lesson7_divide and conquer

1.递归

- **Divide** the problem into a number of sub-problems；　(分解)
- **Conquer** the sub-problems by solving them recursively；(递归)
- **Combine** the solutions to the sub-problems into the solution for the original problem；(合解)

General recurrence:　T(N) = aT(N/b) + f(N)；

例子:

求和最大的子序列: O(NlogN);

tree traversals : O(N)

merge sort and quick sort: O(NlogN)

本章重点是时间复杂度分析，给定T(N) = aT(N/b) + f(N)，求T(N)

2.closest point

找到最近的两个点

- 方法一：遍历 N(N-1)/2 个点，时间复杂度 O(N^2)
- 方法二：根据x坐标分成两部分，找到第一部分，第二部分和第一二部分之间的的最短距离（left , right, middle）

时间复杂度：

1. 有可能left , right, middle情况相同，导致时间复杂度是O(N^2)；
2. 定义 F(N) 找到cross distance 的时间

T(N) = 2 T(N/2) +c F(N) ;

如果F(N) = O(N);

Recall:
$$
\begin{aligned}
T(N) &= 2T(N/2) + cN \\
&= 2[\,2T(N/2^2) + cN/2\,] + cN \\
&= 2^2\,T(N/2^2) + 2cN \\
&= \ldots\ldots \\
&= 2^k\,T(N/2^k) + kcN \\
&= N + c\,N\log N \;= O(\,N\log N\,)
\end{aligned}
$$

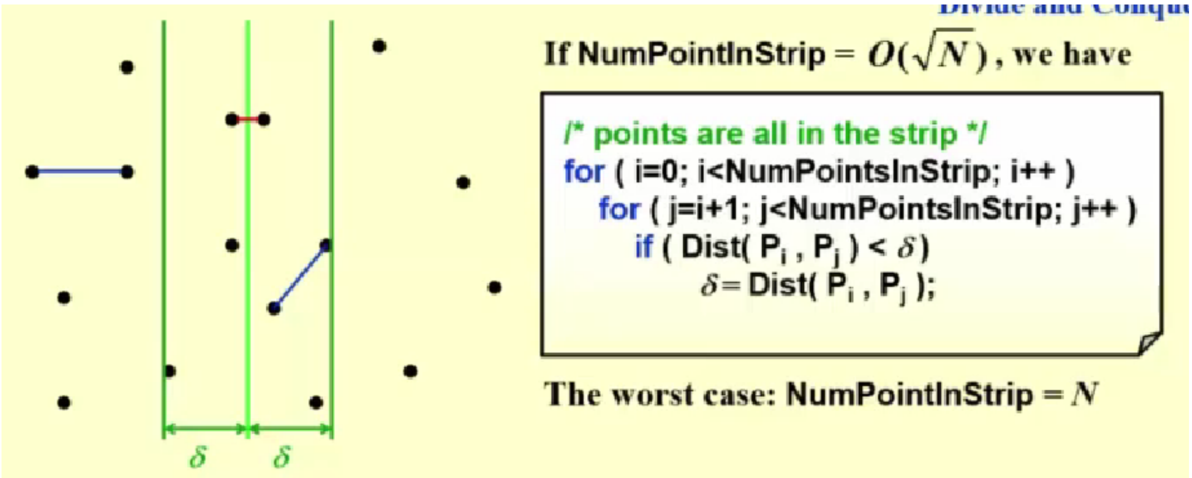
但如果 T(N) = 2(T(N/2)) +c N^2 ; 那么T(N) = O(N^2) ;

所以要想办法降低F(N)；

方法1：设置δ-strip

δ= min(left_distance , right_distance)；

strip内点的数量 O(N^0.5) ,最糟糕的情况是O(N)，那么时间复杂度还是N^2

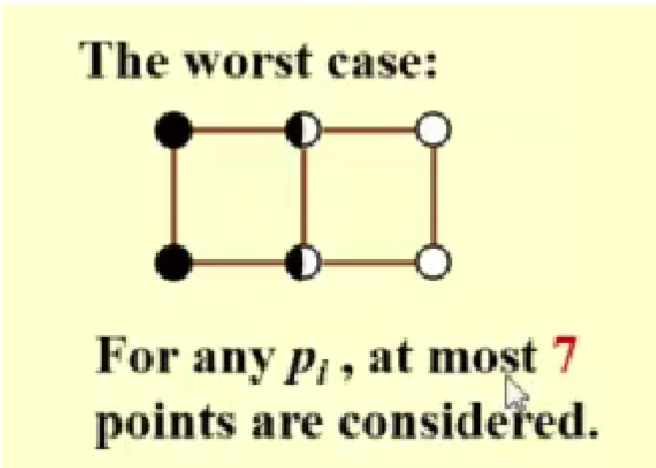


方法2：设置δ-strip （竖 + 横）

遍历竖的范围内的每一个点，遍历时再遍历横的范围内的每一个点，更新δ

```
/* points are all in the strip */
/* and sorted by y coordinates */
for ( i = 0; i < NumPointsInStrip; 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 );
```

