

# Введение в структуры данных. Анализ сложности алгоритмов

---


Для вопросов по курсу:

Иванов Дмитрий Владимирович, [dmitry.ivanov@moevm.info](mailto:dmitry.ivanov@moevm.info)

Префикс в теме письма [CS\_23XX]

# Как сравнивать алгоритмы?

- количество строк кода ? нет
- миллисекунды ? нет
- сложность ! **да**



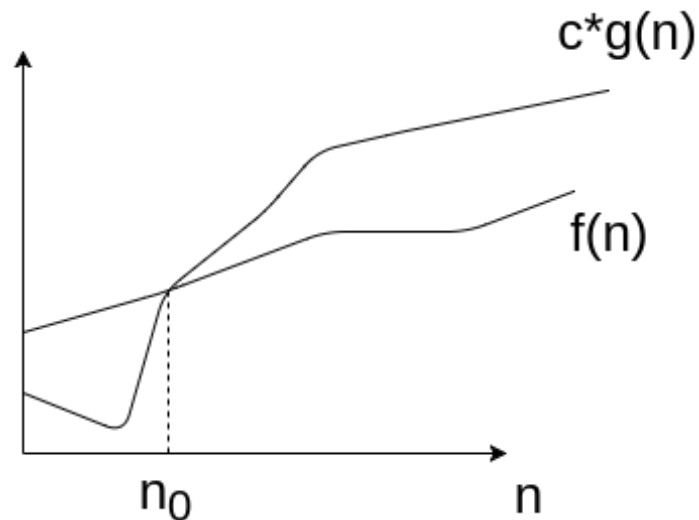
$O$ -символика  
 $\Omega$ -символика  
 $\Theta$ -символика  
...

# O-символика

➤  $f(n) = O(g(n))$

Функция  $f(n)$  ограничена **сверху** функцией  $c \cdot g(n)$  ( $c > 0$ )

для всех  $n \geq n_0$



# O-символика

➤ Пример нахождения с:

$$10x^2 + 5x + 7 = O(x^2)$$

$$10x^2 + 5x + 7 \leq 10x^2 + 5x^2 + 7x^2 = 22x^2$$

таким образом,  $c = 22$ ,  $g(x) = x^2$

## $\Omega$ -символика и $\Theta$ -символика

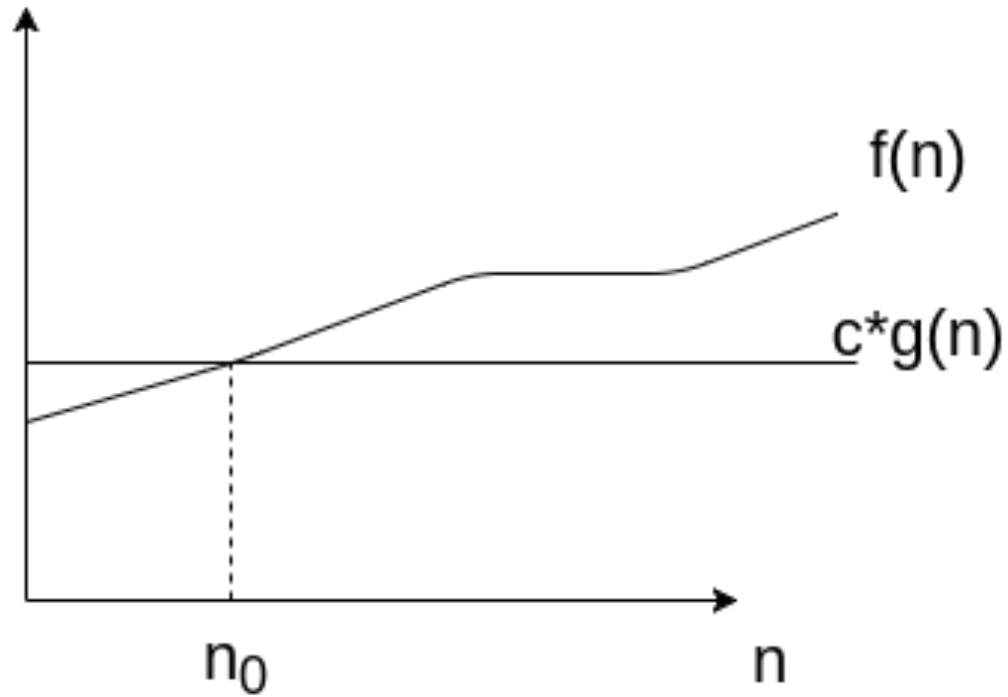
➤  $f(n) = \Omega(g(n))$

Функция  $f(n)$  ограничена **снизу** функцией  $c^*g(n)$  для всех  $n \geq n_0$

