SPb HSE, 1 курс ПМИ, осень 2024/25
Конспект лекций по алгоритмам

Собрано 2 сентября 2024 г. в 10:35

Содержание

Лекция #1: Асимптотика

1-я пара, 2024/25

1.1. О курсе. Хорошие алгоритмы.

Что такое алгоритм, вы представляете. А что такое хороший алгоритм?

1. Алгоритм, который работает на всех тестах. Очень важное свойство. Нам не интересны решения, для которых есть тесты, на которых они не работают.

2. Алгоритм, который работает быстро. Что такое быстро? Время работы программы зависит от размера входных данных. Размер данных часто обозначают за n. Алгоритм, находящий минимум в массиве длины n делает $\approx 4n$ операций. Нам прежде всего важна зависимость от n (пропорционально n), и только во-вторых константа (≈ 4). На самом деле разные операции выполняются разное время, об этом в следующей главе.

Насколько быстро должны работать наши программы? Обычные процессоры для ноутов и телефонов имеют несколько ядер, каждое частотой $\sim$2GHz. Параллельные алгоритмы мы изучать не будем, всё, что изучим, заточено под работу на одном ядре. $\sim$2GHz это 2 000 000 000 элементарных операций в секунду. Если мы пишем на языке `C++` (а мы будем), то это $\approx 10^9$ команд в секунду. Если, например, на `python`, то реально мы успеем выполнить 10^6, в $\approx$1000 раз меньше команд в секунду. За эталон мы берём именно одну секунду — минута по человеческим ощущениям очень медленно, а сотую секунды человек не почувствует.

3. Алгоритм, который использует мало оперативной памяти. Вообще память более дорогой ресурс, чем время, об этом будет в следующей главе, в части про кеш.

4. Простые и понятные алгоритмы. Если алгоритм сложно понять, пересказать (выше порог вхождения), если он содержит много крайних случаев $\Rightarrow$ его сложно корректно реализовать, в нём вероятны ошибки, которые однажды выстрелят.

• **Асимптотика.**

Ближайшие две главы мы будем говорить преимущественно про скорость работы.
Рассмотрим простейший алгоритм, который перебирает все пары i, j: $i \leqslant j \leqslant n$.

```
1 int ans = 0;
2 for (int i = 1; i <= n; i++) // нам дали n
3     for (int j = 1; j <= i; j++)
4         ans++;
5 cout << ans << endl;
```

Мы можем посчитать точное число всех операций (сравнение, присваивание, сложение, ...) в зависимости от n: $1 + 3(1{+}2{+}3{+}\ldots{+}n) + 1$ и получить $f(n) = \frac{3}{2}n(n{+}1) + 2$.

$f(n)$ — время работы программы в зависимости от n, а n — параметр задачи, зачастую «размер входных данных». Ниже мы будем предполагать, что $n \in \mathbb{N}$, $f(n) > 0$. Интересно, насколько быстро растёт время программы в зависимости от n (размера данных). Наша $f(n) \sim n^2$, мы будем говорить «асимптотически работает за n^2» или «за n^2 с точностью до константы», это и есть *асимптотическая часть времени работы*, *асимптотика времени работы*.

Выше мы считали, что все операции работают одно и то же время, просто считали их количество. А потом ещё и забили на константу $\frac{3}{2}$ при n^2. Дальше мы разберёмся с константами и с тем,

какие операции медленнее, какие быстрее. А сейчас сосредоточимся **только** на асимптотике.

1.2. Асимптотика, $\mathcal{O}$-обозначния

Рассмотрим функции $f, g\colon \mathbb{N} \to \mathbb{R}^{>0}$.

