5: Median of Two Sorted Arrays

10: 4Sum

25: Add Two Numbers

34: Copy List with Random Pointer

38: LRU Cache

47: Valid Number

53: Length of Last Word

57: Evaluate Reverse Polish Notation

105: N-Queens

126: Interleaving String

**2.1.5 Median of Two Sorted Arrays**

描述

There are two sorted arrays A and B of size m and n respectively. Find the median of the two sorted

arrays. The overall run time complexity should be *O*(log(*m* + *n*)).

**2.1.10 4Sum**

描述

Given an array *S* of *n* integers, are there elements *a; b; c*, and *d* in *S* such that *a*+*b*+*c*+*d* = *target*?

Find all unique quadruplets in the array which gives the sum of target.

Note:

• Elements in a quadruplet (*a; b; c; d*) must be in non-descending order. (ie, *a \_ b \_ c \_ d*)

• The solution set must not contain duplicate quadruplets.

For example, given array S = {1 0 -1 0 -2 2}, and target = 0.

A solution set is:

(-1, 0, 0, 1)

(-2, -1, 1, 2)

(-2, 0, 0, 2)

**2.2.1 Add Two Numbers**

描述

You are given two linked lists representing two non-negative numbers. The digits are stored in reverse

order and each of their nodes contain a single digit. Add the two numbers and return it as a linked list.

Input: (2 -> 4 -> 3) + (5 -> 6 -> 4)

Output: 7 -> 0 -> 8

**2.2.10 Copy List with Random Pointer**

描述

A linked list is given such that each node contains an additional random pointer which could point to

any node in the list or null.

Return a deep copy of the list.

struct Node {

int data;

Node\* next;

Node\* random;

Node(int d) : data(d), next(nullptr) {}

Node(Node\* n) : data(n->data), next(nullptr) {}

};

Node\* clone(Node\* head)

{

// special cases

if (!head) return nullptr;

// keep a hashmap from node to node

unordered\_map<Node\* ,Node\* > nodeMap;

// clone the main list without consideering random pointers

Node\* head2 = new Node(head);

Node\* p1 = head; Node\* p2 = head2;

nodeMap[p1] = p2;

while (p1->next) {

// create a new node

p2->next = new Node(p1->next);

p2 = p2->next;

p1 = p1->next;

nodeMap[p1] = p2;

}

// scan again

Node\* p1 = head; Node\* p2 = head2;

while (p1) {

p1 = p1->next;

nodeMap[p1]->random = nodeMap[p1->random];

}

}

**2.2.14 LRU Cache**

描述

Design and implement a data structure for Least Recently Used (LRU) cache. It should support the

following operations: get and set.

get(key) - Get the value (will always be positive) of the key if the key exists in the cache, otherwise

return -1.

set(key, value) - Set or insert the value if the key is not already present. When the cache reached its

capacity, it should invalidate the least recently used item before inserting a new item.

#include <set>

struct KeyValue {

int key;

int value;

KeyValue(int k, int v):key(k),value(v) {}

};

class lru {

public:

lru(int cap):capacity(cap) {}

public:

int get(int key) {

if (cacheMap.find(key) == cacheMap.end()) return -1;

// move the item to the head of the list with std::list::splice

cacheList.splice(cacheList.begin(), cacheList, cacheMap[key]);

cacheMap[key] = cacheList.begin();

return cacheMap[key]->value;

}

void set(int key, int value) {

if (cacheMap.find(key) == cahceMap.end()) {

if (cahceList.size() == capacity) {

cacheMap.erase(cacheList.back().key);

caheList.pop\_back();

}

cacheList.push\_front(CacheNode(key, value));

cacheMap[key] = cacheList.begin();

} else {

cacheList.splice(cacheList.begin(), cacheList, cacheMap[key]);

cacheMap[key] = cacheList.begin();

}

// increment usage

}

private:

int capacity;

list<CacheNode> cacheList;

unordered\_map<int, list<CacheNode\*>::iterator> cacheMap;

};

**3.9 Valid Number**

描述

Validate if a given string is numeric.

Some examples:

"0" => true

" 0.1 " => true

"abc" => false

"1 a" => false

"2e10" => true

Note: It is intended for the problem statement to be ambiguous. You should gather all requirements up

front before implementing one.

**3.15 Length of Last Word**

描述

Given a string s consists of upper/lower-case alphabets and empty space characters ' ', return the length

of last word in the string.

If the last word does not exist, return 0.

Note: A word is defined as a character sequence consists of non-space characters only.

For example, Given s = "Hello World", return 5.

**4.1.4 Evaluate Reverse Polish Notation**

描述

Evaluate the value of an arithmetic expression in Reverse Polish Notation.

Valid operators are +, -, \*, /. Each operand may be an integer or another expression.

Some examples:

["2", "1", "+", "3", "\*"] -> ((2 + 1) \* 3) -> 9

["4", "13", "5", "/", "+"] -> (4 + (13 / 5)) -> 6

**10.4 N-Queens**

描述

The n-queens puzzle is the problem of placing n queens on an *n\_n* chessboard such that no two queens

attack each other.

170 第10 章深度优先搜索

图10-2 Eight Queens

Given an integer *n*, return all distinct solutions to the n-queens puzzle.

Each solution contains a distinct board configuration of the n-queens’ placement, where 'Q' and '.'

both indicate a queen and an empty space respectively.

For example, There exist two distinct solutions to the 4-queens puzzle:

[

[".Q..", // Solution 1

"...Q",

"Q...",

"..Q."],

["..Q.", // Solution 2

"Q...",

"...Q",

".Q.."]

]

**13.6 Interleaving String**

描述

Given *s*1*; s*2*; s*3, find whether *s*3 is formed by the interleaving of *s*1 and *s*2.

For example, Given: s1 = ”aabcc”, s2 = ”dbbca”,

When s3 = ”aadbbcbcac”, return true.

When s3 = ”aadbbbaccc”, return false.