实验3-最长公共子序列

问题分析

- 对于最长公共子序列的长度，求解比较简单，采用动态规划即可，定义dp[i][j]为序列a[1-i]部分和序列b[1-j]匹配的最长公共子序列
- 考虑状态转移，分两种情况
  - 当(s1[i-1]==s2[j-1])时,dp[i][j]=dp[i-1][j-1]+1(s1,s2下标从0开始，所以这里减1)
  - 当(s1[i-1]!=s2[j-1])时,那么当前状态分别可从dp[i-1][j]和dp[i][j-1]转移过来，两者取最大值即可

输出全部的最长公共子序列

这是本实验的难点所在，为了能解决该问题，我选择借助序列自动机这一工具(Sequence Auto Machine)

序列自动机

- **自动机**，指将一个字符串构造成一个有向无环图（DAG）的过程
- 序列自动机是一个可以快速判断任意字符串T是否包含子串 S 的算法。
- 假定我们有一个串 s ，和数组 nxt。其中，$nxt_{i,j}$表示在串 s 中，从第 i个位置后的字母表中第 j 个元素首次出现的位置。
- 序列自动机的构造：很显然，对于 nxt 数组，我们需要对它进行构建。我们可以从末尾往前依次计算当前出现的位置。对于每次遍历，$nxt_{i,j}$ 初始等于 $nxt_{i+1,j}$。我们进行二重遍历：外层遍历当前位置，内层遍历当前元素即可构造出序列自动机。

```
class SequenceAM {// 序列自动机 Sequence Auto Machine
    public:
        int nxt[N][M]; // 核心数组,代表从第i个位置开始，字符j出现的第一个位置
        string s;  // 当前串
    void init() {  // 初始化序列自动机
        fill(nxt[s.size()], nxt[s.size()]+M, NPOS);
        for (int i=(int)s.size()-1; i>=0; i--) {  // 倒序初始化
            memcpy(nxt[i], nxt[i+1], sizeof nxt[i]);
            nxt[i][func(s[i])]=i+1;  // 更新数组
        }
    }
};
```

字母表

字母表是符号的有穷非空集合。

为了方便地求出nxt数组，我们要对序列所有可能出现的元素人为规定一个字母表，即字符到下标的映射关系，本题的输入会出现26个大小写英文字母和0-9的10个数字，所以字母表长度可以设为62，映射关系如下

