Αλγόριθμοι και Πολυπλοκότητα

Αρχοντία Γιαννοπούλου
Όλγα Φουρτουνέλλη

Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών

Ασυμπτωτική Πολυπλοκότητα Δεδομένων

Πώς επηρεάζεται η απόδοση ενός αλγόριθμου όταν ο όγκος των δεδομένων που λαμβάνει αυξάνεται;

Απόδοση αλγορίθμου

Ένα πρόγραμμα είναι χρήσιμο όταν

- σε *λογικό* χρόνο και
- σε *λογικό* χώρο μνήμης

εξάγει το σωστό αποτέλεσμα.

Η απόδοση του αλγόριθμου μετριέται με βάση:

- Χρόνο εκτέλεσης
- Απαιτούμενη μνήμη

τα οποία αποτελούν την **πολυπλοκότητά** του.

Η πολυπλοκότητα

Αποτίμηση της αποδοτικότητας του αλγόριθμου σε χρόνο και μνήμη.

- Μονάδα μέτρησης: Στοιχειώδης (ουσιώδης) πράξη, που θεωρούμε ότι γίνεται σε σταθερό χρόνο.

- Συνάρτηση της διάστασης του στιγμιοτύπου.

Για παράδειγμα

Ποιά είναι η πολυπλοκότητα του παρακάτω προγράμματος;

$x = 0$
Για $i = 1$ έως n
    Για $j = 1$ έως $\frac{i+1}{2}$
        $x = x + 1$

Αποδοτικότητα

Έστω πρόβλημα Π με δεδομένα διάστασης n.

Ο αλγόριθμος A_1 το λύνει σε $T(n) = 100n$ βήματα.

Ο αλγόριθμος A_2 το λύνει σε $T(n) = n^2$ βήματα.

$$\text{Για } n \begin{cases} = 100 & \text{οι δύο αλγόριθμοι είναι εξίσου αποδοτικοί} \\ < 100 & \text{ο αλγόριθμος } A_2 \text{ αποδοτικότερος από τον } A_1 \\ > 100 & \text{ο αλγόριθμος } A_1 \text{ αποδοτικότερος από τον } A_2 \end{cases}$$

Τι σημαίνει αυτό στην πράξη;

Χρόνος εκτέλεσης στοιχειώδους εντολής $= 10^{-6}$ sec.

Για $n = 1.000$:

- ο A_1 εκτελείται σε $T(1.000) = 100 \cdot 1.000 \cdot 10^{-6} = 10^{-1}$ sec.
- ο A_2 εκτελείται σε $T(1.000) = 1.000^2 \cdot 10^{-6} = 1$ sec.

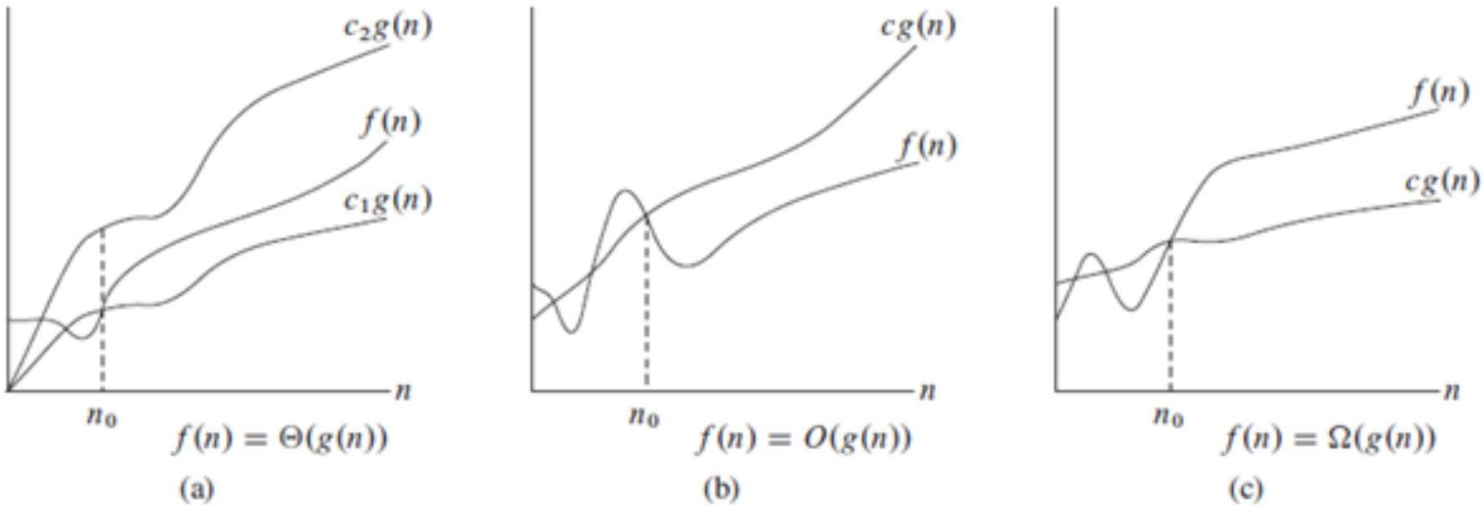
Ο A_2 είναι 10 φορές πιο αργός.

