Алгоритми та складність

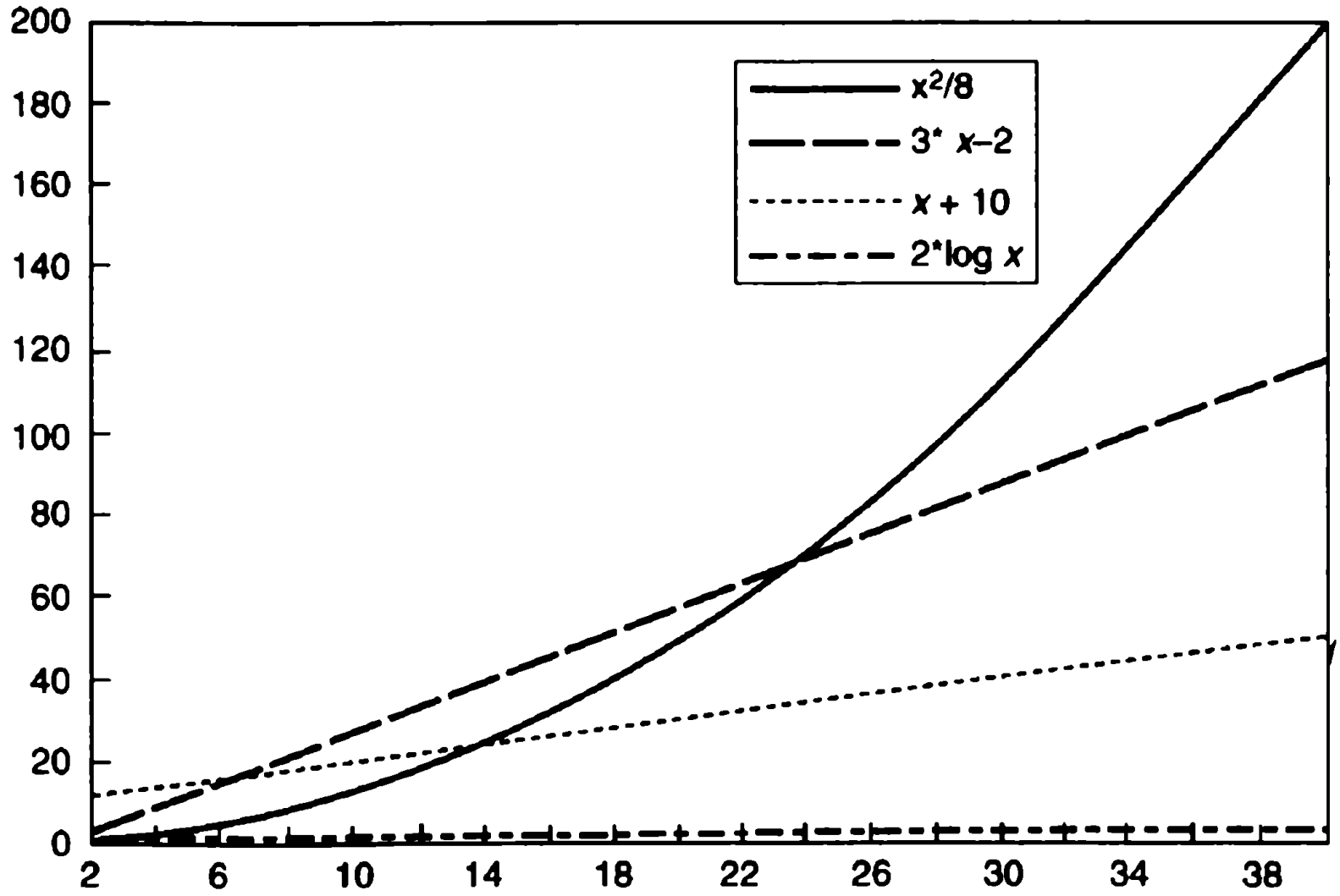
I семестр

Лекція 2

Порядок зростання функцій

- Асимптотична ефективність алгоритму: розглядаємо вхідні дані досить великих розмірів для оцінки тільки такої величини, як порядок зростання часу роботи алгоритму.
- Нас цікавить лише те, як час роботи алгоритму зростає із збільшенням розміру вхідних даних в границі, коли цей розмір збільшується до нескінченності.
- Зазвичай алгоритм, ефективніший в асимптотичному сенсі, буде продуктивнішим для всіх вхідних даних, крім дуже малих.

Порядок зростання функцій



Асимптотичні позначення

- Припустимо для зручності, що для опису асимптотичної поведінки роботи алгоритму будемо використовувати функції, у яких область визначення цілі невід’ємні числа.
- Можна уявити, що такі числа представляють розмір вхідних даних.
- Однак ці припущення можна узагальнити і на область дійсних чисел чи звузити на певну підмножину натуральних чисел.
- Також вважаємо, що всі функції є *асимптотично невід’ємними*: приймають невід’ємне значення на будь-яких досить великих аргументах.

