

第7章 函数 (3)


S.IST@XMU

复习回顾

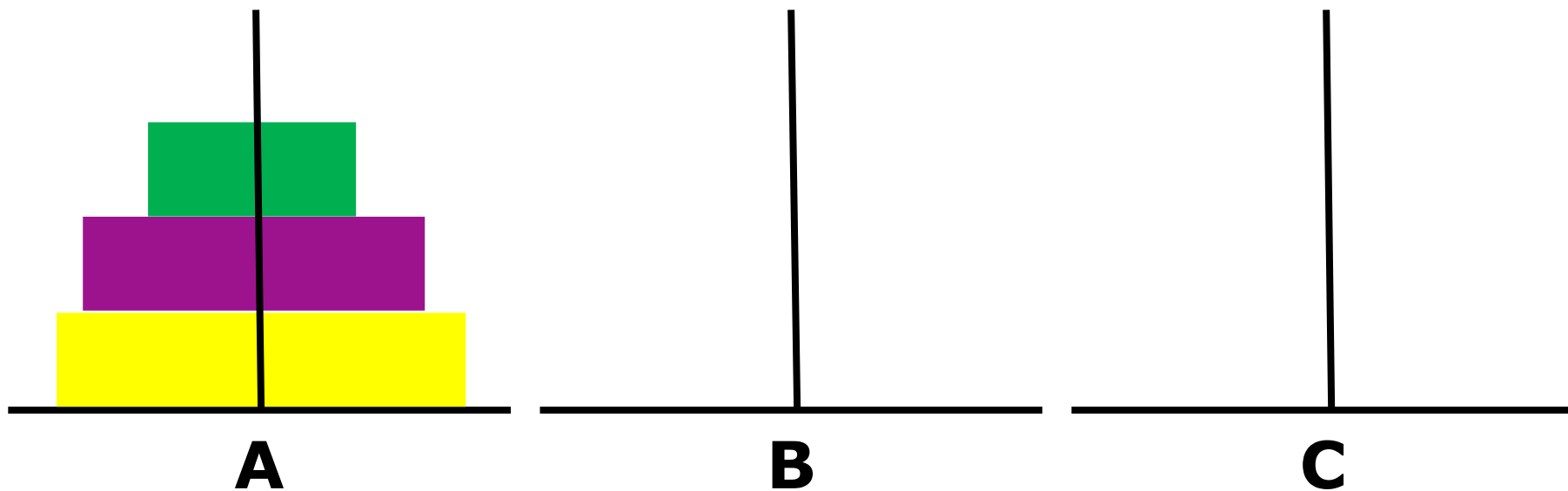
➤ 上次课的内容：

- ◆ 函数声明的例子
- ◆ 函数嵌套调用
- ◆ 递归基本思想
- ◆ 递归函数
- ◆ 递归的简单应用
- ◆ 经典递归：汉诺塔



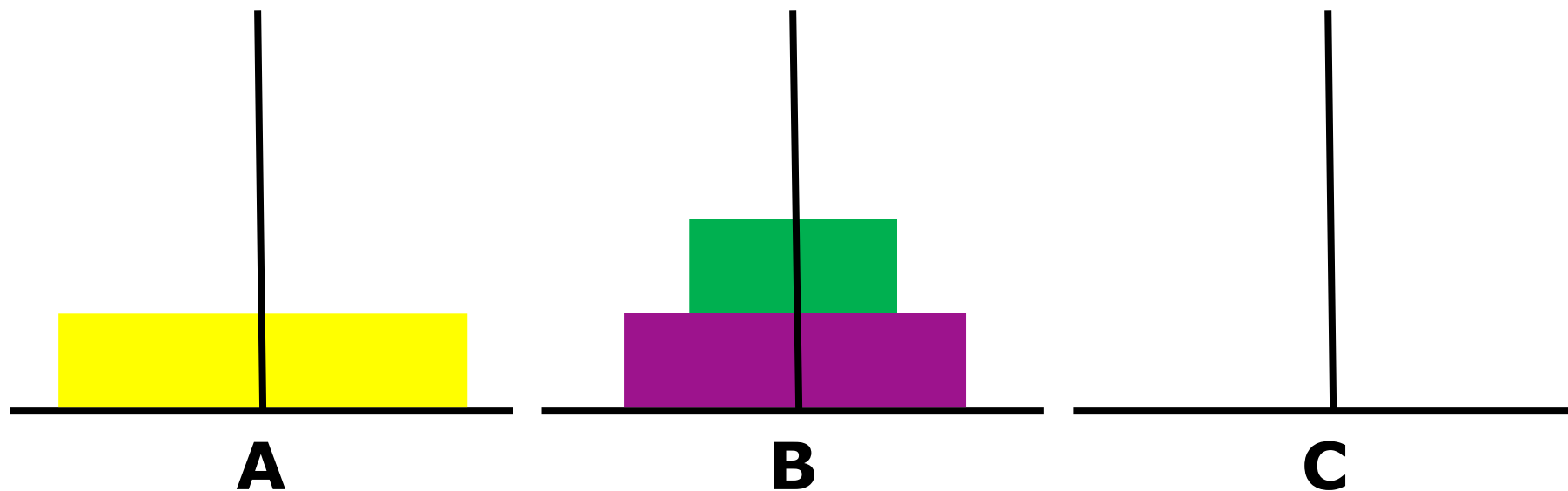
将3个盘子从A移到C的全过程

将2个盘子从A移到B



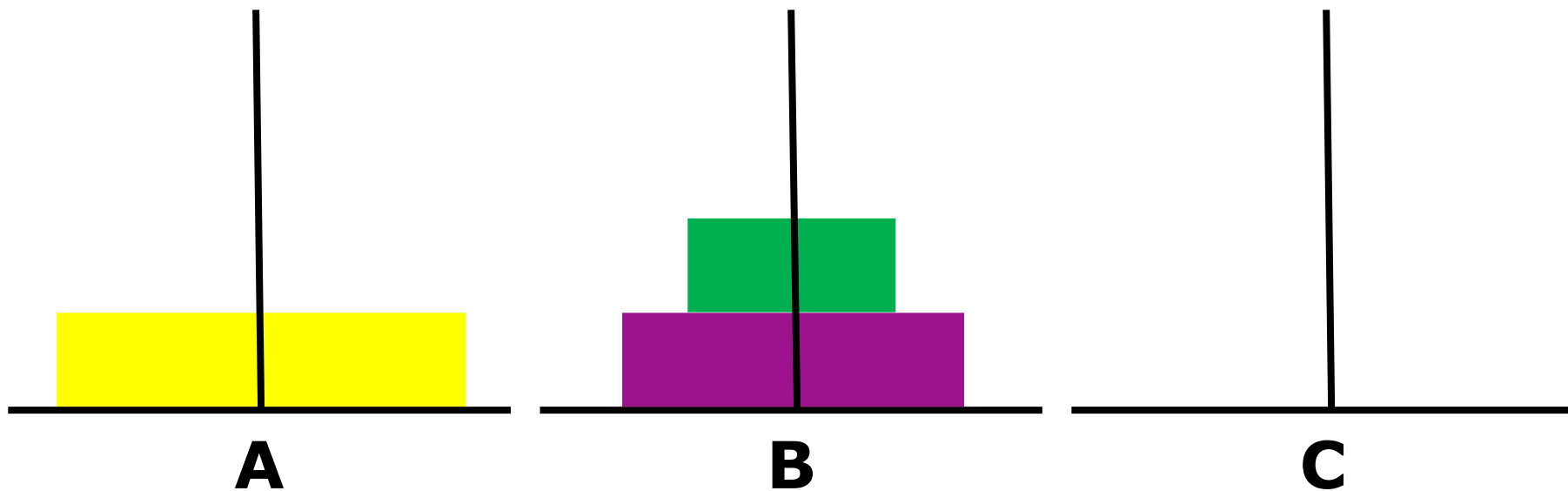
将3个盘子从A移到C的全过程

将2个盘子从A移到B



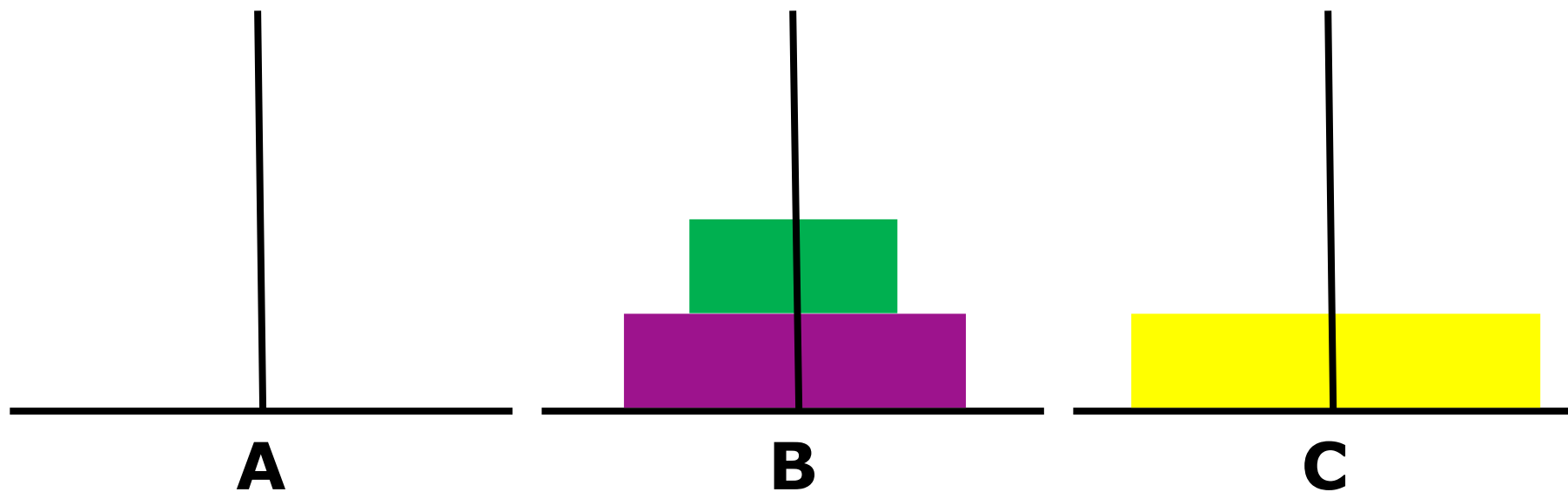
将3个盘子从A移到C的全过程

将1个盘子从A移到C



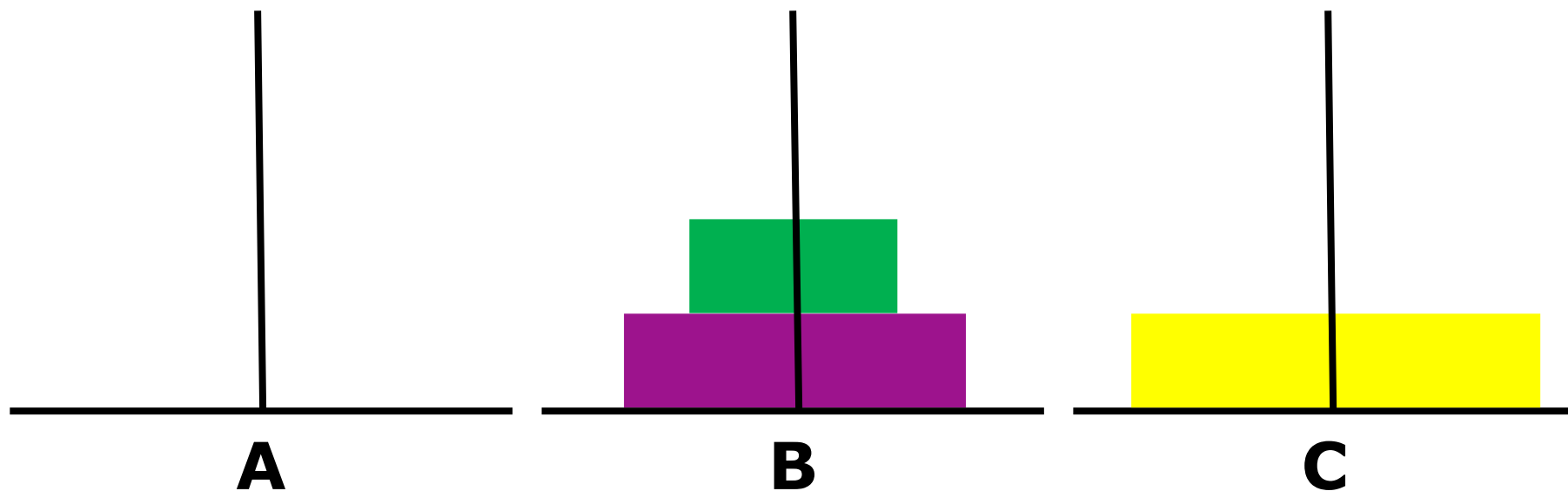
将3个盘子从A移到C的全过程

将1个盘子从A移到C



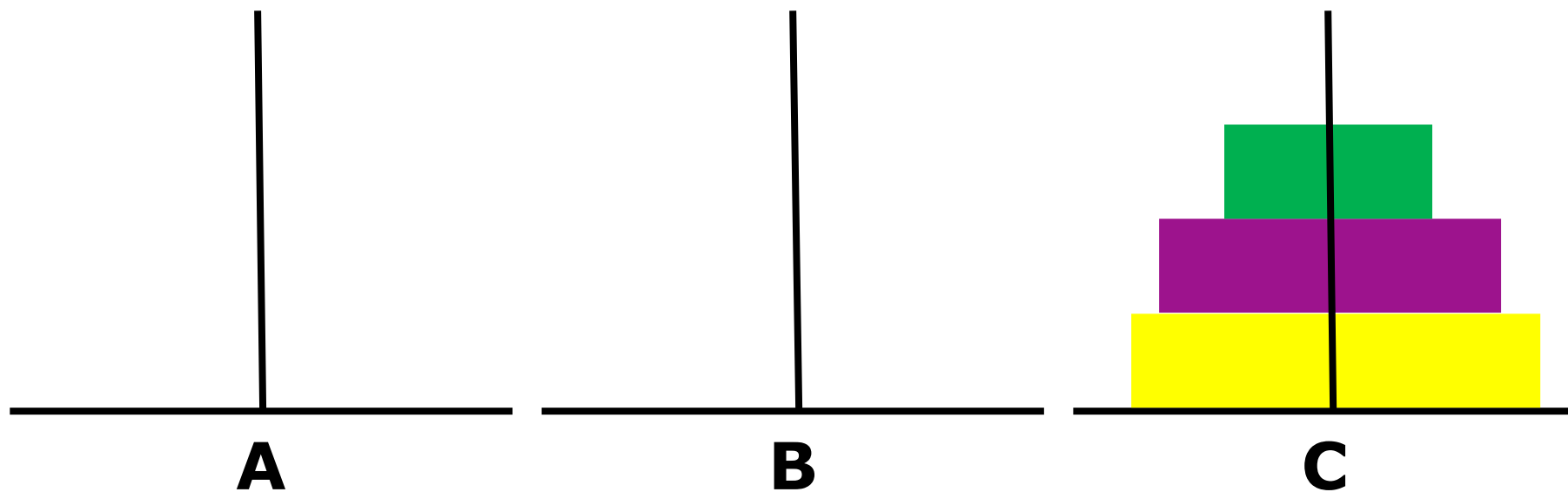
将3个盘子从A移到C的全过程

