560 和位K的子数组

Label：数组、HashMap
给定一个整数数组和一个整数 k，你需要找到该数组中和为 k 的连续的子数组的个数。
    输入:nums = [1,1,1], k = 2
    输出: 2 , [1,1] 与 [1,1] 为两种不同的情况。
数组的长度为 [1, 20,000]。
数组中元素的范围是 [-1000, 1000] ，且整数 k 的范围是 [-1e7, 1e7]。

- 遍历

```
class Solution {
    public int subarraySum(int[] nums, int k) {
        int count = 0;
        for (int i = 0; i < nums.length; i++) {
            int sum = 0;
            for (int j = i; j < nums.length; j++) {
                sum += nums[j];
                if (sum == k) {  //有负数，所以不能剪枝
                    count++;
                    while (j < nums.length-1 && sum + nums[j + 1] == k ) {  // 防止连续值出现
                        count++;
                        j++;
                    }
                }
            }
        }
        return count;
    }
}
```

- 枚举优化

```
public class Solution {
    public int subarraySum(int[] nums, int k) {
        int count = 0;
        for (int start = 0; start < nums.length; ++start) {
            int sum = 0;
            for (int end = start; end >= 0; --end) {
                sum += nums[end];
                if (sum == k) {
                    count++;
                }
            }
        }
        return count;
    }
}
```

- 前缀和

nums[:i]+nums[i:j]=nums[:j]，我们需要的就是中间的nums[i:j]，令nums[i:j]=k，可以获得nums[:j]-k = nums[:i] ，就是当前前缀和减去目标得到的之前的前缀和的数量

```
public class Solution {
    public int subarraySum(int[] nums, int k) {
        int count = 0, pre = 0;
        Map <Integer, Integer> mp = new HashMap <> ();
        mp.put(0, 1);

        for (int i = 0; i < nums.length; i++) {
            pre += nums[i];  // 存储前缀和

            if (mp.containsKey(pre - k)) {
                count += mp.get(pre - k);
            }
            mp.put(pre, mp.getOrDefault(pre, 0) + 1);
        }
        return count;
    }
}
```