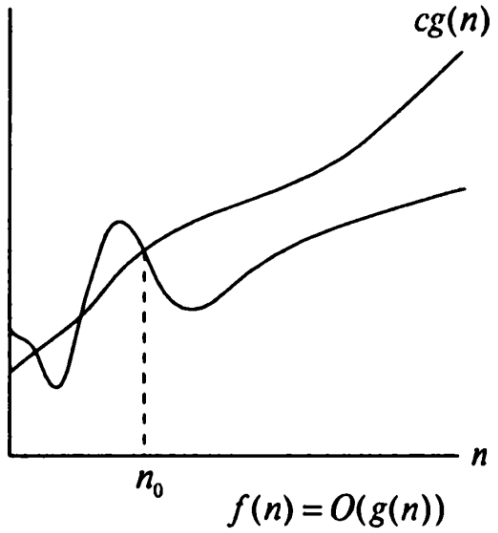
O-позначення

- Асимптотична верхня границя: для функції *g(n)* позначимо через O*(g(n))* множину таких функцій:

$$\mathrm{O}\big(g(n)\big) = \left\{ \begin{matrix} f(n) \text{: існують додатні константи } c \text{ та } n_0, \\ \text{що } 0 \le f(n) \le cg(n) \text{ для всіх } n \ge n_0 \end{matrix} \right\}$$

- Це верхня межа функції *f(n)* з точністю до константи.
- В принципі, при *f(n)*= O*(g(n))*

не важливо , наскільки близько *f(n)* буде знаходитись до своєї верхньої границі.



О-позначення

- Наприклад, будь-яка лінійна функція виду $an+b$ при $a>0$ належатиме $O(n^2)$.

  Покажемо це за означенням. Візьмемо

  $$c = a + |b|,$$

  $$n_0 = \max(1, -b/a).$$

- О-позначення описують час роботи алгоритму в найгіршому випадку.

O-позначення

- Покажемо, що $100n + 5 \in O(n^2)$. Для всіх $n \geq 5$ справедливо

$$100n + 5 \leq 100n + n = 101n \leq 101n^2.$$

Тоді $c = 101$, $n_0 = 5$.

- Можна міркувати й так: для всіх $n \geq 1$ виконується

$$100n + 5 \leq 100n + 5n = 105n,$$

звідки $c = 105$, $n_0 = 1$.

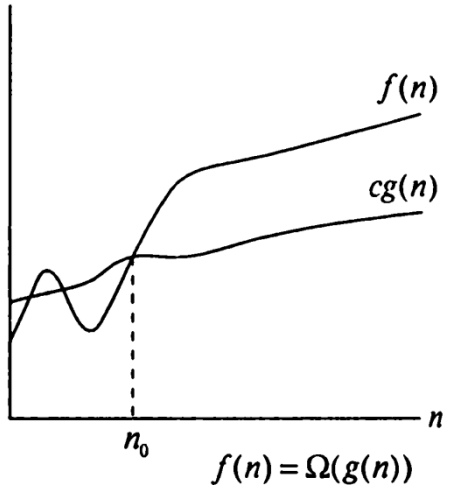
- У визначенні не говориться, яким чином отримувати значення констант. Головне їх вказати.

Ω-позначення

- Асимптотична нижня границя: для функції *g(n)* позначимо через *Ω(g(n))* множину таких функцій:

$$\Omega\big(g(n)\big) = \left\{ \begin{matrix} f(n) \text{: існують додатні константи } c \text{ та } n_0, \\ \text{що } 0 \le cg(n) \le f(n) \text{ для всіх } n \ge n_0 \end{matrix} \right\}$$

- Це нижня межа функції *f(n)* з точністю до константи.



Ω-позначення

- Приклад. Покажемо, що $n^3 \in \Omega(n^2)$.

Справді, для всіх $n \geq 0$ виконується $n^3 \geq n^2$.

Тому можна покласти

$c = 1,$

$n_0 = 0.$

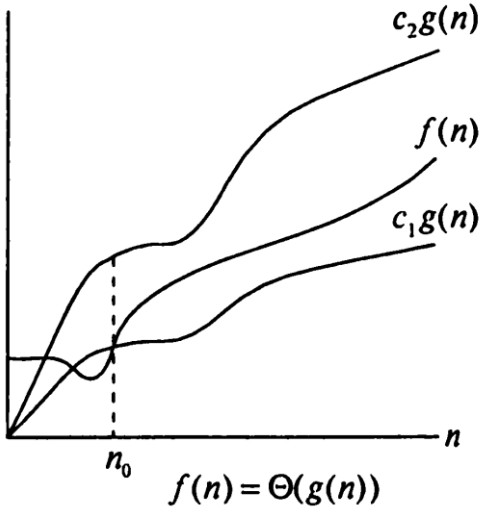
- Ω-позначення дають нижню оцінку часу роботи алгоритму в найкращому випадку.

