例2：快速求和(quicksum)

**问题描述：** 给定一个数字字符串，在字符串中的某些位置插入一些加号。在所有的加号都插入后，就象做普通加法那样来求值，问最少需要插入多少加号可以使得该值等于一个给定的目标数字。

例如，考虑字符串"303"和目标数字 6，最佳方法不是"3+0+3"，而是"3+03"。能这样做是因为一个数的前导零不会改变它的大小。

编程实现这个算法。

**输入格式：**第一行包含一个字符串 S( 1 <= S 的长度 <= 40 ) 和一个整数 N( N <= 1000 )，用单个空格分隔。

**输出格式：**一行一个整数 K，表示需要最少需要插入多少加号。如果无解，则输出 -1。

**输入样例：**

2222 8

**输出样例：**

3

**数据规模：**

对于 80%的数据，1 <= S 的长度 <= 10。

对于 100%的数据，1 <= S 的长度 <= 40,N <= 1000。

分析：对于朴素的回溯搜索，由于字符串长度不超过40，最多有39个位置可以放置加号，这样最多就有239种可能性。显然，不加优化的搜索会出现超时。

如何优化呢？很容易想到的优化有三个：

1. 如果搜索到当前位置i时，已经添加的加号个数p大于最优解ans，则不管以后怎样，都不可能得到最优的解，因此剪枝，返回。
2. 如果搜索到当前位置i时，已经算出的部分和大于N，则以后的数最小，也不可能恰好凑成N，因此剪枝，返回。
3. 如果搜索到当前位置i时，已经算出的部分和再加上i+1～N这一段的最大值（即i+1~N不添加加号），仍然小于N，则再搜索也没有意义，因此剪枝，返回。

另一个要注意的问题是40位的数字串在转化成整数时，会超过int甚至long long的存储范围。而本题的一个特点是要凑的数N本身并不太大（不超过106），所以可以设置一个无穷大INF（值可设为109），当要转化的数字串的值超过INF时，直接把值设为INF就行。这样做不会影响运算结果的正确性，也有效地避免了高精度运算。

有了上述优化，本题完全可以在规定时限内出解。