参考答案

5.1

将数组中相邻元素两两配对，用合并算法将它们排好序，构成 $n/2$ 组长度为 2 的排好序子数组段，然后将它们排序成长度为 4 的排好序子数组段，如此下去直到整个数组排好序。

例子：　5　2　8　4　6　1　3　7

S=1　(5)　(2)　(8)　(4)　(6)　(1)　(3)　(7)

S=2　(2　5)　(4　8)　(1　6)　(3　7)

S=3　(2　4　5　8)　(1　3　6　7)

(1　2　3　4　5　6　7　8)

```
IterMergeSort(A, n)
1       s←1
2       while s<n do
3           MergeAB(A, B, s, n) //合并 A 中大小为 s 的相邻子数组，存放到 B 中
4           s←s+s
5           MergeAB(B, A, s, n)   //合并 B 中大小为 s 的相邻子数组，存放到 A 中
6           s←s+s
```

5.2

$$C=\begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix}=\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}\begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix},$$

$$C=\begin{pmatrix} d_1+d_4-d_5+d_7 & d_3+d_5 \\ d_2+d_4 & d_1+d_3-d_2+d_6 \end{pmatrix}$$

$$d_1=(A_{11}+A_{22})(B_{11}+B_{22})$$
$$d_2=(A_{21}+A_{22})B_{11}$$
$$d_3=A_{11}(B_{12}-B_{22})$$
$$d_4=A_{22}(B_{21}-B_{11})$$
$$d_5=(A_{11}+A_{12})B_{22}$$
$$d_6=(A_{21}-A_{11})(B_{11}+B_{12})$$
$$d_7=(A_{12}-A_{22})(B_{21}+B_{22})$$

$$C=\begin{pmatrix} 12 & -36 \\ 2 & -35 \end{pmatrix}$$

5.3

算法思想：

1、先将问题划分为大小近似相同的两个子问题

2、对子问题递归调用该算法进行处理，递归出口为子问题只含一个元素，这时将其与 x 直接比较，若等于 x 则返回该元素在数组中的位置

3、由于给定数组是有序的，因而可以根据 x 的大小判断 x 位于那个子问题内，而将另一个不含 x 的子问题丢弃

```
Searchx(A, p, r, x)
1       if p>r then
2           return 0
```

```
3       else
4           q← (p+r)/2
5           if x=A[q] then
6               return q
7           else  if  x< A[q] then
8                   return  Searchx(A, p, q-1, x)
9               else
10                  Searchx(A, q+1, r, x)
```

$$T(n)=\begin{cases}\Theta(1) & \text{if } n=1\\ T(\lfloor n/2 \rfloor)+\Theta(1) & \text{if } n\geq 2.\end{cases}$$

由公式法可得 $T(n)=O(\lg n)$

5.4

算法思想：

1、先将数组 A 划分为大小近似相等的两个子数组，令 $k=n/2$，首先做 k 次比较 $a[i]$和 $a[k+i]$，把小的放前面，大的放后面。这样经 k 次比较后我们有 $a[i]<a[k+i]$

2、用 k-1 次比较找出 $A[1..k]$中最小者，再用 k-1 次比较找出 $A[k+1..n]$中最大者。找出的这两个数即为所求的最小值和最大值。

3、总比较次数为 $3k-2=3n/2-2$

```
MaxMin(A,n)
1   k←n/2
2   for i←1 to k do
3       if A[i]>A[k+i] then
4           A[i] ↔A[k+i]
5   min ← A[1]
6   for i←2 to k do
7       if min>A[i] then
8           min←A[i]
9   max←A[k+1]
10  for i←k+2 to n do
11      if  max<A[i] then
12          max←A[i]
13  return (min,max)
```

5.5

算法思想：

1、先将问题划分为大小近似相等的两个子问题

2、对子问题递归调用该算法进行处理，递归出口为子问题只含一个元素，这时元素的和就为该元素自己

3、原问题结果为这两个子问题之和

```
Sum(A, p, r)
1       if p=r then
2           return A[p]
```

```
3       else
4           q←(p+r)/2
5           sum1←Sum(A, p, q)
6           sum2←Sum(A, q+1, r)
7           return  (sum1+sum2)
```

5.6

算法思想：

1、先将问题划分为大小近似相等的两个子问题

2、对子问题递归调用该算法进行处理，递归出口为子问题只含一个元素，若该元素等于 x，则返回 x 的出现次数 1，若该元素不等于 x，则返回 0

3、原问题结果为这两个子问题所得结果之和

```
Countx(A, p, r, x)
1   if p=r then
2       if A[p]=x then
3           return 1
4       else
5           return 0
6   else
7       q←(p+r)/2
8       return  (Countx(A, p, q, x)+Countx(A, q+1, r, x))
```

$$T(n)=\begin{cases}\Theta(1) & \text{if } n=1\\ 2T(\lfloor n/2 \rfloor)+\Theta(1) & \text{if } n\geq 2.\end{cases}$$

