**算法分析与设计实验报告**

**第 二 次实验**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 姓名 | 李平凡 | 学号 | 201907040102 | | 班级 | 计科1905 |
| 时间 | 4.20 | 地点 | 软件大楼 | | | |
| 实验名称 | 半数集问题 | | | | | |
| 实验目的 | 1、理解分治法的概念、原理方法。 2、掌握分治法的使用规律与编程技巧 3.能够应用分治法求解日常生活中问题 | | | | | |
| 实验原理 | 输入一个整数在n的左边加一个自然数，但该自然数不能超过最近添加的数的一半。按此规则进行处理，直到不能再添加自然数为止。考虑n左边添加的数不能超过一半，那么从1到n的一半可以用循环来写出，左边继续添加数，又是各自的一半，通过树形结构绘制出来，发现原为n的半数集问题可以分解为2分之n个从1到2分之n的半数集子问题，然后将这些问题的解求和就可以得到问题的最终解。 | | | | | |
| 实验步骤 | ① 输入整数n，求得n的半数集问题的解就是从1到2分之n的半数集问题的解的和  ② 通过循环遍历从1到2分之n各个整数的半数集的解，通过累加得到结果  ③ 最后输出结果 | | | | | |
| 关键代码 | int banshu(int n)  {  int sum=0;  if(n==1)  {  return 1;  }  else  {  int x=n/2;  for(int i=1;i<=x;i++)  {  sum=banshu(i)+sum;  }  }  return sum+1;  }  这个代码首先当n==1是跳出递归的条件，否则的话就要从1到2分之n分别求出各个子问题的解最终求和得到结果  #include<iostream>  using namespace std;  int main()  {  int a[100];  a[0]=1;  a[1]=1;  int n;  cin>>n;  for(int i=2;i<=n;i++)  {  if(i%2==1)a[i]=a[i-1];  else  {  a[i]=a[i/2]+a[i-2];  }  }  cout<<a[n];  }  优化代码，时间开销只有O(n) | | | | | |
| 测试结果 | 半数集问题数据规模增加时间开销 | | | | | |
| 实验心得 | 半数集问题采用分治法可以很轻松的讲一个大问题化解为小问题然后通过递归返回没一个小问题的结果最后通过汇总得到最终结果，不难发现这个算法的时间开销只和半数集里面的整数有关，里面的整数越大调用的循环和递归次数越多，所以也就相当于最后求解的结果n也就是说时间开销就是最后的结果set(n)，但经过发现这个算法可以将时间复杂度降到O（n）因为这个算法的思想就是如果你要计算set（n）那么就从1到2分之n一直算各自的半数集然后求和，但是如果我将1到2分之n前面一个数看成一个集合，那么他就是set（n-2），并且我们还可已发现如果n为奇数它的半数集就是和n-1的半数集个数相同，那么我们最后可以得到递推公式如代码所示，从而达到时间开销只有n，最后的时间复杂度就是O(n)。 | | | | | |
| 实验得分 |  | 助教签名 | |  | | |

**附录：完整代码**