Assignment #10

Rikiya Takehi
1W21CS04-6
Algorithms and Data Structures
14/06/2022

Problem Statement

Write Euclid algorithm and Tower of Hanoi algorithm in java programming. Euclid algorithm, in this assignment, should be done using recursion. The goal is to find the GCD of two integer values and GCD of all elements in an integer array. For the Tower of Hanoi algorithm, it is required to show the sequence of actions and the total number of steps.

Programs

Euclid algorithm (GCD for two integer values)

```java
import java.util.Scanner;

public class EuclidGCD{
    static int gcd(int x, int y){
        if(y==0){
            return x;
        }else{
            return gcd(y, x % y);
        }
    }

    public static void main(String[] args){
        Scanner stdIn = new Scanner(System.in);
        System.out.println("GCD of two integers");
        System.out.print("Input Integer:");
        int x = stdIn.nextInt();
        System.out.print("Input Integer:");
        int y = stdIn.nextInt();
        System.out.println("GCD :" + gcd(x, y));
    }
}
```

Program explanation:

GCD is short for greatest common divisor. This algorithm is a program that aims to find the greatest common divisor of x and y. In the code, x is a larger number than y.

In lines 4~9, if y is not zero, it replaces y with the remainder of the division. If the remainder is zero, the original “y” is the greatest common divisor. If the remainder is not zero, x is replaced with the original y, and the same sequence is taken until the remainder becomes zero. The recursive function calls itself again and again. The recursive function sorts a smaller part of the array and calls on itself. On the other hand, the non-recursive function does not call itself repeatedly. An example of the non-recursive function is the insertion sort.

Euclid algorithm (all elements in integer array)

```java
import java.util.Scanner;

public class EuclidGCDArray{
    static int gcd(int x, int y){
        if(y==0){
            return x;
        }else{
            return gcd(y, x % y);
        }
    }

    static int gcdArray(int a[], int start, int no){
        if (no==1){
            return a[start];
        }else if (no==2){
            return gcd(a[start], a[start+1]);
        }else{
            return gcd(a[start], gcdArray(a, start+1, no-1));
        }
    }

    public static void main(String[] args){
        Scanner stdIn = new Scanner(System.in);
        System.out.print("How many integers:");
        int num = stdIn.nextInt();
        int[] x = new int[num];
        for (int i = 0; i < num; i++) {
            System.out.print("x[" + i + "]:");
            x[i] = stdIn.nextInt();
        }
        System.out.println("GCD is " + gcdArray(x, 0, num));
    }
}
```

Program Explanation:

This code is able to find GCD of several numbers of values in an array. Lines 4-9 does the exact same sequences as the GCD for two integer values. In lines 13 to 19, it takes two integers from the array, and uses the “gcd” procedure in lines 4-9 in order to find the GCD of the two integers. Then, it finds the GCD of the GCD of the first two integers with the next integer in the array. This procedure keeps on going until the end of the array.

Tower of Hanoi:

```java
import java.util.Scanner;

public class Hanoi{
    static int move(int no, int x, int y){
        int count=0;
        if(no>1)
            count=count+move(no-1, x, 6-x-y);
        System.out.println("move disc["+no+"] from "+x+"to"+y+"");
        count++;
        if(no>1)
            count=count+move(no-1, 6-x-y, y);
        return count;
    }

    public static void main(String[] args){
        Scanner stdIn = new Scanner(System.in);
        System.out.println("Tower of Hanoi");
        System.out.print("The number of disc:");
        int n = stdIn.nextInt();
        int num=move(n, 1, 3);
        System.out.println(""+num+" steps are required");
    }
}
```

Program explanation

In this program, the tower of discs are replaced to another place without changing its order, using three points (source, destination, aux). In line 4, “no” is the number of discs, “x” is the source number, and “y” is the destination number. In lines 5-11, it first moves (n-1) numbers of discs to aux from source. Then, it moves the n^{th} disc (last disc) from source to destination. Last, it moves (n-1) numbers of discs to destination.

```java
        int count=0;
        if(no>1)
            count=count+move(no-1, x, 6-x-y);
        System.out.println("move disc["+no+"] from "+x+"to"+y+"");
        count++;
        if(no>1)
            count=count+move(no-1, 6-x-y, y);
```

This sequence is done unless “no” is not bigger than 1.

Experimental settings

Input files:

None. The algorithms use the data from standard input.

Tested programs:

“EuclidGCD”, “EuclidGCDArray”, “Hanoi”

Results

EuclidGCD: (*the values are random numbers that I have thought of)

Input integers: 16, 8

```
GCD of two integers
Input Integer:16
Input Integer:8
GCD :8
```

Input integers: 1341, 345

```
GCD of two integers
Input Integer:1341
Input Integer:345
GCD :3
```

EuclidGCDArray:

```
How many integers:5
x[0]:142
x[1]:1324
x[2]:26534
x[3]:566
x[4]:134
GCD is 2
```

Tower of Hanoi:

n=3:

```
Tower of Hanoi
The number of disc:3
move disc[1] from 1to3
move disc[2] from 1to2
move disc[1] from 3to2
move disc[3] from 1to3
move disc[1] from 2to1
move disc[2] from 2to3
move disc[1] from 1to3
7 steps are required
```

n=4:

```
The number of disc:4
move disc[1] from 1to2
move disc[2] from 1to3
move disc[1] from 2to3
move disc[3] from 1to2
move disc[1] from 3to1
move disc[2] from 3to2
move disc[1] from 1to2
move disc[4] from 1to3
move disc[1] from 2to3
move disc[2] from 2to1
move disc[1] from 3to1
move disc[3] from 2to3
move disc[1] from 1to2
move disc[2] from 1to3
move disc[1] from 2to3
15 steps are required
```

Discussion of the Results

EuclidGCD:

The EuclidGCD program always functions briefly because any positive integer has a smallest element. When you suppose that you try to figure out the GCD of integer a and b, the following formula stands:

$$a = bq + r$$

Letting q be quotient and r be remainder. Here, any number that divides a and b would also divide r, so it also divides r and b; this can also be said vise versa. This implies that the GCD of a and b is also the GCD of b and r. Therefore, as you carry on changing the integers, the remainder eventually becomes zero, and so GCD is found using Euclid way.

EuclidGCDArray:

This way is like the combination of sorting and GCD, since it tries to find the GCD of the integers in order of the array. Therefore, it may be possible to introduce another order of comparing GCD. For example, it may be possible to find the GCD from the left end and the right end of the array at the same time in order to make the calculation go faster.

Hanoi:

The tower of Hanoi needs 7 steps for n=3 and 15 steps for n=4. Other than this, it needs 31 steps for n=5 and 3 steps for n=2. This implies that as n increases, the number of steps doubles and adds one to it. When letting S be the number of steps, the following formula would come up.

$$S_n = S_{n-1} + 1$$

This is because it needs the same number of steps as n-1 in order to take n-1 discs from source to aux. It also needs the same amount of steps to take n-1 discs from aux to destination. Then, it needs to add a step because it needs to take the n^{th} disc to destination.