Θ-позначення

- Асимптотично точна оцінка: для функції *g(n)* позначимо через Θ*(g(n))* множину таких функцій:

$$\Theta\big(g(n)\big) = \left\{ \begin{array}{c} f(n): \text{існують додатні константи } c_1, c_2 \text{ та } n_0, \\ \text{що } 0 \le c_1 g(n) \le f(n) \le c_2 g(n) \text{ для всіх } n \ge n_0 \end{array} \right\}$$

- Тобто якщо *f(n)*= Θ*(g(n))*, то *f(n)* можна «вставити» між функцій $c_1 g(n)$ та $c_2 g(n)$ при досить великих *n*.



Θ-позначення

- Приклад 1. Покажемо, що $n^2/2 - 3n = \Theta(n^2)$.

Шукаємо константи c_1, c_2, n_0 , для яких для всіх $n>n_0$:

$$c_1 n^2 \leqslant \frac{1}{2} n^2 - 3n \leqslant c_2 n^2$$

Ділимо на n^2:

$$c_1 \leqslant \frac{1}{2} - \frac{3}{n} \leqslant c_2.$$

Ліва нерівність виконується при $c_1 \leq 1/14$ для всіх $n \geq 7$.

Права – при $c_2 \geq 1/2$ для всіх $n \geq 1$.

Звідси отримуємо $c_1 = 1/14$,

$c_2 = 1/2$,

$n_0 = 7$.

Θ-позначення

Можна зауважити, що для виконання нерівностей достатньо взяти за c_1 значення дещо менше за коефіцієнт при старшому члені, а за c_2 – значення трохи більше цього коефіцієнта.

- Приклад 2.

Покажемо, що $6n^3 \neq \Theta(n^2)$.

Супротивне. Нехай існують c_2 та n_0 такі, що для всіх $n>n_0$ буде виконуватися

$$6n^3 \leq c_2 n^2$$

Однак тоді $n \leq c_2/6$, що неможливо для великих n, оскільки c_2 константа.

Θ-позначення

- Приклад 3.

Візьмемо $f(n) = an^2 + bn + c$, причому $a > 0$.

Тоді, взявши $c_1 = a/4$, $c_2 = 7a/4$ та

$$n_0 = 2max(|b|/a, \sqrt{|c|/a}),$$

отримуємо $f(n) = \Theta(n^2)$.

За аналогією такі оцінки будуть справедливі для довільного полінома з додатним коефіцієнтом при старшій степені.

Θ-позначення

- Надалі константу або константну функцію позначатимемо $\Theta(1)$.

Теорема 1. Для будь-яких функцій $f(n)$ та $g(n)$ співвідношення $f(n)=\Theta(g(n))$ виконується тоді і тільки тоді, коли $f(n)=O(g(n))$ та $f(n)=\Omega(g(n))$.

Теорему головним чином використовують для отримання асимптотично точної оцінки за допомогою асимптотичних верхньої та нижньої границь.

Теорема 2. Нехай $t_1(n) = O(g_1(n)), \quad t_2(n) = O(g_2(n))$.

Тоді $t_1(n)+t_2(n) = O(\max \{g_1(n), g_2(n)\})$.

Теорема 2. Нехай $t_1(n) = O(g_1(n)), \quad t_2(n) = O(g_2(n)).$

Тоді $t_1(n)+t_2(n) = O(\max \{g_1(n), g_2(n)\}).$

Доведення.

$t_1(n) = O(g_1(n))$ => існує константа c_1 та невід'ємне ціле n_1 такі, що для всіх $n \geq n_1$ вірно $t_1(n) \leq c_1 g_1(n).$

$t_2(n) = O(g_2(n))$ => існує константа c_2 та невід'ємне ціле n_2 такі, що для всіх $n \geq n_2$ вірно $t_2(n) \leq c_2 g_2(n).$

Теорема 2. Нехай $t_1(n) = O(g_1(n)),\ \ t_2(n) = O(g_2(n)).$

Тоді $t_1(n)+t_2(n) = O(\max\{g_1(n), g_2(n)\})$.

Доведення.

$t_1(n) = O(g_1(n))$ => існує константа c_1 та невід'ємне ціле n_1 такі, що для всіх $n \geq n_1$ вірно $t_1(n) \leq c_1 g_1(n)$.