将2个盘子从B移到C



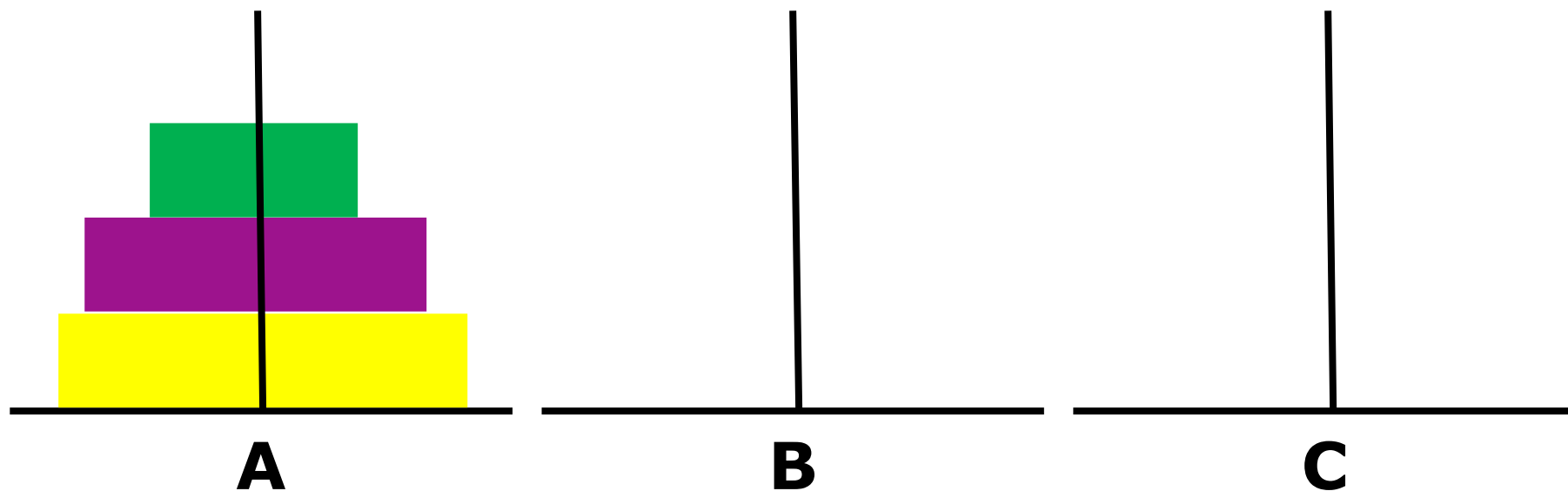
将3个盘子从A移到C的全过程

将2个盘子从B移到C



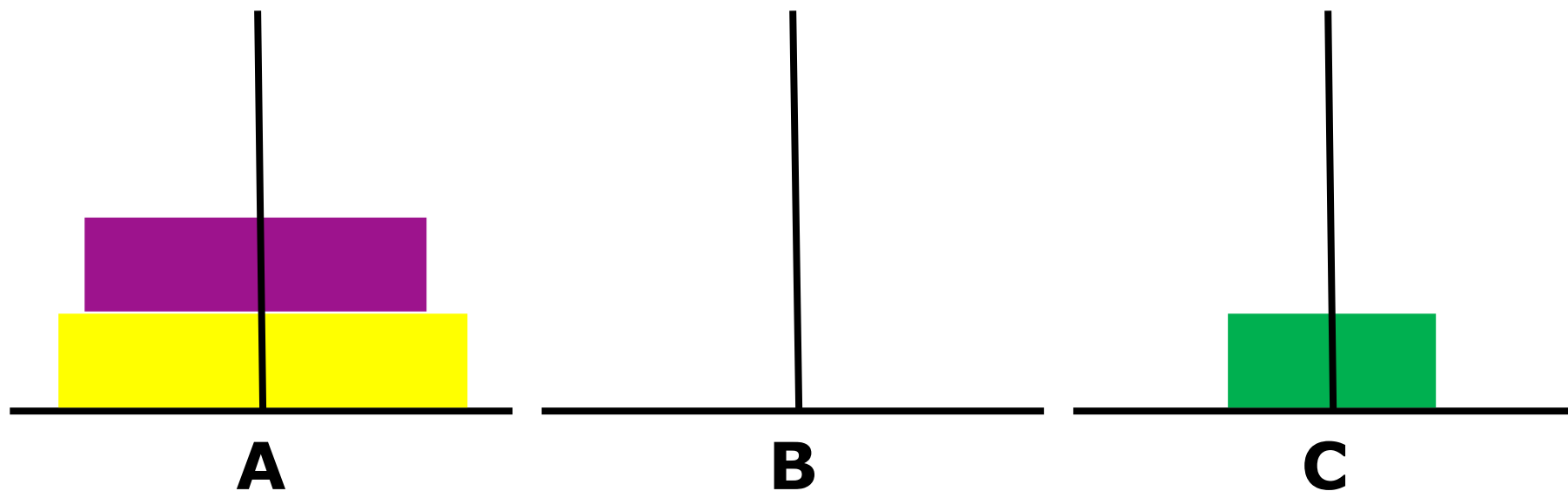
将2个盘子从A移到B的过程

将1个盘子从A移到C



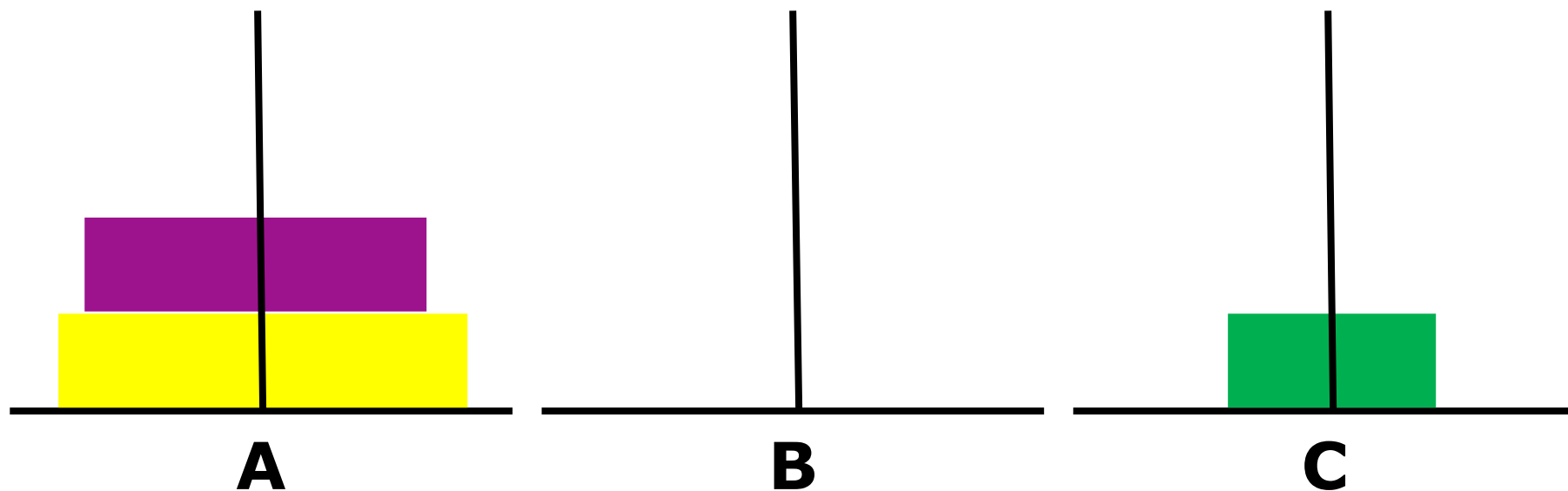
将2个盘子从A移到B的过程

将1个盘子从A移到C



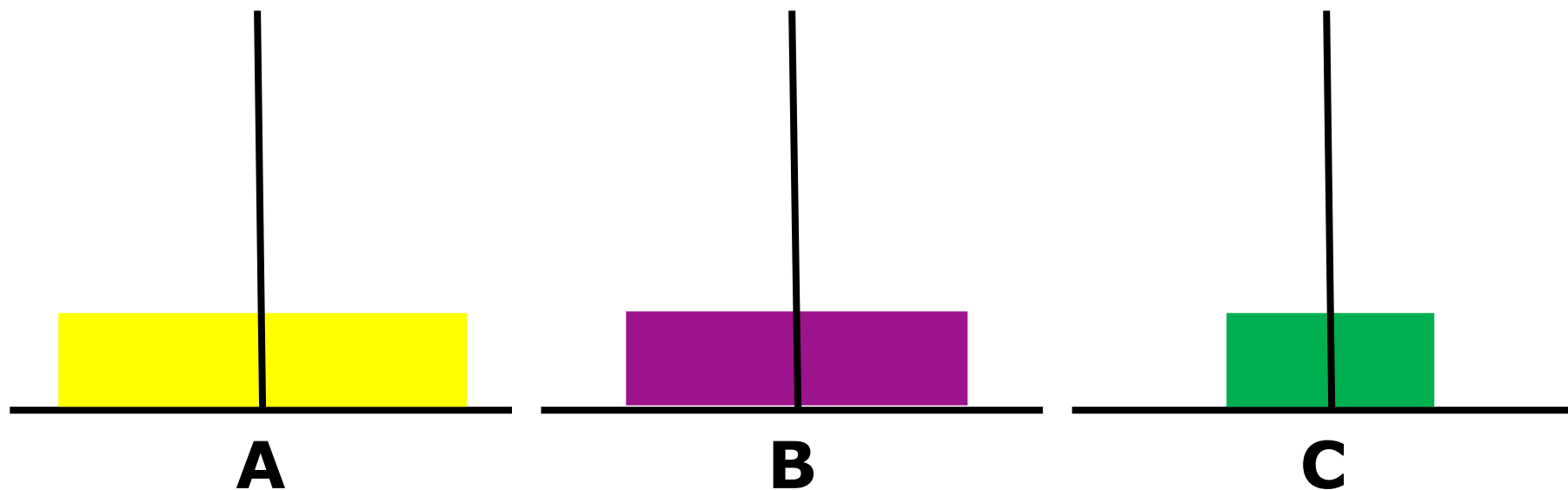
将2个盘子从A移到B的过程

将1个盘子从A移到B



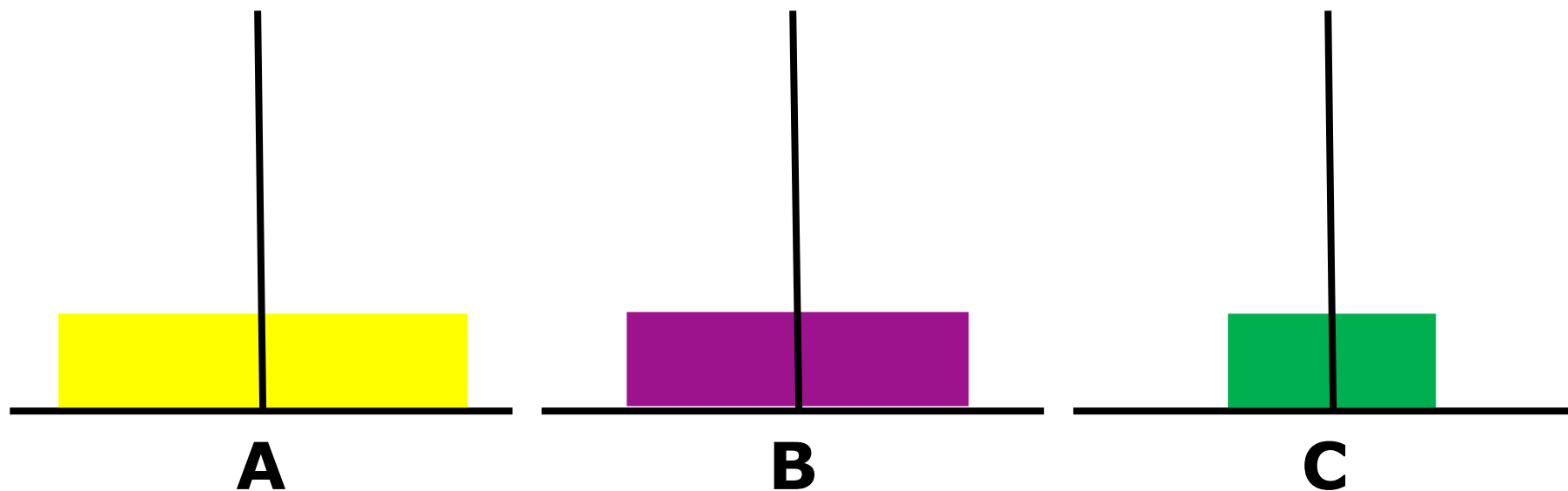
将2个盘子从A移到B的过程

将1个盘子从A移到B



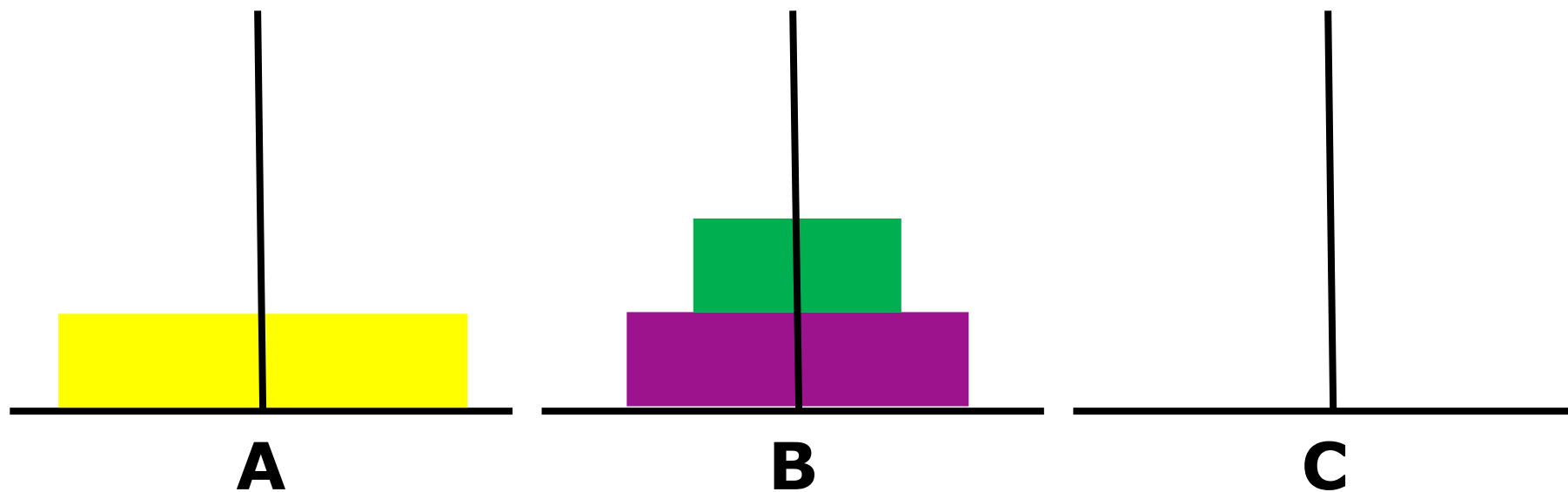
将2个盘子从A移到B的过程

将1个盘子从C移到B

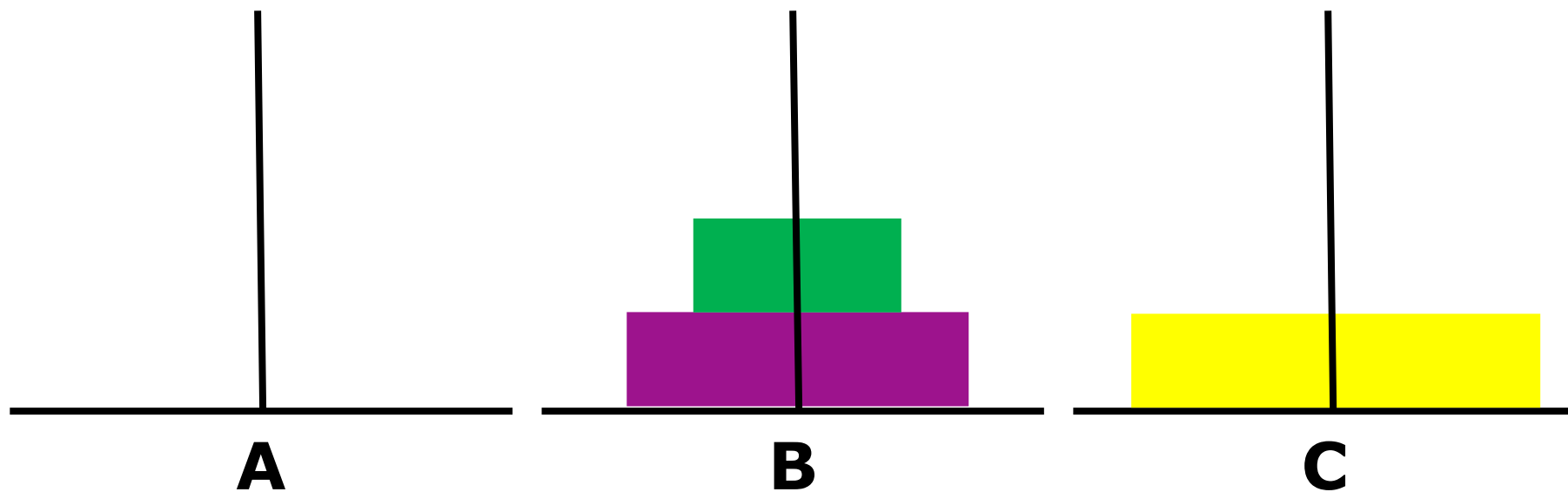


将2个盘子从A移到B的过程

