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：201900150221 | 姓名： 张进华 | | 班级： 智能19 |
| 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：  visual studio code  g++ | | | |
| 1. 实验内容   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。   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）   从集合中挑出子集，那么对集合中的每一个元素每次有选或不选两种方案，分别对应0或是1，所有子集的数量共2^n个。具体的执行步骤是：先对集合中的第n个元素做决策（保留或抛弃），然后然后再对剩下的n-1个元素做决策，直到决策次数累计有n次（递归终止条件）   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）   这道题用递归实现了集合的子集并复习了异或的用法，题目难度还算简单   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 n;  int SumValue = 0; //输出结果  #include <bits/stdc++.h>  using namespace std;  int n;  int SumValue = 0; //输出结果  template <class T>  void GetSumValue(T Array[], int Judge[], int start, int size, int &SumValue) //ArrayJudge为标记数组，start为当前集合位置，size为集合大小  {      if (start == size)      { //所有集合的0/1标记完成          int result = 0;          int j = 0; //标记原数组位置          for (int i = 0; i < size; i++)          {              if (Judge[i] == 1)              {                  result += (j + 1) \* Array[i]; //计算集合价值                  j++;              }          }          SumValue ^= result;          //cout << "result " << result << " sumvalue " << SumValue << endl;          return;      }      Judge[start] = 0; //start位置元素不存在，生成子集      GetSumValue(Array, Judge, start + 1, size, SumValue);      Judge[start] = 1; //start位置元素存在，生成子集      GetSumValue(Array, Judge, start + 1, size, SumValue);  }  int main(void)  {      cin >> n;      int \*ArrayNumber = new int[n + 1]; //输入数组      int \*ArrayJudge = new int[n + 1];  //判断选择元素      for (int i = 0; i < n; i++)      {          cin >> ArrayNumber[i];          ArrayJudge[i] = 0;      }      GetSumValue(ArrayNumber, ArrayJudge, 0, n, SumValue);      cout << SumValue << endl;      return 0;  }  template <class T>  void GetSumValue(T Array[], int Judge[], int start, int size, int &SumValue) //ArrayJudge为标记数组，start为当前集合位置，size为集合大小  {      if (start == size)      { //所有集合的0/1标记完成          int result = 0;          int j = 0; //标记原数组位置          for (int i = 0; i < size; i++)          {              if (Judge[i] == 1)              {                  result += (j + 1) \* Array[i]; //计算集合价值                  j++;              }          }          SumValue ^= result;          //cout << "result " << result << " sumvalue " << SumValue << endl;          return;      }      Judge[start] = 0; //start位置元素不存在，生成子集      GetSumValue(Array, Judge, start + 1, size, SumValue);      Judge[start] = 1; //start位置元素存在，生成子集      GetSumValue(Array, Judge, start + 1, size, SumValue);  }  int main(void)  {      cin >> n;      int \*ArrayNumber = new int[n + 1]; //输入数组      int \*ArrayJudge = new int[n + 1];  //判断选择元素      for (int i = 0; i < n; i++)      {          cin >> ArrayNumber[i];          ArrayJudge[i] = 0;      }      GetSumValue(ArrayNumber, ArrayJudge, 0, n, SumValue);      cout << SumValue << endl;      return 0;  } | | | |