作业6

- 课本48页， 2.3， 2.4， 2.5，2.6(2)，2.7(2)(3)

2.3

2.3　双 *Hanoi* 塔问题是 *Hanoi* 塔问题的一种推广，与 *Hanoi* 塔的不同点在于：　*2n* 个圆盘，分成大小不同的 *n* 对，每对圆盘完全相同. 初始，这些圆盘按照从大到小的次序从下到上放在 *A* 柱上，最终要把它们全部移到 *C* 柱，移动的规则与 *Hanoi* 塔相同.

(1) 设计一个移动的算法并给出伪码描述.

(2) 计算你的算法所需要的移动次数.

2.3

2.3 (1)算法设计思想：分治策略. 先递归地将上面的 $2(n-1)$ 个盘子从 A 柱移到 B 柱；用2次移动将最大的2个盘子从 A 柱移到 C 柱；递归地将 B 柱的 $2(n-1)$ 个盘子从 B 柱移到 C 柱. 伪码描述如下：

```
BiHanoi(A,C,n)          //从A到C移动2n只盘子
1.  if n == 1 then
2.      BiMove(A,C)     //从A到C移动2只盘子
3.   else
4.      BiHanoi(A,B,n-1)
5.      BiMove(A,C)
6.      BiHanoi(B,C,n-1)
7.  end
```

(2) 设 $2n$ 个圆盘的移动次数是 $T(n)$，则第行和第4行的递归调用的子问题规模是 $n-1$，第3行是2次移动，于是有

$$\begin{cases} T(n) = 2T(n-1) + 2 \\ T(1) = 2 \end{cases}$$

解得 $T(n) = 2^{n+1} - 2$

2.4

2.4　给定含有 n 个不同的数的数组 $L=<x_1,x_2,\ldots,x_n>$，如果 L 中存在 x_i，使得 $x_1<x_2<\ldots<x_{i-1}<x_i>x_{i+1}>\ldots>x_n$，则称 L 是单峰的，并称 x 是 L 的“峰顶”. 假设 L是单峰的，设计一个算法找到 L 的峰顶.

因为 L 中存在峰顶元素，因此 $|L|\geq 3$. 使用二分查找算法. 如元素数等于3，则$L[2]$是峰顶元素. 当元素数 n 大于3时，令 $k=\lfloor n/2 \rfloor$，比较 $L[k]$ 与它左边和右边相邻的项. 如果 $L[k]>L[k\text{-}1]$ 且 $L[k]>L[k+1]$，则 $L[k]$ 为峰顶元素；否则，如果$L[k\text{-}1]>L[k]>L[k+1]$，则继续搜索 $L[1\ldots k\text{-}1]$ 的范围；如果 $L[k\text{-}1]<L[k]<L[k+1]$，则继续搜索 $L[k+1\ldots n]$ 的范围．每比较2次，搜索范围减半，直到元素数小于等于3停止递归调用. 时间复杂度函数为：

$$\begin{cases} T(n)=T(\dfrac{n}{2})+2 \\ T(1)=c, \quad c\text{为某个常数} \end{cases}$$

根据主定理，$T(n)=O(\log n)$.

1.是否正确？
2.如何改进？
3.如何验证改进是正确的？

```
1.  l ← 1;  r ← n
2.  While  l ≤ r  do
3.        m ← ⌊(l+r)/2⌋;
4.        if  L[m-1] <L[m] > L[m+1]  then
5.            return  L[m];
6.        elif  L[m-1] < L[m] < L[m+1]  then
7.            l ← m+1;
8.        else          // L[m-1] > L[m] > L[m+1]
9.            r ← m-1;
10.       end
11. end
```

特例：12354
特例：15432

改进一：

特殊情况特殊处理：$l-r>1$

改进二：

$l \leftarrow m$

$r \leftarrow m$

```
1.  l ← 1;   r ← n
2.  While  l ≤ r  do
3.          m ← ⌊(l+r)/2⌋;
4.          if   L[m-1]  <L[m] > L[m+1]   then
5.              return  L[m];
6.          elif  L[m-1] < L[m] < L[m+1]    then
7.              l ← m;
8.          else          //  L[m-1] > L[m] > L[m+1]
9.              r ← m;
10.         end
11. end
```

验证：长度为3，4，5；之后都是重复

改进一：
特殊情况特殊处理：$l-r>1$

改进二：
$l \leftarrow m$
$r \leftarrow m$

改进三（严中圣）：
初始 $l \leftarrow 2$

验证：长度为3，4，5；之后都是重复

```
1.  l ← 2;  r ← n
2.  While  l ≤ r  do
3.        m ← ⌊(l+r)/2⌋;
4.        if  L[m-1] <L[m] > L[m+1]  then
5.            return  L[m];
6.        elif  L[m-1] < L[m] < L[m+1]   then
7.            l ← m+1;
8.        else          // L[m-1] > L[m] > L[m+1]
9.            r ← m-1;
10.       end
11. end
```

