分治　先 divide，在小范围 conquer，最后 combine

⇒ recursively

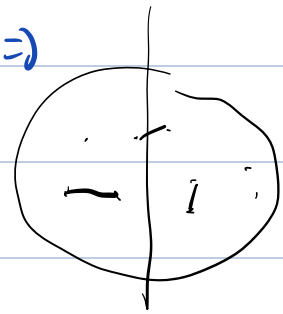
通用 recurrence：$T(N) = aT(N/b) + f(N)$

Closest Points Problems

N points. if the distance between 2 points is 0, then they are the closest points.

→ simple exhaustive search：$N(N-1)/2$ pairs　$T = O(N^2)$

⇒



left, right and cross 分支

left　cross　right　…

recurse

$T(N) = aT(N/b) + f(N)$ cross (if linear)

→ $T(N) = 2T(N/2) + cN$

$= 2[2T(N/2^2) + cN/2] + cN$

$= 2^2 T(N/2^2) + 2cN = \cdots$

$= 2^k T(N/2^k) + kcN$

$2^k = N$, $k = \log(N)$

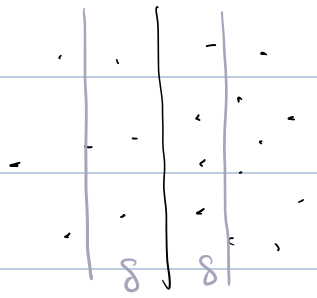
$= NT(1) + cN\log N = O(N\log N)$

if not linear：$T(N) = 2T(N/2) + cN^2$

→ $O(N^2)$

当 cross 太多，$f(N)$ 大，会导致最后算法复杂度较高。

§ Strip 之选取：



我们考虑 δ strip 内上点之间上距离(如果一个点在 δ 外，那么它上 cross 距离一定 > δ)

if NumPointInStrip = O(√N), we have:

↘ 我实用 O(N²) => O(√n) 伴性

```
for (i=0 ; i<NumPointInStrip; i++)
    for (j= i+1 ; j <NumPointInStrip; j++)
        if (Dist(Pi, Pj) < δ)  δ = Dist(Pi, Pj);
```

但实际上我们无能保证在√N内.

the worst case : NumPointInStrip = N

=> 我们对y方向也进行 δ 判断

```
for (i=0; i< NumPointInStrip; i++)
    for (j = i+1 ; j < NumPointsInStrip; j++)
        if (Dist_y(Pi, Pj) > δ)  break;
        else if (Dist(Pi, Pj) < δ)  δ = Dist(Pi, Pj);
```

Three methods for solving $T(N) = aT(N/b) + f(N)$

1. Substitution Method. 猜测，然后对 N, N/2 来验证. (数归)

$T(N) = 2T(\lfloor N/2 \rfloor) + N$

Guess: $T(N) = O(N\log N)$

假设对 all $m < N$ 都成立. 尤其 $m = \lfloor N/2 \rfloor$

$\therefore T(\lfloor N/2 \rfloor) \le c(\lfloor N/2 \rfloor \log \lfloor N/2 \rfloor)$ 假設一个存在 c.

$\therefore T(N) = 2T(N/2) + N \le 2c\lfloor N/2 \rfloor \log \lfloor N/2 \rfloor + N$

