/* Say you have an array for which the ith element is the price of a given stock on day i.

Design an algorithm to find the maximum profit. You may complete as many transactions as you like (ie, buy one and sell one share of the stock multiple times). However, you may not engage in multiple transactions at the same time (ie, you must sell the stock before you buy again).

*/

-
- 思想：
- (1) 方法很巧妙，判断当前值与上一个值与0的关系，若大于0直接加入结果，小于0则舍弃；
- (2) 原理类似于爬山，

情况1： A -- 上坡 -- B -- 下坡 -- C --上坡 -- D

此时 (h(B) - h(A)) + (h(D) - h(C)) 大于 (h(D) - h(A)) (可以画图来看)

情况2： A -- 上坡 -- B -- 上坡 -- C --上坡 -- D

此时 (h(B) - h(A)) + (h(D) - h(C)) 等于 (h(D) - h(A)) (可以画图来看)

- (3) 综上，现算法总能得到最大结果

```
public int maxProfit(int[] prices) {

    int pricesLength = prices.length;

    int result = 0;

    for (int i = 1; i < pricesLength; i++) {

        result += Math.max(prices[i] - prices[i-1], 0);
    }

    return result;
}
```