$t_2(n) = O(g_2(n))$ => існує константа c_2 та невід'ємне ціле n_2 такі, що для всіх $n \geq n_2$ вірно $t_2(n) \leq c_2 g_2(n)$.

Позначимо $c_3 = \max\{c_1, c_2\}$ і візьмемо $n \geq \max\{n_1, n_2\}$.

Складемо отримані вище нерівності:

$$t_1(n) + t_2(n) \leq c_1 g_1(n) + c_2 g_2(n) \leq$$
$$\leq c_3 g_1(n) + c_3 g_2(n) = c_3[g_1(n) + g_2(n)] \leq$$
$$\leq c_3 2 \max\{g_1(n), g_2(n)\}.$$

З ланцюжка нерівностей випливає

$$t_1(n)+t_2(n) = O(\max\{g_1(n), g_2(n)\}).$$

Необхідні константи визначаємо так:

$$c = 2c_3 = 2\max\{c_1,c_2\},$$

$$n_0 = \max\{n_1, n_2\}.$$

Доведення завершено.

З ланцюжка нерівностей випливає

$$t_1(n)+t_2(n) = O(\max\{g_1(n), g_2(n)\}).$$

Необхідні константи визначаємо так:

$$c = 2c_3 = 2\max\{c_1,c_2\},$$

$$n_0 = \max\{n_1, n_2\}.$$

Доведення завершено.

Зауваження. Аналогічні твердження будуть справедливі і для множин Ω та Θ.

Значення доведеної властивості. Загальна ефективність алгоритму залежить від тієї його частини, яка має найбільший порядок зростання, тобто найменш ефективної його частини.

o-позначення

- Верхня асимптотична границя, що подається через O-позначення, може описувати асимптотичну поведінку функції з різною точністю.
- Наприклад, границя $2n^2 = O(n^2)$ дає вірне уявлення про асимптотичну поведінку функції, тоді як границя $2n = O(n^2)$ його не забезпечує.

$$o(g(n)) = \left\{ \begin{matrix} f(n): \text{для довільної додатної константи } c \text{ існує} \\ n_0 > 0, \text{що } 0 \leq f(n) < cg(n) \text{ для всіх } n \geq n_0 \end{matrix} \right\}$$

- Запис *o(g(n))* позначає множину всіх функцій, що зростають відчутно повільніше за *g(n)*.
- Наприклад, $2n = o(n^2)$, але $2n^2 \neq o(n^2)$.

o-позначення

- Визначення O-позначень та o-позначень схожі.
- Основна різниця – в першому випадку функція *f(n)* обмежується *для деякої* додатної константи *c*, а в другому *для всіх* таких констант.

$$\mathrm{O}\big(g(n)\big) = \left\{ \begin{matrix} f(n): \text{існують додатні константи } c \text{ та } n_0, \\ \text{що } 0 \le f(n) \le cg(n) \text{ для всіх } n \ge n_0 \end{matrix} \right\}$$

$$o\big(g(n)\big) = \left\{ \begin{matrix} f(n): \text{для довільної додатної константи } c \text{ існує} \\ n_0 > 0, \text{що } 0 \le f(n) < cg(n) \text{ для всіх } n \ge n_0 \end{matrix} \right\}$$

o-позначення

- Іноді означення дається через границю:

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$$

- Тобто функція *f(n)* мізерно мала в порівнянні з функцією *g(n)* при *n*, що прямує в нескінченність.
- *f(n)=o(g(n))* означає

«*f(n)* асимптотично менше *g(n)*»

ω-позначення

- Через ω-позначення вказується нижня границя, що не є асимптотично точною оцінкою.
- Неформально:

  $f(n) \in \omega(g(n))$ тоді і тільки тоді, коли $g(n) \in o(f(n))$.
- Формальне означення:

$$\omega(g(n)) = \left\{ \begin{matrix} f(n): \text{для довільної додатної константи } c \text{ існує} \\ n_0 > 0, \text{що } 0 \le cg(n) < f(n) \text{ для всіх } n \ge n_0 \end{matrix} \right\}$$

- Наприклад, $n^2/2=\omega(n)$, але $n^2/2 \neq \omega(n^2)$.
- $f(n)=\omega(g(n))$ («$f(n)$ асимптотично більше $g(n)$») означає $\lim\limits_{n\to\infty} \frac{f(n)}{g(n)}=\infty$, якщо границя існує. Тобто $f(n)$ стає як завгодно великою порівняно з $g(n)$ при n, що йде до нескінченності.

Порівняння функцій

Властивості, що аналогічні деяким властивостям дійсним чисел. Вважаємо, що всі функції асимптотично додатні (тобто приймають лише додатні значення).

