General Solution of Hanoi Tower

Edit

Problem statement:

Finding the shortest step of Hanoi Tower problem for Hanoi(tower, disk) and tower>=3, disk>0

Java

```java
/*package whatever //do not write package name here */
import java.util.*;
class hanoiCount {
    static int exetime = 0;
    public static int hanoi(int mem[][], int t, int d)
    {
        int tmp = 0, stepCount = 0;
        if (d < t) {
            return 2 * d - 1;
        }
        else if (t == 3) {
            return (int)Math.pow(2, d) - 1;
            //查表
        }
        else if (mem[t][d] != 0) {
            return mem[t][d];
        }
        else if (t == 4) {
            // count: a1 ~ an; Power2: 2的次方
            int disk = d - 3, i = 3, count = 5, Power2 = 4;
            while (disk - i >= 0) {
                exetime++;
                disk -= i;
                count += Power2 * i;
                i++;
                Power2 *= 2;
            }
            count += disk * Power2;
            mem[t][d] = count;
            return count;
        }
        else {
            for (int mid = d - 1; mid >= t - 2; mid--) {
                //計算執行幾次(時間複雜度估算)
                exetime++;
                if (mid == d - 1) {
```

```
                    stepCount = hanoi(mem, t, d - mid) * 2
                              + hanoi(mem, t - 1, mid);
                }
                else {
                    tmp = hanoi(mem, t, d - mid) * 2
                          + hanoi(mem, t - 1, mid);
                    if (tmp < stepCount) {
                        stepCount = tmp;
                    }
                }
            }
            mem[t][d] = stepCount;
            return stepCount;
        }
    }
    public static void main(String args[])
    {
        Scanner input = new Scanner(System.in);
        //柱
        int tower = input.nextInt();
        //碟子
        int disk = input.nextInt();

        //記錄算過的值以便之後查表
        int[][] mem = new int[tower + 1][disk + 1];

        System.out.println(tower + " tower " + disk
                           + " disk: "
                           + hanoi(mem, tower, disk));
        System.out.println("execution time: " + exetime);
        System.out.println("The hanoi table: ");
        // just to print out a part of table
        for (int i = 1; i < tower; i++) {
            for (int j = 30; j < disk; j++) {
                System.out.print(mem[i][j] + " ");
            }
            System.out.println();
        }
    }
}
```

This program is to count the shortest steps of moving disks from source(a peg) to destination(another peg) in Hanoi Tower.

e discrete mathematics class of the cram school, based on recursive solution, I came up with this idea. Though, this ithm has been discovered and written on Wikipedia.

Practice link

https://ide.geeksforgeeks.org/online-java-compiler/92cdd033-53cd-4e37-bd83-deb678c077c4

https://ide.geeksforgeeks.org/online-java-compiler/aec43be6-b825-4b06-b181-0a2698d6cd7f

https://ide.geeksforgeeks.org/online-java-compiler/a88b9479-2dc8-4c68-ae41-77861604b8c4

patterns

> *1. Hanoi(tower,disk) = 2n-1 (odd)*
>
> *2. Hanoi(m, n) : tower=m , disk=n*
>
> *Hanoi(m, m-1) has unique solution*
>
> *3. Hanoi(2,1) = 1*
>
> *Hanoi(m, n)= min{ (k,n) + (m-k, n-1) + (k, m) }*
>
> *for (m>=3) & (n>m-1) & (n-1>=m-k-1)*

Example

> *Hanoi(3,2)= (2,1) + (2,1) + (2,1)*
>
> *Hanoi(5,5) = min{ (5,1)+(4,4)+(5,1) , (5,2)+(4,3)+(5,2) }*
>
> *//(5,3)+(4,2)+(5,3) is invalid cause Hanoi(4,2) = Hanoi(3,2) and there is a disk idle*
>
> *Hanoi(4,8) = min{ (4,6)+(3,2)+(4,6),*
>
> *(4,5)+(3,3)+(4,5),*
>
> *(4,4)+(3,4)+(4,4),*
>
> *(4,3)+(3,5)+(4,3),*
>
> *(4,2)+(3,6)+(4,2),*
>
> *(4,1)+(3,7)+(4,1) }*

But I think there still some way to improve it,since:

> *Assume Hanoi(m,n):*

[Hanoi(m,k)+Hanoi(m-1,n-k)+Hanoi(m,k)] > [Hanoi(m,k+1)+Hanoi(m-1,n-k-1)+Hanoi(m,k+1)]

*=> [2*Hanoi(m,k)+Hanoi(m-1,n-k)] > [2*Hanoi(m,k+1)+Hanoi(m-1,n-k-1)]*

We already know that:

1. The more tower we use the less step we need

2. The slope :

[Hanoi(m-1,n-k)-Hanoi(m-1,n-k-1)] / 2[Hanoi(m,k+1)- Hanoi(m,k)] > 1*

this mean the growth rate:

2[Hanoi(m,k+1)- Hanoi(m,k)] < [Hanoi(m-1,n-k)-Hanoi(m-1,n-k-1)]*

So, there must exist t for (k > t) that the Hanoi(m,n) will strictly decreasing.

If we could find that {t for (k > t)} in each case, we can reduce the times of comparisons for finding Hanoi(m,n)

The patterns of Hanoi(tower, disk)

Java

```java
/*package whatever //do not write package name here */
import java.util.Scanner;
class hanoiConclusion {
    static int exetime = 0;
    public static int hanoi(int d, int t) {
        if(d < t){
            return 2*d-1;

        }else if(t == 3) {
            return (int)Math.pow(2, d)-1;
        }else {
            int exp=4, step=hanoi(t-1, t), numerator=t-1, coef=(t-1)*(t-2)/2,
                    denominator=2;
            d -= t-1;
            while(d-coef >= 0) {
                exetime++;
                //just to check disk and exp count
                System.out.println("disk:"+d+" exp:"+exp);
```

```
                d -= coef;
                step += exp*coef;
                exp *= 2;
                /*
                4!/2!2! ; 5!/3!2! 5/3
                5!/3!2! ; 6!/4!2! 6/4
                6!/4!2! ; 7!/5!2! 7/5
                */
                numerator++;
                denominator++;
                coef = coef*numerator/denominator;

            }
            step += d*exp;
            return step;
        }
    }

    public static void main(String args[]) {
        Scanner input = new Scanner(System.in);
        //柱
        int tower = input.nextInt();
        //碟子
        int disk = input.nextInt();

        System.out.println(tower + " tower " + disk + " disk: "
                + hanoi(disk,tower));
        System.out.println("execution time: " + exetime);
    }
}
```

This program is the solution of first program's algorithm. I print out the results of first program and try to find the patterns in this program.

Example

consider hanoi(4,n) for n >= 3:

n is number of disk & 4 is number of Tower

a1= hanoi(4,3)=5

a2= hanoi(4,4)=a1+ 4

*a3= hanoi(4,5)=a1+ 4*2*

*a4= hanoi(4,6)=a1+ 4*3*

*a5= hanoi(4,7)=a4+ 8*1*

*a6= hanoi(4,8)=a4+ 8*2*

*a7= hanoi(4,9)=a4+ 8*3*

*a8= hanoi(4,10)=a4+ 8*4*

*a9= hanoi(4,11)=a8+ 16*1*

.

.

.

conclusion:

*a1= 5*1*

*a2= 5*1 +4*1*

*a5= 5*1 +4*3 +8*1*

*a9= 5*1 +4*3 +8*4 + 16*

*a14= 5*1 +4*3 +8*4 + 16*5 +32*

*a20= 5*1 +4*3 +8*4 + 16*5 +32*6 +64*

----

consider hanoi(5,n) for n >= 4:

n is number of disk & 5 is number of Tower

a1= hanoi(5,4)=7

a2= a1+ 4

*a3= a1+ 4*2*

*a4= a1+ 4*3*

...

*a7= a1+ 4*6*

*a8= a7+ 8*1*

*a9= a7+ 8*2*

...

*a17= a7+ 8*10*

*a18=a17+16*1*

.

.

.

