[C++版本] 第一周 周四部分

> 2021-08-19

Introduction

第一周，周四习题。

Content

Table

- [x] 1. 两数之和（简单）
- [x] 1108. IP 地址无效化（简单）
- [x] 344. 反转字符串（简单）
- [x] 剑指 Offer 58 - I. 翻转单词顺序（简单）

- [x] 125. 验证回文串 （简单）
- [x] 9. 回文数（简单）
- [x] 58. 最后一个单词的长度（简单）
- [x] 剑指 Offer 05. 替换空格（简单）
- [x] 剑指 Offer 58 - II. 左旋转字符串（简单）
- [x] 26. 删除排序数组中的重复项（简单）
- [x] 剑指 Offer 67. 把字符串转换成整数（中等）

1. 两数之和

解法一：暴力法

时间：$O(n^2)$

空间：$O(1)$

```
class Solution {
public:
    vector<int> twoSum(vector<int>& nums, int target) {
        for (int i = 0; i < nums.size(); ++i) {
            for (int j = i + 1; j < nums.size(); ++j) {
                if (nums[i] + nums[j] == target) {
                    return {i, j};
                }
            }
        }

        return {};
    }
};
```

解法二：哈希表

时间：$O(n)$

空间：$O(n)$

批注：本周没有讲到哈希表，此题解给有基础的同学看看。后面还会回头重做。

```
class Solution {
  public:
    vector<int> twoSum(vector<int>& nums, int target) {
        unordered_map<int, int> map;
        for (int i = 0; i < nums.size(); ++i) {
```

```
            map.insert(make_pair(nums[i], i));
        }
        for (int i = 0; i < nums.size(); ++i) {
            int diff = target - nums[i];
            if (map.find(diff) != map.end() && i != map[diff]) {
                return {i, map[diff]};
            }
        }
        return {};
    }
};

// 写法2：其实只要一次循环即可
class Solution {
  public:
    vector<int> twoSum(vector<int>& nums, int target) {
        unordered_map<int, int> map;
        for (int i = 0; i < nums.size(); ++i) {
            map.insert(make_pair(nums[i], i));
            int diff = target - nums[i];
            if (map.find(diff) != map.end() && i != map[diff]) {
                return {i, map[diff]};
            }
        }
        return {};
    }
};
```

1108. IP 地址无效化

时间：$O(n)$

空间：$O(1)$

批注：C++和争哥的Java实现在语法上略有区别，故列举了多种写法。

```
// 写法一：使用string
class Solution {
  public:
    string defangIPaddr(string s) {
        string res;
        for (int i = 0; i < s.size(); ++i) {
            if (s[i] != '.') {
                res += s[i];
            } else {
                res += "[.]";
            }
        }
```

```
        return res;
    }
};

// 写法二：使用数组
class Solution {
  public:
    string defangIPaddr(string s) {
        // 栈内存，自动释放
        char str[s.size() + 2 * 3 + 1];// 注意：要多留一个位置给结束符

        int k = 0;
        for (int i = 0; i < s.size(); ++i) {
            if (s[i] != '.') {
                str[k++] = s[i];
            } else {
                str[k++] = '[';
                str[k++] = '.';
                str[k++] = ']';
            }
        }
        str[k] = '\0'; // 必须有结束符
        return str;
    }
};

// 写法三：使用数组
class Solution {
  public:
    string defangIPaddr(string s) {
        // 堆内存，需要最后手动释放
        char *str = new char[s.size() + 2 * 3 + 1];

        int k = 0;
        for (int i = 0; i < s.size(); ++i) {
            if (s[i] != '.') {
                str[k++] = s[i];
            } else {
                str[k++] = '[';
                str[k++] = '.';
                str[k++] = ']';
            }
        }
        str[k] = '\0';
        string res(str);
        delete[] str;// 记得释放指针
        return res;
    }
};
```

```
// 写法四：使用vector<char>
class Solution {
  public:
    string defangIPaddr(string s) {
        vector<char> vec;
        for (int i = 0; i < s.size(); ++i) {
            if (s[i] != '.') {
                vec.push_back(s[i]);
            } else {
                vec.push_back('[');
                vec.push_back('.');
                vec.push_back(']');
            }
        }
        string str(vec.begin(), vec.end());
        return str;
    }
};
```

批注：注意字符数组结尾需要添加结束符号`'\0'`。如果使用堆内存，需要进行手动释放指针。

344. 反转字符串

时间：$O(n)$

空间：$O(1)$

批注：翻转字符串的写法有多重。

```
// 写法一：复刻争哥
class Solution {
public:
    void reverseString(vector<char>& s) {
        int n = s.size();
        for (int i = 0; i < n / 2; ++i) {
            char tmp = s[i];
            s[i] = s[n - i - 1];
            s[n - i - 1] = tmp;
        }
    }
};

// 写法二
class Solution {
public:
    void reverseString(vector<char>& s) {
        int n = s.size();
```

```
        int i = 0;
        int j = n - 1;
        while (i < j) {
            swap(s[i], s[j]);
            ++i;
            --j;
        }
    }
};

// 写法三
class Solution {
public:
    void reverseString(vector<char>& s) {
        for (int i = 0, j = s.size() - 1; i < j; ++i, --j) {
            swap(s[i], s[j]);
        }
    }
};
```

