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（第一题） | | | |
| 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]，定义这个序列的价值为。空序列的价值为0。先给你一个长度为n的序列a，求a中所有子集价值的异或和，要求子集中元素的相对位置保持不变。  异或和: 位运算的一种。如果a、b两个值不相同，则异或结果为1；如果a、b两个值相同，异或结果为0。  **输入输出格式**：  **输入**：第一行，一个整数n  接下来一行有n个非负整数：a1,a2,⋯,an  **输出**：一个整数，表示所有子集价值的异或和。   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）   1.递归，关键步骤是求子集的过程，我的方法是建立一个bool类型的visit数组，大小与输入大小一致，初始化为false，visit[[i]=1表示当前正在建立的子集中包括第i个元素，visit[i]=0表示当前正在建立的子集中不包括第i个元素。在深度小于n的时候将当前遍历到的元素有两种情况，分别是取和不取，分别取两种情况进行递归计算，考虑下一个元素的两种情况。   1. 测试结果（测试输入，测试输出）      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; //最终输出的答案  template <class T>  void sub(T a[], int n, int dep, bool visit[])  {      //对当前子集进行value计算并异或求和      if (dep == n)      {          int temp = 0;          for (int i = 0, j = 1; i < n; i++)          {              if (visit[i])              {                  temp += a[i] \* j;                  j++;              }          }          ans ^= temp;      }      else      {          visit[dep] = true;          sub(a, n, dep + 1, visit);          visit[dep] = false;          sub(a, n, dep + 1, visit);      }  }  int main()  {      int n;      cin >> n;      int a[n];      bool visit[n] = {false};//visit数组表示当前子集是否包含某一个元素，visit[i]=1表示包含，0表示不包含      for (int i = 0; i < n; i++)      {          cin >> a[i];      }      sub(a, n, 0, visit);      cout << ans << endl;      system("pause");      return 0;  } | | | |