**安徽大学集成电路学院**

**《算法设计与分析》第七次报告**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 实验名称 | 算法设计与分析 | | | 日期 |  |
| 专业 |  |  |  |  |  |
| 【实验目的】  本实验的主要目的是通过使用MATLAB语言实现动态规划算法，解决类似于0/1背包问题的最大化货柜价值问题。具体目的包括：  1.理解和掌握动态规划算法的基本原理：  理解动态规划解决问题的思路和步骤，包括状态转移方程的设计和边界条件的处理。  2.熟悉动态规划在实际问题中的应用：  通过解决货柜价值最大化问题，了解动态规划算法在优化问题中的应用及其有效性。  3.提升MATLAB编程能力：  学习和掌握MATLAB语言的基本语法和函数使用，提高用MATLAB编程解决实际问题的能力。  4.培养问题分析与解决能力：  通过对问题的分析，设计并实现相应的算法，培养分析问题和解决问题的综合能力。  5.验证算法的正确性和效率：  通过实验数据记录与处理，验证所实现算法的正确性，并分析算法的时间和空间复杂度。  6.为后续相关课程打下基础：  为后续更加复杂的算法设计与分析课程打下良好的基础，做好理论与实践的结合。 | | | | | |
| 【实验原理（预习）】  在本实验中，我们通过解决一个类似于0/1背包问题的动态规划问题来最大化货柜价值。实验的核心原理包括以下几个方面。  1.动态规划概述  动态规划是一种用于解决最优化问题的算法设计技术。它通过将问题分解为相对简单的子问题来解决复杂问题，并通过存储子问题的结果来避免重复计算，从而提高效率。动态规划特别适用于具有重叠子问题和最优子结构性质的问题。  2.0/1背包问题  0/1背包问题是经典的动态规划问题之一，其目标是在给定的容量限制下，选择若干物品使得物品的总价值最大化。每件物品要么被选择，要么不被选择，这就是“0/1”的含义。  3.问题描述与模型构建  在本实验中，我们的问题类似于0/1背包问题，但描述为货柜选择问题。给定一个长度为.的库房和个货柜，每个货柜有其特定的长度和价值。目标是在库房长度限制内选择货柜，使得所选货柜的总价值最大化。  4. 动态规划状态转移方程  为了应用动态规划解决该问题，我们定义一个二维数组，其中表示在前个货柜中选择，并且库房长度不超过时的最大总价值。  状态转移方程如下：    其中，和分别表示第个货柜的长度和价值。具体来说：  如果不选择第个货柜，那么；  如果选择第个货柜，并且库房剩余长度足够，则，前提是。  5. 边界条件  初始化时，只有第一个货柜的情况：    6. 算法实现步骤  1. 初始化矩阵：创建并初始化二维数组。  2. 填表过程：根据状态转移方程逐步填充二维数组。  3. 结果输出：最终结果保存在中，表示在库房长度为时可获得的最大总价值。  通过上述步骤和原理，我们可以系统地解决货柜价值最大化问题，并验证动态规划算法在实际问题中的应用和有效性。  解决该问题的伪代码如下   |  | | --- | | 伪代码 | | Store  输入：数组L[1..n]，V[1..n]，D //L和V是货柜长度和价值序列，D为库房长度  输出：最大的收益C[n,D]  1. for y←1 to D  2.     C[1,y]=V[1]  3. for k←2 to n  4.     for y←1 to D  5.         C[k,y] ← C[k-1,y]  6.         i[k,y] ← i[k-1,y]  7.         if y ≥ L[k] and C[k-1,y-L[k]]+V[k] > C[k-1,y]  8.             then C[k,y] ← C[k-1,y-L[k]]+V[k]  9.                 i[k,y] ← k | | | | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 【实验内容与记录（题号、操作步骤、数据记录与处理、附图编号、代码等）】  matlab代码   |  | | --- | | Store.m | | function C = Store(L, V, D)      n = length(L);      C = zeros(n, D);      for y = 1:D          if y >= L(1)              C(1, y) = V(1);          end      end      for k = 2:n          for y = 1:D              C(k, y) = C(k-1, y);              if y >= L(k) && (y-L(k)) > 0 && C(k-1, y-L(k)) + V(k) > C(k-1, y)                  C(k, y) = C(k-1, y-L(k)) + V(k);              end          end      end  end |  |  | | --- | | Input.m | | L = [1, 3, 4, 5]; % 货柜长度  V = [1, 4, 5, 7]; % 货柜值  D = 7;            % 库房长度  C = Store(L, V, D);  disp('最大的收益矩阵C:');  disp(C);  disp('最大的收益:');  disp(C(end, end)); |     得到了正确的结果 |

|  |
| --- |
| 【小结与讨论】  在本实验中，我们成功地使用MATLAB语言实现了一个基于动态规划的算法，解决了类似于0/1背包问题的最大化货柜价值问题。通过该实验，我们达到了以下几个目标：  1. 理解和掌握了动态规划算法的基本原理：  学会了如何构建状态转移方程，如何初始化边界条件，以及如何填充动态规划表格。  2. 熟悉了动态规划在实际问题中的应用：  通过解决货柜价值最大化问题，深刻理解了动态规划在优化问题中的优势，尤其是在处理具有重叠子问题和最优子结构的问题时。  3. 提升了MATLAB编程能力：  熟悉了MATLAB的基本语法和函数使用，掌握了如何利用MATLAB实现复杂的算法。  4.培养了问题分析与解决能力：  通过问题的分析、算法设计与实现，以及实验结果的验证，综合锻炼了分析问题和解决问题的能力。  5. 验证了算法的正确性和效率：  实验结果表明，我们所实现的动态规划算法能够正确计算在给定库房长度限制下的最大货柜价值。  在实验过程中，我们遇到了一些问题和挑战，并通过不断调试和改进算法解决了这些问题：  1. 索引超出范围的问题：  在初始实现中，我们在访问数组时遇到了索引超出范围的问题。这是由于未对索引进行有效性检查。通过添加检查条件，确保索引有效后，解决了该问题。  2.算法效率的考虑：  虽然动态规划有效地避免了重复计算，但在实际应用中，仍需关注算法的时间和空间复杂度。尤其是当货柜数量和库房长度较大时，二维数组的存储和计算可能会消耗大量资源。可以考虑通过优化空间复杂度的方法，如使用一维数组进行状态转移，进一步提高算法效率。  3. 边界条件的处理：  边界条件的正确初始化对于动态规划算法的正确性至关重要。在本实验中，我们通过合理初始化第一行数据，确保了算法能够正确进行后续的状态转移。  4. 算法扩展性：  本实验中的动态规划算法可以扩展应用到其他类似的最优化问题中，如资源分配、项目选择等。通过调整状态转移方程和边界条件，可以解决多种不同的实际问题。  通过这次实验，我们不仅掌握了动态规划的理论知识，还通过实际编程加深了对其应用的理解。这为我们在后续课程中解决更复杂的算法问题奠定了坚实的基础。 |