所以最终最多只有七个顶点需要确认



F(N) = O(N)

3.Substitution

三种方法解决 T(N) = aT(N/b) + f(N);

- substitution method 替代法
- recursion tree method 递归树
- master method

tip:N/b 是不是整数无所谓；T(N) = θ （1）对一些小n来说成立

substitution method: 递归法证明

guess + prove by induction

eg:

【Example】 $T(N) = 2\,T(\lfloor N/2 \rfloor) + N$

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

Proof: Assume it is true for all $m < N$, in particular for $m = \lfloor N/2 \rfloor$.

Then there exists a constant $c > 0$ so that $T(\lfloor N/2 \rfloor) \le c \lfloor N/2 \rfloor \log \lfloor N/2 \rfloor$

Substituting into the recurrence:

$$
\begin{aligned}
T(N) &= 2\,T(\lfloor N/2 \rfloor) + N \\
&\le 2\,c \lfloor N/2 \rfloor \log \lfloor N/2 \rfloor + N \\
&\le c\,N(\log N - \log 2) + N \\
&\le c\,N \log N \quad \text{for } c \ge 1
\end{aligned}
$$

tips:

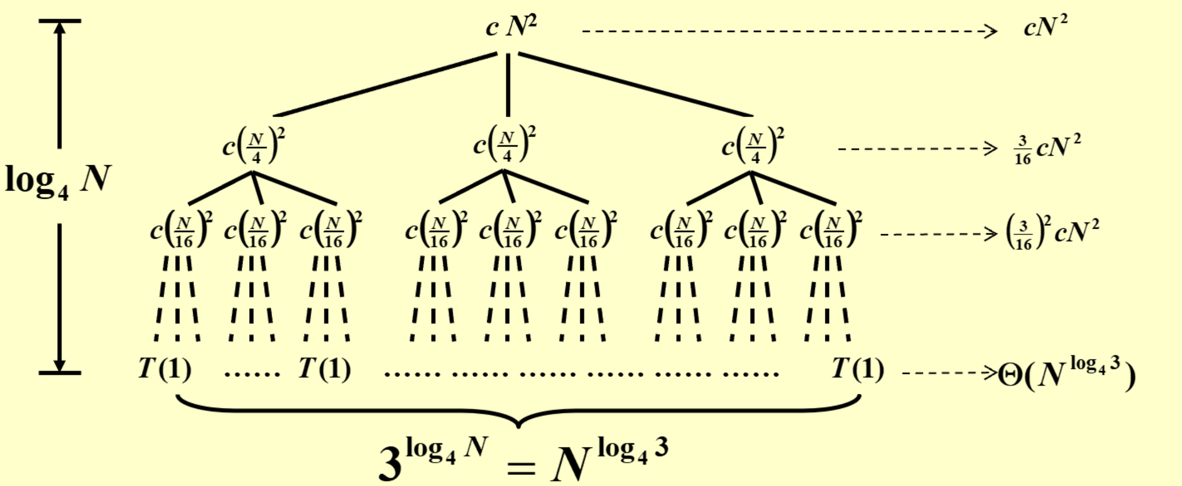
- c can be sufficiently large~
- 不一定要从N=2,3开始，可以设置大一些

如果猜错了，那么证明就会有问题；所以关键是good guess

4.recursion method

eg1 (simple,balanced tree)

【Example】 $T(N) = 3\,T(N/4) + \Theta(N^2)$



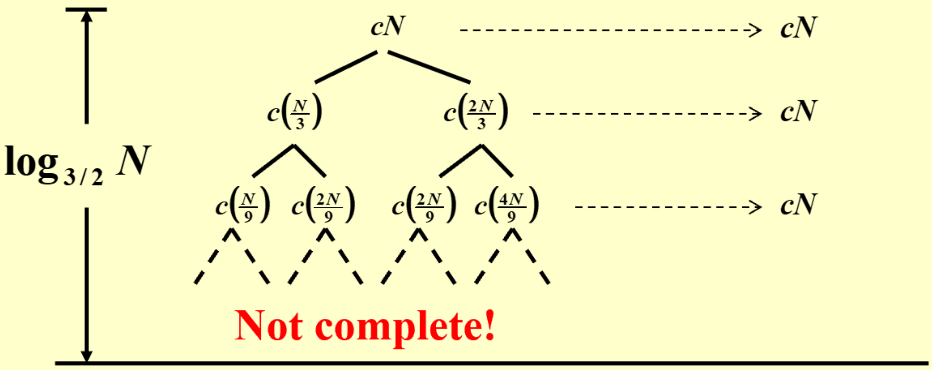
$$T(N) = \sum_{i=0}^{\log_4 N - 1} \left(\tfrac{3}{16}\right)^i cN^2 + \Theta(N^{\log_4 3})$$

height(Tree) = log 4 N

T(N) = O(N^2)

eg2:(unbalanced tree)