Транзитивність

З $f(n) = \Theta\big(g(n)\big)$ та $g(n) = \Theta\big(h(n)\big)$ випливає $f(n) = \Theta\big(h(n)\big)$.
З $f(n) = \mathrm{O}\big(g(n)\big)$ та $g(n) = \mathrm{O}\big(h(n)\big)$ випливає $f(n) = \mathrm{O}\big(h(n)\big)$.
З $f(n) = \Omega\big(g(n)\big)$ та $g(n) = \Omega\big(h(n)\big)$ випливає $f(n) = \Omega\big(h(n)\big)$.
З $f(n) = o\big(g(n)\big)$ та $g(n) = o\big(h(n)\big)$ випливає $f(n) = o\big(h(n)\big)$.
З $f(n) = \omega\big(g(n)\big)$ та $g(n) = \omega\big(h(n)\big)$ випливає $f(n) = \omega\big(h(n)\big)$.

Рефлексивність

$$f(n) = \Theta\left(f(n)\right),$$
$$f(n) = O\left(f(n)\right),$$
$$f(n) = \Omega\left(f(n)\right).$$

Порівняння функцій

Симетричність

$f(n)=\Theta(g(n))$ вірне тоді і тільки тоді, коли $g(n)=\Theta(f(n))$.

Перестановочна симетрія

$f(n)=O(g(n))$ вірне тоді і тільки тоді, коли $g(n)=\Omega(f(n))$;

$f(n)=o(g(n))$ вірне тоді і тільки тоді, коли $g(n)=\omega(f(n))$.

Аналогії з порівняннями чисел

$$f(n) = O(g(n)) \approx a \leqslant b,$$
$$f(n) = \Omega(g(n)) \approx a \geqslant b,$$
$$f(n) = \Theta(g(n)) \approx a = b,$$
$$f(n) = o(g(n)) \approx a < b,$$
$$f(n) = \omega(g(n)) \approx a > b.$$

Порівняння функцій

Властивість **трихотомії**:

для будь-якої пари дійсних чисел a та b має виконуватись одне зі співвідношень: $a<b$, $a = b$, $a>b$

не виконується.

Для двох функцій $f(n)$ і $g(n)$ може не виконуватися ні $f(n)=O(g(n))$, ні $f(n)=\Omega(g(n))$.

Приклад?

Порівняння функцій

Властивість **трихотомії**:

для будь-якої пари дійсних чисел a та b має виконуватись одне зі співвідношень: $a<b$, $a = b$, $a>b$

не виконується.

Для двох функцій $f(n)$ і $g(n)$ може не виконуватися ні $f(n)=O(g(n))$, ні $f(n)=\Omega(g(n))$.

Наприклад, не можна асимптотично порівняти функції n та $n^{1+\sin n}$. Показник останньої буде коливатися між 0 і 2, приймаючи всі значення з цього проміжку.

Використання границь для обчислення порядку зростання функцій

Три основні випадки (*чи є четвертий?*):

$$\lim_{n\to\infty}\frac{t(n)}{g(n)}=\begin{cases}0, & \text{якщо } t(n) \text{ має менший порядок росту, ніж } g(n);\\ c, & \text{якщо } t(n) \text{ та } g(n) \text{ мають однаковий порядок росту;}\\ \infty, & \text{якщо } t(n) \text{має більший порядок росту, ніж } g(n).\end{cases}$$

Перші два випадки: $t(n)=O(g(n))$.

Останні два випадки: $t(n)=\Omega(g(n))$.

Другий випадок: $t(n)=\Theta(g(n))$.

Використання границь для обчислення порядку зростання функцій

При обчисленні границь можна використовувати формулу Лопіталя:

$$\lim_{n\to\infty}\frac{t(n)}{g(n)}=\lim_{n\to\infty}\frac{t'(n)}{g'(n)}$$

та формулу Стірлінга

$$n!=\sqrt{2\pi n}\left(\frac{n}{e}\right)^n\left(1+\Theta\left(\frac{1}{n}\right)\right)$$

З останньої іноді простіше вважати

$$n!\approx\sqrt{2\pi n}\left(\frac{n}{e}\right)^n$$

Використання границь для обчислення порядку зростання функцій

Приклад 1. Порівняємо порядки зростання функцій $n(n-1)/2$ та n^2:

$$\lim_{n\to\infty}\frac{n(n-1)/2}{n^2}=\frac{1}{2}\lim_{n\to\infty}\frac{n^2-n}{n^2}=\frac{1}{2}\lim_{n\to\infty}\left(1-\frac{1}{n}\right)=\frac{1}{2}$$

Границя рівна додатній константі, тому обидві функції мають однаковий порядок зростання,

тобто $n(n-1)/2 = \Theta(n^2)$.

Використання границь для обчислення порядку зростання функцій

Приклад 2. Порівняємо порядки зростання функцій $\log_2 n$ та $\sqrt{n}$:

$$\lim_{n\to\infty} \frac{\log_2 n}{\sqrt{n}} = \lim_{n\to\infty} \frac{(\log_2 n)'}{(\sqrt{n})'} = \lim_{n\to\infty} \frac{(\log_2 e)\frac{1}{n}}{\frac{1}{2\sqrt{n}}} = 2\log_2 e \lim_{n\to\infty} \frac{\sqrt{n}}{n} = 0.$$

Границя рівна нулю, отже верхня функція має менший порядок росту за нижню.

Тому можна записати $\log_2 n$ = o ($\sqrt{n}$).

Використання границь для обчислення порядку зростання функцій

Приклад 3. Порівняємо порядки зростання функцій $n!$ та 2^n. Використовуємо формулу Стірлінга:

$$\lim_{n\to\infty}\frac{n!}{2^n}=\lim_{n\to\infty}\frac{\sqrt{2\pi n}\left(\frac{n}{e}\right)^n}{2^n}=\lim_{n\to\infty}\sqrt{2\pi n}\,\frac{n^n}{2^n e^n}=\lim_{n\to\infty}\sqrt{2\pi n}\left(\frac{n}{2e}\right)^n=\infty.$$

Отже, $n! = \Omega(2^n)$.

Зауваження. З останньої форми запису, на відміну від використання границі, не можна гарантовано стверджувати, що порядок зростання $n!$ вищий, а не, скажімо, дорівнює порядку зростання 2^n.

Декілька корисних співвідношень

- Округлення в більшу та меншу сторони:

$$x - 1 \leq \lfloor x \rfloor \leq x \leq \lceil x \rceil < x + 1$$

- Показникові функції: для всіх дійсних $a > 0$, m, n

$$a^0 = 1,$$
$$a^1 = a,$$
$$a^{-1} = 1/a,$$
$$(a^m)^n = a^{mn},$$
$$(a^m)^n = (a^n)^m,$$
$$a^m a^n = a^{m+n}$$

Для всіх дійсних $a > 1$ та b: $\lim_{n\to\infty} \frac{n^b}{a^n} = 0$, тобто $n^b = o(a^n)$

Всяка показникова функція з основою > 1 зростає швидше за будь-яку поліноміальну функцію.

Декілька корисних співвідношень

- Співвідношення між логарифмами:

$$a = b^{\log_b a},$$

$$\log_c (ab) = \log_c a + \log_c b,$$

$$\log_b a^n = n \log_b a,$$

$$\log_b a = \frac{\log_c a}{\log_c b},$$

$$\log_b (1/a) = -\log_b a,$$

$$\log_b a = \frac{1}{\log_a b},$$

$$a^{\log_b c} = c^{\log_b a}$$

для всіх дійсних $a>0$, $b>0$, $c>0$, n та основах $\neq 1$.

Декілька корисних співвідношень

$$\lim_{n\to\infty} \frac{\lg^b n}{(2^a)^{\lg n}} = \lim_{n\to\infty} \frac{\lg^b n}{n^a} = 0.$$

- Для довільної константи *a>0* буде виконуватись

$$lg^b n = \mathrm{o}(n^a)$$

Отже, будь-яка додатна поліноміальна функція зростає швидше за будь-яку полілогарифмічну (тобто вигляду $\log^k$).

Декілька корисних співвідношень та визначень

- Функціональна ітерація

$f^{(i)}(n)$ означає, що функція послідовно i разів застосовувалася до аргументу n.

Формально, для заданої на множині дійсних чисел $f(n)$ для невід'ємних цілих i :

$$f^{(i)}(n) = \begin{cases} n & \text{при } i = 0, \\ f\left(f^{(i-1)}(n)\right) & \text{при } i > 0. \end{cases}$$

Наприклад, при $f(n) = 2n$, матимемо $f^{(i)}(n) = 2^i n$.

Декілька корисних співвідношень та визначень

- Ітерований логарифм $\lg^* n$

$$\lg^* n = \min \{ i \geq 0 : \lg^{(i)} n \leq 1 \}$$

*(*функція $\lg^{(i)} n$ визначена за умови $\lg^{(i-1)} n > 0$*)*

Тобто, $\lg^* n$ – це мінімальна кількість логарифмувань n, необхідних для отримання значення ≤ 1.

Функція зростає *дуже* повільно:

$\lg^* 2 = 1$ $\lg^* 4 = 2$

$\lg^* 16 = 3$ $\lg^* 65536 = 4$

$\lg^*(2^{65536}) = 5$

Основні асимптотичні класи ефективності

1. Константа O(1)

   Ідеальний варіант – краще не буває.

   Якщо не враховувати ефективність в найкращому випадку, до цього класу потрапляє дуже невелика кількість алгоритмів (наприклад, перевірка парності двійкового числа, використання таблиці пошуку константного розміру). Зазвичай при нескінченному збільшенні розміру вхідних даних час виконання алгоритму також прямує до нескінченності.

Основні асимптотичні класи ефективності

2. O(log log *n*)

   Надзвичайно швидка робота. По суті на практиці час близький до константи.

   Такий час роботи може зокрема виникнути в ітеративних алгоритмах при постійному зменшенні розміру задачі з поточного n до $\sqrt{n}$ або при бінарному пошуці на об’єктах розміром O(log *n*). Приклади – операції над деревом ван Емде Боаса, середній час роботи інтерполяційного пошуку.

Основні асимптотичні класи ефективності

3. Логарифмічна O(log *n*)

   Зазвичай така залежність з'являється у результаті зменшення розміру задачі на стале значення на кожному кроці ітерації алгоритму.

   Складність типова при використанні бінарного пошуку та операціях з двійковими деревами.

Основні асимптотичні класи ефективності

4. Полілогарифмічна $O((\log n)^k)$ (k–константа)

   Непоганий час, якщо неможливо досягти простої логарифмічної складності.

   Приклад – паралельний алгоритм розв’язання задачі про порядок перемноження ланцюжка матриць.

Основні асимптотичні класи ефективності

5. $O(n^p)$, де p – константа та $0 < p < 1$

   Повільніша за будь-яку полілогарифмічну функцію, але краща за лінійну.

   Приклад – пошук в k-d-дереві.

   В алгоритмах, які працюють швидше, ніж за лінійний час, неможлива робота з усіма вхідними даними (і навіть з їх деякою фіксованою частиною). У цьому випадку час виконання будь-якого алгоритму буде знаходитися щонайменше в лінійній залежності від розміру вхідних даних

Основні асимптотичні класи ефективності

6. Лінійна O(*n*)

   Найкращий варіант при необхідності обробки усіх вхідних даних.

   До цього класу належать алгоритми, що виконують сканування списку, що складається з *n* елементів, наприклад алгоритм пошуку методом послідовного перебору.

Основні асимптотичні класи ефективності

7. Лінійно-логарифмічна O($n \log n$)

   «Найпопулярніший» час виконання.

   Ефективніша за будь-яку поліноміальну функцію степені > 1.

   До цього класу належать велика кількість алгоритмів декомпозиції (підхід «розділяй та владарюй»), в т.ч. алгоритми сортування злиттям і швидкого сортування.

Основні асимптотичні класи ефективності

8. Квадратична O(n^2)

   Як правило, подібна залежність характеризує ефективність алгоритмів, що містять два вкладених цикли. Як типові приклади досить назвати простий алгоритм сортування та цілий ряд операцій, виконуваних над матрицями розміром *n x n.*

   Допустимий час, якщо *n* має порядок тисяч, але не мільйонів.

Основні асимптотичні класи ефективності

9. Кубічна O(n^3)

   Головним чином подібна залежність характеризує ефективність алгоритмів, що містять три вкладених цикли. До цього класу належать зокрема кілька досить складних алгоритмів лінійної алгебри.

   Загалом складність вигляду O(n^k), де k – константа, називають поліноміальною, а відповідні алгоритми матимуть практичний сенс при невеликих k.

Основні асимптотичні класи ефективності

10. Експоненціальна $O(2^n)$

    Залежність типова для алгоритмів, що виконують обробку всіх підмножин деякої множини, що складається з n елементів. Термін «експоненціальний» часто використовується в широкому сенсі і означає дуже високі порядки зростання, тобто включає також вищі порівняно з експонентою порядки зростання.