改进一：
特殊情况特殊处理：*l-r*>1

改进二：
l←*m*
r←*m*

改进三（严中圣）：
初始 *l*←2

改进四（罗涛）：
单边判断

验证：长度为3，4，5；之后都是重复

```
1.  faction  Champion(L)
2.  l ←1;  r ←n
3.  While  l < r  do
4.        m ←⌊(l+r)/2⌋;
5.        if  L[m] < L[m+1]  then
6.              l ←m+1 ;
7.        elif  L[m] >L[m+1]  then
8.              r←m ;
9.        end
10. end
11. return  l
```

改进四：
单边判断

广义单峰问题：数列严格递增、严格递减或书上所指单峰。严格递增（峰顶是尾）或者严格递减（峰顶是首）。
显然，如果算法能解决“广义单峰”，那书上的问题也能解决。

idea是：利用分治算法，最简子问题只对两个数或三个数进行比较，*l* 找到满足条件的数，只需要直接返回 *l* 即可。

算法思想史：对于一个满足上述定义的序列：
如果中间的数L[mid]，则返回；
如果中间的数L[mid]比下一个小，那L[mid]绝对不可能是峰顶,峰顶只有可能出现在右边，并且右边序列任满足定义；
如果 L[mid]比下一个小，那峰顶绝不可能在右边，只可能存在于左边包含L[mid]的序列，此时，该序列也满足定义。

改进四：
单边判断

递归表示

```
1.  Function champion(l,r):
2.  if(l==r) then
3.        return l
4.  end
5.        m ← floor((l+r)/2)
6.  if (L[m] < L[m+1])   then
7.        return champion(m+1,r)
8.  else
9.        return champiton(l,m)
10. end
11. end
```

迭代表示

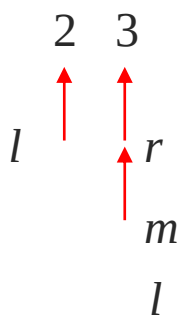
```
1.  faction  Champion(L)
2.  l ←1;  r ←n
3.  While  l <  r  do
4.        m ←⌊(l+r)/2⌋;
5.        if   L[m] < L[m+1]   then
6.            l ←m+1 ;
7.        else
8.            r←m ;
9.       end
10.  end
11. return  l
```

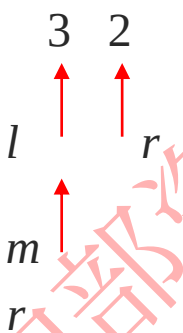
正确性

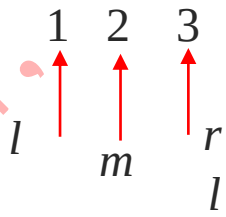
- 最小规模问题只有两种，规模为2，或3（更大规模的都会归约到这两种情况）。
- 结合上诉定义，规模为2只能是递增或递减序列，举例验证，均可以找到封顶；
- 规模为3可能是递增或递减序列，也可能是书上严格定义的单峰，也是均可以找到峰顶。
- 至此“我”认为，不需要再验证更大规模的问题

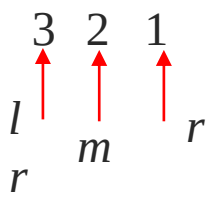
```
1.  faction  Champion(L)
2.  l ←1;  r ←n
3.  While  l <  r  do
4.          m ←⌊(l+r)/2⌋;
5.          if   L[m] < L[m+1]   then
6.              l ←m+1 ;
7.          elif   L[m] >L[m+1]    then
8.              r←m ;
9.          end
10.  end
11.  return   l
```

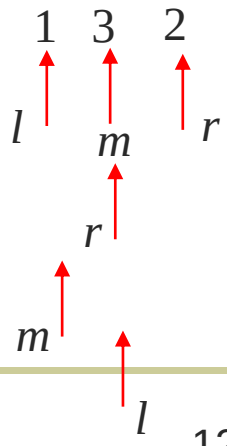
最简子问题只有两种情况，规模为2，或3（更大规模的都会规约到这两种情况）。











2.4 使用直接查找？

2.4 解：

```
Search(L)
s ← 右 L[1];
for i←2 to n do
    if L[i] > s then
        s ← L[i];
    else
        break;
    end
end
return s;
```

2.5

2.5 设 A 是 n 个不同的数排好序的数组，给定数 L 和 U，$L<U$，设计一个算法找到 A 中满足 $L<x<U$ 的所有的数 x.