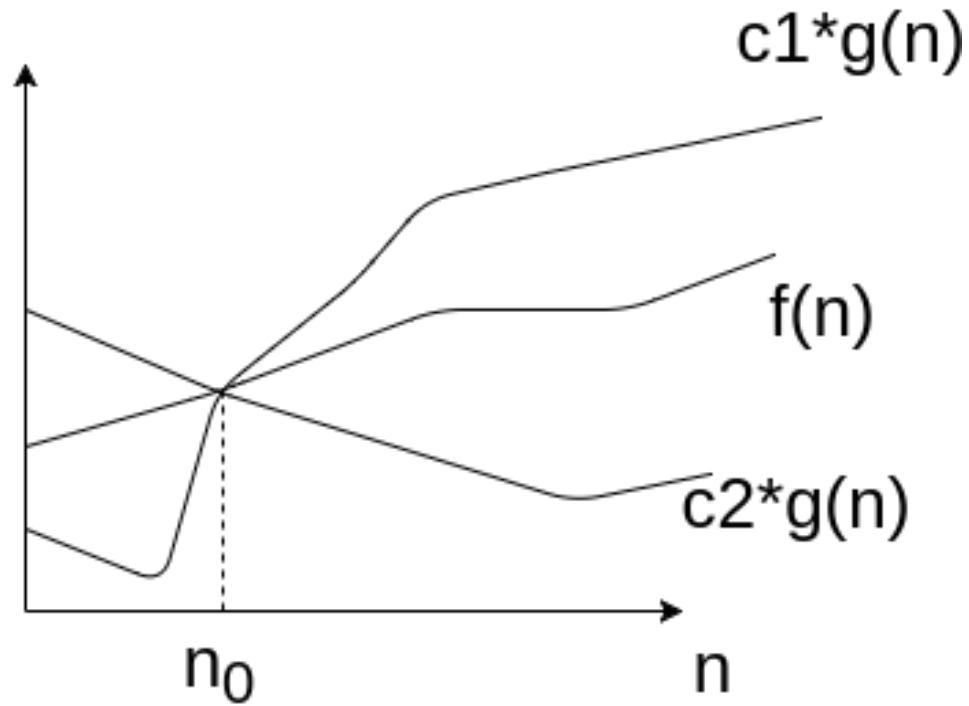
➤  $f(n) = \Theta(g(n))$

Функция  $f(n)$  ограничена **сверху** функцией  $c1^*g(n)$ , а **снизу** функцией  $c2^*g(n)$  для всех  $n \geq n_0$

$$f(n) = \Omega(g(n))$$



$$f(n) = \Theta(g(n))$$



# Скорость роста функций

	$\log n$	$n$	$n \log n$	$n^2$	$2^n$	$n!$
10	0,003 мкс	0,01 мкс	0,033 мкс	0,1 мкс	1 мкс	3,63 мкс
20	0,004 мкс	0,02 мкс	0,086 мкс	0,4 мкс	1 мс	77,1 лет
50	0,006 мкс	0,05 мкс	0,282 мкс	2,5 мкс	13 дней	$8,4 * 10^{15}$ лет
100	0,007 мкс	0,1 мкс	0,644 мкс	10 мкс	$4 * 10^{13}$ лет	
1000	0,010 мкс	1,0 мкс	0,966 мкс	1 мс		
10 000	0,013 мкс	10 мкс	130 мкс	100 мс		
100 000	0,017 мкс	0,1 мс	1,67 мс	10 с		
1 000 000 000	0,03 мкс	1 с	29,90 с	31,7 лет		



# Коллекции. Общие сведения

Коллекции - это специальные хранилища объектов.

