Divide and conquer

Gou Guanglei(苟光磊)
ggl@cqut.edu.cn

Divide-and-Conquer design paradigm

- Divide: divide a problem into subproblems
- Conquer: solve the problems recursively
- Combine: combine the subproblems solutions appropriately

Binary search

Find an element in a **sorted** array:

1. ***Divide:*** Check middle element.
2. ***Conquer:*** Recursively search 1subarray.
3. ***Combine:*** Trivial.

Recurrence for binary search

$$T(n) = 1T(n/2) + \Theta(1)$$

$$n^{\log_b a} = n^{\log_2 1} = n^0 = 1 \Rightarrow \text{CASE } 2\ (k = 0)$$
$$\Rightarrow T(n) = \Theta(\lg n).$$

Powering a number

Problem: Compute a_n, where $n \in \mathbb{N}$.
Naive algorithm: $\Theta(n)$.
Divide-and-conquer algorithm:

$$a^n = \begin{cases} a^{n/2} \cdot a^{n/2} & \text{if } n \text{ is even;} \\ a^{(n-1)/2} \cdot a^{(n-1)/2} \cdot a & \text{if } n \text{ is odd.} \end{cases}$$

$$T(n) = T(n/2) + \Theta(1) \Rightarrow T(n) = \Theta(\lg n).$$

Fibonacci numbers

Recursive definition:

$$F_n = \begin{cases} 0 & \text{if } n = 0; \\ 1 & \text{if } n = 1; \\ F_{n-1} + F_{n-2} & \text{if } n \geq 2. \end{cases}$$

$0 \quad 1 \quad 1 \quad 2 \quad 3 \quad 5 \quad 8 \quad 13 \quad 21 \quad 34 \quad \cdots$

Naive recursive algorithm: $\Omega(\varphi^n)$ (exponential time), where φ= (1+sqrt(5))/2 is the ***golden ratio***.

Recursive Squaring

Theorem: $\begin{bmatrix} F_{n+1} & F_n \\ F_n & F_{n-1} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^n.$

Proof of theorem. (Induction on n.)

Base ($n = 1$): $\begin{bmatrix} F_2 & F_1 \\ F_1 & F_0 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^1.$

Inductive step ($n \geq 2$):

$$\begin{bmatrix} F_{n+1} & F_n \\ F_n & F_{n-1} \end{bmatrix} = \begin{bmatrix} F_n & F_{n-1} \\ F_{n-1} & F_{n-2} \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^{n-1} \cdot \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^{n}$$

$\blacksquare$

multiply two n-bit numbers

- Problem : multiply two n-bit numbers.

    ex) 41× 42 (in binary, 101001 × 101010).

- Add 41 to itself 42 times :

    $\Theta(2^n)$ additions.

- Better algorithm?

```
       101001 = 41
x      101010 = 42
-------------
      1010010
     101001
+ 101001
-------------
  11010111010 = 1722
```

Divide-and-Conquer Multiplication

$X = 2^{n/2}A + B$

A	B

$Y = 2^{n/2}C + D$

C	D

$$XY = 2^n AC + 2^{n/2}BC + 2^{n/2}AD + BD.$$

$$T(n) = 4T(n/2) + cn$$

Karatsuba algorithm

$$XY = 2^n AC + 2^{n/2} BC + 2^{n/2} AD + BD$$

$$(2^n - 2^{n/2})AC + 2^{n/2}(A+B)(C+D) + (1 - 2^{n/2})BD$$

$$T(n) = 3T(n/2) + c'n,$$

$$O(n^{\log_2 3}) \approx O(n^{1.585})$$

Matrix Multiplication

$$\left.\begin{array}{ll}\textbf{Input:} & A = [a_{ij}], B = [b_{ij}]. \\ \textbf{Output:} & C = [c_{ij}] = A \cdot B.\end{array}\right\} \quad i, j = 1, 2, \ldots, n.$$

$$\begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1n} \\ c_{21} & c_{22} & \cdots & c_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ c_{n1} & c_{n2} & \cdots & c_{nn} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \cdot \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1n} \\ b_{21} & b_{22} & \cdots & b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \cdots & b_{nn} \end{bmatrix}$$

$$c_{ij} = \sum_{k=1}^{n} a_{ik} \cdot b_{kj}$$

Standard algorithm

for $i \leftarrow 1$ **to** n
    do for $j \leftarrow 1$ **to** n
        do $c_{ij} \leftarrow 0$
            for $k \leftarrow 1$ **to** n
                do $c_{ij} \leftarrow c_{ij} + a_{ik} \cdot b_{kj}$

Divide-and-Conquer algorithm

IDEA:

$n \times n$ matrix = 2×2 matrix of $(n/2) \times (n/2)$ submatrices:

$$\left[\begin{array}{c|c} r & s \\ \hline t & u \end{array}\right] = \left[\begin{array}{c|c} a & b \\ \hline c & d \end{array}\right] \cdot \left[\begin{array}{c|c} e & f \\ \hline g & h \end{array}\right]$$

$$C \quad = \quad A \quad \cdot \quad B$$

$$\left.\begin{aligned} r &= ae + bg \\ s &= af + bh \\ t &= ce + dg \\ u &= cf + dh \end{aligned}\right\} \begin{array}{l} 8 \text{ mults of } (n/2)\times(n/2) \text{ submatrices} \\ 4 \text{ adds of } (n/2)\times(n/2) \text{ submatrices} \end{array}$$

Analysis

$$T(n) = 8\,T(n/2) + \Theta(n^2)$$

submatrices

submatrix size

work adding submatrices

Strassen's idea

Multiply 2×2 matrices with only 7 recursive mults.

$$
\begin{aligned}
P_1 &= a \cdot (f - h) \\
P_2 &= (a + b) \cdot h \\
P_3 &= (c + d) \cdot e \\
P_4 &= d \cdot (g - e) \\
P_5 &= (a + d) \cdot (e + h) \\
P_6 &= (b - d) \cdot (g + h) \\
P_7 &= (a - c) \cdot (e + f)
\end{aligned}
$$

$$
\begin{aligned}
r &= P_5 + P_4 - P_2 + P_6 \\
s &= P_1 + P_2 \\
t &= P_3 + P_4 \\
u &= P_5 + P_1 - P_3 - P_7
\end{aligned}
$$

Strassen's algorithm

1. ***Divide:*** Partition A and B into $(n/2)\times(n/2)$ submatrices. Form terms to be multiplied using $+$ and $-$.
2. ***Conquer:*** Perform 7 multiplications of $(n/2)\times(n/2)$ submatrices recursively.
3. ***Combine:*** Form C using $+$ and $-$ on $(n/2)\times(n/2)$ submatrices.

$$T(n) = 7\,T(n/2) + \Theta(n^2)$$

$$\boldsymbol{T(n) = \Theta(n^{\lg 7})}$$

7 mults, 18 adds/subs.
Note: No reliance on commutativityof mult!

Conclusion

- Divide and conquer is just one of several powerful techniques for algorithm design.
- Divide-and-conquer algorithms can be analyzed using recurrences and the master method (so practice this math).
- The divide-and-conquer strategy often leads to efficient algorithms.