Tower of Hanoi河內塔

Tower of Hanoi 河內塔是一個數學遊戲或謎題，由三個桿和許多不同直徑的圓盤組成，這些圓盤可以滑到任何桿上。拼圖從將圓盤按尺寸遞減順序堆疊在一根桿上開始，最小的在頂部，因此近似圓錐形。謎題的目標是將整個堆棧移動到最後一根桿，遵守以下規則：

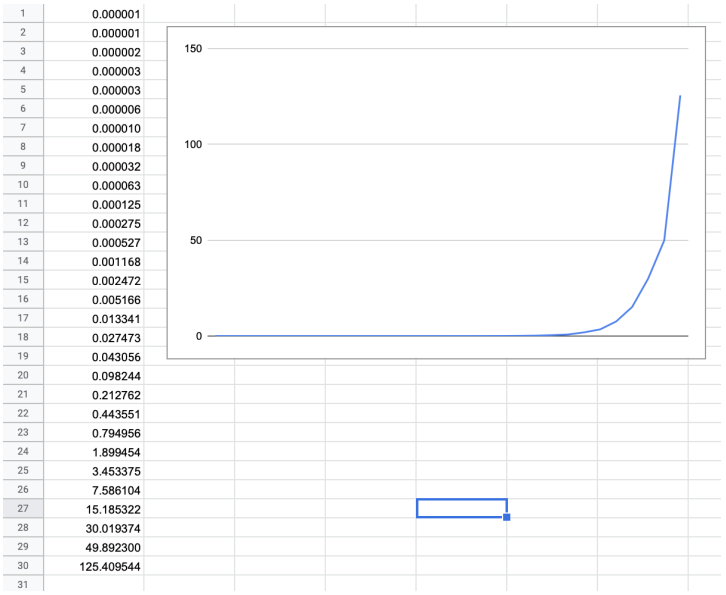
1、一次只能移動一個磁盤。
2，每次移動都包括從其中一個堆棧中取出上盤並將其放置在另一個堆棧的頂部或空桿上。
3、任何磁盤都不能放在比它小的磁盤上。

解決漢諾塔謎題所需的最小移動次數為 2^n - 1，其中 n 是磁盤的數量。

Tower of Hanoi can be solved by using recursion , since the way for solving any number (N) of disk is by , first move disk 0 to n-1 to the auxiliary rod then move the disk n to the target rod , which means we need to move the disk 0 to n-2 to the target rod and the n-1 to the auxiliary rod , so on and so forth until it's disk 1 .

therefore the way we solve Tower of Hanoi is to first set the way to solve N = 1 , then N = 2, N = 3 and when N = 3 , the pattern is already formed and by oscillating the target rod and the auxiliary rod , Tower of Hanoi is solved .

i run all my programs on online GDB , to maximize the efficiency it didn't print out the process . The average time used for 16 disks is 4 millisecond , the max amount of disk in 1 second is 23 disks witch is shown bellow .



This is the graph of the relationship between disk amount and time needed . ***On school's VM ,it takes 5 milliseconds to run 16 disks .***

here is the code i wrote to slove Tower of Hanoi : https://onlinegdb.com/VrnV0RywU

```
#include <stdio.h>
#include <time.h>

void show_pole(int n,int arr[],int amount){
    printf("%d:",n);
    for(int i=0;i<amount;i++){
        if(arr[i]<10){
            printf("0%d,",arr[i]);
        }else{
            printf("%d,",arr[i]);
        }
    }
    printf("\n");
    return ;
}
void show_all_pole(int arr[], int arr2[], int arr3[],int amount){
    show_pole(1,arr,amount);
    show_pole(2,arr2,amount);
    show_pole(3,arr3,amount);
    printf("\n\n\n");
    return ;
}

int get_top_disk(int arr[],int len){
    for(int i=0;i<len;i++){
        if(arr[i]!=0){
            return i;
        }
    }
    return len-1;
}

void Tower_of_Hanoi(int n, int arr[], int arr3[], int arr2[] ,int len){
    if (n == 1){
        int top_disk_index1 = get_top_disk(arr,len);
        int top_disk_index2 = get_top_disk(arr3,len);
        if(arr3[top_disk_index2]==0){
            arr3[top_disk_index2] = arr[top_disk_index1];
        }else{
            arr3[top_disk_index2-1] = arr[top_disk_index1];
        }
        arr[top_disk_index1] = 0;
        return;
    }
    Tower_of_Hanoi(n-1, arr, arr2, arr3,len);
    int top_disk_index1 = get_top_disk(arr,len);
    int top_disk_index2 = get_top_disk(arr3,len);
    if(arr3[top_disk_index2]==0){
        arr3[top_disk_index2] = arr[top_disk_index1];
    }else{
        arr3[top_disk_index2-1] = arr[top_disk_index1];
    }
    arr[top_disk_index1] = 0;
    Tower_of_Hanoi(n-1, arr2, arr3, arr,len);
    return;
}
```

```
int main(){
    int amount = 24;
    printf("amount:");
    scanf("%d",&amount);
    int ploe1[amount];
    int ploe2[amount];
    int ploe3[amount];
    char buff[11];
    char buff2[11];

    for(int i=0;i<amount;i++){
        ploe1[i] = i+1;
        ploe2[i] = 0;
        ploe3[i] = 0;

    }

    show_all_pole(ploe1, ploe2, ploe3,amount);
    time_t now = time(0);
    strftime(buff,10,"%M:%S",localtime(&now));
    Tower_of_Hanoi(amount,ploe1,ploe3,ploe2,amount);
    now = time(0);
    strftime(buff2,10,"%M:%S",localtime(&now));
    show_all_pole(ploe1, ploe2, ploe3,amount);
    printf("start time:%s\nend time:%s",buff, buff2);

    return 0;
}
```