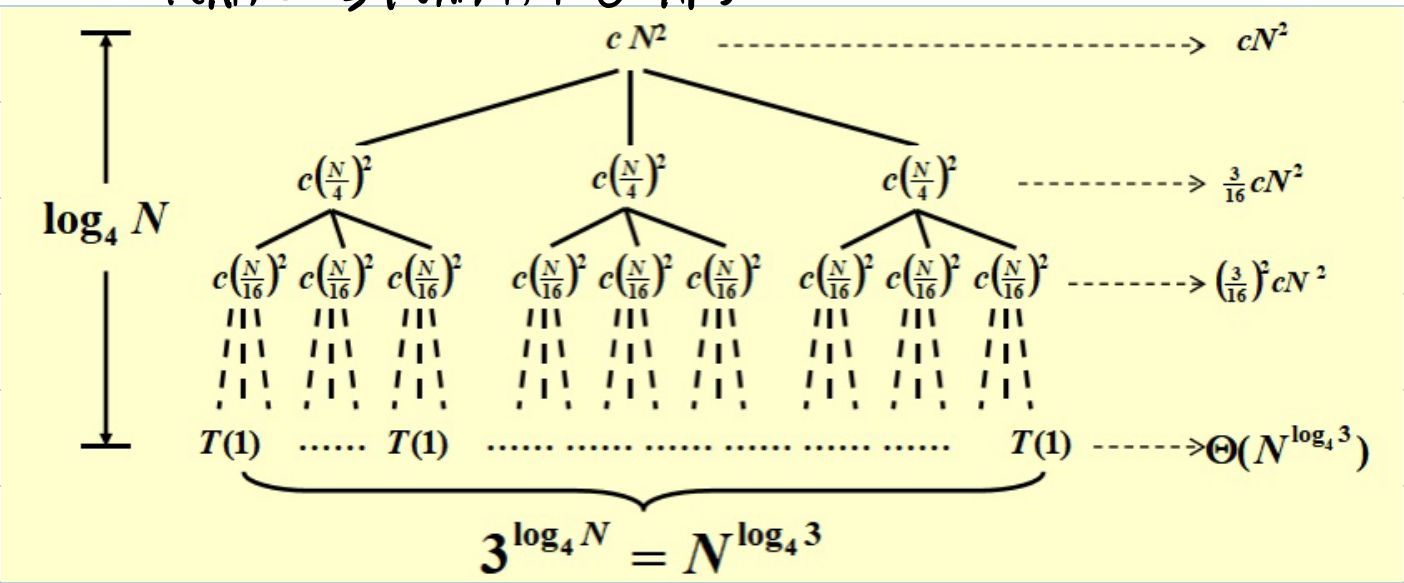
$\le cN \log \lfloor N/2 \rfloor + N$

$\le cN(\log N - \log 2) + N$

$\le cN \log N$ for $c \ge 1$

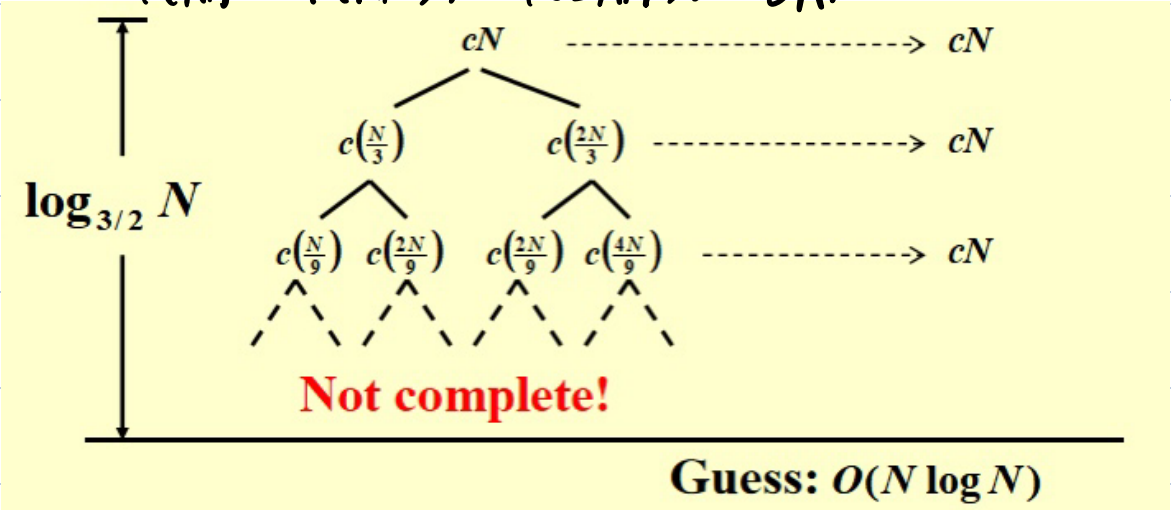
2. Recursion-tree method

$T(N) = 3T(N/4) + \Theta(N^2)$



$T(N) = \sum_{i=0}^{\log_4 N - 1} (\frac{3}{16})^i cN^2 + \Theta(N^{\log_4 3})$

$T(N) = T(N/3) + T(2N/3) + cN$



再用 substitution method:

$T(N) \le O(N\log N)$ $\quad T(N) \le dN\log N$

$T(N) \ge dT(N/3) + dT(2N/3) + cN$

$\le d(N/3)\log(N/3) + d(2N/3)\log(2N/3) + cN$

$= d(N/3)(\log N - \log 3) + d(2N/3)(\log N - \log\frac{3}{2}) + cN$

$= dN\log N - d(\frac{N}{3})\log 3 - d(\frac{2}{3}N)\log\frac{3}{2} + cN$

$\le dN\log N$ for $d \ge c/(\log_2 3 - \frac{3}{2})$

3. Master method. => 这个东西他是怎么做到跟在学数学一样?.

master method（主定理）

假设有递推关系式 $T(n) = aT(\frac{n}{b}) + f(n)$，其中 n 为问题的规模，a 为递推的子问题数量，$\frac{n}{b}$ 为每个子问题的规模（假设每个子问题的基本规模基本一样），$f(n)$ 为递推以外进行的计算工作。

$a \ge 1, b > 1$ 为常数，$f(n)$ 为函数，$T(n)$ 为非负整数。则有以下结果（分类讨论）：

(1) 若 $f(n) = O(n^{\log_b a - \varepsilon})$，$\varepsilon > 0$，那么 $T(n) = \Theta(n^{\log_b a})$

(2) 若 $f(n) = \Theta(n^{\log_b a})$，那么 $T(n) = \Theta(n^{\log_b a}\log n)$

(3) 若 $f(n) = \Omega(n^{\log_b a + \varepsilon})$，$\varepsilon > 0$，且对于某个常数 $c < 1$ 和所有充分大的 n 有 $af(\frac{n}{b}) \le cf(n)$，那么 $T(n) = \Theta(f(n))$

https://blog.csdn.net/qq_43826212/article/details/