一、朴素的动态规划算法

（以下假设所查询的单词串为 S）

求解两个字符串的编辑距离有一个经典的动态规划算法，考虑将这个算法用于S和字典中的每一个单词，得到S和每个单词的编辑距离并且将它们排序，取编辑距离最小的若干个单词就得到了答案。这是一个朴素的基于动态规划的算法。

假设两串为A和B

动态规划算法如下：

令DP[i][j]表示A的长度为i的前缀和B的长度为j的前缀的编辑距离

则有DP[0][x] = DP[x][0] = 0

DP[i][j] = Min(DP[i-1][j]+1,DP[i][j-1]+1,DP[i-1][j-1]+1) A[i]!=B[j]时

= Min(DP[i-1][j]+1,DP[i][j-1]+1,DP[i-1][j-1]) A[i]=B[j]时

//时间复杂度：O(词典单词总长度\*查询串的长度)

//空间复杂度：O(最长的单词长度\*查询串的长度)

优点：能够给出基于编辑距离的精确解

缺点：效率较低，对于大规模的数据，时间无法接受

二、改进的动态规划算法

Ⅰ、Trie树（此处省略图片一张）

Trie树，又称为字典树，是一种用于存储大量字符串的树形数据结构。其每一个节点代表了一个单词或一个单词的前缀。它的优点是：利用单词的公共前缀来减少查询时间，最大限度地减少无谓的字符串比较。

Ⅱ、使用Trie树优化动态规划算法

考虑对词典建立Trie树，并对于树上的每个点表示的字符串都求出与S的每一个前缀的编辑距离。节点P的信息可以用P的父节点的信息算出，这本质上与之前的DP算法相同。

我们可以对于每种编辑L距离求出所有有序对（P，i），其中P表示的串与S的长度为i的前缀编辑距离为L。从小到大枚举L，当查询到的单词数量已经达到要求时，立即停止算法，就可以避免计算编辑距离过大的所有情况。

//时间复杂度：O(词典单词总长度\*查询串的长度)

//空间复杂度：O(词典单词总长度\*查询串的长度)

优点：相较于朴素的动态规划算法，能够提高时间效率

缺点：由于需要提前建出Trie树，消耗的内存空间较大，而且时间效率仍然比较低下