Коллекции предоставляют множество методов для удобной работы:

- Добавление нового объекта в коллекцию
- Удаление объекта из коллекции
- Сортировка объектов в коллекции
- Обращение к объекту коллекции
- ...

а также методы, специфичные для типа коллекции.

# Работа с коллекциями

- Количество элементов: `len()`
- Операторы `in`, `not in`
- Обход коллекции в цикле `for`
- `min()`, `max()` (элементы коллекции должны иметь возможность сравнения)
- `sorted()`

# Списки и массивы

- **Массив** - это упорядоченная коллекция объектов одинакового типа.
- **Список** - это упорядоченная коллекция объектов любого типа.
- Список в языке Python реализован на основе динамического массива указателей языка Си.

# Массивы

- **Массив** - это структура данных, которая хранит набор элементов одинакового типа, доступ к которым осуществляется по индексу.

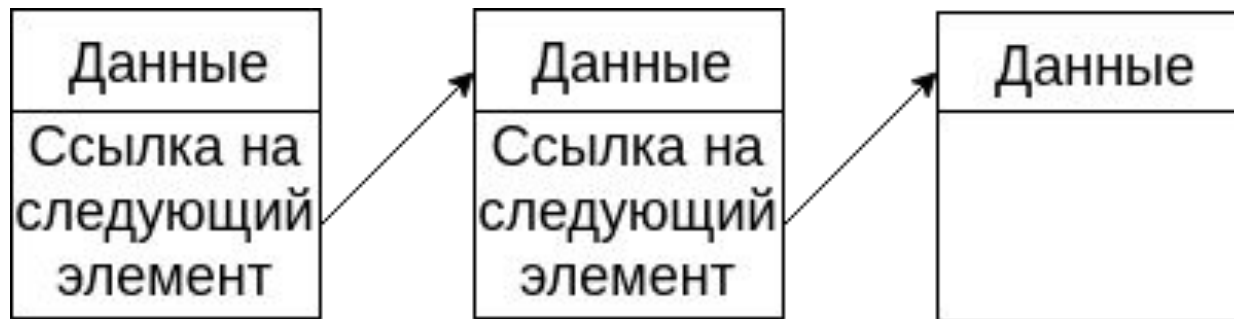
Значение	2.5	3.5	1.7	0.9	10.03	6.45	33.2
Индекс	0	1	2	3	4	5	6

# Массивы

	В начало	В середину	В конец
Вставка элемента	$O(n)$	$O(n)$	$O(1)$
Удаление элемента	$O(n)$	$O(n)$	$O(1)$

# Связные списки

- **Связный список** - структура данных, каждый из элементов которой содержит как собственные данные, так и некоторое количество ссылок на следующий и/или предыдущий узел списка.



# Односвязные списки

	Начало	Середина	Конец
Вставка элемента	$O(1)$	$O(n)$	$O(1)$ или $O(n)$
Удаление элемента	$O(1)$	$O(n)$	$O(n)$

# Двусвязные списки

	В начало	В середину	В конец
Вставка элемента	$O(1)$	$O(1)$ или $O(n)$	$O(1)$ или $O(n)$
Удаление элемента	$O(1)$	$O(1)$ или $O(n)$	$O(1)$ или $O(n)$



# Источники и полезные ссылки

- Курс на Stepik “Алгоритмы: теория и практика. Методы”  
<https://stepik.org/course/217/syllabus>
- Курс на Stepik “Алгоритмы: теория и практика. Структуры данных”  
<https://stepik.org/course/1547>
- Дональд Э. Кнут. Искусство программирования, том 1. Основные алгоритмы/Дональд Э. Кнут //Москва: Вильямс. – 2000. – Т. 712.
- Скиена С. Алгоритмы. Руководство по разработке //СПб.: БХВ-Петербург. – 2011. – Т. 720.
- [https://ru.wikipedia.org/wiki/Временная\\_сложность\\_алгоритма](https://ru.wikipedia.org/wiki/Временная_сложность_алгоритма)

# Вопросы по курсу можно задавать:

---

Иванов Дмитрий Владимирович  
[dmitry.ivanov@moevm.info](mailto:dmitry.ivanov@moevm.info)