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

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()
```

```
{
    unordered_map<string, double> umap;
    umap["PI"] = 3.14;
    umap.insert(make_pair("a", 2.1));

    // find in umap
    string key = "PI";
    if (umap.find(key) == umap.end())
        cout << "cannot find PI" << endl;
    else
        cout << "find " << umap.find(key)->first << " = " << umap.find(key)->second << endl;

    // iterator of umap
    cout << "entire unorded_map is:"<<endl;
    unordered_map<string, double>::iterator itr;
    for (itr = umap.begin(); itr != umap.end(); ++itr)
        cout << "(" << itr->first << "," << itr->second << ")" << endl;
    system("pause");
    return 0;
}
```

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

```
#include <iostream>
#include <unordered_map>
#include <string>
#include <sstream>

using namespace std;

void printWordFreq(const string& str)
{
    unordered_map<string, int> wordFreq;
    string word;
```

```
    stringstream ss(str);
    while (ss >> word)
        wordFreq[word]++;

    cout << "all elements are:" << endl;
    for (auto u : wordFreq)
        cout << "(" << u.first << "," << u.second << ")" << endl;
}

int main()
{
    string str = "studies very very hard";
    printWordFreq(str);
    return 0;
}
```

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

1.1.2 解题代码

```
#include <iostream>
#include <unordered_map>
#include <vector>
using namespace std;

vector<int> twoSum(vector<int>& nums, int target)
{
    unordered_map<int, int> map;
    vector<int> result={};
    int n = (int)nums.size();
    for(int i = 0; i < n; ++i) {
        auto p = map.find(target-nums[i]);
        if(p != map.end()) {
        result.push_back(p->second);
        result.push_back(i);
```

```
            }
        map[nums[i]] = i;
        }
        return result;
}

int main()
{
        vector< int > nums = {2,7,11,15};
        vector<int> result;
        result = twoSum(nums,9);
        cout<<"["<<result[0] << "," <<result[1]<<"]"<<endl;
        return 0;
}
```

§ 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

```
#include <iostream>

using namespace std;

class Node{
```

```
public:
    int data;
    Node* next;
};

int main()
{
    Node* head = nullptr;
    Node* second = nullptr;
    Node* third = nullptr;

    head = new Node();
    head->data = 1;

    second = new Node();
    second->data = 2;

    third = new Node();
    third->data = 3;

    cout << head->data << " " << second->data << " " << third->data << endl;

    delete head;
    delete second;
    delete third;
    return 0;
}
```

output:
1 2 3

Example_2: 打印链表中的所有元素

linked list example_2 code

```
#include <iostream>

using namespace std;

class Node{
public:
    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

```
#include <iostream>

using namespace std;

class Node{
public:
    int data;
    Node* next;
};

// 在链表前面插入节点
void InsertBeforeHead(Node** head_ref, int newData)
{
}

// 在节点后面插入节点
void InsertAfterANode(Node** prev_node, int newData)
{

}

// 在尾节点后插入节点
void InsertAtEnd(Node** head_ref, int newData)
{

}

// 打印链表
void PrintLinkedList(Node* head)
{
    Node* temp = head;
    while (temp != nullptr) {
        cout << temp->data << " ";
        temp = temp->next;
```

```
35      }
36      cout << endl;
37  }
38  int main()
39  {
40      Node* head = nullptr;
41      InsertBeforeHead(&head, 7);
42
43      return 0;
44  }
```

output:
None

1.2.2 解题代码