Για $n = 100.000$:

- ο A_1 εκτελείται σε $T(100.000) = 10$ sec.
- ο A_2 εκτελείται σε $T(100.000) = 10^4$ sec.

Ο A_2 είναι 1000 φορές πιο αργός.

Σύγκριση


$c_2g(n)$
$f(n)$
$c_1g(n)$
n
n_0
$f(n) = \Theta(g(n))$
(a)
$cg(n)$
$f(n)$
n
n_0
$f(n) = O(g(n))$
(b)
$f(n)$
$cg(n)$
n
n_0
$f(n) = \Omega(g(n))$
(c)

Τάξη μεγέθους

Έστω $f : \mathbb{N} \to \mathbb{R}^+$.

Ορίζουμε τις παρακάτω κλάσεις συναρτήσεων σε σχέση με την f.

$$\mathcal{O}(f(n)) = \{T : \exists n_0 \in \mathbb{N} \;\; \exists c \in \mathbb{R}^+ \;\; \forall n \geq n_0 \;\; T(n) \leq c \cdot f(n)\}$$

$$\Omega(f(n)) = \{T : \exists n_0 \in \mathbb{N} \;\; \exists c \in \mathbb{R}^+ \;\; \forall n \geq n_0 \;\; T(n) \geq c \cdot f(n)\}$$

$$\Theta(f(n)) = \mathcal{O}(f(n)) \cap \Omega(f(n))$$

Παραδείγματα

- $T(n) = 3n^2 - 100n + 6 \in \mathcal{O}(n^2)$

  $3n^2 - 100n + 6 < 3n^2$

- $T(n) = 3n^2 - 100n + 6 \in \mathcal{O}(n^3)$

  $3n^2 - 100n + 6 < 0.00001n^3$

- $T(n) = 3n^2 - 100n + 6 \notin \mathcal{O}(n)$

  $cn < 3n^2$ **για** $n > c$

Κλάσεις Πολυπλοκότητας

- $f(n) = 1$ σταθερή πολυπλοκότητα
- $f(n) = \log n$ λογαριθμική πολυπλοκότητα
- $f(n) = n$ γραμμική πολυπλοκότητα
- $f(n) = n \log n$
- $f(n) = n^2$
- $f(n) = n^3$

$\vdots$

- $f(n) = n^p$

Κλάσεις Πολυπλοκότητας

$$\left.\begin{array}{l} f(n) = 2^n \\ f(n) = \alpha^n \end{array}\right\}$$ εκθετική πολυπλοκότητα

