数组形式的整数加法

对于非负整数 `X` 而言，*`X`* 的*数组形式*是每位数字按从左到右的顺序形成的数组。例如，如果 `X = 1231`，那么其数组形式为 `[1,2,3,1]`。

给定非负整数 `X` 的数组形式 `A`，返回整数 `X+K` 的数组形式。

示例 1：

输入： `A = [1,2,0,0], K = 34`

输出： `[1,2,3,4]`

解释： `1200 + 34 = 1234`

示例 2：

输入： `A = [2,7,4], K = 181`

输出： `[4,5,5]`

解释： `274 + 181 = 455`

示例 3：

输入： `A = [2,1,5], K = 806`

输出： `[1,0,2,1]`

解释： `215 + 806 = 1021`

示例 4：

输入： `A = [9,9,9,9,9,9,9,9,9,9], K = 1`

输出： `[1,0,0,0,0,0,0,0,0,0,0]`

解释： `9999999999 + 1 = 10000000000`

解题思路：

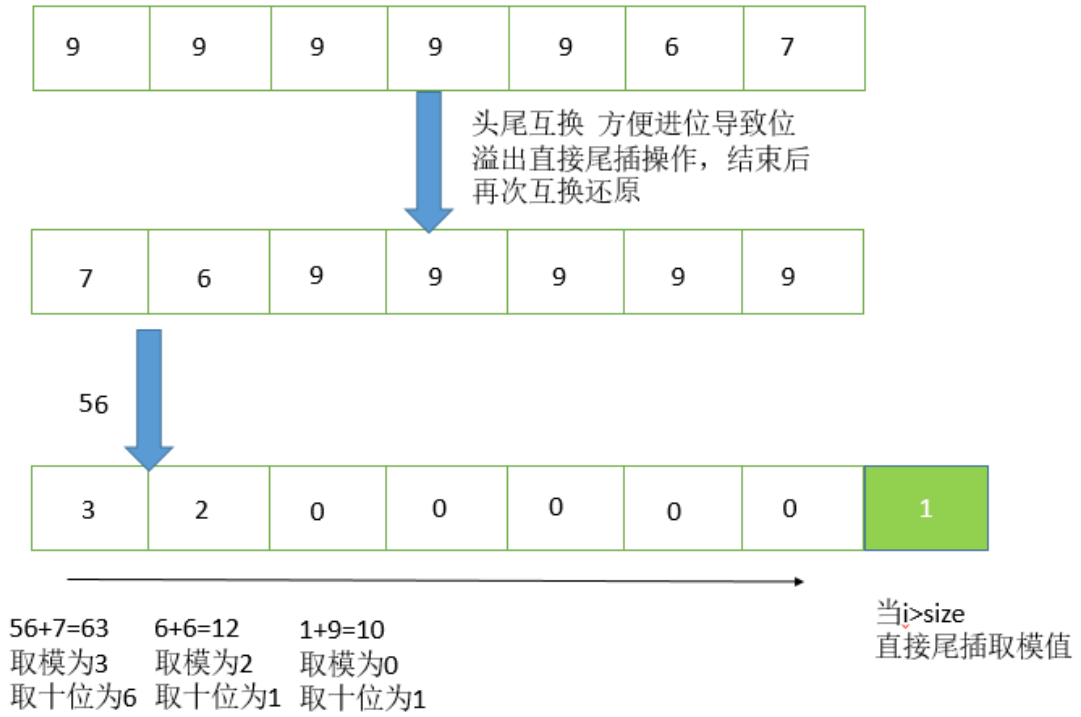
暴力求解；将数组中数据转化为整型，相加操作后在转化为数组

考虑如果数组长度过长，转化整型失败，因为数组即便是 `long long` 类型也有存储上限

对此需要考虑换一种思路

将 `k` 进行逐位分解，与数组末尾对应位置相加，此时需要考虑到进位的情况

比如：




```
/*class Solution {
public:
    vector<int> addToArrayForm(vector<int>& A, int K) {

        int i;
        long long sum=0;
        vector<int> ret;
        for(int i=A.size();i>0;i--)
        {
            sum+=pow(10,i-1)*A[A.size()-i];
        }
        long long res=sum+K;
        if(res==0)
        {
            ret.push_back(0);
        }
        while(res)
        {
            ret.push_back(res%10);
            res=res/10;
        }
        reverse(ret.begin(),ret.end());
        return ret;
    }
```

```
};*/

class Solution {
public:
    vector<int> addToArrayForm(vector<int>& A, int K) {
        reverse(A.begin(),A.end());
        int i=0;
        while(K>0)
        {
            if(i<A.size())
            {
                K=K+A[i];
                A[i]=K%10;
            }
            else
            {
                A.push_back(K%10);
            }
            K=K/10;
            i++;
        }
        reverse(A.begin(),A.end());
        return A;
    }
};
```