改进分治算法的途径1：减少子问题数

减少子问题个数的依据

分治算法的时间复杂度方程

$$W(n) = aW(n/b) + d(n)$$

a：子问题数，n/b：子问题规模，
$d(n)$：划分与综合工作量.

当 a 较大, b 较小, $d(n)$ 不大时,方程的解：

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

减少a是降低函数$W(n)$的阶的途径.

利用子问题的依赖关系，使某些子问题的解通过组合其他子问题的解而得到.

例1：整数位乘问题

输入：X, Y 是 n 位二进制数， $n = 2^k$

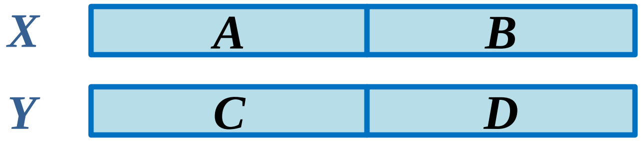
输出：XY

普通乘法：需要 $O(n^2)$ 次位乘运算

简单划分：令

$$X = A2^{n/2} + B, \quad Y = C2^{n/2} + D.$$

$$XY = \underline{AC}\, 2^n + (\underline{AD} + \underline{BC})\, 2^{n/2} + \underline{BD}$$



$$W(n) = 4W(n/2) + O(n) \Rightarrow W(n) = O(n^2)$$

减少子问题个数

子问题间的依赖关系：代数变换

$AD+BC = \underline{(A-B)(D-C)} + \underline{AC} + \underline{BD}$

算法复杂度

$$W(n) = 3\,W(n/2) + cn$$

$$W(1) = 1$$

方程的解

$$W(n) = O(n^{\log 3}) = O(n^{1.59})$$

例2：矩阵相乘问题

输入：A, B 为 n 阶矩阵，$n = 2^k$

输出：$C = AB$

通常矩阵乘法：

C 中有 n^2 个元素

每个元素需要做 n 次乘法

以元素相乘为基本运算

$$W(n) = O(n^3)$$

简单分治算法

分治法　将矩阵分块，得

$$\begin{pmatrix} \boxed{A_{11} \quad A_{12}} \\ A_{21} \quad A_{22} \end{pmatrix} \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix} = \begin{pmatrix} \boxed{C_{11}} & C_{12} \\ C_{21} & C_{22} \end{pmatrix}$$

其中

$$C_{11} = \underline{A_{11}B_{11}} + \underline{A_{12}B_{21}} \qquad C_{12} = \underline{A_{11}B_{12}} + \underline{A_{12}B_{22}}$$

$$C_{21} = \underline{A_{21}B_{11}} + \underline{A_{22}B_{21}} \qquad C_{22} = \underline{A_{21}B_{12}} + \underline{A_{22}B_{22}}$$

递推方程　$W(n) = 8\ W(n/2) + cn^2$

$W(1) = 1$

解　$W(n) = O(n^3)$.

Strassen 矩阵乘法

变换方法：

设计 $M_1, M_2, \ldots, M_7$，对应7个子问题

$$M_1 = A_{11}(B_{12} - B_{22})$$

$$M_2 = (A_{11} + A_{12})B_{22}$$

$$M_3 = (A_{21} + A_{22})B_{11}$$

$$M_4 = A_{22}(B_{21} - B_{11})$$

$$M_5 = (A_{11} + A_{22})(B_{11} + B_{22})$$

$$M_6 = (A_{12} - A_{22})(B_{21} + B_{22})$$

$$M_7 = (A_{11} - A_{21})(B_{11} + B_{12})$$

Strassen 矩阵乘法(续)

利用中间矩阵，得到结果矩阵

$$C_{11} = M_5 + M_4 - M_2 + M_6$$
$$C_{12} = M_1 + M_2$$
$$C_{21} = M_3 + M_4$$
$$C_{22} = M_5 + M_1 - M_3 - M_7$$

时间复杂度函数：

$$\boxed{\begin{array}{l} W(n) = 7\,W(n/2) + 18(n/2)^2 \\ W(1) = 1 \end{array}}$$

解 $W(n) = O(n^{\log 7}) = O(n^{2.8075})$

矩阵乘法的研究及应用

矩阵乘法问题的难度：

- **Coppersmith–Winograd算法：$O(n^{2.376})$**
  目前为止最好的上界
- 目前为止最好的下界是：$\Omega(n^2)$

应用：

- 科学计算、图像处理、数据挖掘等
- 回归、聚类、主成分分析、决策树
  等挖掘算法常涉及大规模矩阵运算

改进途径小结

- 适用于：子问题个数多，划分和综合工作量不太大，时间复杂度函数

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

- 利用子问题依赖关系，用某些子问题解的代数表达式表示另一些子问题的解，减少独立计算子问题个数.

- 综合解的工作量可能会增加，但增加的工作量不影响 $W(n)$的阶.