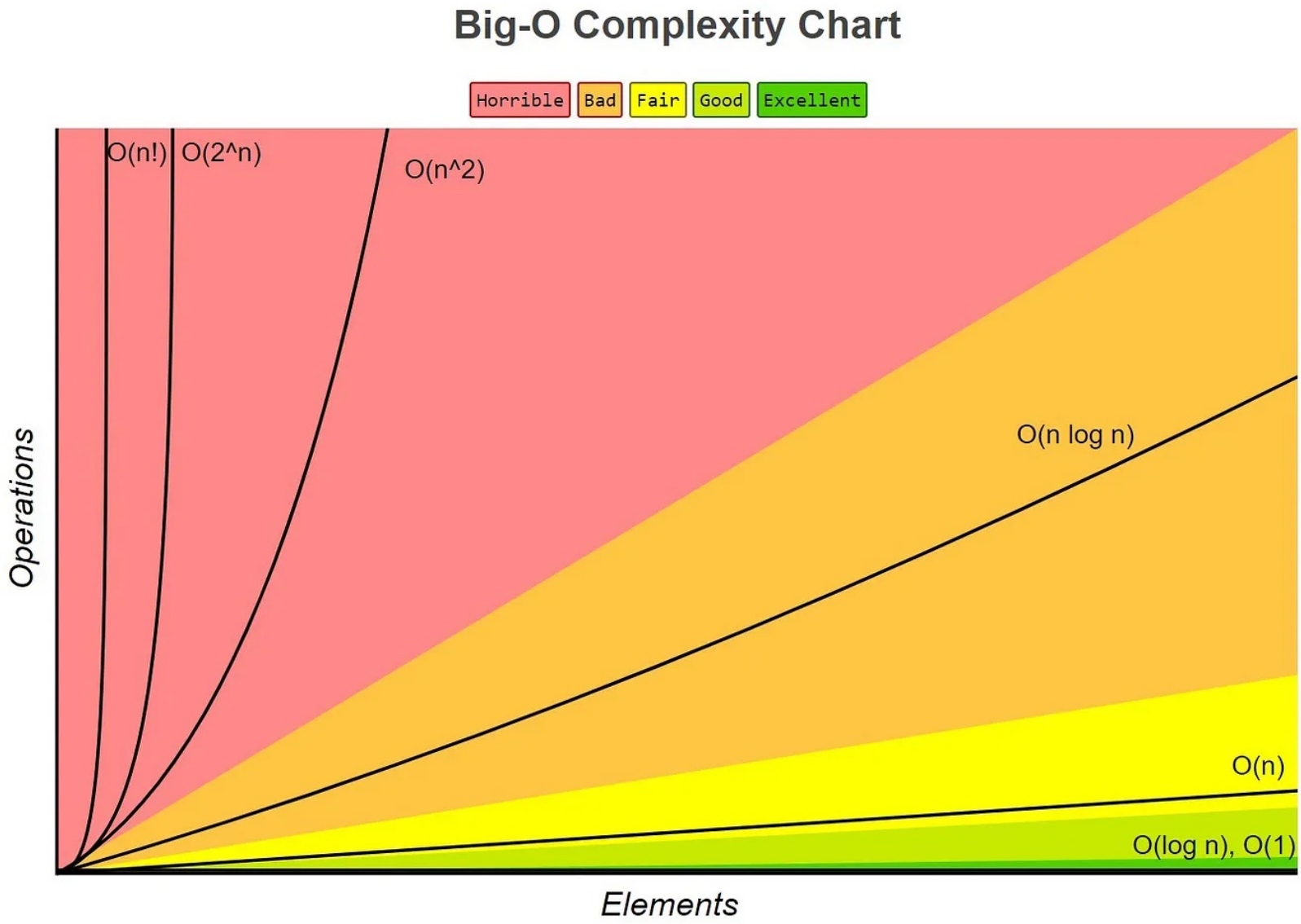
$$f(n) = 2^{2^n}\}$$ διπλά εκθετική πολυπλοκότητα

Wheat and chessboard problem

Wheat and chessboard problem

$$T_{64} = 2^{64} - 1$$

Η πικρή πραγματικότητα



Για εμάς...

