春节7天练讲Day7：贪心、分治、回溯和动态规划

你好，我是王争。今天是节后的第一个工作日，也是我们“春节七天练”的最后一篇。

几种算法思想必知必会的代码实现

回溯

- 利用回溯算法求解八皇后问题
- 利用回溯算法求解0-1背包问题

分治

- 利用分治算法求一组数据的逆序对个数

动态规划

- 0-1背包问题
- 最小路径和（详细可看@Smallfly整理的 Minimum Path Sum）
- 编程实现莱文斯坦最短编辑距离
- 编程实现查找两个字符串的最长公共子序列
- 编程实现一个数据序列的最长递增子序列

对应的LeetCode练习题（@Smallfly 整理）

- Regular Expression Matching（正则表达式匹配）

英文版：https://leetcode.com/problems/regular-expression-matching/

中文版：https://leetcode-cn.com/problems/regular-expression-matching/

- Minimum Path Sum（最小路径和）

英文版：https://leetcode.com/problems/minimum-path-sum/

中文版：https://leetcode-cn.com/problems/minimum-path-sum/

- Coin Change （零钱兑换）

英文版：https://leetcode.com/problems/coin-change/

中文版：https://leetcode-cn.com/problems/coin-change/

- Best Time to Buy and Sell Stock（买卖股票的最佳时机）

英文版：https://leetcode.com/problems/best-time-to-buy-and-sell-stock/

中文版：https://leetcode-cn.com/problems/best-time-to-buy-and-sell-stock/

- Maximum Product Subarray（乘积最大子序列）

英文版：https://leetcode.com/problems/maximum-product-subarray/

中文版：https://leetcode-cn.com/problems/maximum-product-subarray/

- Triangle（三角形最小路径和）

英文版：https://leetcode.com/problems/triangle/

中文版：https://leetcode-cn.com/problems/triangle/

到此为止，七天的练习就结束了。这些题目都是我精选出来的，是基础数据结构和算法中最核心的内容。建议你一定要全部手写练习。如果一遍搞不定，你可以结合前面的章节，多看几遍，反复练习，直到能够全部搞定为止。

学习数据结构和算法，最好的方法就是练习和实践。我相信这在任何知识的学习过程中都适用。

最后，祝你工作顺利！学业进步！

精选留言

kai

听了老师的课程，第一遍的时候，只是在读，现在开始回顾：
课程相关的知识点，做了笔记：https://github.com/guokaide/algorithm/blob/master/summary/algorithm.md
课程涉及的题目，也在逐步总结当中：
https://github.com/guokaide/algorithm/blob/master/questions/questions.md

希望和大家一起进步，欢迎小伙伴们一起来讨论~

2019-02-11 11:15

kai

8皇后问题

```
public class EightQueen {

private static final int QUEEN_NUMBER = 8; // 皇后个数
private int[] columns = new int[QUEEN_NUMBER]; // 每个皇后存储的列 (row, col), row天然不相等
private int total = 0;

public int solution() {
queen(0);
return total;
}

private void queen(int row) {
if (row == QUEEN_NUMBER) {
total++;
} else {
```

```
for (int col = 0; col != QUEEN_NUMBER; col++) {
columns[row] = col;
if (isPut(row)) {
queen(row+1);
}
}
}
}

private boolean isPut(int row) {
for (int i = 0; i != row; i++) {
if (columns[row] == columns[i] || row - i == Math.abs(columns[row]-columns[i])) {
return false;
}
}
return true;
}

}
```

2019-02-11 10:55

kai

动态规划，感觉是面试必考内容，今天跟着这些题目再来复习一遍~

2019-02-11 00:38

李皮皮皮皮皮

每天一道算法题，风雨无阻（过年偷懒不算 ）

2019-02-11 11:15

Richard

老师留的题都很不错，正在刷之前没做过的LeetCode题。

参与下答对三题送课程的活动:

Day 1：

1.求众数(Python)

```
class Solution:
def majorityElement(self, nums):
return sorted(nums)[len(nums) // 2]
```

2.缺失的第一个正数(Golang)

```
func firstMissingPositive(nums []int) int {
if len(nums) == 0 {
return 1
}

var arr = make([]bool, len(nums)+1)
var idx = 1
for i := 0; i < len(nums); i++ {
if nums[i] >= 0 && nums[i] < len(arr) {
arr[nums[i]] = true
}
}

for i := 1; i < len(arr); i++ {
if arr[i] == false {
idx = i
```

```
break
} else {
idx = i + 1
}
}

return idx
}
```

Day 7:

3. 买卖股票的最佳时机(Python)

```
class Solution:
def maxProfit(self, prices):
if not prices:
return 0
min_price = prices[0]
res = 0
for i in prices[1:]:
min_price = min(min_price, i)
if res < i - min_price:
res = i - min_price
return res
```