Def 1.2.1. $f = \Theta(g) \quad \exists N > 0,\, C_1 > 0, C_2 > 0\colon \forall n \geqslant N, C_1 \cdot g(n) \leqslant f(n) \leqslant C_2 \cdot g(n)$

Def 1.2.2. $f = \mathcal{O}(g) \quad \exists N > 0,\, C > 0\colon \forall n \geqslant N, f(n) \leqslant C \cdot g(n)$

Def 1.2.3. $f = \Omega(g) \quad \exists N > 0,\, C > 0\colon \forall n \geqslant N, f(n) \geqslant C \cdot g(n)$

Def 1.2.4. $f = o(g) \quad \forall C > 0\ \exists N > 0\colon \forall n \geqslant N, f(n) \leqslant C \cdot g(n)$

Def 1.2.5. $f = \omega(g) \quad \forall C > 0\ \exists N > 0\colon \forall n \geqslant N, f(n) \geqslant C \cdot g(n)$

Понимание Θ: «равны с точностью до константы», «асимптотически равны».
Понимание $\mathcal{O}$: «не больше с точностью до константы», «асимптотически не больше».
Понимание o: «асимптотически меньше», «для сколь угодно малой константы не больше».

Θ	$\mathcal{O}$	Ω	o	ω
$=$	$\leqslant$	$\geqslant$	$<$	$>$

Замечание 1.2.6. $f = \Theta(g) \Leftrightarrow g = \Theta(f)$

Замечание 1.2.7. $f = \mathcal{O}(g), g = \mathcal{O}(f) \Leftrightarrow f = \Theta(g)$

Замечание 1.2.8. $f = \Omega(g) \Leftrightarrow g = \mathcal{O}(f)$

Замечание 1.2.9. $f = \omega(g) \Leftrightarrow g = o(f)$

Замечание 1.2.10. $f = \mathcal{O}(g), g = \mathcal{O}(h) \Rightarrow f = \mathcal{O}(h)$

Замечание 1.2.11. Обобщение: $\forall \beta \in \{\mathcal{O}, o, \Theta, \Omega, \omega\}\colon f = \beta(g), g = \beta(h) \Rightarrow \boxed{f = \beta(h)}$

Замечание 1.2.12. $\forall C > 0 \quad C \cdot f = \Theta(f)$

Докажем для примера Rem 1.2.6.

Доказательство. $C_1 \cdot g(n) \leqslant f(n) \leqslant C_2 \cdot g(n) \Rightarrow \frac{1}{C_2} f(n) \leqslant g(n) \leqslant \frac{1}{C_1} g(n) \leqslant f(n)$ ■

Упражнение 1.2.13. $f = \mathcal{O}(\Theta(\mathcal{O}(g))) \Rightarrow f = \mathcal{O}(g)$

Упражнение 1.2.14. $f = \Theta(o(\Theta(\mathcal{O}(g)))) \Rightarrow f = o(g)$

Упражнение 1.2.15. $f = \Omega(\omega(\Theta(g))) \Rightarrow f = \omega(g)$

Упражнение 1.2.16. $f = \Omega(\Theta(\mathcal{O}(g))) \Rightarrow f$ может быть любой функцией

Lm 1.2.17. $g = o(f) \Rightarrow f \pm g = \Theta(f)$

Доказательство. $g = o(f)\ \exists N\colon \forall n \geqslant N\ g(n) \leqslant \frac{1}{2} f(n) \Rightarrow \frac{1}{2} f(n) \leqslant f(n) \pm g(n) \leqslant \frac{3}{2} f(n)$ ■

Lm 1.2.18. $n^k = o(n^{k+1})$

Доказательство. $\forall C\, \forall n \geqslant C \quad n^{k+1} \geqslant C \cdot n^k$ ■

Lm 1.2.19. $P(x)$ – многочлен, тогда $P(x) = \Theta(x^{\deg P})$ при старшем коэффициенте > 0.

