

计算机视觉总结

SUMMARY OF COMPUTER VISION

(第1版)

LVSHUAILIN

OPEN SOURCE

BEIJING

VERSION 1

- 一. 数据结构与算法-LeetCode Hot 100
- 二. PYTHON: 1) NUMPY; 2) PANDAS; 3) PYTHON多进程; 4) PYTHON分布式; 5) PYTHON界面;
- 三. 深度学习: TensorFlow 2.0; PYTORCH;
- 四. 图像配准
- 五. 强化学习
- 六. OTHERS: 1) Model INFERENCE by EXE; 2) GIT; 3) DOCKER

LVSHUAILIN

2020年2月

目 录

第1章 绪论	1
1.1 两数之和unordered_map	1
1.1.1 解题思路	1
1.1.2 解题代码	3
1.2 两数相加linked list	4
1.2.1 解题思路	4
1.2.2 解题代码	9
1.3 无重复字符的最长子串(double pointer algorithm, set)	9
1.3.1 解题思路	10
1.3.2 解题代码	10

第1章 绪论

Goals to Achieve

1. unordered_map.

§ 1.1 两数之和unordered_map

HOT100 1.1 问题描述

给定一个整数数组**nums** 和一个目标值**target**, 请你在该数组中找出和为目标值的那两个整数, 并返回他们的数组下标. 你可以假设每种输入只会对应一个答案. 但是, 你不能重复利用这个数组中同样的元素.

示例: 给定nums = [2, 7, 11, 15], target = 9; 因为nums[0] + nums[1] = 2 + 7 = 9;

所以返回[0, 1]

<https://leetcode-cn.com/problems/two-sum>

1.1.1 解题思路

这里用c++的unordered_map来解决, unordered_map内部是一个关联容器, 采用hash 表结构, 有快速检索的功能.

哈希表是通过key关键字直接访问对应value值的数据结构. 特点是键和值一一对应, 查找时间复杂度 $O(1)$.

Example_1: unordered_map插入, 迭代遍历.

unordered_map example_1 code

```
1 #include <iostream>
2 #include <unordered_map>
3 #include <string>
4 using namespace std;
5 int main()
```

```

6 {
7     unordered_map<string, double> umap;
8     umap["PI"] = 3.14;
9     umap.insert(make_pair("a", 2.1));
10
11     // find in umap
12     string key = "PI";
13     if (umap.find(key) == umap.end())
14         cout << "cannot find PI" << endl;
15     else
16         cout << "find " << umap.find(key)->first << " = " << umap.find(key)->second << endl;
17
18     // iterator of umap
19     cout << "entire unorded_map is:" << endl;
20     unordered_map<string, double>::iterator itr;
21     for (itr = umap.begin(); itr != umap.end(); ++itr)
22         cout << " ( " << itr->first << ", " << itr->second << " ) " << endl;
23     system("pause");
24     return 0;
25 }

```

output:

find PI = 3

all elements are:

(PI,3.14)

(a,2.1)

Example_2: 利用unordered_map输出一段文字中重复单词的个数

unordered_map example_2 code

```

1 #include <iostream>
2 #include <unordered_map>
3 #include <string>
4 #include <sstream>
5
6 using namespace std;
7
8 void printWordFreq(const string& str)
9 {
10     unordered_map<string, int> wordFreq;
11     string word;

```

```
12     stringstream ss(str);
13     while (ss >> word)
14         wordFreq[word]++;
15
16     cout << "all elements are:" << endl;
17     for (auto u : wordFreq)
18         cout << " (" << u.first << ", " << u.second << ") " << endl;
19 }
20
21 int main()
22 {
23     string str = "studies very very hard";
24     printWordFreq(str);
25     return 0;
26 }
```

output:

```
all elements are:
(studies, 1)
(very, 2)
(hard, 1)
```

1.1.2 解题代码

```
1  #include <iostream>
2  #include <unordered_map>
3  #include <vector>
4  using namespace std;
5
6  vector<int> twoSum(vector<int>& nums, int target)
7  {
8      unordered_map<int, int> map;
9      vector<int> result={};
10     int n = (int)nums.size();
11     for(int i = 0; i < n; ++i) {
12         auto p = map.find(target-nums[i]);
13         if(p != map.end()) {
14             result.push_back(p->second);
15             result.push_back(i);
16         }
17     }
```

```
16     }
17     map[nums[i]] = i;
18 }
19 return result;
20 }
21
22 int main()
23 {
24     vector<int> nums = {2,7,11,15};
25     vector<int> result;
26     result = twoSum(nums,9);
27     cout<<" ["<<result[0]<<" , "<<result[1]<<" ] "<<endl;
28     return 0;
29 }
```

§ 1.2 两数相加linked list

HOT100 1.2 问题描述

给出两个非空的链表用来表示两个非负的整数。其中，它们各自的位数是按照逆序的方式存储的，并且它们的每个节点只能存储一位数字。如果，我们将这两个数相加起来，则会返回一个新的链表来表示它们的和。

您可以假设除了数字0之外，这两个数都不会以0开头。

示例: 输入(2 → 4 → 3) + (5 → 6 → 4), 输出: 7 → 0 → 8, 原因: 342 + 465 = 807

<https://leetcode-cn.com/problems/add-two-numbers>

1.2.1 解题思路

这里用c++ 链表来解决

Example_1: 创建链表并初始化

linked list example_1 code

```
1 #include <iostream>
2
3 using namespace std;
4
5 class Node{
```



```
6 public:
7     int data;
8     Node* next;
9 };
10
11 int main()
12 {
13     Node* head = nullptr;
14     Node* second = nullptr;
15     Node* third = nullptr;
16
17     head = new Node();
18     head->data = 1;
19
20     second = new Node();
21     second->data = 2;
22
23     third = new Node();
24     third->data = 3;
25
26     cout << head->data << " " << second->data << " " << third->data << endl;
27
28     delete head;
29     delete second;
30     delete third;
31     return 0;
32 }
```

output:

1 2 3

Example_2: 打印链表中的所有元素

linked list example_2 code

```
1 #include <iostream>
2
3 using namespace std;
4
5 class Node{
6 public:
7     int data;
```

```
8     Node* next;
9 };
10
11 void PrintLinkedList(Node* head)
12 {
13     Node* temp = head;
14     while (temp != nullptr) {
15         cout << temp->data << " ";
16         temp = temp->next;
17     }
18     cout << endl;
19 }
20
21 int main()
22 {
23     Node* head = nullptr;
24     Node* second = nullptr;
25     Node* third = nullptr;
26
27     head = new Node();
28     second = new Node();
29     third = new Node();
30
31     head->data = 1;
32     head->next = second;
33
34     second->data = 2;
35     second->next = third;
36
37     third->data = 3;
38     third->next = nullptr;
39
40
41     PrintLinkedList(head);
42
43     delete head;
44     delete second;
45     delete third;
46     return 0;
47 }
```

output:

1 2 3

Example_3: 链表插入节点

linked list example_3 code

```
1 #include <iostream>
2
3 using namespace std;
4
5 class Node{
6 public:
7     int data;
8     Node* next;
9 };
10
11 // 在链表前面插入节点
12 void push(Node** head_ref, int newData)
13 {
14     Node* newNode = new Node();
15     newNode->data = newData;
16     newNode->next = (*head_ref);
17     (*head_ref) = newNode;
18 }
19
20 // 在节点后面插入节点
21 void insertAfter(Node** prev_node, int newData)
22 {
23     if ((*prev_node) == nullptr) {
24         cout << "the previous node cannot be nullptr" << endl;
25         return;
26     }
27
28     Node* newNode = new Node();
29     newNode->data = newData;
30     newNode->next = (*prev_node)->next;
31     (*prev_node)->next = newNode;
32 }
33
34 // 在尾节点后插入节点
```

```
35 void append(Node** head_ref, int newData)
36 {
37     Node* newNode = new Node();
38     newNode->data = newData;
39     newNode->next = nullptr;
40     if ((*head_ref) == nullptr) {
41         (*head_ref) = newNode;
42         return;
43     }
44
45     Node* move = (*head_ref);
46     while (move->next != nullptr) {
47         move = move->next;
48     }
49     move->next = newNode;
50 }
51
52 // 打印链表
53 void PrintLinkedList(Node* head)
54 {
55     Node* temp = head;
56     while (temp != nullptr) {
57         cout << temp->data << " ";
58         temp = temp->next;
59     }
60     cout << endl;
61 }
62
63 void destroyLinkedList(Node** head_ref) {
64     Node* move = (*head_ref);
65     Node* next = nullptr;
66     while (move != nullptr) {
67         next = move->next;
68         delete move;
69         move = next;
70     }
71     (*head_ref) = nullptr;
72 }
73
74 int main()
```

```
75 {  
76     Node* head = nullptr;  
77  
78     append(&head, 6);  
79  
80     push(&head, 7);  
81  
82     push(&head, 1);  
83  
84     append(&head, 4);  
85  
86     insertAfter(&(head->next), 8);  
87  
88     cout << "linked list is: ";  
89     PrintLinkedList(head);  
90     destroyLinkedList(&head);  
91     return 0;  
92 }
```

output:

linked list is: 1 7 8 6 4

1.2.2 解题代码

1 null

§ 1.3 无重复字符的最长子串(double pointer algorithm, set)

HOT100 1.3 问题描述

给定一个字符串, 请你找出其中不含有重复字符的最长子串的长度.

示例1:

输入: "abcabcbb"

输出: 3

解释: 因为无重复字符的最长子串是"abc", 所以其长度为3.

示例2:

输入: “bbbbb”

输出: 1

解释: 因为无重复字符的最长子串是“b”, 所以其长度为1.

示例3:

输入: “pwwkew”

输出: 3

解释: 因为无重复字符的最长子串是“wke”, 所以其长度为3. 请注意, 你的答案必须是子串的长度, “pwke”是一个子序列, 不是子串.

<https://leetcode-cn.com/problems/longest-substring-without-repeating-characters>

1.3.1 解题思路

<https://cloud.tencent.com/developer/article/1377650>

这道题主要用到思路是: 滑动窗口

什么是滑动窗口?

其实就是一个队列, 比如例题中的abcabcbb, 进入这个队列(窗口)为abc满足题目要求, 当再进入a, 队列变成了abca, 这时候不满足要求. 所以, 我们要移动这个队列!

如何移动?

我们只要把队列的左边的元素移出就行了, 直到满足题目要求!

一直维持这样的队列, 找出队列出现最长的长度时候, 求出解!

时间复杂度: $O(n)$

1.3.2 解题代码

```
1 #include <iostream>
2 #include <string>
3 #include <unordered_set>
4 #include <algorithm> // max, min
5
6 using namespace std;
7
8 int lengthOfLongestSubstring(string s) {
9     if (s.size() == 0) return 0;
10    unordered_set<char> lookup;
```

```
11  int maxStr = 0;
12  int left = 0;
13  for (int i = 0; i < s.size(); i++) {
14      while (lookup.find(s[i]) != lookup.end()) {
15          lookup.erase(s[left]);
16          left++;
17      }
18      maxStr = max(maxStr, i - left + 1);
19      lookup.insert(s[i]);
20  }
21  return maxStr;
22 }
23
24 int main()
25 {
26     string str = "abcabcb";
27     cout << lengthOfLongestSubstring(str) << endl;
28     return 0;
29 }
```

output:

3