2019-02-11 10:21

_CountingStars

买卖股票的最佳时机 go 语言实现

```
package main

import "fmt"

func maxProfit(prices []int) int {
max := -1
for i := 0; i < len(prices); i++ {
for j := i + 1; j < len(prices); j++ {
profit := prices[j] - prices[i]
if profit > 0 && profit > max {
max = profit
}
}
}

if max == -1 {
return 0
}

return max
}

func main() {
testData1 := []int{7, 1, 5, 3, 6, 4}
testData2 := []int{7, 6, 4, 3, 1}
```

```
fmt.Println(maxProfit(testData1))
fmt.Println(maxProfit(testData2))
}
```

2019-02-11 08:30

纯洁的憎恶

这冲刺压力有点大了

2019-02-10 21:38

吴...

```
//零钱兑换
#include<iostream>
#include<algorithm>
using namespace std;
int coins[10];
int amount;
int k;//k代表纸币的数目
int dp[20];//代表面值最大，也可以采用动态扩容的方式
int cmax = 32767;
int coinChange(int coins[],int amount){
for(int i = 1;i <= amount;i++){
dp[i] = cmax;
for(int j = 0;j < k;j++){
int t = coins[j];
if(i >= t && coins[i - t] != cmax){
dp[i] = min(dp[i - t] + 1,dp[i]);
}
}
}
if(dp[amount] < cmax && dp[amount] > 0){
return dp[amount];
}
else
return -1;
}
int main(){
k = 0;
while(true){
cin>>k;
for(int i = 0;i < k;i++){
cin>>coins[i];
}
cin>>amount;
cout<<coinChange(coins,amount)<<endl;
}
}
```

2019-02-14 20:18

吴...

```
回溯0-1背包问题
#include<iostream>
using namespace std;
int v[10] = {2,2,4,6,3};
int M;//代表背包的容积
```

```
int n;
int cmax = 0;

void f(int w,int k){
// if(w == 0){
// if(w > max) max = w;
// }
if(w == M || k == n){
if(w > cmax) cmax = w;
return ;
}
f(w,k + 1);
if(w + v[k] <= M){
f(w + v[k],k + 1);
}
}
int main(){
//v[] = {2,2,4,6,3};
M = 9;
n = 5;
f(0,0);
cout<<cmax<<endl;
}
```

2019-02-14 10:48

못...

```
#include<iostream>
#include<cmath>
using namespace std;

int queen[100];
int sum = 0;
int n;
void Print(){
//cout<<"ss"<<endl;
for(int i = 0;i < n;i++){
cout<<"("<<i+1<<","<<queen[i] + 1<<")";
}
sum++;
cout<<endl;
}

void Queen(int queen[],int k){
if(k == n) {
Print();
//return ;
}
int j = 0;
for(int i = 0;i < n;i++){
//j = i;
for( j = 0;j < k;j++){
if((queen[j] == i)||(abs(queen[j] - i) == abs(k - j)))
```

```
break;
}
if(j == k){
queen[k] = i;
Queen(queen,k+1);
}
}
}
int main(){
cin>>n;
Queen(queen,0);
cout<<sum<<endl;
}
```

2019-02-14 10:02

纯洁的憎恶

第一题，把39讲的代码改了一下。。。

```
public class Pattern {
private boolean matched = false;
private char[] pattern; // 正则表达式
private int plen; // 正则表达式长度

public Pattern(char[] pattern, int plen) {
this.pattern = pattern;
this.plen = plen;
}

public boolean match(char[] text, int tlen) { // 文本串及长度
matched = false;
if (pattern[0]='*')
return matched;
int i = 0;
int j = 0;
while (i<=plen&&j<=tlen&&pattern[i]!=text[j]&&pattern[i]!='.')
i++;
rmatch(0, 0, text, tlen);
return matched;
}

private void rmatch(int ti, int pj, char[] text, int tlen) {
if (matched) return; // 如果已经匹配了，就不要继续递归了
if (ti == tlen){ //文本串到结尾了
matched = true;
return;
}
if (pattern[pj] == '*') { // * 匹配任意个字符
for (int k = 0; k <= tlen-ti&&tex[ti+k]==text[ti]; ++k) {
rmatch(ti+k, pj+1, text, tlen);
}
} else if (pattern[pj] == '.') { // . 匹配 0 个或者 1 个字符
rmatch(ti, pj+1, text, tlen);
```

```
rmatch(ti+1, pj+1, text, tlen);
} else if (ti < tlen && pattern[pj] == text[ti]) { // 纯字符匹配才行
rmatch(ti+1, pj+1, text, tlen);
}
}
}
```

2019-02-12 11:13

卡罗

感觉，明天就是专栏的最后一天

2019-02-12 09:42

黄丹

课程的最后一天，也是新年上班的第一天，感谢王老师的教育和陪伴，祝您生活开心，工作顺利。

今天的题目比前几天的都难一点，只做了三题，太累了TaT。对于动态规划和贪心总觉得很巧妙，如果想不到动态转移方程式，就很难做，但要是想到了，真的是豁然开朗。对于这一类题，还是要多锻炼，找动态转移方程式要从最后一个结果出发，去想这个结果可以由什么得到，知道之前所有结点的信息，如何推导出当前结点的信息，其实和高中学的归纳法有一点点像。

