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：202300130183	姓名： 宋浩宇	班级：23 级人工智能班
实验题目：2024 数据结构-数据/智能 实验 1 递归练习		
实验学时：2	实验日期： 2024/9/11	

实验目的：

练习递归代码的编写和使用。

软件开发工具：

1. visual studio code 2022（使用 C/C++、C/C++ Extension Pack、C/C++ Themes 插件）
2. mingw64 工具包

1. 实验内容

使用递归的方式完成 sduoj 上的 2024 数据结构-数据/智能 实验 1 递归练习中 A ： 子集价值和 B ： 全排列问题两个问题。

2. 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）

对于这两道题，有一个前提条件题目未给出，通过离散数学的代数系统的知识可以证明：按位异或计算是满足结合律和交换律的，因此我们无论子集生成的顺序如何、无论全排列生成的顺序如何，问题的答案都是唯一的。

对于 A 题子集价值，该问题的难点主要为生成该集合的子集。由于该集合为一个有序序列，因此在生成子集时不需要考虑变换顺序的问题，由此有序集合的子集生成问题可以转为无序集合的子集生成问题。对于一个有 n 个元素的集合，它的子集的总数量为 2^n，用自然语言描述就是：每个元素都有取和不取两种状态，我们通过递归的方式来完成对于每一个元素取和不取两种状态的更改，第 n 层递归处理第 n 个元素是否取，我们使用一个 bool 数组来存储取的状态，true 即为取，false 即为不取，当递归到达终点，即处理完了所有的 n 个数据的取或不取，就进行一次价值的计算，由于 0 和任何数按位取异或都是那个数，因此我们可以创建一个全局变量 ans，初始为 0，每次计算出来一个子集的价值就和 ans 进行一次按位异或，并再赋值给 ans，由此当所有子集的价值都计算完毕后，我们就可以得到正确的 ans。

对于 B 题全排列问题，该问题的难点主要为如何获取全排列，我们初步的思路是仿照人类计算全排列的方式，来进行求解，具体描述为，对于第一个位置，有 n 种选择，我们可以在这 n 个数字中选取 1 个，剩下 n-1 个数字，对于第二个位置，有 n-1 种选择，我们可以在这 n-1 个数字中选取 1 个，剩下 n-2 个数字.....重复以上过程直到完成一次排列，我们不难计算出，这种方式可以获取 n！个互异的排列，因此该方式可以获取到这串数字的全排列，在实现时同样使用 bool 数组，用来表示哪些元素已经被用于排列，与问题 A 相似，我们同样可以使用初始为 0 的全局变量 ans 来储存每次按位异或的结果。但实际测试下来，这种方式由于多出了对于 bool 数组的操作，因此在时间上会比题目要求的多出较多（题目要求 50ms，这种方式最慢 76ms），但结果是对的，因此我们尝试改进算法，我们将使用 bool 数组来表示挑选,改为了以下逻辑:对于第 i 位上的数,我们分别将该位置的数字与第 i+1、i+2.....n 个数字交换，并在每次交换之后考虑第 i+1 位的数，直到考虑第 n 位的数字，假设第 i 位上的数字与第 i+m 个数字交换了，则在一轮递归返回以后，将第 i 位的数字与第 i+m 位的数字交换回来，再去将第 i 位数字与第 i+m+1 位的数字交换，以此类推，显然，这与我们之前的思路是等价的，但因为少了对于 bool 数组的操作，程序节省下了很多的时间，成功通过了本题（消耗时间最长 34ms）.

3. 测试结果（测试输入，测试输出）
对于 A 题，测试输入为：
5
1 2 3 4 5
输出为：
48
对于 B 题，测试输入为：
6
1 2 3 4 5 6
输出为：
62

4. 分析与探讨（结果分析，若存在问题，探讨解决问题的途径）
就结果来看，本题我们设计的算法在限定的条件内解决了该实验的问题。但仍然具有可以优化的空间存在。B 题相关的代码用时的问题在上文第二部分有过说明。

5. 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）

```cpp
/*2024 数据结构-数据智能 实验 1 递归练习 A 子集价值.cpp*/
#include <iostream>
using namespace std;
class Solution {
public:
    int ans = 0;
    void SubSetValue(int a[], int mark[], int n, int m)
    {
    if (m == n)
    {
        int cnt = 1;
        int sum = 0;
        for (size_t i = 0; i < n;i++)
        {
            sum = sum + cnt * a[i] * mark[i];
            cnt = cnt + mark[i];
        }
        ans = ans ^ sum;
    }
        if (m != n)
        {
            mark[m] = 0;
            SubSetValue(a,mark, n, m + 1);
            mark[m] = 1;
            SubSetValue(a,mark, n, m + 1);
        }
    }
```

```
28.       void solute()
29.       {
30.           int n;
31.           cin >> n;
32.           int a[n];
33.           for (size_t i = 0; i < n; i++)
34.           {
35.               cin >> a[i];
36.           }
37.           int mark[n];
38.           for (size_t i = 0;i < n;i++)
39.           {
40.               mark[i] = 1;
41.           }
42.           SubSetValue(a,mark, n, 0);
43.           cout << ans;
44.           }
45.   };
46.
47.   int main()
48.   {
49.       Solution ans;
50.       ans.solute();
51.       return 0;
52.   }
```

```
1.    /*2024 数据结构-数据智能 实验 1 递归练习 B 全排列问题.cpp*/
2.    #include <iostream>
3.    using namespace std;
4.    class Solution {
5.    public:
6.        int ans = 0;
7.        void swap(int &a , int &b)
8.        {
9.            int temp;
10.           temp = a;
11.           a = b;
12.           b = temp;
13.       }
14.       void SubSetValue(int a[], int n, int m)
15.       {
16.           if (m == n)
17.           {
18.               int sum = 0;
19.               for (int i = 0; i < n;i++)
```

```
            {
                sum = sum + ((i + 1) ^ a[i]);
            }
            ans = ans | sum;
        }
        if (m != n)
        {
            for (int i = m; i < n; i++)
            {
                swap(a[i],a[m]);
                SubSetValue(a,n,m + 1);
                swap(a[i] ,a[m]);
            }
            return;
        }
    }
    void solute()
    {
        int n;
        cin >> n;
        int a[n];
        for (int i = 0; i < n; i++)
        {
            cin >> a[i];
        }
        SubSetValue(a, n, 0);
        cout << ans;
    }
};

int main()
{
    Solution ans;
    ans.solute();
    return 0;
}
```