- 小写英文字母对应下标0-25
- 大写英文字母对应下标26-51
- 数字对应下标52-61

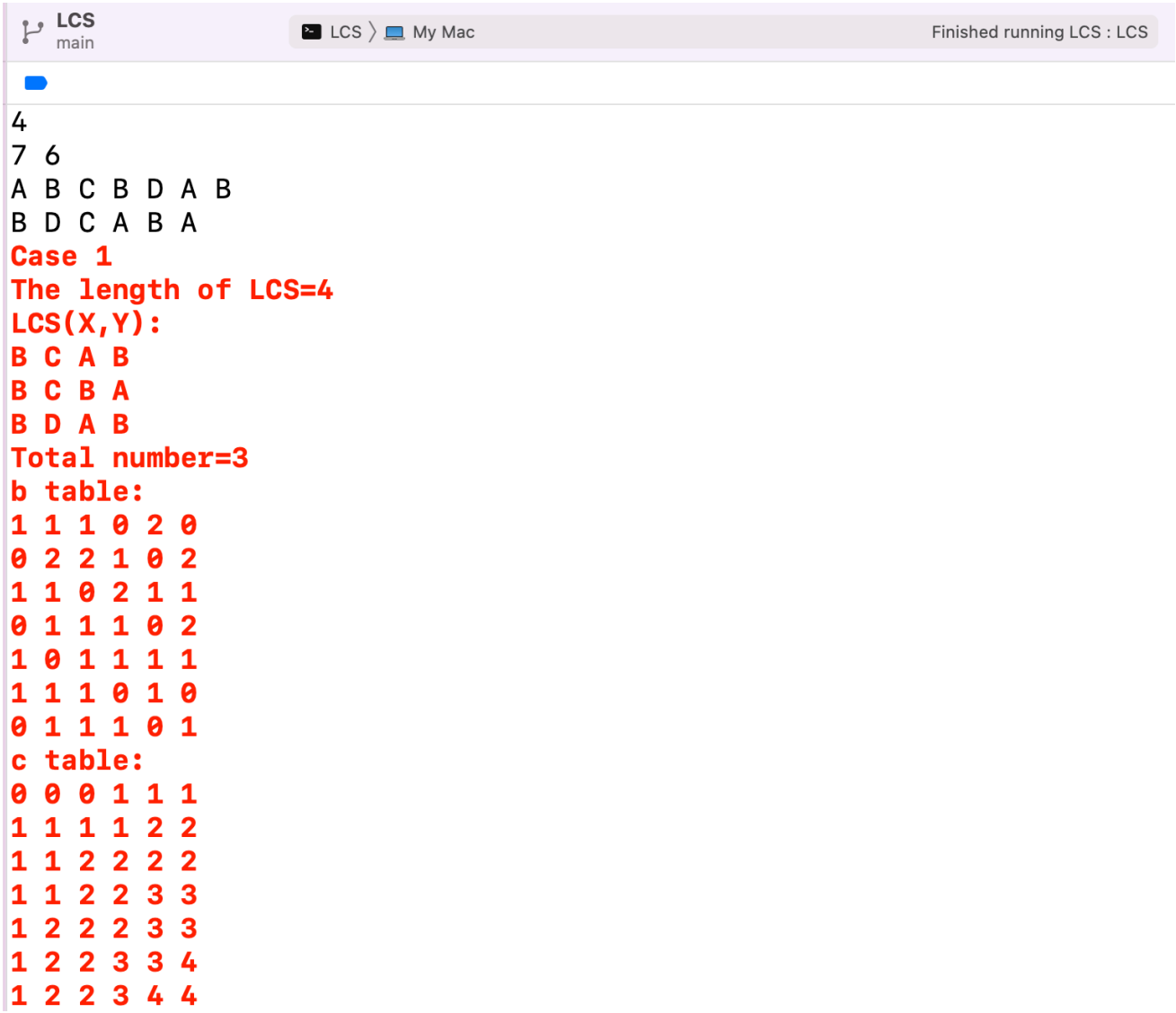
```
int func(char ch){
    if (ch>='a'&&ch<='z') {
        return ch-'a';
    }
    if (ch>='A'&&ch<='Z') {
        return ch-'A'+26;
    }
    if (ch>='0'&&ch<='9') {
        return ch-'0'+52;
    }
    return 0;
}
```

- 有了序列自动机之后，我们便能比较方便的不重不漏地求出一个序列的全部子序列组合。
- 对于两个序列的公共子序列，首先对于两个序列分别建立序列自动机，然后从小到大地去遍历字母表中的元素，当且仅当两个序列都包含当前字母，进一步搜索。
- 序列自动机本质上是构建了序列元素之间的关系并将关系映射到图上，所以对该图进行深度优先搜索，定义字符串ans，在遍历到当前节点的时候把节点的字符加进当前字符串，退出时把该字符删除即可。
- 我们从小到大遍历字母表的过程就是对整个图搜索的过程，在过程中我们可以遍历到两个序列的所有公共子序列，当该公共子序列长度等于最长公共子序列长度时，输出即可。
- NPOS是预先定义的无效值，代表串 s 中，从第 i个位置后不存在字母表中第 j 个元素。
- 从小到大便利字母表的字符，若两个序列都存在该字符，则两边的自动机都将当前位置向前跳到对应位置并进一步搜索（对应的就是分别在图上往下走了一条边）

```
void dfs(int x,int y){
```

```
    if (ans.size()==len) {
        cnt++;
        for (auto x: ans) {
            cout<<x<<' ';
        }
        cout<<'\n';
    }
    for (int i=0; i<62; i++) {
        if (a.nxt[x][i]!=NPOS&&b.nxt[y][i]!=NPOS) {
            ans.push_back(func2(i));
            dfs(a.nxt[x][i], b.nxt[y][i]);
            ans.pop_back();
        }
    }
}
```

