算法笔记

Stay hungry stay foolish



# 前言

本书为个人算法学习笔记，仅作为学习交流，转载请注明，勿用于商业用途,如果发现书中有不足指出，还望纠正，邮箱:kotomifi@gmail.com

本书代码在 <https://github.com/kotomifi/algorithms>上可以下载

本书所有算法采用c++实现，利用了c++的模板机制

编译环境采用g++

本书大部分例子参考《算法导论》，部分例子来自zoj和hdoj，希望感兴趣的同学可以多去做一做上面的题目，对自己算法的 提高还是有很大好处的。

目录

[前言 I](#_Toc384099004)

[第一章dp算法 1](#_Toc384099005)

[1.1 lcs问题： 1](#_Toc384099006)

[1.2 钢条切割问题 3](#_Toc384099007)

[1.3 0/1背包问题 3](#_Toc384099008)

# 第一章dp算法

一个问题能采用dp算法应具备以下条件：

1. 最优子结构
2. 问题重叠

## 1.1 lcs问题：

问题描述：给定两个序列X=<x1, x2, …, xm>, Y=<y1, y2, …, yn>,求X和Y的最长公共子序列。

解题思路：假设Z<z1, z2, …, zk>为X和Y的LCS，那么：

1. 如果xm = yn， 则zk = xm = yn 且Zk-1 是Xm-1 和 Yn-1的一个LCS
2. 如果xm != yn， 则zk != xm且Zk-1 是Xm-1 和 Yn的一个LCS
3. 如果xm != yn， 则zk != yn 且Zk-1 是Xm 和 Yn-1的一个LCS

问题求解：

用数组 c[m+1][n+1] 保存 Xm和Yn的LCS，数组b[m][n]记录每一次比较的结果，如果不要求输入公共子序列而只输出公共子序列的长度，则可以去掉数组b。

代码如下：

#include <iostream>

#include <stack>

**using** **namespace** std;

/\*\*

 \* 计算两个数组的最长公共子序列

 \* m, n分别为两数组长度

 \* b 用来储公共子序列标识

 \*/

**template**<**typename** T>

**int** lcs(T \*x, **const** **int** &m, T \*y, **const** **int** &n, **int** \*\*b) {

**int** \*\*c = **new** **int**\*[m+1]; // 保存最长公共子序列

**for** (**int** t = 0; t <=m; ++t) {

c[t] = **new** **int**[n+1];

}

// 如果一个数组为空，则最长公共子序列为0

**for** (**int** i = 0; i <= m; ++i) {

c[i][0] = 0;

}

**for** (**int** j = 0; j <= n; ++j) {

c[0][j] = 0;

}

// dp求解

**for** (**int** k = 0; k != m; ++k) {

**for** (**int** r = 0; r != n; ++r) {

**if** (x[k] == y[r]) {

c[k+1][r+1] = c[k][r] + 1;

b[k][r] = 0;

} **else** **if** (c[k][r+1] < c[k+1][r]) {

c[k+1][r+1] = c[k+1][r];

b[k][r] = 1;

} **else** {

c[k+1][r+1] = c[k][r+1];

b[k][r] = 2;

}

}

}

**return** c[m][n];

}

/\*\*

 \* m, n 为数组长度

 \*/

**template**<**typename** T>

**void** printLCS(**const** **int** &m, **const** **int** &n, **int** \*\*b, T \*x) {

stack<T> sta;

**if** (0 == m || 0 == n)

**return**;

**if** (0 == b[m-1][n-1]) {

sta.push(x[m-1]);

printLCS(m-1, n-1, b, x);

}**else** **if** (1 == b[m-1][n-1]) {

printLCS(m, n-1, b, x);

} **else** {

printLCS(m-1, n, b, x);

}

**while** (!sta.empty()) {

cout << sta.top() << "\t";

sta.pop();

}

}

**int** main(**int** argc, **char** \*\*argv) {

**const** **int** m = 7;

**const** **int** n = 6;

**char** x[m] = {'a', 'b', 'c', 'b', 'd', 'a', 'b'};

**char** y[n] = {'b', 'd', 'c', 'a', 'b', 'a'};

**int** \*\*b = **new** **int**\*[m];

**for** (**int** i = 0; i != m; ++i) {

b[i] = **new** **int**[n];

}

**int** r = lcs(x, m, y, n, b);

**for** (**int** i = 0; i != m; ++i) {

**for** (**int** j = 0; j != n; ++j) {

cout << b[i][j] << "\t";

}

cout << endl;

}

printLCS(m, n, b, x);

cout << endl;

**return** 0;

}

## 1.2 钢条切割问题

问题描述：给定一根长度为n英寸的钢条和一个价格表pi，求切割钢条方案，使得销售收益rn最大，注意，如果长度为n英寸的钢条价格为pn足够大，最优解可能是不需要切割。

## 1.3 0/1背包问题

问题描述：已知有一个可容纳重量为C的背包以及n种物品，其中第i件物品的重量为wi，每件物品的效益是pi（pi > 0）。每个物品只能选择全部装入或者不装入，求最大效益。

解题思路：

先分析问题的约束条件，我们知道一次只能取一个物品或者不取，且总容量不能超过C，即有如下约束条件：



最优解：



# 第二章 贪心算法

# 第三章 图的搜索算法

3.3 回溯法