Доказательство. $P(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_k x^k$. По леммам Rem 1.2.12, Lm 1.2.18 имеем, что все слагаемые кроме $a_k x^k$ являются $o(x^{\deg P})$. Поэтому по лемме Lm 1.2.17 вся сумма является $\Theta(x^k)$. ■

1.3. Рекуррентности и Карацуба

• Алгоритм умножения чисел в столбик

Рассмотрим два многочлена $A(x) = 5 + 4x + 3x^2 + 2x^3 + x^4$ и $B(x) = 9 + 8x + 7x^2 + 6x^3$.
Запишем массивы `a[] = {5, 4, 3, 2, 1}`, `b[] = {9, 8, 7, 6}`.

```
for (i = 0; i < an; i++) // an = 5
    for (j = 0; j < bn; j++) // bn = 4
        c[i + j] += a[i] * b[j];
```

Мы получили в точности коэффициенты многочлена $C(x) = A(x)B(x)$.

Теперь рассмотрим два числа $A = 12345$ и $B = 6789$, запишем те же массивы и сделаем:

```
// Перемножаем числа без переносов, как многочлены
for (i = 0; i < an; i++) // an = 5
    for (j = 0; j < bn; j++) // bn = 4
        c[i + j] += a[i] * b[j];
// Делаем переносы, массив c = [45, 76, 94, 100, 70, 40, 19, 6, 0]
for (i = 0; i < an + bn; i++)
    if (c[i] >= 10)
        c[i + 1] += c[i] / 10, c[i] %= 10;
// Массив c = [5, 0, 2, 0, 1, 8, 3, 8, 0], ответ = 83810205
```

Данное умножение работает за $\Theta(nm)$, или $\Theta(n^2)$ в случае $n = m$.

Следствие 1.3.1. Чтобы умножать длинные числа достаточно уметь умножать многочлены.

Многочлены мы храним, как массив коэффициентов. При программировании умножения, нам важно знать не степень многочлена d, а длину этого массива $n = d + 1$.

• Алгоритм Карацубы

Чтобы перемножить два многочлена (или два длинных целых числа) $A(x)$ и $B(x)$ из n коэффициентов каждый, разделим их на части по $k = \frac{n}{2}$ коэффициентов – A_1, A_2, B_1, B_2.
Заметим, что $A \cdot B = (A_1 + x^k A_2)(B_1 + x^k B_2) = A_1B_1 + x^k(A_1B_2 + A_2B_1) + x^{2k}A_2B_2$.
Если написать рекурсивную функцию умножения, то получим время работы:

$$T_1(n) = 4T_1(\tfrac{n}{2}) + \Theta(n)$$

Из последующей теоремы мы сделаем вывод, что $T_1(n) = \Theta(n^2)$. Алгоритм можно улучшить, заметив, что $A_1B_2 + A_2B_1 = (A_1 + A_2)(B_1 + B_2) - A_1B_1 - A_2B_2$, где вычитаемые величины уже посчитаны. Итого три умножения вместо четырёх:

$$T_2(n) = 3T_2(\tfrac{n}{2}) + \Theta(n)$$

Из последующей теоремы мы сделаем вывод, что $T_2(n) = \Theta(n^{\log_2 3}) = \Theta(n^{1.585\ldots})$.
Данный алгоритм применим и для умножения многочленов, и для умножения чисел.

Псевдокод алгоритма Карацубы для умножения многочленов:

```
Mul(n, a, b): // n = 2^k, c(w) = a(w)*b(w)
    if n == 1: return {a[0] * b[0]}
    a --> a1, a2
    b --> b1, b2
    x = Mul(n / 2, a1, b1)
    y = Mul(n / 2, a2, b2)
    z = Mul(n / 2, a1 + a2, b1 + b2)
    // Умножение на w^i - сдвиг массива на i вправо
    return x + y * w^n + (z - x - y) * w^(n/2);
```

Чтобы умножить числа, сперва умножим их как многочлены, затем сделаем переносы.

