《算法分析与设计》第 2 次作业答案

（答案摘自部分同学的优秀解答）

算法分析题

题目 1：求下列递推关系表示的算法复杂度 (当 $n=1$ 时，$T(n)=1$)。

(1)$T(n)=7T(\frac{n}{7})+n$

(2)$T(n)=4T(\frac{n}{2})+n^2$

(3)$T(n)=8T(\frac{n}{6})+n^{\frac{3}{2}}\log n$ (用主定理证明，选做)

答：

(1) 公式法：$T(n)=7T\left(\frac{n}{7}\right)+n$中，$k=7$，$m=7$，$f(n)=n$

$$\begin{aligned}T(n)&=n^{\log_m k}+\sum_{i=1}^{\log_m n-1}k^i f\left(\frac{n}{m^i}\right)\\&=n^{\log_7 7}+\sum_{i=1}^{\log_7 n-1}\left(\frac{7}{7}\right)^i n\\&=n^{\log_7 7}+\mathrm{nlog}_7(n-1)\\&=O(nlogn)\end{aligned}$$

(2) 公式法：$T(n)=4T\left(\frac{n}{2}\right)+n^2$中，$k=4$，$m=2$，$f(n)=n^2$

$$\begin{aligned}T(n)&=n^{\log_m k}+\sum_{i=1}^{\log_m n-1}k^i f\left(\frac{n}{m^i}\right)\\&=n^{\log_2 7}+\sum_{i=1}^{\log_2 n-1}\left(\frac{4}{4}\right)^i n^2\\&=n^{\log_2 7}+n^2\log_2(n-1)\\&=O(n^2\log n)\end{aligned}$$

(3) 因为存在一个n_0,使得对任意$n\geq n_0$,有

$$f(n)=n^{\frac{3}{2}}logn\geq n^{\frac{3}{2}}\geq n^{log_6 8+\varepsilon}$$

其中ε 存在且大于0.

所以可得：

$$f(n)=\Omega(n^{log_6 8+\varepsilon})$$

而对所有充分大的n,假设存在一个小于1的常数c，对不等式$8\times(\frac{n}{6})^{\frac{3}{2}}log\frac{n}{6}\leq c\times n^{\frac{3}{2}}\log n$,有：

$$\begin{aligned}&8\times(\frac{n}{6})^{\frac{3}{2}}log\frac{n}{6}\leq c\times n^{\frac{3}{2}}\log n\\\Leftrightarrow&(\frac{8}{6^{\frac{3}{2}}}-c)logn\leq\frac{8}{6^{\frac{3}{2}}}log6\\\Leftrightarrow&(8-6^{\frac{3}{2}}c)logn\leq 8log6\end{aligned}$$

显然可知当$c=\frac{8}{6^{\frac{3}{2}}}<1$时，上述不等式成立.
所以由主定理可知

$$T(n)=\Theta(n^{\frac{3}{2}}\log n)$$

题目 2：设 X[0:n-1] 和 Y[0:n-1] 为两个数组，每个数组中含有 n 个已排好序的数。试设计一个 $O(\log n)$ 时间的分治算法，找出 X 和 Y 的 $2n$ 个数的中位数，并证明算法的时间复杂性为 $O(\log n)$。

答：包括算法思路，伪代码和时间复杂度分析

算法思路：

对有序的X，Y数组，若需找到两数组的中位数，则需先对两数组各自的中位数X_{median},Y_{median}进行比较，若相同则该数即为所求，若$X_{median}<Y_{median}$则所求即为$[X_{median},X[n-1]]$与$[Y[0],Y_{median}]$中位数，而这两个数组同样是有序的，大于则同理。收敛条件为数组长度为1。以此可由递推解决问题。

伪代码：

算法 1 用分治法求两有序数组的中位数

```
function FindMedian(doubleX[], intXLeft, intXRight, doubleY[], intYLeft, intYRight)
    if XLeft == XRight and YLeft == YRight then
        return (XLeft + YLeft)/2
    end if
    XMedian ← (XLeft + XRight)/2
    YMedian ← (YLeft + YRight)/2
    XResult ← X's median number
    YResult ← Y's median number
    if XResult == YRsult then
        return XResult
    else if XResult > YRsult then
        return FindMedian(X, XLeft, XMedian, Y, YMedian, YRight)
    else
        return FindMedian(X, XMedian, XRight, Y, YLeft, YMedian)
    end if
end function
```

时间复杂度分析：

由递推可得该算法递推关系为：

$$T(n)=T(\frac{n}{2})+O(1)$$

所以由公式法易得，该算法时间复杂度为$\Theta(logn)$

算法实现题

题目1：问题描述：在与联盟的战斗中连续失败之后，帝国退居其最后据点。帝国依靠其强大的防御系统，击退了联盟的六次进攻。经过数次不眠之夜的思考，联盟将军亚瑟注意到防御系统的唯一弱点是其能源供应。该系统由N 个核电站供电，任何一个发生故障都将导致系统瘫痪。

这位将军很快就派N名特工袭击据点。 不幸的是，由于帝国空军的袭击，他们未能降落在预期的位置。作为一名经验丰富的将军，亚瑟很快意识到他需要重新安排计划。他现在想知道的第一件事是哪个特工离任何一个核电站最近。你是否可以帮助将军计算特工与核电站之间的最小距离？

算法思路：

该题可以抽象为求两个独立点集之间的最短距离的问题。所以类似最近点对问题，将点集合并后按横坐标大小排序，将合并后的点集均分为两份，问题即可转化为求在左侧点集的最短距离、在右侧点集的最短距离和在划分线两边的点的最短距离三者中最小的一个，若所取的点出自原来的同一点集，则认为其距离为无穷大。收敛条件为该点集只有一个点。由此可递归求解该问题。

