南京邮电大学

实 验 报 告

（2024 / 2025 学年 第 一 学期）

课程名称	离散数学
实验名称	偏序关系中盖住关系的求取及格论中有补格的判定
实验时间	2024 年 11 月 13 日
指导单位	计算机学院计算机科学与技术系
指导教师	柯昌博

学生姓名	于明宏	班级学号	B23041011
学院(系)	计算机学院	专　　业	信息安全

实 验 报 告

实验名称	偏序关系中盖住关系的求取及格论中有补格的判定			指导教师	柯昌博
实验类型	验证	实验学时	4	实验时间	2024.11.13

一、 实验目的和要求

为了进一步理解偏序关系中盖住关系及格论中有补格的判定，具体的实验要求如下：

输入：给定具体的偏序关系，如：“整除关系”，根据盖住关系的定义求取对应的盖住关系，以及判定是否为有补格；

输出：盖住关系和有补格的判定结果。

二、实验环境(实验设备)

硬件：微型计算机

软件：Windows 操作系统、Microsoft Visual C++ 2022

三、实验原理及内容

1、这个程序实现了一个用于分析偏序关系中整除关系的工具。在输入该整数后，判断盖住关系和有补格的判定结果并进行输出。

2、C++源代码：

```
#include <iostream>  // Include the iostream library for input/output operations
#include <vector>    // Include the vector library to use the vector container
#include <algorithm> // Include the algorithm library for functions like find

using namespace std; // Use the standard namespace

// Function to check if a number is prime
bool Is(long long num) {
    if (num <= 1) // Numbers less than or equal to 1 are not prime
        return false;
    for (long long i = 2; i * i <= num; i++) { // Loop from 2 to the square root of num
        if (num % i == 0) // If num is divisible by i, it's not prime
            return false;
    }
    return true; // If no divisors were found, the number is prime
}

// Function to compute the greatest common divisor (GCD) of two numbers
long long gcd(long long a, long long b) {
    while (b != 0) { // Repeat until b becomes 0
        long long temp = b; // Store b in a temporary variable
        b = a % b;          // Update b with the remainder of a divided by b
        a = temp;             // Update a with the value of b from the previous iteration
    }
    return a; // When b is 0, a is the GCD
}

// Function to compute the least common multiple (LCM) of two numbers
long long lcm(long long a, long long b) {
```

```
        return a * b / gcd(a, b); // LCM is calculated as (a * b) / GCD(a, b)
    }

    int main() {
        long long num = 0, n;                    // Declare variables for input number and a copy of the input
        bool isLattice = true;                   // Boolean flag to check if the number forms a lattice with complements
        vector<long long> arr, arrab, son, none; // Vectors to store prime factors, distinct prime factors, divisors, and non-complement elements

        cout << "Please enter the number: ";   // Prompt the user to enter a number
        cin >> num;                            // Read the input number
        n = num;                               // Store a copy of the input in n

        if (Is(num)) {                         // Check if the input number is prime
            cout << "<1," << num << ">\nIt is a lattice with complements\n";
            return 0;                          // If prime, output that it's a lattice with complements and exit
        }

        for (long long i = 1; i <= num; i++) { // Loop through all numbers from 1 to num
            if (num % i == 0)                  // If i is a divisor of num
                son.push_back(i);              // Add i to the list of divisors
        }

        // Factorize the number into its prime factors and store in arr and arrab
        while (!Is(num)) {                       // Repeat until num becomes a prime number
            for (long long i = 2;; i++) {        // Loop to find the smallest prime factor of num
                if (!Is(i)) continue;            // Skip if i is not prime
                if (num % i == 0) {              // If i is a prime factor of num
                    num /= i;                    // Divide num by i
```

```
                    arr.push_back(i);          // Add i to the list of prime factors
                    if (arrab.empty() || arrab.back() != i) // If arrab is empty or
the last element is not i
                        arrab.push_back(i);    // Add i to the list of distinct prime
factors
                    break;                       // Break to reset the loop with the
new num
                }
            }
        }
        arr.push_back(num);                       // Add the final prime factor (num
itself) to arr
        if (arrab.back() != num)                // If the last element in arrab is not
num
            arrab.push_back(num);               // Add num to arrab as a distinct prime
factor

        cout << "\nPartial order relations:\n"; // Output label for partial order
relations
        for (auto p = son.begin(); p != son.end(); p++) { // Loop through each divisor
in son
            bool hasComplement = false;        // Flag to check if a complement exists
for the current divisor
            for (auto j = son.begin(); j != son.end(); j++) { // Loop through each
divisor in son again
                if ((find(arrab.begin(), arrab.end(), *j / *p) != arrab.end()) && (*j %
*p == 0) && (j != p)) {
                    cout << "<" << *p << "," << *j << "> "; // Print partial order
relation if the condition holds
                }
                else if (gcd(*p, *j) == 1 && lcm(*p, *j) == n) { // Check if p and j
are complements
                    hasComplement = true;      // Set the flag to true if a complement
is found
                }
```

```
            }
            if (!hasComplement) {              // If no complement was found for p
                none.push_back(*p);               // Add p to the list of elements without complements
                isLattice = false;              // Set isLattice to false as not all elements have complements
                if (*p != n)                      // If p is not equal to the input number
                    cout << "(" << *p << " has no complement)"; // Output that p has no complement
            }
            cout << endl;                         // Print a newline after each set of relations
        }

        if (!isLattice) {                     // If not all elements have complements
            cout << "It is not a lattice with complements."; // Output that it is not a lattice with complements
        }
        else {
            cout << "It is a lattice with complements.\n"; // Output that it is a lattice with complements
        }

        return 0;                             // End the program
    }
```