1.4. Теоремы о рекуррентных соотношениях

Теорема 1.4.1. *Мастер Теорема* (теорема о простом рекуррентном соотношении)
Пусть $T(n) = aT(\frac{n}{b}) + f(n)$, где $f(n) = n^c$. При этом $a > 0, b > 1, c \geqslant 0$. Определим глубину рекурсии $k = \log_b n$. Тогда верно одно из трёх:

$$\begin{cases} T(n) = \Theta(a^k) = \Theta(n^{\log_b a}) & a > b^c \\ T(n) = \Theta(f(n)) = \Theta(n^c) & a < b^c \\ T(n) = \Theta(k \cdot f(n)) = \Theta(n^c \log n) & a = b^c \end{cases}$$

Доказательство. Раскроем рекуррентность:
$T(n) = f(n) + aT(\frac{n}{b}) = f(n) + af(\frac{n}{b}) + a^2 f(\frac{n}{b^2}) + \cdots = n^c + a(\frac{n}{b})^c + a^2(\frac{n}{b^2})^c + \ldots$
Тогда $T(n) = f(n)(1 + \frac{a}{b^c} + \left(\frac{a}{b^c}\right)^2 + \cdots + \left(\frac{a}{b^c}\right)^k)$. При этом в сумме $k+1$ слагаемых.
Обозначим $q = \frac{a}{b^c}$ и оценим сумму $S(q) = 1 + q + \cdots + q^k$.
Если $q = 1$, то $S(q) = k + 1 = \log_b n + 1 = \Theta(\log_b n) \Rightarrow T(n) = \Theta(f(n) \log n)$.
Если $q < 1$, то $S(q) = \frac{1-q^{k+1}}{1-q} = \Theta(1) \Rightarrow T(n) = \Theta(f(n))$.
Если $q > 1$, то $S(q) = q^k + \frac{q^k - 1}{q-1} = \Theta(q^k) \Rightarrow T(n) = \Theta(a^k(\frac{n}{b^k})^c) = \Theta(a^k)$. ■

Теорема 1.4.2. *Обобщение Мастер Теоремы*
Мастер Теорема верна и для $f(n) = n^c \log^d n$
$T(n) = aT(\frac{n}{b}) + n^c \log^d n$. При $a > 0, b > 1, c \geqslant 0, d \geqslant 0$.

$$\begin{cases} T(n) = \Theta(a^k) = \Theta(n^{\log_b a}) & a > b^c \\ T(n) = \Theta(f(n)) = \Theta(n^c \log^d n) & a < b^c \\ T(n) = \Theta(k \cdot f(n)) = \Theta(n^c \log^{d+1} n) & a = b^c \end{cases}$$

Без доказательства. ■

Теорема 1.4.3. *Об экспоненциальном рекуррентном соотношении*
Пусть $T(n) = \sum b_i T(n - a_i)$. При этом $a_i > 0, b_i > 0, \sum b_i > 1$.
Тогда $T(n) = \Theta(\alpha^n)$, при этом $\alpha > 1$ и является корнем уравнения $1 = \sum b_i \alpha^{-a_i}$, его можно найти бинарным поиском.

Доказательство. Предположим, что $T(n) = \alpha^n$, тогда $\alpha^n = \sum b_i \alpha^{n-a_i} \Leftrightarrow 1 = \sum b_i \alpha^{-a_i} = f(\alpha)$.
Теперь нам нужно решить уравнение $f(\alpha) = 1$ для $\alpha \in [1, +\infty)$.
Если $\alpha = 1$, то $f(\alpha) = \sum b_i > 1$, если $\alpha = +\infty$, то $f(\alpha) = 0 < 1$. Кроме того $f(\alpha) \searrow [1, +\infty)$.
Получаем, что на $[1, +\infty)$ есть единственный корень уравнения $1 = f(\alpha)$ и его множно найти бинарным поиском.

