求两个字符串的所有公共最长子序列

代码：

class Solution{

public:

void search\_route(string &a,string &b,int xlabel,int ylabel,int \*\*&p,stack<char>&stk,vector<stack<char>>&result){

if (xlabel == 0 || ylabel == 0)return;

bool flag = false;

if (a[xlabel-1] == b[ylabel-1]){

stk.push(a[xlabel-1]);

if (stk.size() == p[a.size()][b.size()]){

result.push\_back(stk);

stk.pop();

return;

}

flag = true;

}

if (p[xlabel][ylabel] == p[xlabel - 1][ylabel] || p[xlabel][ylabel] == p[xlabel][ylabel - 1]){ **//如果可以往上走或是往左走**

if (p[xlabel][ylabel] == p[xlabel - 1][ylabel])search\_route(a, b, xlabel - 1, ylabel, p, stk, result);

if (p[xlabel][ylabel] == p[xlabel][ylabel - 1])search\_route(a, b, xlabel, ylabel - 1, p, stk, result);

}

else search\_route(a, b, xlabel-1, ylabel - 1, p, stk, result); **//只能往对角线走**

if(flag)stk.pop();

}

vector<stack<char>> LCSlen(string a,string b){

ofstream fout("c:\\users\\dell\\desktop\\cin.txt");

int sa = a.size(), sb = b.size();

if (sa == 0 || sb == 0)return result;

int \*\*p= new int \*[sa+1];

for (int i = 0; i < sa+1;i++)p[i] = new int[sb+1];

//填表

for (int i = 0; i < sa+1; i++)p[i][0] = 0;

for (int i = 0; i < sb+1; i++)p[0][i] = 0;

for (int i = 0; i < sa;i++){

for (int j = 0; j < sb;j++){

if (a[i] == b[j]){

p[i+1][j+1] = p[i][j] + 1;

}

else{

p[i+1][j+1] = max(p[i][j+1],p[i+1][j]);

}

}

}

for (int i = 0; i <=sa;i++){

for (int j = 0; j<= sb;j++)fout << setw(3)<<p[i][j];

fout << endl;

}

fout.close();

int lena = sa, lenb = sb;

search\_route(a,b,lena,lenb,p,stk,result);

for (int i = 0; i < sa; i++)delete p[i];

delete p;

return result;

}

stack<char>stk;

vector<stack<char>>result;

};

int main(void){

string b = "bdcaba";

string a = "abcbdab";

Solution s1;

vector<stack<char>> re = s1.LCSlen(a,b);

return 0;