剑指 Offer 58 - I. 翻转单词顺序

方法一：两次翻转

时间：$O(n^2)$

空间：$O(1)$

```
class Solution {
 public:
  int trim(string& str) {
      int i = 0;
      int n = str.size();
      int k = 0;
      while (i < str.size() && str[i] == ' ') {
          i++;
      }
      while (i < n) {
          if (str[i] == ' ') {
              if (i + 1 < str.size() && str[i + 1] != ' ') {
                  str[k++] = ' ';
              }
          } else {
              str[k++] = str[i];
          }
          i++;
      }
```

```
        return k;
    }

    string reverseWords(string s) {
        int n = trim(s);
        if (n == 0) return "";
        reverse(s, 0, n - 1);
        int p = 0;
        while (p < n) {
            int r = p;
            while (r < n && s[r] != ' ') {
                r++;
            }
            reverse(s, p, r - 1);
            p = r + 1;
        }
        return s.substr(0, n);
    }

    void reverse(string & str, int p, int r) {
        int mid = (p + r) / 2;
        for (int i = p; i <= mid; ++i) {
            char tmp = str[i];
            str[i] = str[r - (i - p)];
            str[r - (i - p)] = tmp;
        }
    }
};
```

方法二：争哥解法

时间：$O(n)$

空间：$O(n)$

批注：该解法争哥课上提了一嘴，但是没有给代码，这里把它实现

```
// 1. 按空格切分放入vector中
// 2. vector各个元素翻转
// 3. vector转换为string
class Solution {
  public:
  int trim(string& str) {
        int i = 0;
        int n = str.size();
        int k = 0;
        while (i < str.size() && str[i] == ' ') {
            i++;
        }
        while (i < n) {
```

```
            if (str[i] == ' ') {
                if (i + 1 < str.size() && str[i + 1] != ' ') {
                    str[k++] = ' ';
                }
            } else {
                str[k++] = str[i];
            }
            i++;
        }
        return k;
    }

    string reverseWords(string str) {
        vector<string> vec;
        int n = trim(str);
        int p = 0;
        for (int i = p; i < n; ++i) {
            if (str[i] == ' ') {
                vec.push_back(str.substr(p, i - p));
                p = i + 1;
            } else {
                if (i == n - 1) {
                    vec.push_back(str.substr(p, i + 1 - p));
                }
            }
        }

        string result;
        reverse(vec.begin(), vec.end());
        for (int i = 0; i < vec.size(); ++i) {
            result += vec[i];
            if (i < vec.size() - 1) result += ' ';
        }
        return result;
    }
};
```

####

125. 验证回文串

时间：$O(n)$

空间：$O(1)$

```
class Solution {
 public:
  char toLower(char c) {
```

```
        if (c >= 'a' && c <= 'z') return c;
        if (c >= '0' && c <= '0') return c;
        return (char) c + 32;
    }

    bool isAlpha(char c) {
        if (c >= 'a' && c <= 'z') return true;
        if (c >= 'A' && c <= 'Z') return true;
        if (c >= '0' && c <= '9') return true;
        return false;
    }

    bool isPalindrome(string s) {
        int i = 0;
        int j = s.size() - 1;
        while (i < j) {
            if (!isAlpha(s[i])) {
                ++i;
                continue;
            }
            if (!isAlpha(s[j])) {
                --j;
                continue;
            }
            if (toLower(s[i]) != toLower(s[j])) {
                return false;
            } else {
                i++;
                j--;
            }
        }
        return true;
    }
};
```

9. 回文数

方法一：放入数组

时间：$O(n)$

空间：$O(n)$

```
class Solution {
 public:
  bool isPalindrome(int x) {
        int digits[10];
```

```
        if (x < 0) return false;
        int k = 0;
        while (x != 0) {
            digits[k] = x % 10;
            x /= 10;
            k++;
        }

        for (int i = 0; i < k / 2; ++i) {
            if (digits[i] != digits[k - i - 1]) {
                return false;
            }
        }
        return true;
    }
};
```

方法二：一边拆原始整数，一边还原新整数

时间：$O(n)$

空间：$O(1)$

```
class Solution {
public:
    bool isPalindrome(int x) {
        if (x < 0) return false;
        long backupX = x;
        long y = 0;
        while (x != 0) {
            y = y * 10 + x % 10;
            x /= 10;
        }
        return backupX == y;
    }
};
```

批注：整数在翻转过程中，可能溢出，因此采用long。

批注：java中就算整型溢出了，也能AC，溢出之后变为负数，肯定和原先输入不同。但是C++不行，会直接报错

58. 最后一个单词的长度

时间：$O(n)$

空间：$O(1)$

```
class Solution {
```

```
 public:
  int lengthOfLastWord(string s) {
      int n = s.size();
      int i = n - 1;
      while (i >= 0 && s[i] == ' ') {
          --i;
      }
      if (i < 0) return 0;

      int len = 0;
      while (i >= 0 && s[i] != ' ') {
          ++len;
          --i;
      }
      return len;
  }
};
```

