```
class Solution {
    public int maxSubArrayLen(int[] nums, int k) {
        Map<Integer, Integer> mp = new HashMap<>();
        int sum = 0, res = 0;
        mp.put(0, -1);
        for (int i = 0; i < nums.length; i++) {
            sum += nums[i];
            if (mp.containsKey(sum - k)) {
                res = Math.max(res, i - mp.get(sum - k));
            }
            if (!mp.containsKey(sum)) {
                mp.put(sum, i);
            }
        }
        return res;
    }
}
```

given an array *nums* and a target value *k*, find the maximum length of a subarray that sums to *k*. If there isn't one, return 0 instead.

Note:
The sum of the entire *nums* array is guaranteed to fit within the 32-bit signed integer range.

Example 1:

```
Input: nums = [1, -1, 5, -2, 3], k = 3
Output: 4
Explanation: The subarray [1, -1, 5, -2] sums to 3 and is the longest.
```