}

输出：

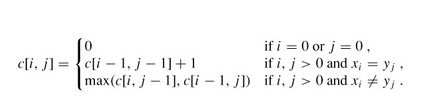
bcba

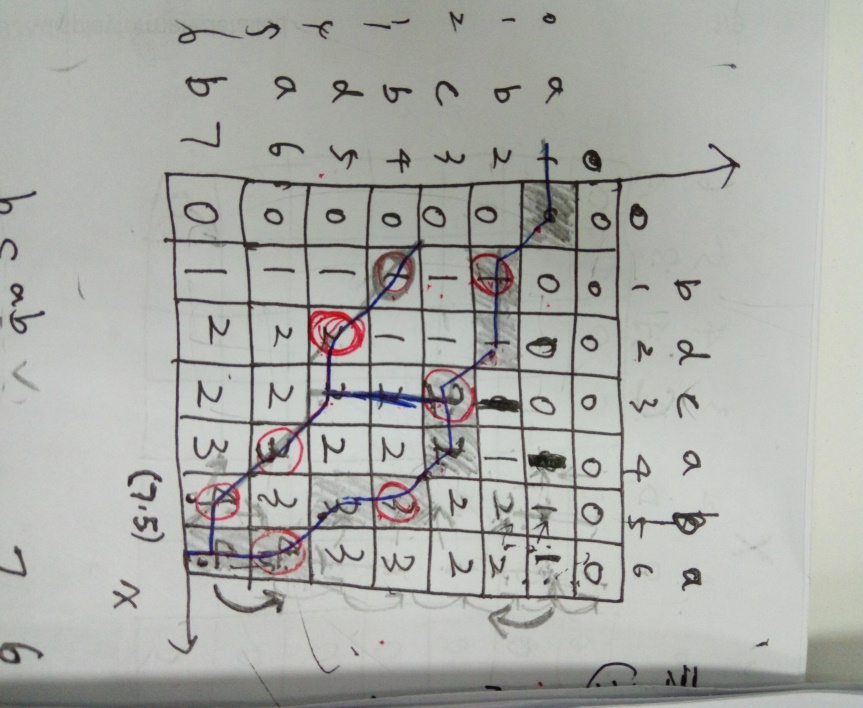
bdab

bcab

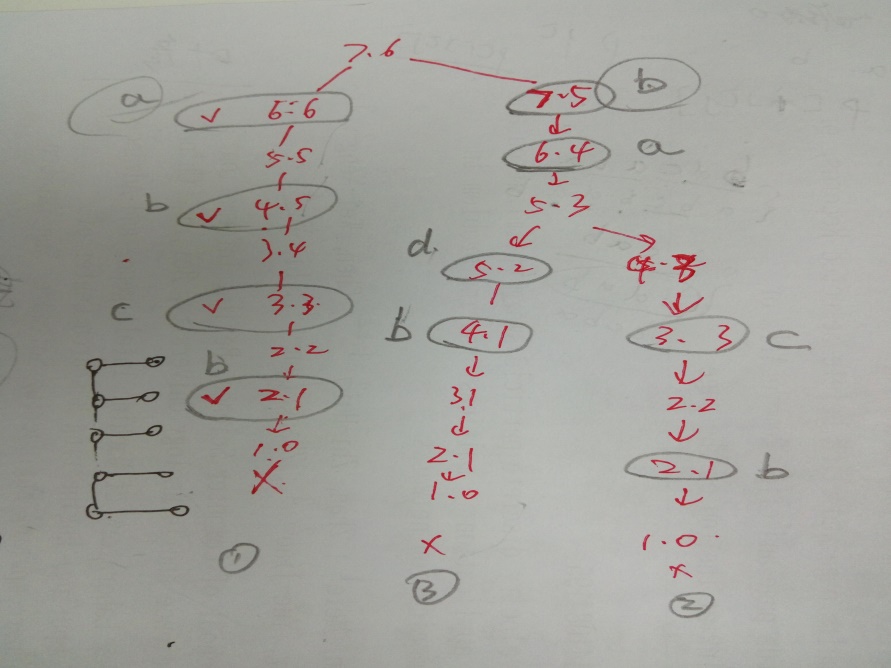
**分析**(结构如下图):

动态规划方程为：





1. 因为表格有sa+1行和sb+1列,所以在函数递归到xlabel和ylabel的时候，如果要判断字符是否相等，都要进行减1操作if (a[xlabel-1] == b[ylabel-1])；
2. 递归回溯从(7,6)开始，如图共有三条路径



3.在代码中

if (stk.size() == p[a.size()][b.size()]){

result.push\_back(stk);

**//stk.pop();**

**//return;**

}

若注释掉红色两行，则会出现以下这种情况：

在stk中已经有四个元素的情况下，还会继续添加元素,如下图所示从(4,1)继续向(2,1)移动，并将(2,1)对应的元素入栈。

if (stk.size() == p[a.size()][b.size()]){

result.push\_back(stk);

**stk.pop();**

**return;**

}

若立即出栈，并返回，在其后续语句

if (p[xlabel][ylabel] == p[xlabel - 1][ylabel] || p[xlabel][ylabel] == p[xlabel][ylabel - 1]){

**//如果可以往上走或是往左走**

if (p[xlabel][ylabel] == p[xlabel - 1][ylabel])search\_route(a, b, xlabel - 1, ylabel, p, stk, result);

if (p[xlabel][ylabel] == p[xlabel][ylabel - 1])search\_route(a, b, xlabel, ylabel - 1, p, stk, result);

}

else search\_route(a, b, xlabel-1, ylabel - 1, p, stk, result); **//只能往对角线走**

将不会执行，并直接返回到父函数。