Мы показали, откуда возникает уравнение $1 = \sum b_i \alpha^{-a_i}$. Доказали, что у него $\exists!$ корень α. Теперь докажем по индукции, что $T(n) = \mathcal{O}(\alpha^n)$ (оценку сверху) и $T(n) = \Omega(\alpha^n)$ (оценку снизу). Доказательства идентичны, покажем $T(n) = \mathcal{O}(\alpha^n)$. База индукции:

$$\exists C \colon \forall n \in B = [1 - \max_i a_i, 1] \;\; T(n) \leqslant C\alpha^n$$

Переход индукции:

$$T(n) = \sum b_i T(n - a_i) \overset{\text{по индукции}}{\leqslant} C \sum b_i \alpha^{n-a_i} \overset{(*)}{=} C\alpha^n$$

(*) Верно, так как α – корень уравнения. ■

1.5. Доказательства по индукции

Lm 1.5.1. *Доказательство по индукции*
Есть простой метод решения рекуррентных соотношений: угадать ответ, доказать его по индукции. Рассмотрим на примере $T(n) = \max\limits_{x=1..n-1}\big(T(x) + T(n-x) + x(n-x)\big)$.
Докажем, что $T(n) = \mathcal{O}(n^2)$, для этого достаточно доказать $T(n) \leqslant n^2$:
База: $T(1) = 1 \leqslant 1^2$.
Переход: $T(n) \leqslant \max\limits_{x=1..n-1}\big(x^2 + (n-x)^2 + x(n-x)\big) \leqslant \max\limits_{x=1..n-1}\big(x^2 + (n-x)^2 + 2x(n-x)\big) = n^2$

• Примеры по теме рекуррентные соотношения

1. $T(n) = T(n-1) + T(n-1) + T(n-2)$.
   Угадаем ответ 2^n, проверим по индукции: $2^n = 2^{n-1} + 2^{n-1} + 2^{n-2}$.
2. $T(n) = T(n-3) + T(n-3) \Rightarrow T(n) = 2T(n-3) = 4T(n-6) = \cdots = 2^{n/3}$
3. $T(n) = T(n-1) + T(n-3)$. Применяем Thm 1.4.3, получаем $1 = \alpha^{-1} + \alpha^{-3}$, находим α бинпоиском, получаем $\alpha = 1.4655\ldots$.

1.6. Числа Фибоначчи

Def 1.6.1. $f_1 = f_0 = 1, f_i = f_{i-1} + f_{i-2}$. f_n – *n-е число Фибоначчи.*

• Оценки снизу и сверху

$f_n = f_{n-1} + f_{n-2}$, рассмотрим $g_n = g_{n-1} + g_{n-1}$, $2^n = g_n \geqslant f_n$.
$f_n = f_{n-1} + f_{n-2}$, рассмотрим $g_n = g_{n-2} + g_{n-2}$, $2^{n/2} = g_n \leqslant f_n$.
Воспользуемся Thm 1.4.3, получим $1 = \alpha^{-1} + \alpha^{-2} \Leftrightarrow \alpha^2 - \alpha - 1 = 0$, получаем $\alpha = \frac{\sqrt{5}+1}{2} \approx 1.618$.
$f_n = \Theta(\alpha^n)$.

1.7. (*) $\mathcal{O}$-обозначения через пределы

Def 1.7.1. $f = o(g)$ *Определение через предел:* $\lim\limits_{n\to+\infty} \frac{f(n)}{g(n)} = 0$

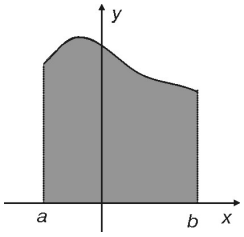
Def 1.7.2. $f = \mathcal{O}(g)$ *Определение через предел:* $\overline{\lim\limits_{n\to+\infty}} \frac{f(n)}{g(n)} < \infty$

Здесь необходимо пояснение: $\overline{\lim\limits_{n\to+\infty}} f(n) = \lim\limits_{n\to+\infty}(\sup\limits_{x\in[n..+\infty]} f(x))$, где sup – верхняя грань.

