

198. 打家劫舍

已解答 ✓

中等

🔖 相关标签

🏢 相关企业

Ax

你是一个专业的小偷，计划偷窃沿街的房屋。每间房内都藏有一定的现金，影响你偷窃的唯一制约因素就是相邻的房屋装有相互连通的防盗系统，**如果两间相邻的房屋在同一晚上被小偷闯入，系统会自动报警。**

给定一个代表每个房屋存放金额的非负整数数组，计算你 **不触动警报装置的情况下**，一夜之内能够偷窃到的最高金额。

示例 1:

输入: [1,2,3,1]

输出: 4

解释: 偷窃 1 号房屋 (金额 = 1) , 然后偷窃 3 号房屋 (金额 = 3)。
偷窃到的最高金额 = 1 + 3 = 4 。

示例 2:

输入: [2,7,9,3,1]

输出: 12

解释: 偷窃 1 号房屋 (金额 = 2), 偷窃 3 号房屋 (金额 = 9), 接着偷窃 5 号房屋 (金额 = 1)。
偷窃到的最高金额 = 2 + 9 + 1 = 12 。

提示:

解题思路一:

动态规划:

保留每个解，最后一个解是最优解。设置 `dp` 数组来存储解（即当前能得到的最大值），观察数组可以发现，第一个解一定是 `dp[0]=nums[0]`,第二个解一定是 `dp[1]=max(dp[0],nums[1])`;从第三个解开始，一定是前一个解的值与前一个解的前一项与当前价值之和的较大值，即有方程 `dp[i]=max(dp[i-1],dp[i-2]+nums[i])`,最后输出 `dp[nums.size()-1]`即是所求的最大值

代码:

```
class Solution {
public:
    int rob(vector<int>& nums) {
        if (nums.empty()) {
            return 0;
        }
        if (nums.size() == 1) {
```

```

        return nums[0];
    }
    //主要方法是找到 dp 的方程
    vector<int> dp(nums.size());
    dp[0] = nums[0];
    dp[1] = max(nums[0], nums[1]);
    for (int i = 2; i < nums.size(); i++) {
        dp[i] = max(dp[i - 1], dp[i - 2] + nums[i]);
    }
    return dp[nums.size() - 1];
}
};

```

解题思路二：

滑动窗口：

在解题思路一的基础上优化，不必保留每个解，每次只保存两个解，即当前 i 前两个解，每次比较选取最大级可

```


class Solution {
public:
    int rob(vector<int>& nums) {
        if (nums.empty()) {
            return 0;
        }
        int size = nums.size();
        if (size == 1) {
            return nums[0];
        }
        //解的初始化
        int first = nums[0], second = max(nums[0], nums[1]);
        //只保留前两个最大解
        for (int i = 2; i < size; i++) {
            int temp = second;
            second = max(first + nums[i], second);
            first = temp;
        }
        return second;
    }
};


```

2288. 价格减免

已解答 

中等

 相关标签

 相关企业

 提示

Aa

句子 是由若干个单词组成的字符串，单词之间用单个空格分隔，其中每个单词可以包含数字、小写字母、和美元符号 '\$'。如果单词的形式为美元符号后跟着一个非负实数，那么这个单词就表示一个 **价格**。

- 例如 "\$100"、"\$23" 和 "\$6" 表示价格，而 "100"、"\$" 和 "\$1e5" 不是。

给你一个字符串 `sentence` 表示一个句子和一个整数 `discount`。对于每个表示价格的单词，都在价格的基础上减免 `discount%`，并 **更新** 该单词到句子中。所有更新后的价格应该表示为一个 **恰好保留小数点后两位** 的数字。

返回表示修改后句子的字符串。

注意：所有价格 **最多** 为 10 位数字。

示例 1:

输入: `sentence = "there are $1 $2 and 5$ candies in the shop"`,
`discount = 50`

输出: `"there are $0.50 $1.00 and 5$ candies in the shop"`

解释:

表示价格的单词是 "\$1" 和 "\$2"。

- "\$1" 减免 50% 为 "\$0.50"，所以 "\$1" 替换为 "\$0.50"。
- "\$2" 减免 50% 为 "\$1"，所以 "\$2" 替换为 "\$1.00"。

示例 2:

解题思路一:

按空格分割然后一一替换，先裁成每个单词在一起，对每个单词进行处理，满足要求的根据输入的折扣进行计算，然后更新单词，每个单词更新结束后，重新写入字符串，每个单词之间用空格隔开

代码:

```
#include <iostream>
#include <sstream>
#include <vector>
#include <iomanip>

using namespace std;

class Solution {
public:
```

```

string discountPrices(string sentence, int discount) {
    istringstream iss(sentence);
    vector<string> words;
    string word;

    // 分割句子成单词，并存储到 words 向量中
    while (iss >> word) {
        words.push_back(word);
    }

    // 处理每个单词
    for (string& w : words) {
        // 检查单词是否以$开头且长度大于 1
        if (w[0] == '$' && w.size() > 1) {
            bool isPrice = true;
            // 检查$符号后面的字符是否全是数字或小数点
            for (int i = 1; i < w.size(); ++i) {
                if (!isdigit(w[i]) && w[i] != '.') {
                    isPrice = false;
                    break;
                }
            }
            if (isPrice) {
                // 转换为价格并计算折扣后的价格
                double price = stod(w.substr(1)); // 将字符串转换为 double
                double discountedPrice = price * (1 - discount / 100.0); // 计算折扣后的价格

                // 格式化为两位小数的字符串
                ostringstream oss;
                oss << fixed << setprecision(2) << discountedPrice; // 保留两位小数

                w = "$" + oss.str(); // 重新组合成带$符号的字符串
            }
        }
    }

    // 重新组合句子

```

```

        ostream result;
        for (int i = 0; i < words.size(); ++i) {
            if (i > 0) result << " "; // 在单词之间添加空格
            result << words[i];
        }

        return result.str(); // 返回处理后的句子
    }
};

int main() {
    Solution solution;
    string sentence = "there are $1 $2 and 5$ candies in the shop";
    int discount = 50;
    string result = solution.discountPrices(sentence, discount);
    cout << result << endl; // 输出处理后的句子
    return 0;
}

```

解题思路二：

模拟：

对每一种情况进行模拟，对于不满足要求的字符正常遍历存储，当满足要求时，当前字符到最后一个连续数字全部用 `long` 保存后使用 `double` 升级得到折扣结果，这个结果可以用 `sprintf` 写入字符串，然后使存储原字符串的 `ans` 拼接这段结果即可

代码：

```

class Solution {
public:
    string discountPrices(string sentence, int discount) {
        string ans;

        for (int i = 0; i < sentence.size(); i++) {

            // 非$之后的数字一律先放进数组
            ans.push_back(sentence[i]);

            if (sentence[i] == '$') {

                // 如果$前一个不是空格

```

```

        if (i - 1 >= 0 && sentence[i - 1] != ' ')
            continue;

        // 如果$后一个不是数字
        if (i + 1 >= sentence.size() || sentence[i + 1] > '9' ||
            sentence[i + 1] < '0') {
            continue;
        }

        // 记录当前下标
        int tmp = i;

        // 从下一个开始，找到全部连续数字
        long num = 0;
        i++;
        while (i < sentence.size() && sentence[i] >= '0' &&
            sentence[i] <= '9') {
            num = num * 10 + sentence[i] - '0';
            i++;
        }

        // 如果数字的结尾不是空格，则返回
        if (i < sentence.size() && sentence[i] != ' ') {
            i = tmp;
            continue;
        }

        // 将数字打折转换成字符串返回
        double ret = num * (1.0 - discount / 100.0);
        char str[40];
        sprintf(str, "%.2f", ret);
        ans += string(str);

        // 由于之前循环里多加了一次，减回来
        i--;
    }
}

return ans;

```