output

```
amount:16
1:01,02,03,04,05,06,07,08,09,10,11,12,13,14,15,16,
2:00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,
3:00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,



1:00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,
2:00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,
3:01,02,03,04,05,06,07,08,09,10,11,12,13,14,15,16,



start time 54:12
end time  54:12
```

I also write a few more versions with different methods to measure time spent in milliseconds and print output .

V2 : **used <sys/time.h>** https://onlinegdb.com/jq0AzCEXm

```
#include <stdio.h>
#include <sys/time.h>

void show_pole(int n,int arr[],int amount){
    // printf("%d:",n);
    for(int i=0;i<amount;i++){
        if(arr[i]<10){
            printf("0%d,",arr[i]);
        }else{
            printf("%d,",arr[i]);
        }
    }
    printf("\n");
    return ;
}
void show_all_pole(int arr[], int arr2[], int arr3[],int amount){
    show_pole(1,arr,amount);
    show_pole(2,arr2,amount);
    show_pole(3,arr3,amount);
    printf("\n\n\n");
    return ;
}

long long current_timestamp(){
    struct timeval te;
    gettimeofday(&te, NULL);
    long long milliseconds = te.tv_sec*1000LL + te.tv_usec/1000;
    return milliseconds;
}

int get_top_disk(int arr[],int len){
    for(int i=0;i<len;i++){
        if(arr[i]!=0){
            return i;
        }
    }
    return len-1;
}

void Tower_of_Hanoi(int n, int arr[], int arr3[], int arr2[] ,int len){
    if (n == 1){
        int top_disk_index1 = get_top_disk(arr,len);
        int top_disk_index2 = get_top_disk(arr3,len);
        if(arr3[top_disk_index2]==0){
            arr3[top_disk_index2] = arr[top_disk_index1];
        }else{
            arr3[top_disk_index2-1] = arr[top_disk_index1];
        }
        arr[top_disk_index1] = 0;


        // show_all_pole(arr, arr2, arr3, len);
        return;
    }

    Tower_of_Hanoi(n-1, arr, arr2, arr3,len);
    // show_all_pole(arr, arr2, arr3, len);

    int top_disk_index1 = get_top_disk(arr,len);
    int top_disk_index2 = get_top_disk(arr3,len);
    if(arr3[top_disk_index2]==0){
        arr3[top_disk_index2] = arr[top_disk_index1];
    }else{
        arr3[top_disk_index2-1] = arr[top_disk_index1];
    }
    arr[top_disk_index1] = 0;

    // show_all_pole(arr, arr2, arr3, len);

    Tower_of_Hanoi(n-1, arr2, arr3, arr,len);
```

```
    // show_all_pole(arr, arr2, arr3, len);
    return;
}


int main(){
    int amount =16;
    printf("amount:");
    scanf("%d",&amount);
    int ploe1[amount];
    int ploe2[amount];
    int ploe3[amount];


    for(int i=0;i<amount;i++){
        ploe1[i] = i+1;
        ploe2[i] = 0;
        ploe3[i] = 0;

    }

    show_all_pole(ploe1, ploe2, ploe3,amount);
    long int p_time = current_timestamp();
    Tower_of_Hanoi(amount,ploe1,ploe3,ploe2,amount);
    long int n_time = current_timestamp();
    show_all_pole(ploe1, ploe2, ploe3,amount);
    printf("time span : %ld milliseconds",n_time - p_time);

    return 0;
}
```

output

---

amount:16
01,02,03,04,05,06,07,08,09,10,11,12,13,14,15,16,
00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,
00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,