Lm 1.7.3. Определения o эквивалентны

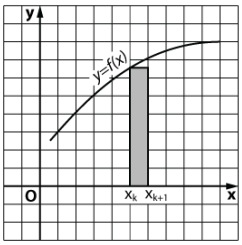
Доказательство. Вспомним, что речь о положительных функциях f и g.
Распишем предел по определению: $\forall C > 0 \quad \exists N \quad \forall n \geqslant N \quad \frac{f(n)}{g(n)} \leqslant C \Leftrightarrow f(n) \leqslant Cg(n)$. ■

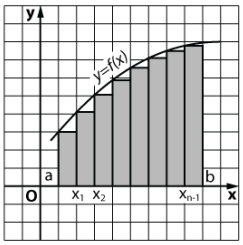
1.8. (*) Замена сумм на интегралы



Def 1.8.1. *Определённый интеграл* $\int\limits_a^b f(x)dx$ *положительной функции* $f(x)$ *– площадь под графиком* f *на отрезке* $[a..b]$.

Lm 1.8.2. $\forall f(x) \nearrow [a..a{+}1] \Rightarrow f(a) \leqslant \int\limits_a^{a+1} f(x)dx \leqslant f(a{+}1)$





Lm 1.8.3. $\forall f(x) \nearrow [a..b{+}1] \Rightarrow \sum\limits_{i=a}^{b} f(i) \leqslant \int\limits_a^{b+1} f(x)dx$

Доказательство. Сложили неравенства из Lm 1.8.2 ■

Lm 1.8.4. $\forall f(x) \nearrow [a..b], f > 0 \Rightarrow \int\limits_a^b f(x)dx \leqslant \sum\limits_{i=a}^{b} f(i)$

Доказательство. Сложили неравенства из Lm 1.8.2, выкинули $[a{-}1, a]$ из интеграла. ■

Теорема 1.8.5. *Замена суммы на интеграл #1*

$\forall f(x) \nearrow [1..\infty), f > 0, S(n) = \sum\limits_{i=1}^{n} f(i), I_1(n) = \int\limits_1^n, I_2(n) = \int\limits_1^{n+1}, I_1(n) = \Theta(I_2(n)) \Rightarrow \boxed{S(n) = \Theta(I_1(n))}$

Доказательство. Из лемм Lm 1.8.3 и Lm 1.8.4 имеем $I_1(n) \leqslant S(n) \leqslant I_2(n)$.
$C_1 I_1(n) \leqslant I_2(n) \leqslant C_2 I_1(n) \Rightarrow I_1(n) \leqslant S(n) \leqslant I_2(n) \leqslant C_2 I_1(n)$ ■

Теорема 1.8.6. *Замена суммы на интеграл #2*

$\forall f(x) \nearrow [a..b], f > 0 \quad \int\limits_a^b f(x)dx \leqslant \sum\limits_{i=a}^{b} f(i) \leqslant f(b) + \int\limits_a^b f(x)dx$

Доказательство. Первое неравенство – лемма Lm 1.8.3. Второе – Lm 1.8.4, применённая к $\sum\limits_{i=a}^{b-1}$. ■

Следствие 1.8.7. Для убывающих функций два последних факта тоже верны.
Во втором ошибкой будет не $f(b)$, а $f(a)$, которое теперь больше.

• Как считать интегралы?

Формула Ньютона-Лейбница: $\int\limits_a^b f'(x)dx = f(b) - f(a)$

Пример: $\ln'(n) = \frac{1}{n} \Rightarrow \int\limits_1^n \frac{1}{x}dx = \ln n - \ln 1 = \ln n$

1.9. Примеры по теме асимптотики

• Вложенные циклы for

```
#define forn(i, n) for (int i = 0; i < n; i++)
int counter = 0, n = 100;
forn(i, n)
  forn(j, i)
    forn(k, j)
      forn(l, k)
        forn(m, l)
          counter++;
cout << counter << endl;
```

Чему равен `counter`? Во-первых, есть точный ответ: $\binom{n}{5} \approx \frac{n^5}{5!}$. Во-вторых, мы можем сходу посчитать число циклов и оценить ответ как $\mathcal{O}(n^5)$, правда константа $\frac{1}{120}$ важна, оценка через $\mathcal{O}$ не даёт полное представление о времени работы.

