tags: 演算法

演算法 程式作業4 報告

LCS

pseudo code

```
#def s1, s2: sequence 1, 2
#def map[i][j]: i is s1 length, j is s2 length
#def previoud[i][j]: keep the direction where map[i][j] come from
map[i][j] = 0, if i=0 or j=0
          = map[i-1][j-1] + 1, if s1[i] = s2[j], i, j != 0
            previous[i][j] = 0. //左上
          = max( map[i-1][j], map[i][j-1] ), if s1[i] != s2[j], i, j != 0
            previous[i][j] = 1, if map[i-1][j] bigger, //上
                           = 2, if map[i][j-1] bigger. //左

#def LCS: a sequence save LCS
void getLCS(int i, int j){    //建立LCS字串
    if(i == 0 || j == 0) return;
    if(previous[i][j] == 0){
        LCS.insert(LCS.begin(), s1[i]);
        getLCS(j-1, i-1);
    }
    else if(previous[i][j] == 1)    //上
        getLCS(j-1, i);
    else if(previous[i][j] == 2)    //左
        getLCS(j, i-1);
}
```

時間複雜度

- 假設s1長度n，s2長度m，花費O(nm)建立map，其餘程式複雜度皆比O(nm)低，因此是O(nm)

LIS

pseudo code

```
#def struct{
    character: 儲存字元
    index: 儲存字元位於那些位置，是vector，順序由大到小
} word
#def vector<word> s1:將字串1以上述規則儲存
#def s2: user輸入的字串2
#def LIS_element: 儲存準備LIS處理的數列
#def LIS: 儲存LIS數列
#def printIndex: 儲存要印出的s1序數
#def sIndex: 用來儲存DP對應到的序數
for i from 0 to s2.length
    if s2.at(i) == one of s1 character
        LIS_element.add(s1.this_character.index);


int maxLength = 0
vector<int> LIS_element_DP;
    for(int i=0;i<LIS_element.size();i++){
        int cur = LIS_element.at(i);
        if(i==0){
            LIS_element_DP.push_back(0);
            LIS.push_back(LIS_element.at(0));
        }
        else if(cur > LIS.back()){
            LIS.push_back(cur);
            LIS_element_DP.push_back(LIS.size()-1);
            maxLength = LIS.size()-1;
        }
        else{
            int temp = cur;
            int temp_small = MAX_INT;
            int temp_index = 0;
            for(int j=LIS.size()-1;j>=0;j--){
                if(LIS.at(j) >= temp && LIS.at(j) < temp_small){
                    temp_small = LIS.at(j);
                    temp_index = j;
                }
            }
            LIS.at(temp_index) = cur;
            LIS_element_DP.push_back(temp_index);
        }
    }

for i from LIS_element_DP.size()-1 to 0
    if LIS_element_DP.at(i) == maxLength
        sIndex.insert(sIndex.begin(), i);
        maxLength--;

for i from 0 to sIndex.size()-1
        int temp = LIS_element.at(sIndex.at(i));
        printIndex.push_back(temp);
```

時間複雜度

- 假設s2長度n，LIS長度L，最主要花費在處理DP，每個s2元素都要處理，而往前看時的花費平均為O(logL)，因此總花費為O(nlogL)

比較LCS

- 由於LCS會跑過所有的s1和s2來建立map，消耗時間較多，LIS則處理s2，大部分時間花在LIS的DP上，當LIS過長時會消耗較多時間，但總體較LCS快速。

時間差

電腦問題無法計算