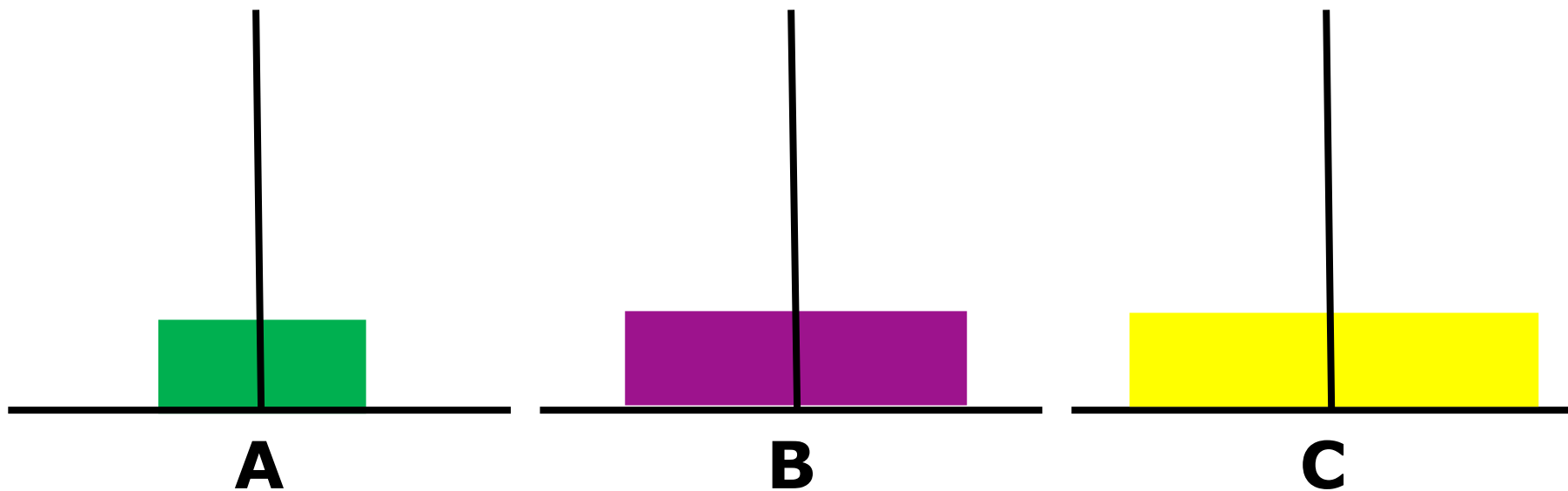
将1个盘子从C移到B



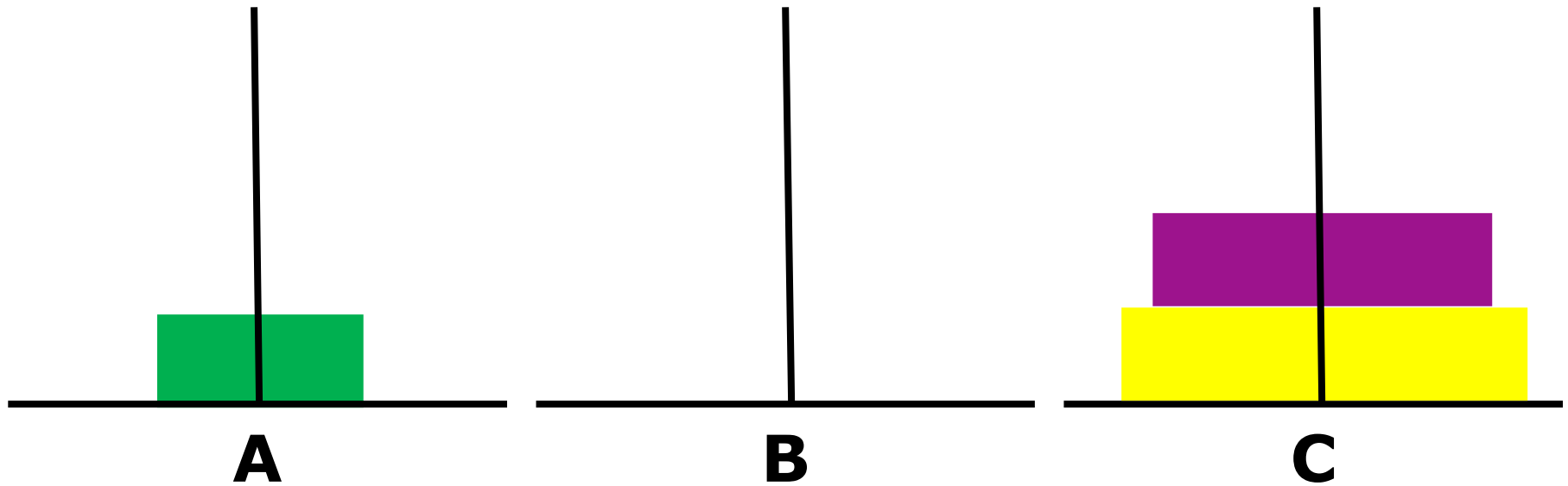
将2个盘子从B移到C的过程



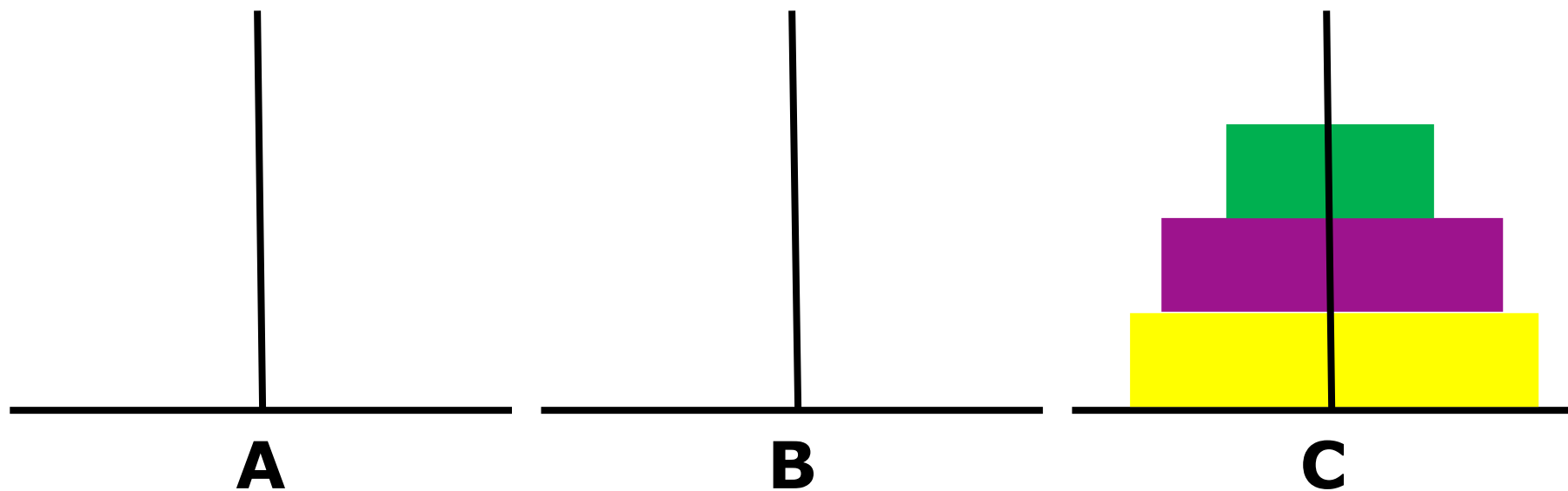
将2个盘子从B移到C的过程



将2个盘子从B移到C的过程



将2个盘子从B移到C的过程



经典递归问题：汉诺塔

- 由上面的分析可知：将 n 个盘子从A座移到C座可以分解为以下3个步骤：
- (1) 将 $n-1$ 个盘从A座借助C座先移到B座上
 - (2) 把A座上剩下的一个盘移到C座上
 - (3) 将 $n-1$ 个盘从B座借助于A座移到C座上

经典递归问题：汉诺塔

- 可以将第(1)步和第(3)步表示为：
 - ◆ 将“from”座上 $n-1$ 个盘移到“transfer”座(借助“to”座)。
 - ◆ 在第(1)步和第(3)步中，from、transfer、to和A、B、C的对应关系不同。
 - ◆ 对第(1)步，对应关系是from 对应A，to 对应B，transfer 对应C。