• За сколько вычисляется n-е число Фибоначчи?

```
f[0] = f[1] = 1;
for (int i = 2; i < n; i++)
    f[i] = f[i - 1] + f[i - 2];
```

Казалось бы за $\mathcal{O}(n)$. Но это в предположении, что «+» выполняется за $\mathcal{O}(1)$. На самом деле мы знаем, что $\log f_n = \Theta(n)$, т.е. складывать нужно числа длины $n \Rightarrow$ «+» выполняется за $\Theta(i)$, а n-е число Фибоначчи считается за $\Theta(n^2)$.

• Задача из теста про $a^2 + b^2 = N$

```
int b = sqrt(N);
for (int a = 1; a * a <= N; a++)
    while (a * a + b * b >= N; b--)
        ;
    if (a * a + b * b == N)
        cnt++;
```

Время работы $\Theta(N^{1/2})$, так как в сумме b уменьшится лишь $N^{1/2}$ раз. Здесь мы первый раз использовали так называемый «метод двух указателей».

• Число делителей числа

```
vector<int> divisors[n + 1]; // все делители числа
for (int a = 1; a <= n; a++)
    for (int b = a; b <= n; b += a)
        divisors[b].push_back(a);
```

За сколько работает программа?

$$\sum_{a=1}^{n} \lceil \tfrac{n}{a} \rceil = \mathcal{O}(n) + \sum_{a=1}^{n} \tfrac{n}{a} = \mathcal{O}(n) + n \sum_{a=1}^{n} \tfrac{1}{a} \overset{Thm\ 1.8.5}{=} \mathcal{O}(n) + n \cdot \Theta(\int_{1}^{n} \tfrac{1}{x} dx) = \Theta(n \log n)$$

• **Сумма гармонического ряда**

Докажем более простым способом, что $\sum_{i=1}^{n} \frac{1}{i} = \Theta(\log n)$

$$1 + \lfloor \log_2 n \rfloor \geqslant \tfrac{1}{1} + \overbrace{\tfrac{1}{2} + \tfrac{1}{2}}^{1} + \overbrace{\tfrac{1}{4} + \tfrac{1}{4} + \tfrac{1}{4} + \tfrac{1}{4}}^{1} + \overbrace{\tfrac{1}{8} + \ldots}^{1\ldots} \geqslant \sum_{k=1}^{n} \tfrac{1}{k} = \tfrac{1}{1} + \tfrac{1}{2} + \tfrac{1}{3} + \tfrac{1}{4} + \tfrac{1}{5} + \tfrac{1}{6} + \tfrac{1}{7} + \tfrac{1}{8} + \ldots \geqslant$$
$$\tfrac{1}{1} + \underbrace{\tfrac{1}{2}}_{1/2} + \underbrace{\tfrac{1}{4} + \tfrac{1}{4}}_{1/2} + \underbrace{\tfrac{1}{8} + \tfrac{1}{8} + \tfrac{1}{8} + \tfrac{1}{8}}_{1/2} + \ldots \geqslant 1 + \tfrac{1}{2}\lfloor \log_2 n \rfloor \Rightarrow \sum_{k=1}^{n} \tfrac{1}{k} = \Theta(\log n)$$

1.10. Сравнение асимптотик

Def 1.10.1. *Линейная сложность* $\mathcal{O}(n)$

Def 1.10.2. *Квадратичная сложность* $\mathcal{O}(n^2)$

Def 1.10.3. *Полиномиальная сложность* $\exists\, k > 0\colon \mathcal{O}(n^k)$

Def 1.10.4. *Полилогарифм* $\exists\, k > 0\colon \mathcal{O}(\log^k n)$

Def 1.10.5. *Экспоненциальная сложность* $\exists\, c > 0\colon \mathcal{O}(2^{cn})$

