Homework3 – Hao Wu 001443861

1. The steps of recursive stack operation:
2. factorial (6): push factorial(6) = 6\*factorial(5);
3. push factorial (5) = 5\* factorial(4)
4. push factorial (4) = 4\* factorial(3)
5. push factorial (3) = 3\*factorial (2)
6. push factorial (2) = 2 \* factorial(1)
7. push factorial(1) =1 ;
8. pop() \*= count ,count = 1;
9. pop()\*=count ,count = 2;
10. pop() \*= count, count =6;
11. pop() \*=count ,count = 24;
12. pop() \*=count, count = 120;
13. pop() \* =count ,count = 720;
14. stack is empty, return count;

Because every recursive operation with only one calculation, so the time complexity of N! is O(n)

1. The step of recursive stack operation of Fibonacci:
2. stack.push(Fibonacii(5))
3. stack.pop() = Fibonacii(5), = Fibonacii(4)+F(3),push(F(4)),push(F(3))
4. stack.pop() = F(4) = F(3)+F(2), F(3)=F(2)+F(1)
5. stack.push(F(3)), because F(2)=F(1)=1,so count=F(2)+F(2)+F(1) = 3
6. stack.pop() = F(3) = F(2)+F(1) = 2 count = 3+2 =5 return 5.
7. The step of Iterative of Fibonacci:
8. Given n = 5,define int pre = 1, int cur =1,int res =0;
9. For loops:

While n>2:

res = pre+ cur

pre = cur

cur = res

n=n-1;

when loop ends, return res;

1. Hanoi:

Implement a function that move n disks from a to c and helped by b:

move(5,a,b,c),

if I need move 5 disks from a to c ,I need put top 4 disks from a to b for transferring.

And move the bottom disks from a to c and move those 4 disks from b to c. So there is recursive function :

n=5

move(n,a,b,c){

if(n==1) System.out.println(“disks” n “from” a “to” c)

move(n-1,a,c,b);

move(n-1,b,a,c)

It means that :I’ll move 5 disks, because I need another tower for transferring, so the first I’will move 4disks from a to b through c, and when it finish then move this 4disks from b to c through a.

And when move 4 disks ,I move top of 3disks in the same way. So there is recursive function. The time complexity is O(2^n).

The Java program is attached in to file.

1. No description
2. The grey code creation is here:
3. when n ==1 , the first one is 0, second is 1; have String temp1[2] store it
4. n ==2, find String temp2[4],
   1. temp2[0] = “0” +temp[0];
   2. temp2[1] = “0” +temp[1];

and here when we finish the code started from 0 , we reverse the order of reading last String array,

* 1. temp2[2] =”1”+temp[1]
  2. temp2[3] = “1”+temp[0]

finally we get the Grey code when n ==2: { 00, 01, 11 ,10}

1. when n ==3 get the same way create the Grey code from the result of n=2,

when it starts from 0, by order:

000 , 001,011,010,

and when it starts from “1”, reverse the order:

110,111,101,100;

Finally we get the Grey code of size 3:{000, 001,011,010,110,111,101, 100}

1. so n==4, we can create Grey code from the size of 3:

begins with “0”,

0000, 0001, 0011,0010,0110,0111,0101,0100,

begins with “1”, reverse the order:

1100,1101,1111,1110,1010,1011,1001,1000.

So here is the Grey code when size is 4{

0000,0001,0011,0010,0110,0111,0101,0100,

1100,1101,1111,1110,1010,1001,1001,1000}