Η πολυωνυμική πολυπλοκότητα είναι καλή!

Η εκθετική πολυπλοκότητα είναι κακή!

Η αιρετική άποψη...

Η ανάγκη ανάπτυξης αποτελεσματικών αλγορίθμων
θα εκλείψει μπροστά στις
αυξητικές αποδόσεις των αυριανών υπολογιστών!

What's the catch?

Για υπολογιστή 1000 φορές ταχύτερο από το σημερινό

Πολυπλοκότητα	**Τωρινό μέγεθος**	**Μελλοντικό μέγεθος**
$\mathcal{O}(n)$	n	$n \cdot 1000$
$\mathcal{O}(n^2)$	n	
$\mathcal{O}(n^3)$	n	
$\mathcal{O}(n^4)$	n	
$\mathcal{O}(2^n)$	n	$n + 10$
$\mathcal{O}(3^n)$	n	
$\mathcal{O}(n!)$	n	

Ιδιότητες στις τάξεις μεγέθους

$f, f_1, f_2 : \mathbb{N} \to \mathbb{R}^+$.

- $f(n) \in \mathcal{O}(f(n))$
- $f(n) \in \Theta(f(n))$
- $\mathcal{O}(cf(n)) = \mathcal{O}(f(n))$
- $\mathcal{O}(f(n) + f(n)) = \mathcal{O}(f(n))$
- $\mathcal{O}(f_1(n) + f_2(n)) = \mathcal{O}(\max\{f_1(n), f_2(n))\})$
- $\mathcal{O}(f_1(n)) \cdot \mathcal{O}(f_2(n)) = \mathcal{O}(f_1(n) \cdot f_2(n)))$

Έστω $g(n) \in \mathcal{O}(f_1(n))$ και $g(n) \in \mathcal{O}(f_2(n))$.

Τότε:

- $\exists c_1\ \exists n_1\ \forall n \geq n_1\ g(n) \leq c_1 f_1(n) + c_2 f_2(n)$.
- $\exists c_2\ \exists n_2\ \forall n \geq n_2\ g(n) \leq c_2 f_2(n) + c_1 f_1(n)$.

Παίρνουμε $c_0 = \max\{c_1, c_2\}$ και $n_0 = \max\{n_1, n_2\}$ και τότε:

$$g(n) \leq c_0(f_1(n) + f_2(n)) \leq 2c_0 \max\{f_1(n), f_2(n)\}$$

για κάθε $n \geq n_0$.

Ιδιότητες

- $f(n) \in \mathcal{O}(g(n))$ αν και μόνο αν $g(n) \in \Omega(f(n))$
- $f(n) \in \Theta(g(n))$ αν και μόνο αν $f(n) \in \mathcal{O}(g(n))$ και $g(n) \in \mathcal{O}(f(n))$
- Αν $f_1(n) \in \mathcal{O}(g_1(n))$ και $f_2(n) \in \mathcal{O}(g_2(n))$ τότε $(f_1 \cdot f_2)(n) \in \mathcal{O}(g_1(n)g_2(n))$
- $f(n) \in \Theta(g(n))$ αν και μόνο αν $g(n) \in \Theta(f(n))$.
- Αν $f(n) \in \Theta(g(n))$ και $g(n) \in \Theta(h(n))$ τότε $f(n) \in \Theta(h(n))$.

Είναι κάθε δύο συναρτήσεις ασυμπτωτικά συγκρίσιμες;

Όχι!

π.χ. $f(n) = n$ και $g(n) = n^{1+\sin(n)}$ (ο εκθέτης είναι στο $[0, 2]$)

Άσκηση

Να δειχθεί ότι αν $a > 1$ και $f(n) \in \mathcal{O}(\log_a n)$ τότε $f(n) \in \mathcal{O}(\log_2 n)$.
(Με άλλα λόγια, δε μας απασχολεί η βάση του λογάριθμου.)

$$f(n) \leq c \log_a n = c \frac{\log_2 n}{\log_2 a} \leq c' \log_2 n$$

Άσκηση

Να δειχθεί ότι $\log(n!) \in \mathcal{O}(n \log n)$.

$n! = 1 \cdot 2 \cdots n \leq n \cdot n \cdots n = n^n$.

Αφού η λογαριθμική συνάρτηση είναι αύξουσα:

$\log(n!) \leq \log n^n = n \log n$.

Ασκήσεις

Να βρεθεί ο καλύτερος $\mathcal{O}$ συμβολισμός για τα ακόλουθα:

- $2 \log n - 4n + 3n \log n$
- $2 + 4 + \cdots + 2n$
- $2 + 4 + 8 + \cdots + 2^n$

Ασκήσεις

Να δειχθεί ότι:

- $\sum_{k=1}^{n} k \cdot n \in \mathcal{O}(n^3)$
- Αν $f(n) = \mathcal{O}(n)$ τότε $(f(n))^2 \in \mathcal{O}(n^2)$
- $3^n \notin \mathcal{O}(2^n)$.

Εναλλακτικά

$f, g : \mathbb{N} \to \mathbb{R}^+$

- $\lim_{n\to\infty} \frac{f(n)}{g(n)} = \alpha \neq 0 \Rightarrow f \in \Theta(g)$
- $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0 \Rightarrow f \in \mathcal{O}(g)$ και $f \notin \Theta(g)$
- $\lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty \Rightarrow g \in \mathcal{O}(f)$ και $g \notin \Theta(f)$

Ρυθμός Αύξησης

Κάθε εκθετική συνάρτηση με βάση μεγαλύτερη του 1 αυξάνεται γρηγορότερα από κάθε πολυωνυμική συνάρτηση.

Τυπικά, για κάθε $a, b \in \mathbb{R}$ σταθερές με $a > 1$ ισχύει ότι:

$$\lim_{n \to \infty} \frac{n^b}{a^n} = 0.$$

Συμβολισμός: $n^b = o(a^n)$.

Υπενθύμιση - Κανόνας de L'Hôpital

Χρήσιμο εργαλείο για να υπολογίζουμε το όριο ενός λόγου συναρτήσεων $\frac{f(x)}{g(x)}$ όταν προκύπτει απροσδιοριστία της μορφής $\frac{\infty}{\infty}$.

Αν $\lim_{x\to\infty} f(x) = +\infty$ και $\lim_{x\to\infty} g(x) = +\infty$ και f, g παραγωγίσιμες τότε

$$\lim_{x\to\infty} \frac{f(x)}{g(x)} = \lim_{x\to\infty} \frac{f'(x)}{g'(x)}.$$

Απόδειξη

Με επανηλειμμένες χρήσεις του Κανόνα de L'Hôpital έχουμε ότι:

$$\lim_{x\to\infty} \frac{x^b}{a^x} = \lim_{x\to\infty} \frac{b \cdot x^{b-1}}{a^x \cdot \ln a} = \cdots = \lim_{x\to\infty} \frac{b!}{a^x \ln^b a} = 0$$

Ρυθμός αύξησης

Μία $f(n)$ καλείται πολυλογαριθμικά φραγμένη αν υπάρχει κάποια σταθερά $c \in \mathbb{R}$ τέτοια ώστε $f(n) \leq \log^c n$.

Κάθε πολυωνυμική συνάρτηση αυξάνεται γρηγορότερα από κάθε πολυλογαριθμική συνάρτηση.

