = 0; // len为x的长度
    int num[11];
    while (x)
    {
        num[++len] = x % 10;
        x /= 10;
    }
    LL ans = 0;
    // 这里可以考虑假如x是一位n位数，其小于n位的所有数字都可以通过dp[i][j]来求出
    for (int i = 1; i < len; ++i)
    {
        for (int j = 1; j <= 9; ++j) // 枚举第二维的时候要注意从1开始枚举，否则存在前导为零的数字，不
        {
            ans += dp[i][j];
        }
    }
    // 最后将位数与x相等的windy数加入ans，这里需要注意使用num数组来枚举
    for (int i = 1; i < num[len]; ++i)
    {
        ans += dp[len][i];
    }
    // 如果要枚举的最后一位与x相同
    for (int i = len - 1; i >= 1; --i)
```

```
    {
        for (int j = 0; j < num[i]; ++j)
        {
            if (abs(j - num[i + 1]) >= 2)
                ans += dp[i][j];
        }
        if (abs(num[i] - num[i + 1]) < 2)
            break;
        if (i == 1)
            ans++;
    }
    return ans;
}
```

这里的$init$函数用来初始化dp数组，该问题的dp状态转移方程为

$$dp[i][j] = \sum dp[i-1][k] \ (abs(j-k) \geq 2 \ and \ 0 \leq k \leq 9)$$