Теорема 1.10.6. $\forall x, y > 0, z > 1\ \exists N\ \forall n > N\colon\ \log^x n < n^y < z^n$

Доказательство. Сперва докажем первую часть неравенства через вторую.
Пусть $\log n = k$, тогда $\log^x n < n^y \Leftrightarrow k^x < 2^{ky} = (2^y)^k = z^k \Leftarrow n^y < z^n$ ■
Докажем вторую часть исходного неравенства $n^y < z^n \Leftrightarrow n < 2^{\frac{1}{y} n \log z}$
Пусть $n' = \frac{1}{y} n \log z$, обозначим $C = 1/(\frac{1}{y}\log z)$, пусть $C \leqslant n'$ (возьмём достаточно большое n),
тогда $n^y < z^n \Leftrightarrow n < 2^{\frac{1}{y} n \log z} \Leftrightarrow C \cdot n' < 2^{n'} \Leftarrow (n')^2 < 2^{n'}$

Осталось доказать $n^2 < 2^n$. Докажем по индукции.
База: для любого значения из интервала [10..20) верно,
так как $n^2 \in [100..400) < 2^n \in [1024..1048576)$.
Если n увеличить в два раза, то $n^2 \to 4 \cdot n^2$, а $2^n \to 2^{2n} = 2^n \cdot 2^n \geqslant 4 \cdot 2^n$ при $n \geqslant 2$.
Значит $\forall n \geqslant 2$ если для n верно, то и для $2n$ верно.
Переход: $[10..20) \to [20..40) \to [40..80) \to \ldots$ ■

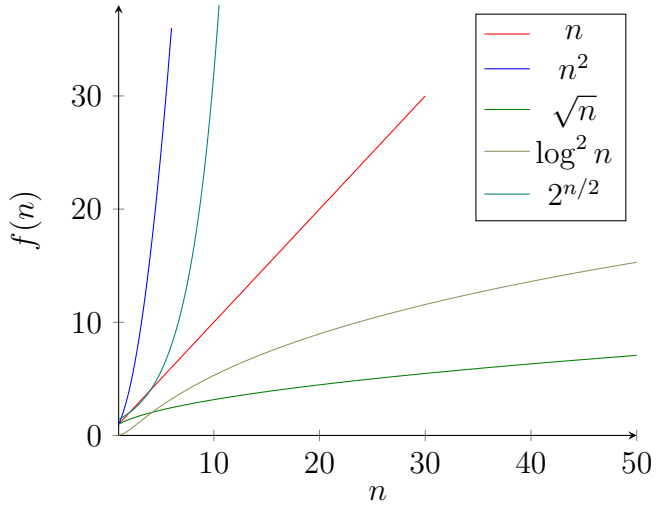
Следствие 1.10.7. $\forall x, y > 0, z > 1\colon\ \log^x n = \mathcal{O}(n^y), n^y = \mathcal{O}(z^n)$

Доказательство. Возьмём константу 1. ■

Следствие 1.10.8. $\forall x, y > 0, z > 1\colon\ \log^x n = o(n^y), n^y = o(z^n)$

Доказательство. Достаточно перейти к чуть меньшим y, z и воспользоваться теоремой.
$\exists N\, \forall n \geqslant N\ \log^x n < n^{y-\varepsilon} = \frac{1}{n^\varepsilon} n^y,\ \frac{1}{n^\varepsilon} \underset{n\to\infty}{\to} 0 \Rightarrow \log^x = o(n^y)$.
$\exists N\, \forall n \geqslant N\ n^y < (z-\varepsilon)^n = \frac{1}{(z/(z-\varepsilon))^n} z^n,\ \frac{1}{(z/(z-\varepsilon))^n} \underset{n\to\infty}{\to} 0 \Rightarrow n^y = o(z^n)$. ■

• **Посмотрим как ведут себя функции на графике**



Заметим, что $2^{n/2}, n^2$ и $\log^2 n, \sqrt{n}$ на бесконечности ведут себя иначе: