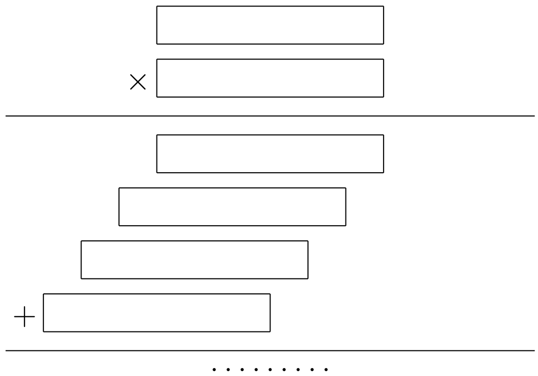
Divide and conquer

How to multiply 2 large integers (of n digits each)?
Elementary method : use the grid method for written multiplication :



$: \Theta(n^2)$

We can do better!

$$
\begin{aligned}
x \times y &= (\underbrace{x_0}_{n/2 \text{ digits}} + 10^{\lfloor n/2 \rfloor} \underbrace{x_1}_{n/2}) \times (\underbrace{y_0}_{n/2} + 10^{\lfloor n/2 \rfloor} \underbrace{y_1}_{n/2}) \\
&= x_0 \times y_0 + 10^{\lfloor n/2 \rfloor}(x_1 \times y_0 + x_0 \times y_1) + 10^{2\lfloor n/2 \rfloor}(x_1 \times y_1)
\end{aligned}
$$

Complexity : $t_n = 4t_{\lfloor n/2 \rfloor} + \Theta(n)$
Solution : if we assume $n \approx 2^k$, we have :

$$
\begin{aligned}
t_n &= 4t_{n/2} + \Theta(n) \\
&= 16t_{n/4} + 4\Theta(n/2) + \Theta(n) \\
&= 64t_{n/8} + \underbrace{16\Theta(n/4)}_{\Theta(4n)} + \underbrace{4\Theta(n/2)}_{\Theta(2n)} + \underbrace{\Theta(n)}_{\Theta(n)} \\
&= \dots \\
&= 2^{2k} \underbrace{t_{n/2^k}}_{\mathcal{O}(1)} + \Theta(\underbrace{(2^{k-1} + 2^{k-2} + \dots + 1)}_{2^k - 1} n) = \Theta(n^2)
\end{aligned}
$$

This is not better.
This kind of recurrence is solved by the *master theorem.*

Theorem 1 (Master)
For the recurrence $t_n = at_{n/b} + f(n)$ *(or* $t_{\lfloor n/b \rfloor}$ *or* $t_{\lceil n/b \rceil}$*) :*

a) If $f(n) \in \mathcal{O}\left(n^{\log_b a - \epsilon}\right)$ *for some* $\epsilon > 0$*, then* $t_n \in \Theta\left(n^{\log_b a}\right)$*;*

b) If $f(n) \in \Theta\left(n^{\log_b a}\right)$*, then* $t_n \in \Theta\left(n^{\log_b a} \log n\right)$*;*

c) If $f(n) \in \Omega\left(n^{\log_b a + \epsilon}\right)$ *for some* $\epsilon > 0$*,* $af(n/b) \leq cf(n)$ *for some* $c < 1$ *and* n *is large enough , then* $t_n \in \Theta(f(n))$*.*

Examples

- $t_n = 2t_{n/2} + \Theta(n)$: $a = 2$, $b = 2$ and $\log_2 2 = 1$
  By ??), $f(n) \in \Theta\left(n^{\log_b a}\right) \to t_n \in \Theta(n \log n)$

- $t_n = 4t_{n/2} + \Theta(n)$: $a = 4$, $b = 2$ and $\log_2 4 = 2$
  By ??), $f(n) = \Theta\left(n^{\log_2 4 - 1}\right) \to t_n = \Theta\left(n^{\log_b a}\right) = \Theta\left(n^2\right)$

It is possible to make only 3 multiplications of $\frac{n}{2}$-digits numbers.

`Multiply(x,y)` (Karastuba-Ofman, 1962)

$$\begin{aligned} x &= x_0 + 10^{\lfloor n/2 \rfloor} x_1 \\ y &= y_0 + 10^{\lfloor n/2 \rfloor} y_1 \\ M &= (x_0 + y_0) \times (x_1 + y_1) \\ x \times y &= x_0 \times y_0 + 10^{\lfloor n/2 \rfloor}(M - x_0 \times y_0 - x_1 \times y_1) + 10^{2\lfloor n/2 \rfloor} x_1 \times y_1 \end{aligned}$$

Time complexity :

$$t_n = 3t_{n/2} + \Theta(n) \Rightarrow t_n = \Theta\left(n^{\log_2 3}\right) \approx \Theta\left(n^{1.585}\right)$$

We can do even better!→ Schönhage-Strassen (1971) : $\mathcal{O}(n \log(n) \log\log(n))$

How can we multiply two n-by-n matrices?
Elementary algorithm : $\Theta(n^3)$ elementary operations.
Is block multiplication better?

$$A = \left(\begin{array}{c|c} A_{11} & A_{12} \\ \hline A_{21} & A_{22} \end{array}\right) \qquad B = \left(\begin{array}{c|c} B_{11} & B_{12} \\ \hline B_{21} & B_{22} \end{array}\right)$$

$$\to AB = \begin{pmatrix} A_{11}B_{11} + A_{12}B_{21} & A_{11}B_{12} + A_{12}B_{22} \\ A_{21}B_{11} + A_{22}B_{21} & A_{21}B_{12} + A_{22}B_{22} \end{pmatrix}$$

Time complexity : 8 products of $\frac{n}{2}$-by-$\frac{n}{2}$ matrices.

$$t_n = 8t_{n/2} + \Theta(n^2) = \Theta\left(n^{\log_2 8}\right) = \Theta\left(n^3\right)$$

This is not better. We can do it with only 7 products (Strassen's algorithm, 1969) :

$$\begin{cases} M_1 = (A_{21} + A_{22} - A_{11})(B_{22} - B_{12} + B_{11}) \\ M_2 = A_{11}B_{11} \\ M_3 = A_{12}B_{21} \\ M_4 = (A_{11} - A_{21})(B_{22} - B_{12}) \\ M_5 = (A_{21} + A_{22})(B_{12} - B_{11}) \\ M_6 = (A_{12} - A_{21} + A_{11} - A_{22})B_{22} \\ M_7 = A_{22} \end{cases} \Rightarrow AB = \begin{pmatrix} M_2 + M_3 & M_1 + M_2 + M_5 + M_6 \\ M_1 + M_2 + M_4 - M_7 & M_1 + M_2 + M_4 + M_5 \end{pmatrix}$$

Time complexity : $t_n = 7t_{n/2} + \Theta(n^2) = \Theta\left(n^{\log_2 7}\right) = \Theta\left(n^{2.81}\right)$
We can do even better!→ Coppersmith-Winograd (1981) : $\Theta(n^{2.376})$
Best one? Unknown but $\Omega(n^2)$

What is the cost of matrix inversion?
Elementary methods : $\Theta(n^3)$

Theorem 2
Matrix inversion and multiplication have the same complexity

Proof Let $I(n)$ be the (worst-case) complexity of inversion (i.e. of the best possible algorithm) and let $M(n)$ be the (worst-case) complexity of multiplication (i.e. of the best possible algorithm).
We want to show that $I(n) \in \Theta(M(n))$.

a) $M(n) \in \mathcal{O}(I(n))$? We reduce multiplication to inversion.

$$D = \underbrace{\begin{pmatrix} I_n & A & 0 \\ 0 & I_n & B \\ 0 & 0 & I_n \end{pmatrix}}_{3n*3n} \to D^{-1} = \begin{pmatrix} I_n & -A & AB \\ 0 & I_n & -B \\ 0 & 0 & I_n \end{pmatrix}$$

$$\to M(n) = \mathcal{O}(I(3n)) \qquad\qquad \Rightarrow M(n) \in \mathcal{O}(I(n))$$

Since $I(n) \in \mathcal{O}(n^3)$, $I(3n) \in \mathcal{O}(I(n))$. (Because n is multiplied by 27)

$(*)$ $I(n) \in \mathcal{O}(M(n))$?

We reduce inversion to "a few" multiplications.

$(*)$ First assume $A = A^T \succcurlyeq 0$ (symetric, positive-definite).

$$A = \begin{pmatrix} B & C^T \\ C & D \end{pmatrix} \qquad , \qquad \begin{matrix} B = B^T \succcurlyeq 0 \\ C = C^T \succcurlyeq 0 \end{matrix}$$

$$A^{-1} = \begin{pmatrix} B^{-1} + B^{-1}C^TS^{-1}CB^{-1} & -B^{-1}C^TS^{-1} \\ -S^{-1}CB^{-1} & S^{-1} \end{pmatrix}$$

with $S =$ Schur's Complement $= D - CB^{-1}C^T$
There are:

- 2 inversions: B^{-1}, S^{-1} of size $\dfrac{n}{2}$
- 4 multiplications: CB^{-1}, $C^T(CB^{-1})$, $S^{-1}(CB^{-1})$, $(CB^{-1})^T\,[S^{-1}(CB^{-1})]$

$$\begin{aligned} \Rightarrow I(n) &\leq 2I(\frac{n}{2}) + 4M(\frac{n}{2}) + \Theta(n^2) \\ &= 2I\left(\frac{n}{2}\right) + \Theta(M(n)) && (\text{because} n^2 \in \mathcal{O}(M(n))) \\ &= \Theta(M(n)) && (\text{Master theorem with } a = b = 2) \end{aligned}$$

$\Rightarrow I(n) \in \mathcal{O}(M(n))$

$(*)$ For general invertible matrices:
$A^{-1} = \underbrace{(A^TA)^{-1}}_{\succcurlyeq 0} A^T$ $\to$we can apply the trick to (A^TA). There is one more multiplication but it does not change anything.

Another D&Q algorithm
How to find the ith smallest entry of an array?
E.g:

- $i = 1$ minimum: $\Theta(n)$
- $i = n$ maximum: $\Theta(n)$
- $i = \lfloor \frac{n}{2} \rfloor$ median: $\mathcal{O}(n \log n)$: sort then read $T(\lfloor \frac{n}{2} \rfloor)$

Can we do better?
Idea: Do random Quicksort but save effort by not sorting one of the two subarrays.

Selection (T,i) Assumption: all entries are different

Pivot := random entry of $T = T(j)$ for random $j \in \{1, ..., n\}$
T_{low}:=entries¡Pivot
T_{high}:=entries¿Pivot
If $|T_{low}|$=$i-1$ then Selection(T,i):=Pivot
If $|T_{low}| < i-1$ then Selection(T,i):=Selection($T_{high}, i - |T_{low}| - 1$)
If $|T_{low}| > i-1$ then Selection(T,i):=Selection(T_{low}, i)

(Worst-case) expected time= t_n

$$
\begin{aligned}
t_n &\leq \mathbb{E}t_{\max(|T_{low}|,|T_{high}|)} + \Theta(n) && \Theta(n) \rightarrow \text{finding } T_{low}, T_{high} \text{ and } |T_{low}| \\
&= \sum_{k=0}^{n-1} Prob(|T_{low}| = k) t_{\max(k,n-k-1)} + \Theta(n) && \Theta(n) \leq an \text{ (check following theorem)} \\
&= 2 \sum_{k=\lfloor (n-1)/2 \rfloor}^{n-1} \frac{1}{n} t_k + \Theta(n)
\end{aligned}
$$

Clever way to solve it: guess and check if it is correct (many terms otherwise).

Theorem 3
$t_n = \Theta(n)$

Proof Assume the term $\Theta(n) \leq an$ for n large enough.
Assume $t_n \leq cn$ for some $c > 0$ (to be chosen later), all large enough n.
Proof by induction: assume it's time for $k \leq n-1$: $t_k \leq c_k$.

Show that it is true for n: $t_n \leq c_n$?

$$\begin{aligned}
t_n &\leq 2 \sum_{k=(n-1)/2}^{n-1} \frac{t_k}{n} + an \\
&\leq 2 \sum_{k=(n-1)/2}^{n-1} \frac{ck}{n} + an \\
&= 2\frac{c}{n} \overbrace{\left[\frac{(n-1)(n-2)}{2} - \frac{\frac{n-1}{2}\left(\frac{n-1}{2} - 1\right)}{2} \right]}^{\leq \frac{3n^2}{8}} + an
\end{aligned}$$

$$= \frac{3cn}{4} + a_n$$

which is $\leq cn$ if we choose c such that $\dfrac{3c}{4} + a < c \Rightarrow c > 4a$