3、运行结果：

```
Please enter the number: 100

Partial order relations:
<1,2> <1,5>
<2,4> <2,10> (2 has no complement)
<4,20>
<5,10> <5,25> (5 has no complement)
<10,20> <10,50> (10 has no complement)
```

<20,100> (20 has no complement)

<25,50>

<50,100> (50 has no complement)

It is not a lattice with complements.

四、实验小结（包括问题和解决方法、心得体会、意见与建议等）

说明：这部分内容主要包括：在编程、调试或测试过程中遇到的问题及解决方法、本次实验的心得体会、进一步改进的设想等。

（一）实验中遇到的主要问题及解决方法

1. 问题：在实现盖住关系的求取时，由于对偏序关系的定义理解不够深入，导致部分关系判定错误。

解决方法：通过查阅教材和相关资料，重新梳理了偏序关系和盖住关系的概念，结合具体实例加深了理解，从而确保代码逻辑的正确性。

2. 问题：在进行有补格的判定时，未考虑所有可能的补元素组合，导致判定结果不准确。

解决方法：调整算法结构，增加对所有补元素的检验，确保判断过程的完整性，并通过多组测试验证了代码的正确性。

3. 问题：在实现因数分解时，因对算法效率的考虑不周，导致程序运行时间较长。

解决方法：优化因数分解算法，引入素数筛选方法，从而减少了不必要的运算，提升了程序效率。

（二）实验心得

通过本次实验，我对偏序关系和格论中的有补格的概念有了更深入的理解。实验过程中，编程实现和调试的过程提升了我分析问题的能力，并增强了我在处理复杂关系时的严谨性。尤其是通过代码实现盖住关系和有补格的判定，让我更加体会到算法设计和数学理论的结合之处，进一步拓展了我对离散数学的应用理解。

（三）意见与建议（没有可省略）

可以提供更多的时间上机操作，以确保更多程序设计思路得以实现，提升掌握程度和编程能力。

五、支撑毕业要求指标点

支撑毕业要求的指标点为：

☐ 1-4 掌握计算机科学与技术领域的专业知识，能将专业知识用于分析和解决计算机领域复杂工程问题。

☑ 2-1 能够应用数学、自然科学和工程科学的基本知识，识别和分析计算机领域复杂工程问题的特征。

六、指导教师评语 (含学生能力达成度的评价)					
成　绩		批阅人		日　期	

如果不太想写太多字，“指导教师评语”也可以设计为如下的各选择项用打勾形式（仅仅作为一个简单示例，请各课程负责人根据课程和实验情况以及支撑的指标点来自行设定选择项，同一门课程的不同实验评分细则项允许存在不同）：

	评分项	优秀	良好	中等	合格	不合格
评	遵守实验室规章制度					
	学习态度					
	算法思想准备情况					
	程序设计能力					
分	解决问题能力					
	课题功能实现情况					
	算法设计合理性					
	算法效能评价					
	回答问题准确度					
细	报告书写认真程度					
	内容详实程度					
	文字表达熟练程度					
	其它评价意见					
则	本次实验能力达成评价（总成绩）					