7.2-1
见《算法导论》7.4.1。

> 我的方法：
> T(n) = T(n-1) + O(n)
> T(n-1) = T(n-2) + O(n-1)
> …… = …… + ……
> T(2) = T(1) + O(2)
>
> T(n) = T(1) + O(n) + O(n-1) + …… + O(2) = $O(n^{2})$

7.2-6

> 由于是一个随机的序列，不妨先对这个序列进行排序，然后随机的挑选主元。题目要求划分比1-a：a要好，则说明主元在这个排好序的序列里处的位置在na个元素之后,而在n（1-a）个元素之前，因此主元能选择的可能性有n（1-a）-na个，总共有n个选择，故概率为n（1-a）-na/n=1-2a，得证。

7.4-3

> 令$f(q) = q^2 + (n-q-1)^2 = 2q^2 + 2(1-n)q + (n-1)^2$
> 这是一个关于q的抛物线，且开口向上。因此q的取值离对称轴越远，f(q)的值就越大。
> 对称轴为q = -b/2a = (n-1)/2
> 当q=0或q=n-1时取得最大值

7.4-4

> 见P7.4.2

7.4-5

```
//7.4-5利用插入排序改善快排
int k = 4;
//划分
int Partition(int *A, int p, int r)
{
//选择A[r]作为主元
int x = A[r];
int i = p - 1, j;
bool flag = 0;
for(j = p; j < r; j++)
{
//小于主元的放在左边
if(A[j] < x || (A[j] == x && flag))
{
i++;
//把大于主元的交换到右边
swap(A[i],A[j]);
if(A[j] == x)flag = !flag;
}
}
swap(A[i+1], A[r]);
//返回最终主元的位置
return i+1;
}
//快速排序
void QuickSort(int *A, int p, int r)
{
//长度小于k的子数组不排序
if(r - p >= k)
{
//以某个主元为标准，把数组分为两部分，左边都比主元小，右边都比主元大
int q = Partition(A, p, r);
//分别对左边和右边排序
```

```
QuickSort(A, p, q-1);
QuickSort(A, q+1, r);
}
}
//插入排序
void InsertSort(int *A, int p, int r)
{
int i, j;
for(i = p + 1; i <= r; i++)
{
int temp = A[i];
j = i;
while(A[j-1] > temp)
{
A[j] = A[j-1];
j--;
}
A[j] = temp;
}
}
void Sort(int *A, int p, int r)
{
//先进行粗粒度的快排
QuickSort(A, p, r);
//逐个进行插入排序
InsertSort(A, p, r);
}
```