*a32=a17+16*15*

conclusion:

*a1= 7*1*

*a2= 7*1 +4*1*

*a8= 7*1 +4*(1+2+3) +8*1*

*a18= 7*1 +4*6 +8*(6+4) + 16*1*

*a33= 5*1 +4*6 +8*(6+4) + 16*(6+4+5) +32*

*a54= 5*1 +4*6 +8*(6+4) + 16*(6+4+5) +32*(6+4+5+6) +64*

*a82= 5*1 +4*6 +8*(6+4) + 16*(6+4+5) +32*(6+4+5+6) +64*(6+4+5+6+7) +128...*

*= 5*1 +4*(3+hanoiAcc(2,4)) +8*(hanoiAcc(2,5)+hanoiAcc(3,4)) +16*(hanoiAcc(3,5)+hanoiAcc(4,4)) +...+128* (hanoiAcc(6,5)+hanoiAcc(7,4))*

hanoiAcc(n,m): n is exponent of 2, m is number of towers

---

consider hanoi(6,n) for n >= 5:

n is number of disk & 6 is number of Tower

a1 a2 a11 a31 a66 a150 a270

*9 +4*10 +8*20 +16*35 +32*56 +64*84 + 128*120*

=9 +4(1+2+3+4) +8*(10+4*5/2) +16*(20+5*6/2) +32*(35+6*7/2) +64*(56+7*8/2) +128*(84+8*9/2)*

= 9 +4(4+hanoiAcc(2,5)) +8*(hanoiAcc(2,6)+hanoiAcc(3,5)) +16*(hanoiAcc(3,6)+hanoiAcc(4,5)) +...+128* (hanoiAcc(6,6)+hanoiAcc(7,5))*

---

consider hanoi(7,n) for n >= 6:

n is number of disk & 7 is number of Tower

a1 a2 a17 a52 a122 a248

11 +4(1+2+3+4+5) +8*(15+20) +16*(35+35) +32*(70+56) +64*(126+84) +...*

= 11 +4(5+hanoiAcc(2,6)) +8*(hanoiAcc(2,7)+hanoiAcc(3,6)) +16*(hanoiAcc(3,7)+hanoiAcc(4,6)) +...+128* (hanoiAcc(6,7)+hanoiAcc(7,6))*

---

consider hanoi(m,n) for n >= m-1:

m is number of Tower & n is number of disk

*(m-1)*2-1*

+(2^2)((m-2)*(m-1)/2)*

+(2^3)((m-2)*(m-1)/2+hanoiAcc(3,m-1))*

+(2^4)(hanoiAcc(3,m)+hanoiAcc(4,m-1))*

+...+(2^t)(hanoiAcc(t-1,m)+hanoiAcc(t,m-1))*

hanoiAcc(a,b) = hanoiAcc(a-1,b) + hanoiAcc(a,b-1)

---

beautiful number, coeffecient of exponent of 2

tower = 3: 1.....1.....1.....1.....1.....1.....1.....1.....1

tower = 4: 3.....4.....5.....6.....7.....8.....9.....10

tower = 5: 6.....10.....15.....21.....28.....36.....45

tower = 6: 10.....20.....35.....56.....84.....120

tower = 7: 15.....35.....70.....126....210

tower = 8: 21.....56.....126....252

tower = 9: 28.....84.....210

tower =10: 36....120

tower =11: 45

(/) -> Pascal triangle{(1,3,1),(1,4,6,4,1),(1,5,10,10,5,1)...} -> combination

Example:

*hanoi(40,5) = 7 + 2^2*C(4,2) + 2^3*C(5,2) + 2^4*C(6,2) + 2^5*(40-6-10-15)*

= 1 + 2^1(5-2)/1 + 2^2*(5-2)*(5-1)/(1*2) + 2^3*(5-2)*(5-1)*5/(1*2*3) + 2^5*(5-2)*(5-1)*5*(5+1)/(1*2*3*4) + 2^5*(40-6-10-15)*

*hanoi(200,7) = 11 + 2^2*C(6,4) + 2^3*C(7,4) + 2^4*C(8,4) + 2^5*(200-15-35-70)*

Practice link

https://ide.geeksforgeeks.org/online-java-compiler/6f9af76b-a0ef-4303-8cad-d608a88a47f1

https://ide.geeksforgeeks.org/online-java-compiler/15edc6bb-69ab-4779-8153-187b5c10e680

https://ide.geeksforgeeks.org/online-java-compiler/b16bfb71-9ba4-4cc0-a757-28054b5edf63

If you want to know more, please visit my Github below

https://github.com/murasaki-saya/HanoiTower/tree/main