Ch3

1.思想 :Divide :将 $2n$ 个数字分为 2 组有序的 n 个数字。Conquer:分别选出 X 和 Y 的中位数 midnum1、midnum2 进行比较，若二者相等，则返回 midnum1。若 midnum1 大于 midnum2，则 X 和 Y 整体的中位数应在 $X[1,\lceil n/2\rceil]$和 $Y[\lceil n/2\rceil,n]$中，否则在 X $[\lceil n/2\rceil,n]$, Y $[1,\lceil n/2\rceil]$中，反复递归。Merge:输出 midnum1。

算法 **FindMidNum**(X, Y)

输入 :有序数组 $X[1{:}n]$, 有序数组 $Y[1{:}n]$

输出 :X 和 Y 中 $2n$ 个数字的中位数

If n=odd midnum1=$X[(n+1)/2]$, midnum2=$Y[(n+1)/2]$

Else midnum1=($X[n/2]$+$X[(n+1)/2]$)/2, midnum2=($Y[n/2]$+$Y[(n+1)/2]$)/2

If midnum1==midnum2 **return** midnum1

Else if midnum1>midnum2 **FindMidNum**($X[1,\lceil n/2\rceil]$, $Y[\lceil n/2\rceil,n]$)

Else **FindMidNum**($X[\lceil n/2\rceil,n]$, $Y[1,\lceil n/2\rceil]$)

时间复杂度:Divide 和 Merge 阶段的复杂度为 $O(1)$，Conquer 阶段每次递归后删去了一半的元素，总的递归方程为 $T(n)=O(1)$，$n=1$；$T(n)=T(n/2)+O(1)$，$n>1$，故时间复杂度为 $O(\log n)$。

2.思想:由于 2*$A[k]$为偶数，可以在 $A[k]$的左边全部放奇数，在 $A[k]$的右边全部放偶数即符合题设。A[k]由 A[$\lfloor k/2\rfloor$]和 A[$\lfloor (k+1)/2\rfloor$]组成，奇数的个数为$\lfloor (k+1)/2\rfloor$个，偶数个数为$\lfloor k/2\rfloor$个。对 A[$\lfloor (k+1)/2\rfloor$]可取 1 到$\lfloor (k+1)/2\rfloor$的连续正整数，将其乘 2 减 1 则必为奇数；对 A[$\lfloor k/2\rfloor$]可取 1 到$\lfloor k/2\rfloor$的连续正整数，将其乘 2 则必为偶数。显然，两部分的并集恰好为 1 到 N 的正整数集。

算法 **BeautifulArray**(N)

输入:正整数 N

输出:满足性质的漂亮数组 A

If N=1 **return** $A[1]$=1

BeautifulArray($\lfloor (N+1)/2\rfloor$)

For i=1 **TO** $\lfloor (N+1)/2\rfloor$ **DO**

  $A[i]$=2$A[i]$-1

BeautifulArray($\lfloor N/2\rfloor$)

For i=1 **TO** $\lfloor N/2\rfloor$ **DO**

  $A[i]$=2$A[i]$

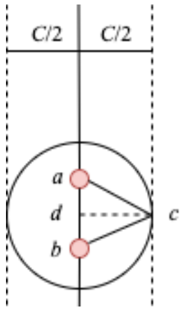
return A

时间复杂度:Divide 阶段复杂度为 $O(1)$，Conquer 阶段复杂度为 $2T(n/2)$，Merge 阶段复杂度为 $O(n)$，总的递归方程为 $T(n)=1$，$n=1$，$T(n)=2T(n/2)+O(n)$，$n>1$，故时间复杂度为 $O(n\log n)$。

3.思想:类似于寻找二维空间中的最小距离点对。Preprocessing:如果 S 中仅包含 3 个点，算法结束，否则把 S 中点分别按 x 和 y 坐标值排序。Divide:计算 S 中各点 x 坐标的中位数 m，用垂线 $L{:}x=m$ 把 S 分成两个大小相等的子集合 S_L 和 S_R，S_L 中的点在 L 左边，S_R 中的点在 L 右边。Conquer:递归地在 S_L 和 S_R 中找出周长最小的三角形点组合$(p_1, p_2, p_3)\in S_L$,$(q_1, q_2, q_3)\in S_R$，$C=min\{C(p_1, p_2, p_3), C(q_1, q_2, q_3)\}$。Merge:在 $L{:}x=m$ 左右划分 $x=m+C/2$ 和 $x=m-C/2$ 的临界区，如下图所示。由于左右临界区对称，因此分析一边，再将复杂度乘 2 即可。对左临界区，遍历所有点对，剔除距离大于 C 的点对，得到 a 和 b，d 是 a 和 b 连线的中点。由于 $ac+bc>2dc$，当 $dc=C/2$，ab 趋近于 0 时，