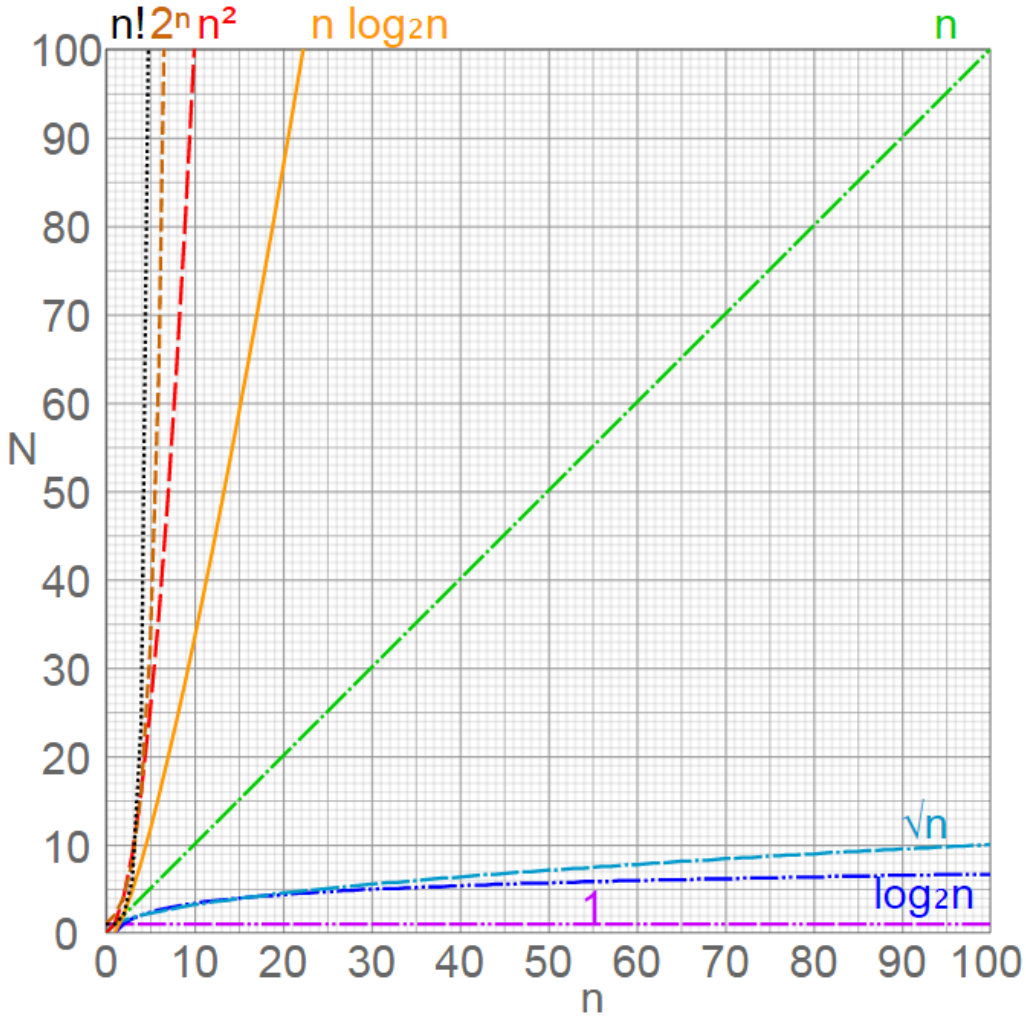
Основні асимптотичні класи ефективності

11. Факторіальна O(*n*!)

Така залежність типова для алгоритмів, що виконують обробку всіх перестановок деякої множини, що складається з *n* елементів.

На практиці використання будь-яких експоненціальних алгоритмів буде обмеженим зовсім малими значеннями *n* (не більше 10-20).

Основні асимптотичні класи ефективності



Запитання і завдання

- Доведіть, що для будь-яких дійсних констант *a*, *b* (*b* > *0*) виконується

$$(n+a)^b = \Theta(n^b).$$

- Доведіть , що множина $o(g(n)) \cap \omega(g(n))$ порожня.

- Доведіть

$$n! = o\left(n^n\right),$$
$$n! = \omega\left(2^n\right),$$
$$\lg(n!) = \Theta(n \lg n)$$

Запитання і завдання

- Для кожної пари (A, B) наведених в таблиці функцій встановіть, чи перебуває A з B у відповідному відношенні. Запишіть «так» чи «ні» в комірках. Тут k, ε, c – константи, причому $k \geq 1$, $\varepsilon > 0$ та $c > 1$.

A	B	O	o	Ω	ω	Θ
$\lg^k n$	n^ε					
n^k	c^n					
$\sqrt{n}$	$n^{\sin n}$					
2^n	$2^{n/2}$					
$n^{\lg c}$	$c^{\lg n}$					
$\lg(n!)$	$\lg(n^n)$					

Запитання і завдання

- Використовуючи визначення, доведіть або спростуйте, навівши контрприклад:

a) Якщо $t(n) \in O\big(g(n)\big)$, то $g(n) \in \Omega(t(n))$.

b) $\Omega\big(\alpha g(n)\big) = \Omega\big(t(n)\big)$, де $\alpha > 0$.

c) $\Theta\big(g(n)\big) = O(g(n)) \cap \Omega\big(g(n)\big)$.

d) Для довільних двох невід'ємних функцій $t(n)$ та $g(n)$, визначених на множині невід'ємних чисел, буде виконуватися або $t(n) \in O(g(n))$, або $t(n) \in \Omega(g(n))$, або обидва цих вирази одночасно.

- Розташуйте функції відповідно до порядку їх зростання:

$$(n-2)!,\ 5\lg(n+100)^{10},\ 2^{2n},\ 0.001n^4+3n^3+1,\ \ln^2 n,\ \sqrt[3]{n},\ 3^n$$