Τυπικά, για σταθερές $a, b \in \mathbb{R}$ με $a > 0$ ισχύει ότι:

$$\lim_{n \to \infty} \frac{\log^b n}{n^a} = 0.$$

Όπως πρίν: $\log^b n = o(n^a)$.
Απόδειξη: Παρόμοια

Απόδειξη

Αρκεί να δείξουμε ότι υπάρχει n_0 τέτοιο ώστε για κάθε $n \geq n_0$:

$$(\ln n)^b \leq n^a \Leftrightarrow b \ln \ln n \leq a \ln n \Leftrightarrow \ln \ln n \leq \frac{a}{b} \ln n.$$

Όπως πριν με Κανόνα de L'Hôpital ισχύει ότι:

$$\lim_{x \to \infty} \frac{\ln \ln x}{\frac{a}{b} \ln x} = \frac{b}{a} \lim_{x \to \infty} \frac{\frac{1}{\ln x} \cdot \frac{1}{x}}{\frac{1}{x}} = 0$$

Όπου θυμόμαστε ότι $(f \circ g)' = (f' \circ g) \cdot g'$ και $(\ln x)' = \frac{1}{x}$.

Συμβολισμοί εναλλακτικά

$f, g : \mathbb{N} \to \mathbb{R}^+$

- $\lim_{n\to\infty} \frac{f(n)}{g(n)} = \alpha \neq 0 \Rightarrow f \in \Theta(g)$
- $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0 \Rightarrow f \in \mathcal{O}(g)$ και $f \notin \Theta(g)$ $\qquad f \in o(g)$
- $\lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty \Rightarrow g \in \mathcal{O}(f)$ και $g \notin \Theta(f)$ $\qquad f \in \omega(g)$

Διαφορετικά

$$o(f(n)) = \{g(n) \mid \forall c > 0 \quad \exists n_0 \in \mathbb{N} \quad \forall n \geq n_0 : \quad g(n) < cf(n)\}$$

$$\omega(f(n)) = \{g(n) \mid \forall c > 0 \quad \exists n_0 \in \mathbb{N} \quad \forall n \geq n_0 : \quad g(n) > cf(n)\}$$

Η έννοια του βέλτιστου αλγόριθμου

Για ένα πρόβλημα Π να βρεθεί ο καλύτερος αλγόριθμος που το επιλύει.

Δηλαδή ο αλγόριθμος με την καλύτερη πολυπλοκότητα.

Αν **οποιοσδήποτε** αλγόριθμος που επιλύει το Π έχει ασυμπτωτική πολυπλοκότητα $\Omega(f(n))$ τότε **ένας** αλγόριθμος με πολυπλοκότητα $\mathcal{O}(f(n))$ είναι (ασυμπτωτικά) βέλτιστος!

Γινόμενο τετραγωνικών πινάκων

Είσοδος: Δύο $n \times n$ πίνακες $A = (a_{ij}), B = (b_{ij})$.
Ζητούμενο: Ο πίνακας $C = A \cdot B$.

Γνωρίζουμε ότι $C = A \cdot B = (c_{ij})$, όπου

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

Αλγόριθμος πολλαπλασιασμού πινάκων

Για $i = 1$ έως n
  Για $j = 1$ έως n
    $c_{ij} \leftarrow 0$
    Για $k = 1$ έως n
      $c_{ij} \leftarrow c_{ij} + a_{ik} \cdot b_{kj}$

Ανάλυση πολυπλοκότητας

Ο προηγούμενος αλγόριθμος απαιτεί n^3 πολλαπλασιασμούς.

Χρειαζόμαστε να υπολογίσουμε τουλάχιστον n^2 στοιχεία.

Ωστόσο δεν γνωρίζουμε αλγόριθμο πολυπλοκότητας $\mathcal{O}(n^2)$.

- Strassen, 1969 $\mathcal{O}(n^{2.81})$
- Coppersmith & Winograd, 1986 $\mathcal{O}(n^{2.379})$
- V. V. Williams, 2011 $\mathcal{O}(n^{2.373})$
- V. V. Williams & Y. Xu & Z. Xu & R. Zhou, 2023 $\mathcal{O}(n^{2.371552})$
- J. Alman, R. Duan, V. V. Williams, Y. Xu, Z. Xu, and R. Zhou, 2025 $\mathcal{O}(n^{2.371339})$