具体算法思路如下：

假设合并之后的点集为$Q = \{p_0, p_0, \cdots, p_0\}$，我们需要输出距离最近的两个点$(p_r, p_s)$，因为要求两个点必须是不同集合的，所以计算距离时加上一个判断条件，如果属于相同集合返回无穷大即可。

$Dis(p_i, p_j)$为 Euclidean 距离，令$p_i = (x_i, y_i)$，$p_j = (x_j, y_j)$，则：

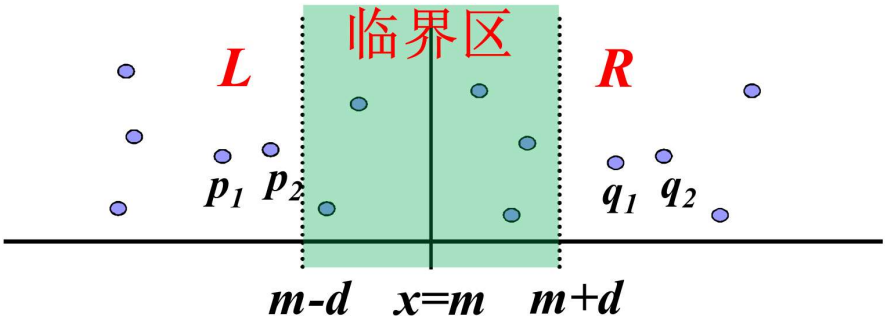
$$Dis(p_i, p_j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$$

然后将 Q 中的点按 x-坐标值进行排序；

分割过程：

1、 计算 Q 中各点 x-坐标的中位数 m。

2、 用垂线 $x=m$ 把点集合划分成两个大小相等的子集合 L 和 R， L 中点在 $x=m$ 左边，R 中点在 $x=m$ 右边。如下图所示：



合并过程：

1、 递归地在 L、R 中找出最接近点对：$(p1, p2) \in L$ ，$(q1, q2) \in R$。

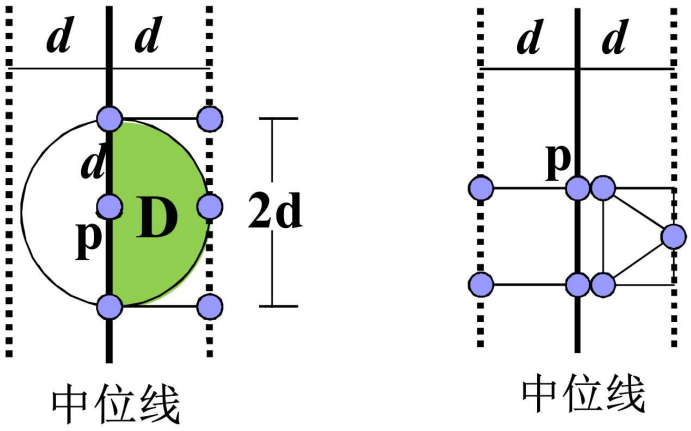
2、 $d = min\{Dis(p1, p2), Dis(q1, q2)\}$;

3、 在临界区查找距离小于 d 的点对 (pl, qr), $pl \in L$, $qr \in R$;

4、 如果找到，则(pl, qr)是 Q 中最近点对；

5、 否则(p1, p2)和(q1, q2) 中距离最小者为 Q 中最接近点对。

如何在临界区查找距离小于 d 的点对(pl, qr)，$pl \in L$，$qr \in R$？



由上图左图可见，形成的宽为 2d 的带状区间，最多可能有 n 个点，合并时间最坏情况下为 n^2。但是，中位线左侧和右侧中的点具有以下稀疏的性质，对于中位线左侧中的任意一点，中位线右侧中的点若与中位线左侧中的点的距离小于等于 d，则其必定落在绿色的半圆区域中 D 中；若把 D 扩大为 d*2d 的矩形区域，这样的区域内最多只需检查六个点，这六个点之间每两点的距离都大于等于 d（鸽巢原理，极端情况如上图右图所示），即对于任一点 p，最多只需计算 6 次距离可以了。因此，先将带状区间的点按 y 轴坐标排序，然后进行线性扫描，时间复杂度就近似线性的了。

一般的最近点对算法伪代码示例：

```
O(nlogn) 对Q按x轴从小到大排序
O(nlogn) 对Q中点按y轴从小到大排序放放Y中，并记录各点对应Q中的下标
        CPair(i, j, Q, Y) //返回数组Q中i到j的最近距离d，以及最近两点p,q
           n = j-i+1
           if m < 3 then return 最近距离d,及其点对(p,q);
           YL = Q中 i 到 i+⌈n/2⌉-1点按y轴从小到大排好的序;//不需重新排序了
           YR = Q中 i+⌈n/2⌉到 j 点按y轴从小到大排好的序;//不需重新排序了
T(n/2)     (d1, pr, ps) = CPair(i, i+⌈n/2⌉-1, Q, YL);
T(n/2)     (d2, pr', ps') = CPair(i+⌈n/2⌉, j, Q, YR);
           (d, pr, ps)记录d1和d2更小值
           x中位线=(Qx[i+⌈m/2⌉-1]+ Qx[i+⌈m/2⌉])/2
           Yd = 落于[x中位线-d, x中位线+d]带中按y轴从小到大排好的序;//不需重新排序
           对于Yd任一点，与前6个、后6个比较找最小的(d',p,q)
           if d'< d return (d',p,q)
           else return  (d, pr, ps)
```

一般的最近点对算法时间复杂度：

$$T(n) = \begin{cases} O(1) & n = 2 \\ 2T\left(\frac{n}{2}\right) + O(n) & n \geq 3 \end{cases}$$

求解可得$T(n) = O(nlogn)$