c 离 $L{:}x=m$ 最远。因此，对 d 点来说，至多需要考虑右临界区中的 6 个区域(参考二维空间

计算最近点对的算法)。对 d 点遍历右临界区的 6 个区域，组成三角形计算周长，若结果小于 C，则输出 abc 三个点。否则，输出 $C(p_1, p_2, p_3)$和 $C(q_1, q_2, q_3)$中更小的三个点。



对于任意点对 ab，我们只需计算右临界区中 6 个区域中的点和 ab 中点 d 的距离。

Merge 算法

把临界区中所有点集合投影到分割线 L 上
对于左临界区中的每个长度小于 C 的点对 ab，考察 a 和 b 的中点 d 和右临界区的每个点 c(这样的点至多 6 个)，组成三角形，计算出周长 C'，若 $C'<C$，则 $C=C'$
如果 C 发生过变化，与最后的 C 对应的点即为 abc，否则不存在 abc

时间复杂度: Preprocessing 阶段排序至少需要 $O(n\log n)$，Divide 阶段需要 $O(n)$，Conquer 阶段需要 $2T(n/2)$，Merge 阶段找出点对 ab 的复杂度为 $O(n^2/4)$，依次寻找 c 的复杂度为 $O(6n^2/4)$，对右临界区执行相同操作，因此 Merge 阶段总的复杂度为 $O(3n^2)$，总的递归方程为 $T(n)=O(1)$，$n=3$；$T(n)=2T(n/2)+ O(3n^2)+O(n\log n)+O(n)$，n>3，故时间复杂度为 $O(n^2)$。

4.思想:Divide:找出二叉树的重心，去掉它分成 2 个新二叉树。Conquer:递归求解每个新二叉树中满足 $\mathrm{dis}(x, y)<\tau$ 的顶点对个数。Merge:计算重心的父节点和子节点到它的距离，若小于 τ，则加入计算结果，然后输出最终的顶点对个数。

算法:**CalVertexNum**(T)

输入:含有 n 个顶点的加权二叉树 T 和正数 τ

输出: 满足 $\mathrm{dis}(x, y)<\tau$ 的顶点对个数

If 二叉树节点数为 2 **If** $\mathrm{dis}(x, y)<\tau$ count++
Else
找到二叉树的重心，去掉它划分成 2 个新二叉树 T_1,T_2
计算重心的父节点到它的距离 dist, **If** dist<τ count++
遍历重心的子节点，计算到它的距离 dist, **If** dist<τ count++
CalVertexNum(T_1)
CalVertexNum(T_2)
return count

时间复杂度:Divide 阶段寻找重心需要 $O(n)$，Conquer 阶段需要 $2T(n/2)$，Merge 阶段需要 O(1)，总的递推方程为 $T(n)=O(n)+2T(n/2)$，故时间复杂度为 $O(n\log n)$。

5.思想:Divide:找出数组 $A[1:n]$的中位数 $A[\lfloor n/2 \rfloor]$，据此分成集合 A_U 和 A_L，分别对应比它大的集合和比它小的集合，遍历 A_U 中元素，若下标比 m 的下标小，则个数+1。Conquer:递归在 A_U 和 A_L 中继续根据各自的中位数划分集合，当集合元素仅有 2 个时，若反序，则个数+1。Merge:输出反序个数。

算法 **CountReverse**(A)

输入:无序数组 A

输出:数组 A 的反序个数

If A.size=2 **If** $A[0]>A[1]$ count++
Else
比中位数 $A[\lfloor n/2 \rfloor]$大的元素组成集合 A_{U}，小的元素组成集合 A_{L}。
For i=1 **TO** A_{U}.size **DO**
If A.at[$A_{\mathrm{U}}[i]$]>A.at[m] count++ (A.at[n]返回的是 n 对于 A 的下标数)
CountReverse(A_{U})
CountReverse(A_{L})
return count

时间复杂度 :Divide 阶段查找中位数和划分均需要 $O(n)$，Conquer 阶段需要 $2T(n/2)$，Merge 阶段遍历 A_{U} 需要 $O(n/2)$，输出 count 需要 $O(1)$，总的递归方程为 $T(n)=2T(n/2)+O(2n)+O(n/2)$，故时间复杂度为 $O(n\log n)$。