利用公式法可得 $T(n)=\Theta(n)$

5.7

利用二分搜索的思想。

```
BinarySearch(A,left, right, ix);

BinarySearch(A, p, r,x1,x2,y1,y2)
1   if x1>x2 then
2       x1↔ x2
3   beswap←1
4   while p≤r do
5       q←(p+r)/2
6       if A[q]<x1 then   p←q+1;
7       else if A[q]=x1 then
8           y1←q;
9           y2←BinarySearch(A,q+1,r,x2);
10      else if A[q]>x1 and A[q]<x2 then
11          y1←BinarySearch(A,p,q-1,x1);
12          y2←BinarySearch(A,q+1,r,x2);
```

```
13        else if A[q]=x2 then
14              y2←q;
15              y1←BinarySearch(A,l,q-1,x1);
16        else
17                    r←q-1;
18  if beswap=1 then
19                    y1↔y2
```

5.8

5.9

算法思想：

1、选择一个元素 $a[r]$作为支点，将数组 A 划分成三个部分：A1、A2、A3，它们分别包含小于、等于和大于 $a[r]$的元素

2、根据数组的长度可以判断第 k 小元素出现在三个数组中的哪一个中，对于包含第 k 小元素的那个数组递归调用算法进行处理，丢弃不含的两部分

Selectimin(A, p, r, k)

1 **if** $p = r$

2 **then return** $A[p]$

3 $q \leftarrow$Partition (A, p, r)

4 $i \leftarrow q - p + 1$

5 **if** $k = i$ // the pivot value is the answer

6 **then return** $A[q]$

7 **elseif** $k < i$

8 **then return** Selectimin ($A, p, q-1, k$)

9 **else return** Selectimin ($A, q+1, r, k-i$)

5.10

参见上一习题答案

5.11

有两种办法，可使用栈，也可以不使用栈，网上都可以搜到，这里略。

5.12

算法思想：

1、利用算法 QuickSort 中的划分过程 Partition 将数组 A 划分成两部分，假设划分过程

返回支点元素 $A[q]$，则 A1=$A[1..q-1]$、A2=$A[q+1..r]$，其中 A1 中元素均比 $A[q]$小，A2 中元素均比 $A[q]$大

2、若 $q=k+1$，则 A1 就为前 k 个最小元素

若 $q>k+1$，则可将 A2 丢弃，继续递归调用算法在 A1 中寻找前 k 个最小元素

若 $q<k+1$，则 A1 为结果的一部分，结果的另一部分为递归调用算法在 A2 中寻找前 $k-q+1$ 个最小元素

算法过程同习题 5.9

5.13

5.14

5.15

算法思想：

1、首先从 T1 和 T2 的根节点开始判断，如果两二叉树的根节点相同，则两二叉树有可能相同

2、递归调用算法，分别判断根的左右子树是否相同，如果根节点相同且左右子树都分别相同，则这两个二叉树相同

```
IsEqual(T1,T2)
1.  if T1=Null and T2=Null then
2.      return true
3.  else if T1≠Null and T2≠Null then
4.      if T1->data=T2->data then
5.          return IsEqual(T1->lchild,T2->lchild) and IsEqual(T1->rchild,T2->rchild)
8.      else
9.          return flase
10.     end if
11. else
12.     return false
13. end if
```

5.16

算法思想：

1、要计算一棵二叉树的高，先分别计算其左右子树的高度，二叉树的高度等于左右子树高度中大的再加 1

2、在计算左右子树高度时递归调用算法，递归出口为遍历到叶子节点时，其高度为 1

```
GetHeight(T)
1.  if T=Null then return 0
2.  else
3.      return max{GetHeight(T－>lchild),
4.                 GetHeight(T－>rchild)}+1
5.  end if
```

5.17

类似习题 5.9 求第 k 大元素，也可以如下求解。

```
FindSecond(A, p, r)
1   if   p=r then return (-∞, A[r])
3   else if r- p =1 then
4            if A[p]<A[r] then return (A[p], A[r])
5            else return (A[r], A[p])
6       else if r- p >1 then
7                q←(p+r)/2
8                (x1,y1)←FindSecond(A, p, q)
9                (x2,y2)←FindSecond(A, q +1, r)
10               if y1>y2 then
11                    y←y1    // 子数组的最大元素
12                    x←max(x1,y2) // 子数组的次大元素
13               else
14                    y←y2
15                    x←max(x2,y1)
16               return (x, y)
17          else
18               return(-∞,-∞)
```

5.18

实验题

5.19 编程序实现残缺棋盘游戏算法，并能用图形演示。

5.20 分别将插入排序、选择排序、合并排序和快速排序编程实现，并使用实验分析方法比较各种算法的效率。

5.21 完成 XOJ 如下题目：1004，1007，1017，1018，1022，1057。

5.22 完成 POJ 如下题目：1002，1947，2082，2282，2299，2318，2379，2418，2726，3122。

3.1