 - ◆ 对第(3)步，对应关系是from 对应B，to 对应C，transfer 对应A。

经典递归问题：汉诺塔

➤ 把上面3个步骤分成两类操作：

- (1) 将 $n-1$ 个盘从一个座移到另一个座上 ($n > 1$)。这就是大和尚让小和尚做的工作，它是一个递归的过程，即和尚将任务层层下放，直到第64个和尚为止。
- (2) 将1个盘子从一个座上移到另一座上。这是大和尚自己做的工作。

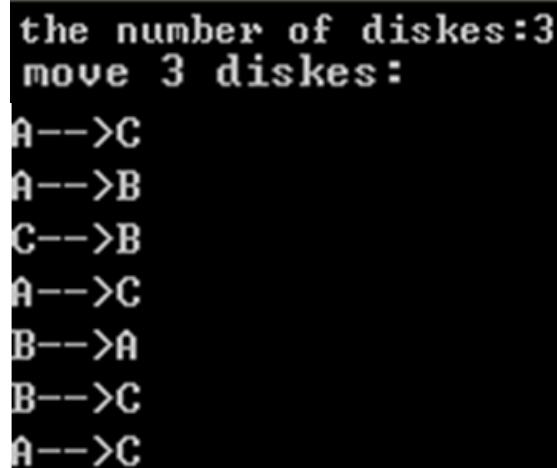
经典递归问题：汉诺塔

➤ 编写程序。

- ◆ 用hanoi函数实现第1类操作（即模拟小和尚的任务）
- ◆ 用move函数实现第2类操作（模拟大和尚自己移盘）
- ◆ 函数调用hanoi(n,from,transfer,to)表示将n个盘子从“from”座移到“to”座的过程(借助“transfer”座)
- ◆ 函数调用move(x,y)表示将1个盘子从x座移到y座的过程。x和y是代表A、B、C座之一，根据每次不同情况分别取A、B、C代入

经典递归问题：汉诺塔

```
