**《算法设计与分析》实验报告**

实验名称 ： 实验4 动态规划 2

实验日期 ： 　 2025.4.1

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. 深入了解动态规划算法的思想 2. 熟练编写动态规划算法程序 |
| **实验内容和要求** |
| 1、0-1背包问题  问题描述：  给定n种物品和一个背包，物品i的重量是wi，其价值为vi，背包的容量为C。背包问题是如何选择装入背包的物品，使得装入背包中物品的总价值最大。  输入内容（格式不限）:  有5个物品，  其体积分别是{2, 5, 4, 2,3}，价值分别为{ 6, 3, 5, 4,6}，  背包的容量为10  输出格式:  放入背包的物品为1,3,5，最大总价值为17。  2、最大子段和  问题描述：  输入一个整形数组，数组里有正数也有负数，数组中连续的一个或多个整数组成一个子数组，每个子数组都有一个和，求所有子数组的和的最大值。要求时间复杂度为O(n)。  输入样例  1,-2,3,10，-4,7,2，-5  输出样例  和最大子数组为3,10，-4,7,2,对应数组和为18。  3. 最长递增子序列  问题描述：  给出一个序列a1,a2,a3,a4,a5,a6,a7....an，求它的一个子序列（设为s1,s2,...sn），使得这个子序列满足这样的性质，s1<s2<s3<...<sn，并且这个子序列的长度最长。  输入  1 7 3 5 9 4 8  输出  长度为4  序列为1 3 5 9和1 3 4 8….. |
| **运行结果（写清题号）** |
| |  | | --- | | Task1.cpp | | // encoding:utf-8  //1、0-1背包问题  // 问题描述：  // 给定n种物品和一个背包，物品i的重量是wi，其价值为vi，背包的容量为C。  // 背包问题是如何选择装入背包的物品，使得装入背包中物品的总价值最大。  // 输入内容（格式不限）:  // 5  // 2 5 4 2 3  // 6 3 5 4 6  // 10  // 有5个物品，  // 其体积分别是{2, 5, 4, 2,3}，价值分别为{ 6, 3, 5, 4,6}，  // 背包的容量为10  // 输出格式:  // 放入背包的物品为1,3,5，最大总价值为17。  #include <bits/stdc++.h>  using namespace std;  void solve() {  int n;  cin >> n;  std::vector<int> v(n);  std::vector<int> c(n);  int V;  for (int i = 0; i < n; ++i)  cin >> v[i];  for (int i = 0; i < n; ++i)  cin >> c[i];  cin >> V;  vector<vector<int> > dp(n + 1, vector<int>(V + 1, 0));  for (int i = 1; i <= n; i++) {  for (int j = 1; j <= V; j++) {  if (v[i - 1] <= j)  dp[i][j] = max(dp[i - 1][j], dp[i - 1][j - v[i - 1]] + c[i - 1]);  else  dp[i][j] = dp[i - 1][j];  // cout << dp[i][j] << "\t";  }  // cout << endl;  }  cout << "max:" << dp[n][V] << endl;  vector<int> way;  int i = n, j = V;  while (i > 0 && j > 0) {  if (dp[i][j] != dp[i - 1][j]) {  way.push\_back(i);  j -= v[i-1];  }  i--;  }  reverse(way.begin(), way.end());  cout << "way:";  for (auto item : way)  cout << item << " ";  }  int main() {  std::ios::sync\_with\_stdio(0);  std::cin.tie(0);  int T;  T = 1;  while (T--) {  solve();  }  return 0;  } |  |  | | --- | | Task2.cpp | | // encoding:utf-8  #include <bits/stdc++.h>  using namespace std;  // 2、最大子段和  // 问题描述：  // 输入一个整形数组，数组里有正数也有负数，数组中连续的一个或多个整数  // 组成一个子数组，每个子数组都有一个和，求所有子数组的和的最大值。  // 要求时间复杂度为O(n)。  // 输入样例  // 8  // 1 -2 3 10 -4 7 2 -5  // 1,-2,3,10，-4,7,2，-5  // 输出样例  // 和最大子数组为3,10，-4,7,2,对应数组和为18。  void solve() {  int n;  cin >> n;  std::vector<int> v(n);  std::vector<int> pre(n + 1, 0);  int max = -1, maxi = -1, nsi = -1, maxsi = -1;  for (int i = 0; i < n; i++)  cin >> v[i];  for (int i = 1; i <= n; i++) {  if (pre[i - 1] < 0) {  nsi = i;  pre[i] = v[i - 1];  }  else  pre[i] = pre[i - 1] + v[i - 1];  if (pre[i] > max) {  max = pre[i];  maxi = i;  maxsi = nsi;  }  }  cout << "max:" << max << endl;  cout << "way:";  for (int i = maxsi; i <= maxi; i++)  cout << v[i - 1] << " ";  }  int main() {  std::ios::sync\_with\_stdio(0);  std::cin.tie(0);  int T;  T = 1;  while (T--) {  solve();  }  return 0;  } |  |  | | --- | | Task3.cpp | | // encoding:utf-8  #include <bits/stdc++.h>  using namespace std;  // 3. 最长递增子序列  // 问题描述：  // 给出一个序列a1,a2,a3,a4,a5,a6,a7....an，  // 求它的一个子序列（设为s1,s2,...sn），使得这个子序列满足这样的性质，  // s1<s2<s3<...<sn，并且这个子序列的长度最长。  // 输入  // 7  // 1 7 3 5 9 4 8  // 输出  // 长度为4  // 序列为1 3 5 9和1 3 4 8  struct Node {  int length;  int max;  };  void solve() {  int n;  cin >> n;  std::vector<int> v(n);  std::vector<Node> dp(n + 1);  for (int i = 0; i < n; i++)  cin >> v[i];  dp[0].length = 0;  dp[0].max = INT\_MIN;  int max = -1;  for (int i = 1; i <= n; i++) {  dp[i].length = 1;  dp[i].max = v[i - 1];  for (int j = 0; j < i; j++) {  if (dp[j].max < dp[i].max && dp[j].length + 1 > dp[i].length) {  dp[i].max = v[i - 1];  dp[i].length = dp[j].length + 1;  }  }  if (dp[i].length > max)  max = dp[i].length;  }  cout << "max:" << max << endl;  vector<int> way;  int i = n, nmax = INT\_MAX;  while (i > 0 && max > 0) {  if (dp[i].length < nmax) {  way.push\_back(dp[i].max);  nmax = dp[i].length;  max--;  }  i--;  }  reverse(way.begin(), way.end());  cout << "way:";  for (auto item : way)  cout << item << " ";  }  int main() {  std::ios::sync\_with\_stdio(0);  std::cin.tie(0);  int T;  T = 1;  while (T--) {  solve();  }  return 0;  } | |
| **实验的体会与建议** |
| 进一步了解了动态规划的编写与理解，有利于解决更多问题，寻找最优结果，深刻理解了空间换时间的理念。 |