下面给出我今天做的三题的解题思路和代码

1. Problem 121. Best Time to Buy and Sell Stock

解题思路：这道题很久以前做的，我们可以维持两个变量 - 分别对应于最小谷值和最大利润（销售价格和最低价格之间的最大差异）的minprice 和maxprofit。

代码：https://github.com/yyxd/leetcode/blob/master/src/leetcode/array/easy/Problem121.java

2. Problem 120. Triangle

解题思路：这道题给一个由整数组成的三角形，自上而下寻找顶点到三角形边的最短的一条路径，设置一个数组A[0...n-1][0...n-1]，A[i][j]代表到达第i行第j列结点的最短路径* DP转移方程式为：A[i][j]=min(A[i-1][j-1],A[i-1][j])+triangle[i][j]* 其中二维数组可以简化为一维数组，因为我们只需要上一行结点的信息* 然后遍历到达最后一行的节点的路径，找到最短路径的值

代码：https://github.com/yyxd/leetcode/blob/master/src/leetcode/dp/Problem120_Triangle.java

3. Problem 322. Coin Change

解题思路：这道题是给定具有n个不同金额的硬币（硬币个数无限）coins[0...n-1]，给一个整数amount，是否给的硬币能正好达到整数，给出能组成整数最少需要的硬币个数.解法是设置一个数组A[0...amount],进行初始化A[0]=0;A[1...amount] = -1;保存的是当给定金额为i时，所需要的最少的硬币。* dp转移方程式为 A[k] = 1+min(A[k-coins[0]],A[k-coins[1]],....A[k-coins[n-1]]).* 这里要注意的是判断A[k]是否有解

代码：https://github.com/yyxd/leetcode/blob/master/src/leetcode/dp/Problem322_CoinChange.java

课程完结撒花，真的学到好多，自己以后还会反复回顾的，再次感谢王争老师，还有每天负责朗读的声音好好听的修阳小哥哥。

2019-02-11 21:39

C_love

Best Time to Buy and Sell Stoc...

```
class Solution {
public int maxProfit(int[] prices) {
if (prices == null || prices.length == 0 || prices.length == 1) {
return 0;
}

int max = 0;
int start = 0;

for (int i = 1; i < prices.length; i++) {
if (prices[i] < prices[start]) {
start = i;
} else {
max = Math.max(max, prices[i] - prices[start]);
```

```
}
}
return max;
}
}
```

2019-02-11 14:38

ext4

最小路径和

```
class Solution {
public:
int minPathSum(vector< vector<int> >& grid) {
int m = grid.size();
if (m == 0) return 0;
int n = grid[0].size();
if (n == 0) return 0;
int matrix[m][n];
int accum = 0;
for (int i = 0; i < m; i++) {
accum += grid[i][0];
matrix[i][0] = accum;
}
accum = 0;
for (int j = 0; j < n; j++) {
accum += grid[0][j];
matrix[0][j] = accum;
}
for (int i = 1; i < m; i++) {
for (int j = 1; j < n; j++) {
matrix[i][j] = min(matrix[i - 1][j], matrix[i][j - 1]) + grid[i][j];
}
}
return matrix[m - 1][n - 1];
}
};
```

2019-02-11 13:11

Kyle Liu

有兴趣朋友可以将思路分析提交到https://github.com/kylesliu/awesome-golang-leetcode，欢迎大家提issue

2019-02-11 11:31

幻月剑

一刷完成，题目没有做，算是草草学了一遍，等第二遍将题目都做完

2019-02-11 07:41

峰

七七八八跟着老师复习了遍算法，印象最深刻的无疑是老师结合具体的应用场景讲利用的数据结构与算法！棒棒哒

2019-02-11 00:05

小美

这期题目好难，而且在Web网页上找对应的课程有点麻烦，能否在每个Day的文章中加上对应文章的链接呢？

2019-02-10 22:24

虎虎

正则表达式

```
public boolean isMatch(String s, String p) {
```

```
if (s == null || p == null) {
return false;
}
boolean[][] dp = new boolean[s.length()+1][p.length()+1];
dp[0][0] = true;
for (int i = 0; i < p.length(); i++) {
if (p.charAt(i) == '*' && dp[0][i-1]) {
dp[0][i+1] = true;
}
}
for (int i = 0 ; i < s.length(); i++) {
for (int j = 0; j < p.length(); j++) {
if (p.charAt(j) == '.') {
dp[i+1][j+1] = dp[i][j];
}
if (p.charAt(j) == s.charAt(i)) {
dp[i+1][j+1] = dp[i][j];
}
if (p.charAt(j) == '*') {
if (p.charAt(j-1) != s.charAt(i) && p.charAt(j-1) != '.') {
dp[i+1][j+1] = dp[i+1][j-1];
} else {
dp[i+1][j+1] = (dp[i+1][j] || dp[i][j+1] || dp[i+1][j-1]);
}
}
}
}
return dp[s.length()][p.length()];
}
```

leetcode的排名第一的答案，搬过来了

2019-02-10 21:35