动态规划问题

输入一个整型数组，数组里有正数也有负数，数组中的一个或连续多个整数组成一个子数组，求所有子数组的和的最大值。要求时间复杂度为0(n)

利用动态规划来解决

max(dp[i - 1] + nums[i], nums[i])就保证是连续的，因为从1开始，dp[i - 1]就是连续的，dp[i]的意思是这个nums[i]必须在内的

```
public class continusArrayMaxSum {

    public int maxSum(int [] nums) {

        if(nums == null || nums.length == 0)
            return 0;

        int [] dp = new int [nums.length];
        dp[0] = nums[0];
        int max = nums[0];

        for(int i = 1; i < nums.length; i ++) {
            dp[i] = Math.max(dp[i - 1] + nums[i], nums[i]);
            if(dp[i] > max)
                max = dp[i];
        }

        return max;
    }

    public static void main(String [] args) {
        int [] nums = new int [] {1, -2, 3, 10, -4, 7, 2, -5};
        System.out.println(new continusArrayMaxSum().maxSum(nums));
    }

}
```