**《算法设计与分析》实验报告**

实验名称 ： 实验3 动态规划

实验日期 ： 　 2025.3.25

姓 名 ： 　 高心阳

学 号 ： 　 084623237

班 级 ： 　 计算机232

成 绩 ：

**人工智能与信息技术学院**

**南京中医药大学**

|  |
| --- |
| **实验目的：** |
| 1. 掌握动态规划算法的思想 2. 能够编写动态规划算法程序 |
| **实验内容和要求** |
| 1、数字金字塔问题  老王到埃及看金字塔，想从金字塔底向上走到顶。可是每一条路所花问题的时间都不一样长。请帮他找到登上金字塔最快的路(所费时间为路径上数字的总和)。注意每一个向上的路径只有左上和右上的两条路。  样例输入（结构不限）:    样例输出:  从山脚到山顶最短路径长度为92，  路径为09->18->20->45    2、最长公共子序列问题  假设爸爸对应的基因序列为X={x1, x2，x3 . . . xm}，  孩子对应的基因序列为Y={y1, y2,y3 . . . yn},  那么怎么找到他们有多少相似的基因呢?  样例输入：  请输入待检测的两段字符信息：  ABCADAB  BACDBA  样例输出：  两段字符信息的最长公共子序列长度为4，  对应的序列为ACDA。  3. 最长公共子串问题桥杯题目  题目描述  Alex输入了hish，那么他原本要输入的是fish还是vista呢？  样例输入：  请输入原输入字符：hish  请输入预判断字符1：fish  请输入预判断字符2：vista样例输出  样例输出：  原本应输入的字符为fish! |
| **运行结果（写清题号）** |
| |  | | --- | | Task1.cpp | | #include <bits/stdc++.h>  #define endl '\n'  using namespace std;  typedef long long ll;  void solve() {  int n;  cin >> n;  std::vector<vector<int> > v(n, vector<int>(n));  for (int row = 0; row < n; row++)  for (int col = 0; col <= row; col++)  cin >> v[row][col];  for (int row = n - 2; row >= 0; row--)  for (int col = 0; col <= row; col++)  v[row][col] += min(v[row + 1][col], v[row + 1][col + 1]);  cout << v[0][0] << endl;  }  int main() {  int T;  // cin >> T;  T = 1;  while (T--) {  solve();  }  return 0;  } |  |  | | --- | | Task2.cpp | | #include <bits/stdc++.h>  #define endl '\n'  using namespace std;  typedef long long ll;  void solve() {  string x, y;  cin >> x >> y;  std::vector<vector<int> > v(x.length() + 3, vector<int>(y.length() + 3, 0));  std::vector<vector<int> > m(x.length() + 3, vector<int>(y.length() + 3, 0));  for (int i = 0; i < x.length() + 3; ++i) v[i][0] = 0;  for (int i = 0; i < y.length() + 3; ++i) v[0][i] = 0;  for (int i = 1; i <= x.length(); ++i) {  for (int j = 1; j <= y.length(); ++j) {  if (x[i] == y[j]) {  m[i][j] = 1;  v[i][j] = v[i - 1][j - 1] + 1;  }  else if (v[i - 1][j] > v[i][j - 1]) {  v[i][j] = v[i - 1][j];  m[i][j] = 2;  } else {  v[i][j] = v[i][j - 1];  m[i][j] = 3;  }  }  }  cout << v[x.length()][y.length()] << endl;  string str;  int row = x.length(), col = y.length();  while (1) {  if (row == 0 || col == 0)  break;  if (str.length() == v[x.length()][y.length()]) {  cout << str << endl;  return;  }  if (m[row][col] == 1) {  str = str + x[row - 2];  row--;  col--;  }  else if (m[row][col] == 2) {  row--;  } else if (m[row][col] == 3) {  col--;  }  }  for(int i=str.length()-1;i>=0;i--)  cout << str[i];  }  int main() {  int T;  // cin >> T;  T = 1;  while (T--) {  solve();  }  return 0;  } |  |  | | --- | | Task3.cpp | | #include <bits/stdc++.h>  #define endl '\n'  using namespace std;  typedef long long ll;  int check(string x, string y) {  std::vector<vector<int> > v(x.length() + 3, vector<int>(y.length() + 3, 0));  for (int i = 0; i < x.length() + 3; ++i) v[i][0] = 0;  for (int i = 0; i < y.length() + 3; ++i) v[0][i] = 0;  for (int i = 0; i < x.length(); ++i) {  for (int j = 0; j < y.length(); ++j) {  if (x[i] == y[j])  v[i+1][j+1] = v[i][j] + 1;  else  v[i+1][j+1] = 0;  }  }  int ans = -1;  for (int i = 1; i <= y.length(); ++i)  if (v[x.length()][i] > ans)  ans = v[x.length()][i];  return ans;  }  void solve() {  string x, y, z;  cin >> x >> y >> z;  if (check(x, y) > check(x, z))  cout << y << endl;  else  cout << z << endl;  }  int main() {  int T;  // cin >> T;  T = 1;  while (T--) {  solve();  }  return 0;  } | |
| **实验的体会与建议** |
| 学习了动态规划的基础知识，理解了空间换时间的基本理念，提升了实际问题解决的能力。 |