1. #include <stdio.h>
2. int main()
3. {
4.     void hanoi(int n, char from, char transfer, char to);
5.     int m;
6.     printf("the number of disks:");
7.     scanf("%d",&m);
8.     printf("move %d disks:\n",m);
9.     hanoi(m, 'A', 'B', 'C');
10.    return 0;
11.}
12.void hanoi(int n, char from, char transfer, char to)
13.{
14.    void move(char x,char y);
15.    if (n==1)
16.        move(from,to);
17.    else
18.    {
19.        hanoi(n-1,from,to,transfer);
20.        move(from,to);
21.        hanoi(n-1,transfer,from,to);
22.    }
23.}
24.void move(char from,char to) { printf("%c-->%c\n",from,to); }
```



the number of disks:3
move 3 disks:
A-->C
A-->B
C-->B
A-->C
B-->A
B-->C
A-->C

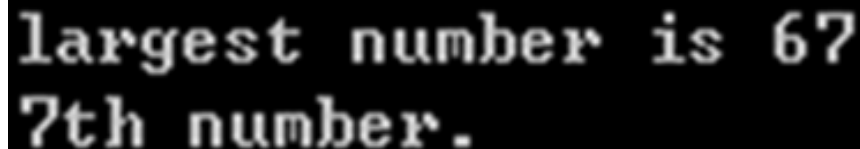
数组元素做函数参数

- 输入10个数，要求输出其中值最大的元素和该数是第几个数。
- 解题思路：
 - ◆ 定义数组a，用来存放10个数
 - ◆ 设计函数max，用来求两个数中的大者
 - ◆ 在主函数中定义变量m，初值为a[0]，每次调用max函数后的返回值存放在m中
 - ◆ 用“打擂台”算法，依次将数组元素a[1]到a[9]与m比较，最后得到的m值就是10个数中的最大者

数组元素做函数参数

```
1. #include <stdio.h>
2. int main()
3. {
4.     int max(int x,int y);
5.     int a[10],m,n,i;
6.     printf("10 integer numbers:\n");
7.     for (i=0;i<10;i++)
8.         scanf("%d",&a[i]);
9.     printf("\n");
10.    for (i=1,m=a[0],n=0;i<10;i++)
11.    {
12.        if (max(m,a[i]) > m)
13.        {
14.            m = max(m,a[i]);
15.            n = i;
16.        }
17.    }
18.    printf("largest number is %d\n",m);
19.    printf("%dth number.\n",n+1);
20.}

21.int max(int x, int y) { return(x>y?x:y); }
```



largest number is 67
7th number.

