练习2 – 最大连续子序列和

■ 给定一个长度为 n 的整数序列，求它的最大连续子序列和

□ 比如 –2、1、–3、4、–1、2、1、–5、4 的最大连续子序列和是 4 + (–1) + 2 + 1 = 6

■ 状态定义

□ 假设 dp(i) 是以 nums[i] 结尾的最大连续子序列和（nums是整个序列）

✓ 以 nums[0] –2 结尾的最大连续子序列是 –2，所以 dp(0) = –2

✓ 以 nums[1] 1 结尾的最大连续子序列是 1，所以 dp(1) = 1

✓ 以 nums[2] –3 结尾的最大连续子序列是 1、–3，所以 dp(2) = dp(1) + (–3) = –2

✓ 以 nums[3] 4 结尾的最大连续子序列是 4，所以 dp(3) = 4

✓ 以 nums[4] –1 结尾的最大连续子序列是 4、–1，所以 dp(4) = dp(3) + (–1) = 3

✓ 以 nums[5] 2 结尾的最大连续子序列是 4、–1、2，所以 dp(5) = dp(4) + 2 = 5

✓ 以 nums[6] 1 结尾的最大连续子序列是 4、–1、2、1，所以 dp(6) = dp(5) + 1 = 6

✓ 以 nums[7] –5 结尾的最大连续子序列是 4、–1、2、1、–5，所以 dp(7) = dp(6) + (–5) = 1

✓ 以 nums[8] 4 结尾的最大连续子序列是 4、–1、2、1、–5、4，所以 dp(8) = dp(7) + 4 = 5

最大连续子序列和 – 状态转移方程和初始状态

■ 状态转移方程

□ 如果 dp(i – 1) ≤ 0，那么 dp(i) = nums[i]

□ 如果 dp(i – 1) > 0，那么 dp(i) = dp(i – 1) + nums[i]

■ 初始状态

□ dp(0) 的值是 nums[0]

■ 最终的解

□ 最大连续子序列和是所有 dp(i) 中的最大值 max { dp(i) }，i ∈ [0, nums.length)

最大连续子序列和 – 动态规划 – 实现

```
int maxSubArray(int[] nums) {
    if (nums == null || nums.length == 0) return 0;
    int[] dp = new int[nums.length];
    int max = dp[0] = nums[0];
    for (int i = 1; i < dp.length; i++) {
        int prev = dp[i - 1];
        if (prev > 0) {
            dp[i] = prev + nums[i];
        } else {
            dp[i] = nums[i];
        }
        max = Math.max(max, dp[i]);
    }
    return max;
}
```

- 空间复杂度：O(n)，时间复杂度：O(n)

最大连续子序列和 – 动态规划 – 优化实现

```
int maxSubArray(int[] nums) {
    if (nums == null || nums.length == 0) return 0;
    int dp = nums[0];
    int max = dp;
    for (int i = 1; i < nums.length; i++) {
        if (dp > 0) {
            dp = dp + nums[i];
        } else {
            dp = nums[i];
        }
        max = Math.max(max, dp);
    }
    return max;
}
```

- 空间复杂度：O(1)，时间复杂度：O(n)