一．问题描述及规格说明（需求分析）

给定两个等长的数字序列(原始串和目标串)，类似于密码锁的结构，我们可以对数字进行转动(也就是加 1 减 1)，可以同时旋转最多连续三个数字，要求最小的旋转次数。

二．约定

为了方便说明，我们把原始串记作 arr，目标串记作 pat，串长度记作 n。
由于旋转的时候，数字不是递增的，而是会循环变换，所以给出 fun()函数的说明，对于给定的数字 arr，返回它与 x 进行加减操作之后变成的另外一个数。

```
inline int fun(const int arr, const int x, const char sign)//sign是符号位，+代表正转，-代表倒转
{
    if (sign == '+')
        return (arr + x) % 10;
    else
        return (arr - x + 10) % 10;
}
```

三．搜索算法思路

要求最小的旋转次数，我第一时间想到了搜索算法。DFS（深度优先搜索算法）和 BFS（广度优先搜索算法）是两个比较常见的搜索算法。一般的处理手段是把序列的变化状态用多叉树的形式表现出来。很明显，此题对数字进行操作一次之后，状态的变化数是有限的，经过计算，可以得到下列的结果：

$$
\begin{aligned}
&\text{对于正转来说}(positive\_sequence)\ \text{也就是只对数字进行}+1\text{取模}\\
&\text{选择一个数的操作：}ans1=n\\
&\text{选择两个数的操作：}ans2=n-1\\
&\text{选择三个数的操作：}ans3=n-2\\
&total1=ans1+ans2+ans3=3n-3\\
&\text{同理：对于逆转来说}(negative\_sequence)\\
&total\_2=total\_1\\
&\therefore total=total\_1+total\_2=6n-6
\end{aligned}
$$

此时把原始串作为多叉树的根节点，那么就会有 6n-6 个子节点(它们代表可以由该根节点经过一次操作之后产生的中间状态串)。可以经过计算得出:

$$\text{对于第}h\text{层来说}(h \geq 1)$$
$$\text{节点数量：}f(h) = f(h-1) * (6n-6)$$
$$\therefore f(h) = (6n-6)^h = O(n^h)$$
$$\text{节点总个数：}\Sigma f(h) = n^{h_{max}+1}$$

若是采用 DFS 算法，我们需要遍历所有的节点才能找到最短的路径，时间复杂度与空间复杂度都很高。
若是采用 BFS 算法，只需要在建树的过程中第一次遇见目标串就可以停止搜索了。可以看到，时间复杂度是幂次级别的，而且 h 是难以确定的，若数字序列增大则 h 也必定增大，在有限的时间内很难计算出结果。

通过前面的分析，我们发现搜素算法扩展的节点实在是很多。有效的解决办法是已经走过的节点就不要继续往下走，这样我们就可以对多叉树进行剪枝，实际却是，由于数字序列的多样性，剪枝不能有效降低时间复杂度，而且检查是否为已走过的节点需要遍历所有数组，时间开销很大。
而原问题具有最优子结构以及无后效性，其实可以通过递推来做，这里采用动态规划的思想。

四．设计(目标：有效的组织和处理数据)

数据结构设计：设置了三维数组 dp[1001][10][10]（**其中 1001 是因为序列最长是 1000 位，后两个 10 代表数字只能从 0-9 进行变化**）用来存储最小的旋转次数，其中 dp[i][x][y]表示前 i 位数字已经旋转好，第 i+1 位为 x,i+2 位为 y 时候的最小次数。由于要求最小值，我们把数组里的元素全部初始化为 max_num。并根据初始状态把 dp[0][arr[0]-‘0’][arr[1]-‘0’]设为 0。

五．算法设计

既然满足题目满足动态规划的特点，那么下一步就是要找到动态转移方程。
即 dp[i+1][][]与 dp[i][x][y]的关系。
下面先考虑正序旋转的情况（即 positive_sequence）,枚举第 i 位的值 x 和第 i+1 位的 y，要把第 i 位复原，则必须对第 i 位旋转 **positive_seq=(pat[i - 1] - '0' - x + 10) % 10** 次,并且在转动的过程，可以进行三种操作：
●转动第 i 位置。
●转动第 i 位和第 i+1 位
●转动第 i 位，第 i+1 位，第 i+2 位。
以上三种情况都可以视作以 i 为主体转动，i+1 和 i+2 从动，其约束条件为 i 位转的次数>=i+1 位置转的次数>=i+2 位置转的次数，可得如下的状态转移方程：

$$dp[i][fun(y,m,'+')][fun(arr[i+1]-'0',n,'+')]$$
$$=min(dp[i][fun(y,m,'+')][fun(arr[i+1]-'0',n,'+')],$$
$$dp[i-1][x][y]+positive\_seq);$$

同理可得，向下的旋转次数：**negative_seq = 10 - positive_seq**；状态转移方程如下：

$$dp[i][fun(y,m,'-')][fun(arr[i+1]-'0',n,'-')]$$
$$=min(dp[i][fun(y,m,'-')][fun(arr[i+1]-'0',n,'-')],$$
$$dp[i-1][x][y]+negative\_seq);$$

注意：由于数字串最多一下子转三个，在输入数字序列长度小于等于 3 时，三维数组的含义就表示不出来，于是把输入进来的两个序列串末尾加上“00”，让数组保持三位数及以上。

六. 算法复杂度分析

空间复杂度;只使用了一个三维数组，空间复杂度为 1001*10*10，为 O(1)
时间复杂度下面有详细的阐述：

时间复杂度由两部分构成

初始化dp数组 and 主程序运行部分

$T_1(n)=(n+1)*10*10=O(n)$

$\because positive_seq$为$0\ \rightarrow 10$, 在正转和逆转的情况下，外面三层循环都一样

即$n*10*10=100n$

不妨令 $positive\_seq=k$

$\therefore$ 内层的两层循环为$g(k)=\Sigma_{m=0}^{k}\Sigma_{n=0}^{m}1+\Sigma_{m=0}^{10-k}\Sigma_{n=0}^{m}1$

$=k^2-10k+67$

平均的时间复杂度：$G(k)=\dfrac{\Sigma_{k=0}^{9}g(k)}{10}=50.5=O(1)$

$\therefore T_2(n)=G(k)*100n=5050n$

$\therefore T(n)=T_1(n)+T_2(n)=5050n+100n+100=5150n+100=O(n)$

七．测试

```
请输入两个数字序列:
a
b
请重新输入两个数字序列:
a3
34
请重新输入两个数字序列:
12
74
请重新输入两个数字序列:
1235
124
请重新输入两个数字序列:
1236
8971
8
```