00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,
00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,
01,02,03,04,05,06,07,08,09,10,11,12,13,14,15,16,

time span : 3 milliseconds

V3: **use <time.h> clock()** https://onlinegdb.com/TO7iHzP1wG

```
#include <stdio.h>
#include <time.h>

void show_pole(int n,int arr[],int amount){
    printf("%d:",n);
    for(int i=0;i<amount;i++){
        if(arr[i]<10){
            printf("0%d,",arr[i]);
        }else{
            printf("%d,",arr[i]);
        }
```

```
    }
    printf("\n");
    return ;
}
void show_all_pole(int arr[], int arr2[], int arr3[],int amount){
    show_pole(1,arr,amount);
    show_pole(2,arr2,amount);
    show_pole(3,arr3,amount);
    printf("\n\n\n");
    return ;
}

int get_top_disk(int arr[],int len){
    for(int i=0;i<len;i++){
        if(arr[i]!=0){
            return i;
        }
    }
    return len-1;
}

void Tower_of_Hanoi(int n, int arr[], int arr3[], int arr2[] ,int len){
    if (n == 1){
        int top_disk_index1 = get_top_disk(arr,len);
        int top_disk_index2 = get_top_disk(arr3,len);
        if(arr3[top_disk_index2]==0){
            arr3[top_disk_index2] = arr[top_disk_index1];
        }else{
            arr3[top_disk_index2-1] = arr[top_disk_index1];
        }
        arr[top_disk_index1] = 0;
        return;
    }
    Tower_of_Hanoi(n-1, arr, arr2, arr3,len);
    int top_disk_index1 = get_top_disk(arr,len);
    int top_disk_index2 = get_top_disk(arr3,len);
    if(arr3[top_disk_index2]==0){
        arr3[top_disk_index2] = arr[top_disk_index1];
    }else{
        arr3[top_disk_index2-1] = arr[top_disk_index1];
    }
    arr[top_disk_index1] = 0;
    Tower_of_Hanoi(n-1, arr2, arr3, arr,len);
    return;
}


int main(){
    int amount = 24;
    double time_spent = 0.0;
    printf("amount:");
    scanf("%d",&amount);
    int ploe1[amount];
    int ploe2[amount];
    int ploe3[amount];


    for(int i=0;i<amount;i++){
        ploe1[i] = i+1;
        ploe2[i] = 0;
        ploe3[i] = 0;

    }

    show_all_pole(ploe1, ploe2, ploe3,amount);
    clock_t begin = clock();
    Tower_of_Hanoi(amount,ploe1,ploe3,ploe2,amount);
    clock_t end = clock();
    show_all_pole(ploe1, ploe2, ploe3,amount);
    time_spent += (double)(end - begin) / CLOCKS_PER_SEC;
    printf("The elapsed time is %f seconds\n", time_spent);

    return 0;
}
```

output

---

amount:16
1:01,02,03,04,05,06,07,08,09,10,11,12,13,14,15,16,
2:00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,

3:00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,

1:00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,
2:00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,
3:01,02,03,04,05,06,07,08,09,10,11,12,13,14,15,16,

The elapsed time is 0.003993 seconds

V4 : **output instruction on how to solve**
https://onlinegdb.com/l8LqDqs5Y

This one is output instruction in text form.

```
#include <stdio.h>

void Tower_of_Hanoi(int n, char from, char to, char aux){
    if (n == 1){
        printf("\n Move disk 1 from rod %c to rod %c", from, to);
        return;
    }
    Tower_of_Hanoi(n-1, from, aux, to);
    printf("\n Move disk %d from rod %c to rod %c", n, from, to);
    Tower_of_Hanoi(n-1, aux, to, from);
}

int main(){
    int n = 10;
    Tower_of_Hanoi(n, 'A', 'C', 'B');
    return 0;
}
```

output : *for 5 disk*

Move disk 1 from rod C to rod B
Move disk 3 from rod A to rod C
Move disk 1 from rod B to rod A
Move disk 2 from rod B to rod C
Move disk 1 from rod A to rod C
