**算法设计与分析 实验报告**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 姓名 | 高嘉 | | 院系 | 软件学院 | | 学号 | 2021211948 |
| 任课教师 | | 王金宝 | | | 指导教师 | 王金宝 | |
| 实验地点 | | 研究院 | | | 实验时间 | 2023年 4月 13 日 | |
| 实验名称 | | 使用动态规划技术的算法设计与分析 | | | | | |
| 实验内容 | | | | | | | |
| **1、最长公共增长子序列问题定义如下：**  输入：由*n*个数组成的一个序列*S*：*a1,a2,…,an*,  由*m*个数组成的一个序列*T*: *b1*,*b2*,…*bm*.  输出：*S*和*T*的公共子序列*X=c1c2,…,ck* ，满足：  (1) *c1≤c2 ≤ … ≤ ck* ，  (2) |*X*|最大  使用动态规划技术设计算法求解最长公共增长子序列问题。请分析优化解的结构，递归地定义优化解的代价，给出算法的伪代码和实现过程中的主要代码，并分析算法的时间复杂性. 随机生成两个数组，并展示运行结果。  **2、问题求解思路**  2.1、最优解的构成分析  设=()为最长公共增长子序列,d[i][j]存放最长序列的最后一位：   1. 若 , ，1<=j<=m，   1)，d[n][m]=   1. 若 , 如果，那么d[n][m]=d[n-1][m]，否则d[n][m]=d[n][m-1]。   2.2、优化解的递归方程  C[i,j]为S和T的最长公共增长子序列  递归方程为：  C[i,j]=0,d[i][j]=0, if i=0或j=0;  C[i,j]=max(C[i-1,j-k])+1,d[i][j]=ai  if d[i][j]>=d[i-1][j-k] and i,j>0and ai=bj  C[i,j]=max(C[i-1,j],C[i][j-1])  If(C[i,j]=C[i-1,j]) d[i][j]=d[i-1][j],else d[i][j]=d[i][j-1]  If aibj and i,j>0  2.3、自顶向下的划分和自底向上的求解过程简述   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | C | ai | 3 | 5 | 3 | 4 | 5 | | bj | 0 | 0 | 0 | 0 | 0 | 0 | | 3 | 0 | 1 | 1 | 1 | 1 | 1 | | 2 | 0 | 1 | 1 | 1 | 1 | 1 | | 5 | 0 | 1 | 2 | 2 | 2 | 2 | | 3 | 0 | 1 | 2 | 2 | 2 | 2 | | 4 | 0 | 1 | 2 | 2 | 3 | 3 | |  |  |  |  |  |  |  | | d | ai | 3 | 5 | 3 | 4 | 5 | | bj | 0 | 0 | 0 | 0 | 0 | 0 | | 3 | 0 | 3 | 3 | 3 | 3 | 3 | | 2 | 0 | 3 | 3 | 3 | 3 | 3 | | 5 | 0 | 3 | 5 | 5 | 5 | 5 | | 3 | 0 | 3 | 5 | 3 | 5 | 5 | | 4 | 0 | 3 | 5 | 3 | 4 | 4 |   **3、算法伪代码**  LCSM—length(S,T)  n=length(S),m=length(T)  For i=0 to n Do C[i,0]=0,d[i,0]=0;  For j=0 to m Do C[0,j]=0,d[0,j]=0;  For i=1 to n Do  For j=1 to m Do  If()  Then k=1 to j Do  If(d[i-1][j-k]<=and c[i,j]<=c[i-1,j-k]+1)  Then c[i,j]=c[i-1,j-k]+1  d[i][j]=  Else  IF(C[i-1][j]>=C[i][j-1])  Then C[i][j]=C[i-1][j];d[i][j]=d[i-1][j];  Else C[i][j]=C[i][j-1];d[i][j]=d[i][j-1];  **4、算法复杂性分析**  计算代价的时间：（i,j,k）三重循环，O(m\*m\*n)  空间复杂度：需要数组C和d，O(m\*n)  **5、测试结果**（说明程序中使用的主要数据结构及其上的操作）  数据结构：  数组C[n][m]记录序列最大长度，  数组d[n][m]记录最长序列最后一位的值  先计算最上面（i,0）和最左面(0,j)，在一行一行计算 | | | | | | | |
| 实验结论（结果分析、遇到的困难和解决方法等） | | | | | | 备注 |  |
| 结果分析：    遇到的问题：  怎样让序列跳过当前最大值，再向前搜索。  解决方案：  再设一个变量k，全部遍历一遍。 | | | | | | | |