山东大学 计算机科学与技术 学院

数据结构与算法 课程实验报告

|  |  |  |  |
| --- | --- | --- | --- |
| 学号：201900161140 | 姓名： 张文浩 | | 班级： 19级人工智能 |
| 实验题目：递归练习 | | | |
| 实验学时：2 | | 实验日期： 9.14 | |
| 实验目的：   1. 熟悉开发工具的使用。 2. 掌握递归的实现思想。 | | | |
| 软件开发工具：  Vscode | | | |
| 1. 实验内容   **题目描述**：  现有一个有n个元素的序列a=[a1,a2,⋯,an]，定义其价值为  给出这样一个序列，求其所有排列的价值vi或  其中∣为位运算或操作，⊕为位运算异或操作。  **输入输出格式**：  **输入**：输入的第一行是一个整数n (2<=n<=10)，表示需排列的数的个数。  接下来一行是n个整数，数的范围是0到100000，每两个相邻数据间用一个空格分隔。  **输出**：  一个整数，代表所有排列价值的或。   1. 数据结构与算法描述 （整体思路描述，所需要的数据结构与算法）   第二题的关键步骤是计算全排列的过程，我的方法是考虑n个元素的全排列=（n-1个元素的全排列）+（另一个元素作为前缀）；  出口：如果递归到只有一个元素的全排列，则说明已经排完，则输出数组；不断将每个元素放作第一个元素，然后将这个元素作为前缀，并将其余元素继续全排列，等到出口，出口出去后还需要还原数组；   1. 测试结果（测试输入，测试输出）   1     1. 分析与探讨（结果分析，若存在问题，探讨解决问题的途径）   通过本次实验，我熟悉并掌握了递归算法，学会了子集和全排列问题的代码实现。  A要求的难点在于标记的设置。如果想要获得全部子集，必须考虑到所有的情况，而这一点可以采用设定布尔类型的变量来实现，0表示元不存在，1表示存在，这样即可无遗漏的获取所有的子集。除此之外，new的使用也可以节省大量的实验时间，在堆上开辟数组无需考虑数组容量，不会出现数组容量不够用的情况。  B要求中需要注意交换元素获取可能排列之后，需要将数组恢复原状，以保证之后递归函数的调用不受影响。如果数组内的数值的顺序发生了实际上的改变，可能导致后续的操作产生重复的结果，从而得到错误的全排列。   1. 附录：实现源代码（本实验的全部源程序代码，程序风格清晰易理解，有充分的注释）   }  #include <bits/stdc++.h>  using namespace std;  int ans = 0;  template <class T>  void permutation(T a[], int dep, int n)  {      if (dep == n)      {          //找到了一种排列方式          int temp = 0;          for (int i = 0; i < n; i++)          {              temp += a[i] ^ (i + 1);          }          ans |= temp;      }      else      {          for (int j = dep; j < n; j++)          {              swap(a[j], a[dep]);              permutation(a, dep + 1, n);              swap(a[j], a[dep]);          }      }  }  int main()  {      int n;      cin >> n;      int a[n];      for (int i = 0; i < n; i++)      {          cin >> a[i];      }      permutation(a, 0, n);      cout << ans << endl;      system("pause");      return 0;  } | | | |