重庆大学本科学生实验项目任务书

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 实验题目 | 动态规划算法实践 | | | |
| 实验时间 |  | 实验地点 | |  |
| 实验性质 | □验证性 √设计性 □综合性 | | | |
| 实验目的  1. 掌握动态规划算法的基本原理  2. 训练使用动态规划算法设计技术，通过编程解决不同难度问题的实践能力 | | | | |
| 实验内容   1. 动态算法基本原理及时空复杂度分析方法 2. 在线编程解决2道不同难度的编程题，共40分   3． 在线编程时间（答题时长）150分钟，从开始答题时刻起算  4． 在线程序测评平台采用PTA (https://pintia.cn/)  5． 每人必须独立完成编程，可查阅教科书、PPT等资料，不得相互抄袭以及抄袭网上已有的程序  6． 实验课结束后，会对所有程序进行查重，如检测出有抄袭的程序，成绩计零分处理  注意：最后提交完整的实验报告，包括对每道题的算法思路、代码描述、复杂度分析等内容并回答思考题。 | | | | |
| 参考资料：   * Data Structures and Algorithm Analysis (C++ Version) Clifford A. Shaffer   + Introduction to Algorithms, 3rd Edition, MIT Press, T.H. Cormen, et al.   + 《数据结构（ C 语言版）》，严蔚敏，吴伟民编著，清华大学出版社 | | | | |
| 任务下达日期 年 月 日 | | | 完成日期 年 月 日 | |

