```
                }
            } else if (!this.dogQ.isEmpty()) {
                return this.dogQ.poll().getPet();
            } else if (!this.catQ.isEmpty()) {
                return this.catQ.poll().getPet();
            } else {
                throw new RuntimeException("err, queue is empty!");
            }
        }

        public Dog pollDog() {
            if (!this.isDogQueueEmpty()) {
                return (Dog) this.dogQ.poll().getPet();
            } else {
                throw new RuntimeException("Dog queue is empty!");
            }
        }

        public Cat pollCat() {
            if (!this.isCatQueueEmpty()) {
                return (Cat) this.catQ.poll().getPet();
            } else
                throw new RuntimeException("Cat queue is empty!");
        }

        public boolean isEmpty() {
            return this.dogQ.isEmpty() && this.catQ.isEmpty();
        }

        public boolean isDogQueueEmpty() {
            return this.dogQ.isEmpty();
        }

        public boolean isCatQueueEmpty() {
            return this.catQ.isEmpty();
        }

    }
```

用一个栈实现另一个栈的排序

【题目】

一个栈中元素的类型为整型，现在想将该栈从顶到底按从大到小的顺序排序，只许申请一个栈。除此之外，可以申请新的变量，但不能申请额外的数据结构。如何完成排序？

【难度】

士 ★☆☆☆

【解答】

将要排序的栈记为 stack，申请的辅助栈记为 help。在 stack 上执行 pop 操作，弹出的元素记为 cur。

- 如果 cur 小于或等于 help 的栈顶元素，则将 cur 直接压入 help；
- 如果 cur 大于 help 的栈顶元素，则将 help 的元素逐一弹出，逐一压入 stack，直到 cur 小于或等于 help 的栈顶元素，再将 cur 压入 help。

一直执行以上操作，直到 stack 中的全部元素都压入到 help。最后将 help 中的所有元素逐一压入 stack，即完成排序。

```
public static void sortStackByStack(Stack<Integer> stack) {
        Stack<Integer> help = new Stack<Integer>();
        while (!stack.isEmpty()) {
                int cur = stack.pop();
                while (!help.isEmpty() && help.peek() > cur) {
                        stack.push(help.pop());
                }
                help.push(cur);
        }
        while (!help.isEmpty()) {
                stack.push(help.pop());
        }
}
```

用栈来求解汉诺塔问题

【题目】

汉诺塔问题比较经典，这里修改一下游戏规则：现在限制不能从最左侧的塔直接移动到最右侧，也不能从最右侧直接移动到最左侧，而是必须经过中间。求当塔有 N 层的时候，打印最优移动过程和最优移动总步数。

例如，当塔数为两层时，最上层的塔记为 1，最下层的塔记为 2，则打印：

```
Move 1 from left to mid
Move 1 from mid to right
Move 2 from left to mid
```