数组名做函数参数实例

```
1. #include <stdio.h>
2. int main()
3. {
4.     float average(float array[10]);
5.     int i; float aver, score[10];
6.     printf("input 10 scores:\n");
7.     for (i=0;i<10;i++)
8.         scanf("%f",&score[i]);
9.     printf("\n");
10.    aver=average(score);
11.    printf("%5.2f\n",aver);
12.    return 0;
13.}
14.float average(float array[10])
15.{
16.    int i;
17.    float aver,sum = array[0];
18.    for (i=1;i<10;i++)
19.        sum = sum+array[i];
20.    aver = sum/10;
21.    return(aver);
22.}
```

定义实参数组

```
input 10 scores:
100 56 78 98 67.5 99 54 88.5 76 58
77.50
```

定义形参数组

相当于score[0]

相当于score[i]

数组名做函数参数实例

```
1. #include <stdio.h>
2. int main()
3. {
4.     float average(float array[],int n);
5.     float score1[5]={98.5,97,91.5,60,55};
6.     float score2[10]={67.5,89.5,99,69.5,
                        77,89.5,76.5,54,60,99.5};
7.     printf("%6.2f\n",average(score1,5));
8.     printf("%6.2f\n",average(score2,10));
9.     return 0;
10. }
```

数组名做函数参数实例

调用形式为average(score1,5)时

```
float average(float array[ ],int n)
```

```
{
```

```
    int i;
```

相当于5

```
    float aver, sum=array[0];
```

```
    for (i=1;i<n;i++)
```

```
        sum=sum+array[i];
```

相当于score1[0]

```
    aver=sum/n;
```

```
    return(aver);
```

相当于score1[i]

```
}
```

数组名做函数参数实例

调用形式为average(score2,10)时

```
float average(float array[ ], int n)
{
    int i;
    float aver, sum=array[0];
    for (i=1;i<n;i++)
        sum=sum+array[i];
    aver=sum/n;
    return (aver);
}
```

相当于10

相当于score2[0]

相当于score2[i]

80.40
78.20

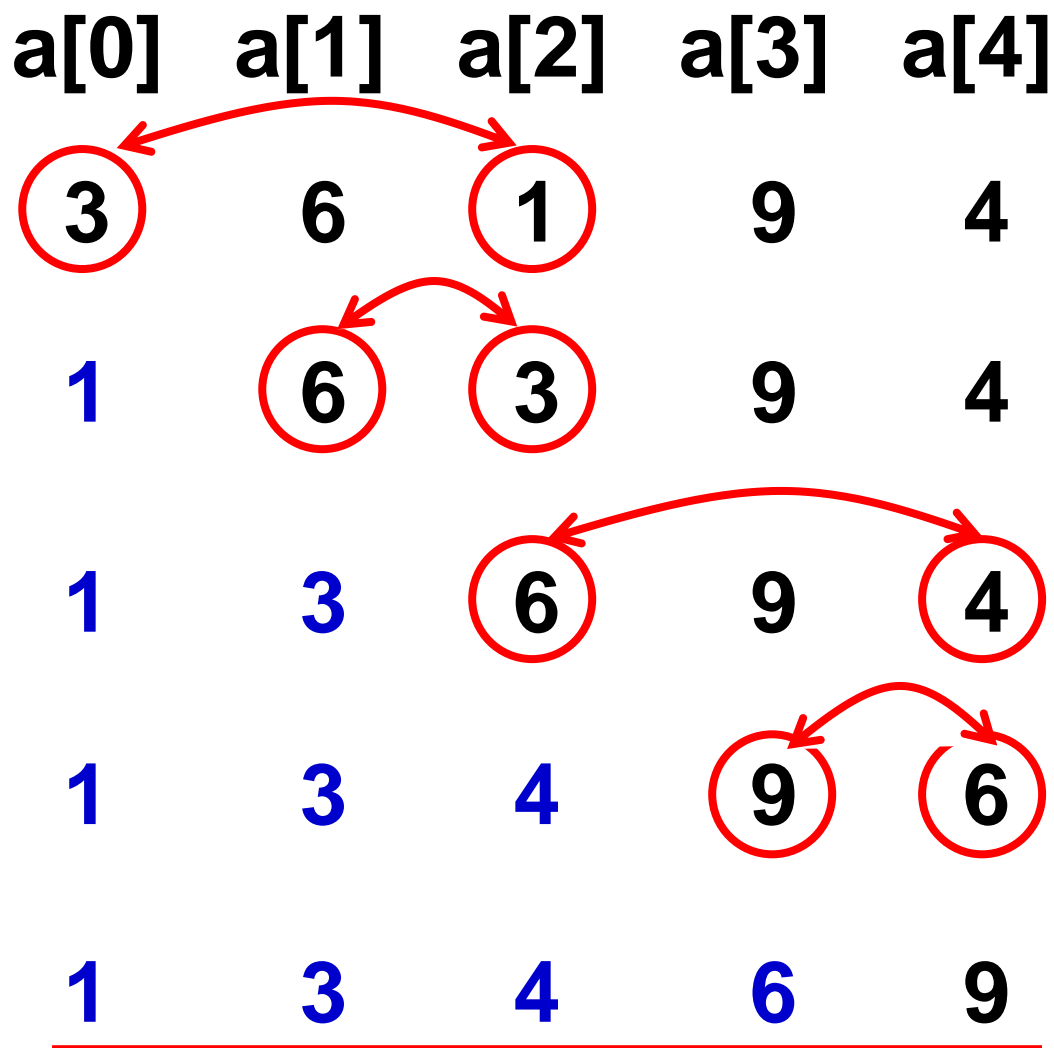
一维数组名做函数参数实例

用选择法对数组中10个整数按由小到大排序

➤ 解题思路：

◆ 所谓**选择法**就是先将10个数中最小的数与a[0]对换；再将a[1]到a[9]中最小的数与a[1]对换……每比较一轮，找出一个未经排序的数中最小的一个

◆ 共比较9轮



小到大排序

一维数组名做函数参数实例

```
1. #include <stdio.h>
2. int main()
3. {
4.     void sort(int array[],int n);
5.     int a[10], i;
6.     printf("enter array:\n");
7.     for (i=0;i<10;i++)
8.         scanf("%d",&a[i]);
9.
10.    sort(a,10);
11.
12.    printf("The sorted array:\n");
13.    for (i=0;i<10;i++)
14.        printf("%d ",a[i]);
15.    printf("\n");
16.    return 0;
17.}
```


选择法排序的实现

```
1. void sort(int array[],int n)
2. {
3.     int i,j,k,t;
4.     for (i=0;i<n-1;i++)
5.     {
6.         k = i;
7.         for (j=i+1;j<n;j++)
8.         {
9.             if (array[j]<array[k])
10.                k = j;
11.         }
12.         t = array[k];
13.         array[k] = array[i];
14.         array[i] = t;
15.     }
16. }
```

在`sort[i+1]~sort[n-1]`中，
找到最小数，下标为`k`

作业 2017/11/24

➤ 按下列要求编写程序，提交手写源代码

1. 输入一个整数 n ($0 < n < 10000$)和一个整数 m ，接着输入 n 个整数，然后

- ① 对输入的 n 个整数做从小到大的排序
- ② 在对排完序的整数中查找 m ，若找到，输出 m 的位置即排第几个，否则，输出“Not Found!”。要求用“二分法”查找，用“非递归方式”实现
- ③ 用递归方式实现上面的“二分法”查找。