说明：学院、专业、年级均填全称，如：计算机学院、计算机科学与技术、2021。

**《数据结构与算法》实验报告**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **年级、专业、班级** | |  | | | **姓名** |  |
| **实验题目** | **动态规划算法实践** | | | | | |
| **实验时间** |  | | **实验地点** |  | | |
| **实验成绩** |  | | **实验性质** | **□验证性** √**设计性 □综合性** | | |
| 教师评价：  □算法/实验过程正确； □源程序/实验内容提交 □程序结构/实验步骤合理；  □实验结果正确； □语法、语义正确； □报告规范；  其他：  评价教师签名： | | | | | | |
| 实验目的  1. 掌握动态规划算法的基本原理  2. 训练使用动态规划算法设计技术，通过编程解决不同难度问题的实践能力 | | | | | | |
| 二、实验项目内容  注：每道题按下面的格式分别描述  实验课题1：  题目内容：  **7-1 真实的背包故事**  **0-1背包问题是经典的动态规划问题，这个问题大多用这样的故事开场：一个小偷溜进了一家博物馆，博物馆里排列着*N*件珍稀古董，每件古董都有重量和价值，而小偷带来的背包有重量限制*W*。因此，小偷的目的就是要选择部分古董，使其总重量不超过*W*且总价值最大。这故事听上去就像小偷在逛超市一样能轻松自如地挑选，而真实的情况是小偷提心吊胆，尤其是每拿下一件古董，随时都有触动警报的危险，所以小偷想尽可能少带几件古董立马跑路，但他的职业“素养”又不允许他不把背包装满。你能帮他解决这个困境吗？**  **输入格式:**  **第一行中给出2个整数值N和W,其中1<N≤100表示古董的数量，1<W≤2000表示背包的重量限制。**  **接下来N行数据，每行两个正整数。第i行（1≤i≤N)的整数vi和wi分别表示第i件古董的价值和重量。vi和wi的值均不超过2000。**  **输出格式:**  **在一行中输出两个整数值，用空格分开。第一个整数表示装背包能获得的最大总价值，第二个整数表示在获得最大价值的条件下装入背包里的古董的最少数量。**  解题思路（含递推方程）：  在经典的01背包基础上,对最大价值进行状态转移的同时,对所对应的物品件数进行同步转移.当两种转移来源的最大价值相同时,取物品件数较小的一种.  f[j][0]=f[j-w][0]+v;f[j][1]=f[j-w][1]+1; f[j][0]<=f[j-w][0]+v;  f[j][0]=f[j][0];f[j][1]=f[j][1]; f[j][0]>f[j-w][0]+v;  源代码（格式正确，对关键步骤加注释）：  #include<bits/stdc++.h>  using namespace std;  typedef pair<int,int> P;  typedef long long ll;  typedef unsigned long long ull;  const int N=1e5+10,inf=0x3f3f3f3f,mod=1e9+7;  int f[2001][2],n,m; //循环进行到第i次时，f[j][0]表示总重量不超过j的物品的最大价值,f[j][1]表示此时携带最少的物品数  int main()  {  scanf("%d%d",&n,&m);  for(int i=1;i<=n;i++)  {  int v,w;  scanf("%d%d",&v,&w);  for(int j=m;j>=w;j--)  if(f[j][0]<=f[j-w][0]+v)//总重量不超过j时,携带第i件物品所获得的总价值更大  {  f[j][0]=f[j-w][0]+v;  f[j][1]=f[j-w][1]+1;//f[j][0]=f[j-w][0]+v时,f[j][1]=min(f[j-w][1],f[j][1])+1.容易证明,不等式f[j-w][1]<=f[j][1]恒成立.  }  }  printf("%d %d",f[m][0],f[m][1]);  return 0;  }  时间与空间复杂度分析  时间复杂度O(n\*W):  外循环执行n次,内循环最坏情况下执行W次.故动态规划操作的时间复杂度为O(n\*W);输入和输出操作的时间复杂度均为O(n).总体的时间复杂度为O(n\*W)+O(n)=O(n\*W).  空间复杂度O(W):  用于保留动态规划过程量的数组空间占用为W\*2,临时变量v,w的空间占用为O(1),总体的空间复杂度为O(W\*2)+O(1)=O(W).  实验课题2：  题目内容：  **7-2 动态规划**  **马里奥又在编愚蠢的游戏，而不是在学校读陀思妥耶夫斯基。他最新的游戏规则就是这样。 首先，他从书中随便选了一个词。然后他把这个在任意两个位置分割，得到三个独立的词。 之后，他颠倒了这三个单词中每个字母的顺序（交换第一个和最后一个字母，第二个和第二个最后一个字母等等）。 最后，他把这三个词按照分开前的顺序重新组合起来。游戏的目标是获得尽可能最小的单词。换句话说，在通过上述过程可以获得的所有单词中，找出字典中最早的一个。 编写一个程序，完美地玩马里奥的游戏。**  **输入格式:**  **输入的第一行也是唯一一行包含马里奥选择的单词，这是英语字母表中不带空格的小写字母串。 输入字的长度在3到50个字符之间（含）。**  **输出格式:**  **输出最好的单词。**  解题思路（含递推方程）：  暴力穷举分割位置,颠倒并组合成新单词.取每次操作后的最小值单词,即ans=min(ans,s1+s2+s3).  源代码（格式正确，对关键步骤加注释）：  #include<bits/stdc++.h>  using namespace std;  typedef pair<int,int> P;  typedef long long ll;  typedef unsigned long long ull;  const int N=1e5+10,inf=0x3f3f3f3f,mod=1e9+7;  string str,ans;  int main()  {  cin>>str;  ans.append(int(str.size()),'z');//初始化单词为无穷大  for(int i=0;i<int(str.size())-2;i++)  for(int j=i+1;j<int(str.size())-1;j++)//穷举分割位置  {  string s1,s2,s3;  s1=str.substr(0,i+1);reverse(s1.begin(),s1.end());  s2=str.substr(i+1,j-i);reverse(s2.begin(),s2.end());  s3=str.substr(j+1);reverse(s3.begin(),s3.end());//截取对应子串并翻转  ans=min(ans,s1+s2+s3);//组合各子串并保留最小值  }  cout<<ans;  return 0;  }  时间与空间复杂度分析  时间复杂度O(n^3):  循环一共进行n\*n次,每次循环中字符串截取操作和翻转操作的时间复杂度均为O(n),故总的时间复杂度为O(n\*n)\*O(n)=O(n^3).  空间复杂度O(n):  全局变量str和ans的空间复杂度为O(n),每次循环新建变量s1,s2,s3的空间复杂度也为O(n),生存周期为本次循环.故总的空间复杂度为O(2\*n+3\*n)=O(n). | | | | | | |
| 三、思考题  0-1背包问题，当商品的重量和价值满足以下哪些条件，就可用比动态规划效率更好的算法解决？ 描述解题思路并分析时空复杂度。如果没有更好的算法， 说明理由（举反例）。   1. 每件商品的价值相同   贪心算法:每次取重量最小的商品，直至背包不能再装入新物品.时空复杂度取决于所用的排序算法.   1. 每件商品的重量相同   贪心算法:每次取价值最大的商品，直至背包不能再装入新物品.时空复杂度取决于所用的排序算法.   1. 每件商品的性价比相同 （性价比 = 价值/重量）   贪心算法:每次取重量最小的商品，直至背包不能再装入新物品.时空复杂度取决于所用的排序算法.   1. 重量越大，性价比越低   贪心算法的反例:  每次重量取最小:背包总容量为2,物品1重量为1,价值为100;物品2重量为2,价值为101.方案1:取物品1;方案2:取物品2.显然方案2优于方案1.  每次性价比取最高:等价于每次重量取最小.  每次价值取最大:背包总容量为2,物品1重量为1,价值为100;物品2重量为1,价值为100;物品3重量为2,价值为101;方案1:取物品1和物品2;方案2:取物品2.显然方案1优于方案2.   1. 重量越大，价值越小   贪心算法:每次取重量最小的商品，直至背包不能再装入新物品.时空复杂度取决于所用的排序算法. | | | | | | |