运行结果截图



```
8 9
b a a b a b a b
a b a b b a b b a
Case 2
The length of LCS=6
LCS(X,Y):
a a b a b a
a a b a b b
a a b b a b
a b a b a b
b a a b b a
b a b a b a
b a b a b b
b a b b a b
Total number=8
b table:
1 0 2 0 0 2 0 0 2
0 1 0 2 2 0 2 2 0
0 1 0 1 1 0 2 2 0
1 0 1 0 0 1 0 0 2
0 1 0 1 1 0 1 1 0
1 0 1 0 0 1 0 0 1
0 1 0 1 1 0 1 1 0
1 0 1 0 0 1 0 0 1
c table:
0 1 1 1 1 1 1 1 1
1 1 2 2 2 2 2 2 2
1 1 2 2 2 3 3 3 3
1 2 2 3 3 3 4 4 4
1 2 3 3 3 4 4 4 5
1 2 3 4 4 4 5 5 5
1 2 3 4 4 5 5 5 6
1 2 3 4 5 5 6 6 6
```

```
11 11
b a a b a b 0 7 1 3 5
a b a b b a 0 1 7 4 3
Case 3
The length of LCS=7
LCS(X,Y):
a a b a 0 1 3
a a b a 0 7 3
a a b b 0 1 3
a a b b 0 7 3
a b a b 0 1 3
a b a b 0 7 3
b a b a 0 1 3
b a b a 0 7 3
b a b b 0 1 3
b a b b 0 7 3
Total number=10
b table:
1 0 2 0 0 2 2 2 2 2 2
0 1 0 2 2 0 2 2 2 2 2
0 1 0 1 1 0 2 2 2 2 2
1 0 1 0 0 1 1 1 1 1 1
0 1 0 1 1 0 2 2 2 2 2
1 0 1 0 0 1 1 1 1 1 1
1 1 1 1 1 1 0 2 2 2 2
1 1 1 1 1 1 1 1 0 2 2
1 1 1 1 1 1 1 0 1 1 1
1 1 1 1 1 1 1 1 1 1 0
1 1 1 1 1 1 1 1 1 1 1
```

```
c table:
0 1 1 1 1 1 1 1 1 1 1
1 1 2 2 2 2 2 2 2 2 2
1 1 2 2 2 3 3 3 3 3 3
1 2 2 3 3 3 3 3 3 3 3
1 2 3 3 3 4 4 4 4 4 4
1 2 3 4 4 4 4 4 4 4 4
1 2 3 4 4 4 5 5 5 5 5
1 2 3 4 4 4 5 5 6 6 6
1 2 3 4 4 4 5 6 6 6 6
1 2 3 4 4 4 5 6 6 6 7
1 2 3 4 4 4 5 6 6 6 7
```

```
Case 4
The length of LCS=4
LCS(X,Y):
G A C T
G C C T
G T C T
Total number=3
b table:
0 2 2 2 2 2
1 1 1 0 0 2
1 1 0 1 1 0
1 0 1 1 1 1
1 1 1 0 0 1
1 1 0 1 1 0
c table:
1 1 1 1 1 1
1 1 1 2 2 2
1 1 2 2 2 3
1 2 2 2 2 3
1 2 2 3 3 3
1 2 3 3 3 4
Program ended with exit code: 0
```

设计调试中的问题

- nxt[N][M]数组中M值开小啦，导致该数组初始化失败，将M调大即可
- dfs的过程中忘记在回溯的时候忘记恢复字符串ans的状态，导致答案错误，正确代码应为如下。在搜索结束后，应使用pop_back()函数将末尾的字符给删除掉

```
ans.push_back(func2(i));
dfs(a.nxt[x][i], b.nxt[y][i]);
ans.pop_back();
```

实验体会

本次实验相比前面两个实验难度有所增加。在最开始的时候，我只能找到一个最长的公共子序列，不知道怎么去简便地找出所有的最长公共子序列。在查阅资料之后，我学习到了序列自动机这个可以巧妙处理序列有关问题的有力工具。之后便顺利地实现了找出所有最长公共子序列的程序。这也提示了我，遇到困难不应畏惧，而是要想方设法地去解决它。本次实验加深了我对动态规划的理解，也教会了我字符串的一些相关知识，利用自动机把字符串字符之间的关系转换成图的形式进而巧妙地解决一些问题，本次实验让我受益匪浅。