剑指 Offer 05. 替换空格

时间：$O(n)$

空间：$O(1)$

```
// 写法一
class Solution {
public:
    string replaceSpace(string s) {
        string res;
        for (int i = 0; i < s.size(); ++i) {
            if (s[i] == ' ') {
                res += "%20";
            } else {
                res += s[i];
            }
        }

        return res;
    }
};

// 写法二
class Solution {
public:
    string replaceSpace(string s) {
        vector<char> res;
        for (int i = 0; i < s.size(); ++i) {
```

```
            if (s[i] == ' ') {
                res.push_back('%');
                res.push_back('2');
                res.push_back('0');
            } else {
                res.push_back(s[i]);
            }
        }

        string str(res.begin(), res.end());
        return str;
    }
};
```

剑指 Offer 58 - II. 左旋转字符串

方法一：每次向左挪移1位，挪移n轮

时间：$O(n)$

空间：$O(1)$

```
class Solution {
public:
    string reverseLeftWords(string s, int n) {
        for (int i = 0; i < n; ++i) {
            char tmp = s[0];
            for (int j = 1; j < s.size(); ++j) {
                s[j - 1] = s[j];
            }
            s[s.size() - 1] = tmp;
        }
        return s;
    }
};
```

方法二

时间：$O(n)$

空间：$O(1)$

```
// 写法一：
class Solution {
public:
    string reverseLeftWords(string s, int n) {
        char tmp[s.size() + 1];
        for (int i = 0; i < n; ++i) {
```

```
            tmp[i + s.size() - n] = s[i];
        }
        for (int i = n; i < s.size(); ++i) {
            tmp[i - n] = s[i];
        }
        tmp[s.size()] = '\0';
        return tmp;
    }
};

// 写法二
class Solution {
public:
    string reverseLeftWords(string s, int n) {
        vector<char> vec(s.size());
        for (int i = 0; i < n; ++i) {
            vec[i + s.size() - n] = s[i];
        }
        for (int i = n; i < s.size(); ++i) {
            vec[i - n] = s[i];
        }
        string str(vec.begin(), vec.end());
        return str;
    }
};

// 写法三
class Solution {
public:
    string reverseLeftWords(string s, int n) {
        if (s.size() <= n) return s;
        return s.substr(n, s.size() - n) + s.substr(0, n);
    }
};

// 写法四
class Solution {
public:
    string reverseLeftWords(string s, int n) {
        string s1;
        string s2;
        for (int i = 0; i < n; ++i) {
            s2 += s[i];
        }
        for (int i = n; i < s.size(); ++i) {
            s1 += s[i];
        }
        return s1 + s2;
    }
```

```
};
```


26. 删除有序数组中的重复项

方法一

时间：$O(n)$

空间：$O(1)$

```
class Solution {
public:
    int removeDuplicates(vector<int>& nums) {
        if (nums.empty()) return 0;
        int n = nums.size();
        int k = 0;
        for (int i = 1; i < n; ++i) {
            if (nums[i] != nums[k]) {
                k++;
                nums[k] = nums[i];
            }
        }
        return k + 1;
    }
};
```

方法二

时间：$O(n)$

空间：$O(n)$

批注：非原地实现，开辟新数组

```
class Solution {
 public:
    int removeDuplicates(vector<int> &nums) {
        if (nums.empty()) return 0;
        vector<int> arr(nums.size(), nums[0]);

        int k = 0;
        for (int i = 1; i < nums.size(); ++i) {
            if (nums[i] != arr[k]) {
                arr[++k] = nums[i];
            }
        }

        for (int i = 0; i <= k; ++i) {
```

```
            nums[i] = arr[i];
        }

        return k + 1;
    }
};
```

剑指 Offer 67. 把字符串转换成整数

时间：$O(n)$

空间：$O(1)$

```
class Solution {
 public:
  int strToInt(string s) {
      int n = s.size();
      if (n == 0) return 0;

      int i = 0;
      // 读入字符串并丢弃无用的前导空格
      while (i < n && s[i] == ' ') {
          i++;
      }
      if (i == n) return 0;

      // 判断第一个非空字符
      int sign = 1;
      char c = s[i];
      if (c == '-') {
          sign = -1;
          i++;
      } else if (c == '+') {
          sign = 1;
          i++;
      }

      int intAbsHigh = 214748364;
      int result = 0;
      while (i < n && s[i] >= '0' && s[i] <= '9') {
          int d = s[i] - '0';

          // 处理特殊情况
          if (result > intAbsHigh) {
              if (sign == 1) return INT_MAX;
              else return INT_MIN;
          }
```

```
            if (result == intAbsHigh) {
                if (sign == 1 and d >= 7) return INT_MAX;
                if (sign == -1 and d >= 8) return INT_MIN;
            }

            result = result * 10 + d;
            i++;
        }
        return sign * result;
    }
};
```