【Example】 $T(N) = T(N/3) + T(2N/3) + cN$



Guess: $O(N \log N)$

Proof by substitution:

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

$$= dN\log N - dN(\log_2 3 - \tfrac{2}{3}) + cN \le dN\log N$$

$$\text{for } d \ge c/(\log_2 3 - \tfrac{2}{3})$$

height(Tree) = height of right most path = log 1.5 N

再用递归法证明 T(N) = O(N logN)

5.master method

form1:

【Master Theorem】 Let $a \ge 1$ and $b > 1$ be constants, let $f(N)$ be a function, and let $T(N)$ be defined on the nonnegative integers by the recurrence $T(N) = aT(N/b) + f(N)$. Then:

1. If $f(N) = O(N^{\log_b a - \varepsilon})$ for some constant $\varepsilon > 0$, then $T(N) = \Theta(N^{\log_b a})$
2. If $f(N) = \Theta(N^{\log_b a})$, then $T(N) = \Theta(N^{\log_b a} \log N)$
3. If $f(N) = \Omega(N^{\log_b a + \varepsilon})$ for some constant $\varepsilon > 0$, and if $af(N/b) < cf(N)$ for some constant $c < 1$ and all sufficiently large N, then $T(N) = \Theta(f(N))$

regularity condition

f(N)代表combine的时间, N^ (log b a) 代表divide的时间，两者共同决定时间复杂度，

如果两者等价，那么考虑层数 log N ,时间复杂度就是O(N^ log b a *(log N)),

如果两者不等价，时间复杂度就由大的那个决定

eg1:

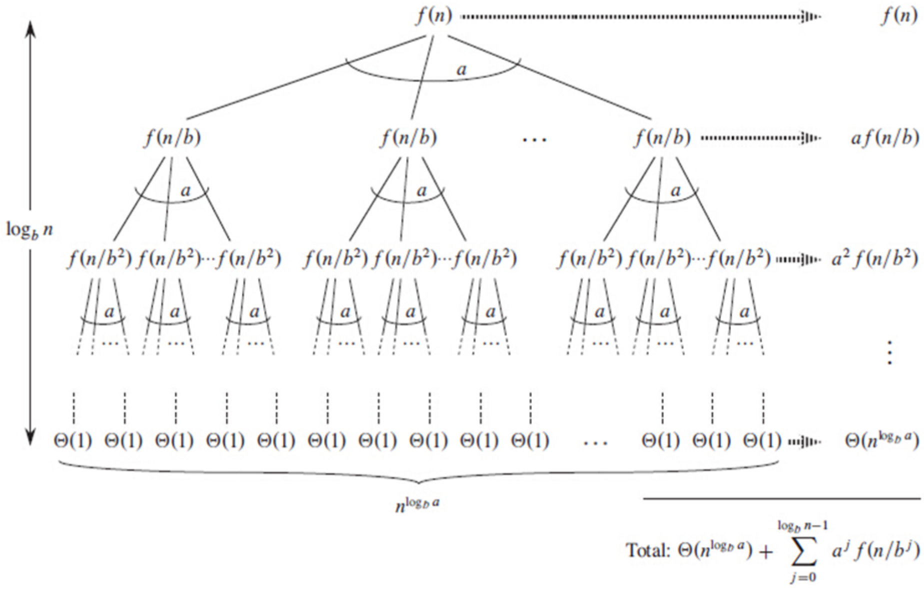
merge sort : a=b=2 and case 2,T(N)= θ (N log N)

eg2:

a=b=2 and f(N)=N log N,不适用任何一个case,case2 θ不对， case3 找不到合适的ε使得N^ε等价log N . Master theorem不适用该情况

证明:

假设 n=b^k



case1 证明:

For case 1 where $f(N) = O(N^{\log_b a-\varepsilon})$

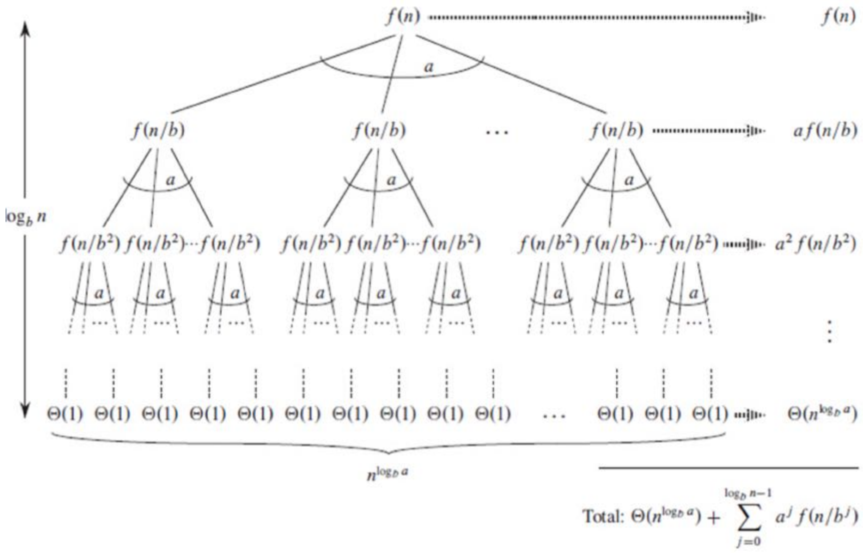
$$\sum_{j=0}^{\log_b N-1} a^j f(N/b^j) = O(\sum_{j=0}^{\log_b N-1} a^j \left(\frac{N}{b^j}\right)^{\log_b a-\varepsilon}) = O(N^{\log_b a-\varepsilon} \sum_{j=0}^{\log_b N-1} \left(\frac{ab^\varepsilon}{\underbrace{b^{\log_b a}}_{=a}}\right)^j)$$

$$= O(N^{\log_b a-\varepsilon} \sum_{j=0}^{\log_b N-1} (b^\varepsilon)^j) = O(N^{\log_b a-\varepsilon} \frac{b^{\varepsilon \log_b N}-1}{b^\varepsilon-1})$$

$$= O(N^{\log_b a-\varepsilon} N^\varepsilon) = O(N^{\log_b a})$$

$$T(N) = \Theta(N^{\log_b a}) + O(N^{\log_b a}) = \Theta(N^{\log_b a})$$

case2 证明:



for case 2 where $f(n) = \Theta(N^{\log_b a})$

$$\sum_{j=0}^{\log_b n-1} a^j f(n/b^j) = \sum_{j=0}^{\log_b n-1} a^j \, \Theta\left(\left(\frac{n}{b^j}\right)^{\log_b a}\right)$$

$$= \sum n^{\log_b a} \cdot \Theta\left(\frac{a^j}{(b^j)^{\log_b a}}\right)$$

$$= \sum n^{\log_b a} \cdot \Theta\left(\frac{a^j}{a^j}\right)$$

$$= \Theta(\log_b n \cdot n^{\log_b a})$$

$$T(N) = \Theta(n^{\log_b a}) + \Theta(\log_b n \cdot n^{\log_b a})$$

$$= \Theta((1+\log_b n) \cdot n^{\log_b a})$$

$$= \Theta(n^{\log_b a} \cdot \log n)$$

$\Rightarrow$

$$T(N) = \Theta(N^{\log_b a} \log N)$$

form2:

【Master Theorem】The recurrence $T(N) = aT(N/b) + f(N)$ can be solved as follows:

1. **If $af(N/b) = \kappa f(N)$ for some constant $\kappa < 1$, then $T(N) = \Theta(f(N))$**
2. **If $af(N/b) = K f(N)$ for some constant $K > 1$, then $T(N) = \Theta(N^{\log_b a})$**
3. **If $af(N/b) = f(N)$, then $T(N) = \Theta(f(N)\log_b N)$**

反例:不可以被该形式cover,但可以被之前的形式cover

【Example】 $a = 4, b = 2, f(N) = N\log N$

$$af(N/b) = 4(N/2)\log(N/2) = 2N\log N - 2N \ ?$$

$$f(N) = N\log N \qquad O(N^{\log_b a - \varepsilon}) = O(N^{2-\varepsilon})$$

$\rightarrow T = O(N^2)$

form3:

针对特定形式

【Theorem】 The solution to the equation

$$T(N) = a\,T(N/b) + \Theta(N^k \log^p N),$$

where $a \geq 1$, $b > 1$, and $p \geq 0$ is

$$T(N) = \begin{cases} O(N^{\log_b a}) & \text{if } a > b^k \\ O(N^k \log^{p+1} N) & \text{if } a = b^k \\ O(N^k \log^p N) & \text{if } a < b^k \end{cases}$$

三个例子:

【Example】 Mergesort has $a = b = 2$, $p = 0$ and $k = 1$.

$\rightarrow T = O(N\log N)$

【Example】 Divide with $a = 3$, and $b = 2$ for each recursion;
Conquer with $O(N)$ – that is, $k = 1$ and $p = 0$.

$\rightarrow T = O(N^{1.59})$

If conquer takes $O(N^2)$ then $T = O(N^2)$.

【Example】 $a = b = 2$, $f(N) = N\log N$ $\rightarrow T = O(N\log^2 N)$