ИНФОРМАТИКА

```
~/polytech $ date
осень 2019
~/polytech $ name
константин кориков
~/polytech $ topic
алгоритмы и структуры данных. начало
```

О ЧЁМ ПОГОВОРИМ?

1. Как учиться (бонус)
2. Определения
3. Как сравнивать алгоритмы
4. Задачи

Как учиться?

Определения

Алгоритм (algorithm) — это формально описанная вычислительная процедура, получающая **исходные данные** (input), называемые также входом алгоритма или его аргументом, и выдающая **результат** вычислений на выход (output).

~последовательность действий

Абстрактный тип данных — это математическая модель плюс различные операторы, определенные в рамках этой модели.

~интерфейс работы с данными

Для представления АТД используются *структуры данных,* которые представляют собой набор переменных, возможно, различных типов данных, объединенных определенным образом.

~конкретная реализация АТД

Хотя термины *тип данных* (или просто *тип*), *структура данных* и *абстрактный тип данных* звучат **похоже**, но имеют они различный смысл. В языках программирования *тип данных* переменной **обозначает** множество значений, которые может принимать эта переменная.

(👨🏾‍🏫 на доске)

Что читать?


ТОМАС КОРМЕН
ЧАРЛЬЗ ЛЕЙЗЕРСОН
РОНАЛЬД РИВЕСТ
КЛИФФОРД ШТАЙН
АЛГОРИТМЫ
ПОСТРОЕНИЕ И АНАЛИЗ
ТРЕТЬЕ ИЗДАНИЕ


Альфред В. Ахо, Джон Э. Хопкрофт, Джеффри Д. Ульман
Структуры ДАННЫХ и АЛГОРИТМЫ
Классическое издание

Что читать?


С. Дасгупта
Х. Пападимитриу
У. Вазирани
Алгоритмы


Грокаем
алгоритмы
Иллюстрированное пособие
для программистов и любопытствующих
Адитья Бхаргава
«Грокнуть» означает понять так полно, что наблюдатель становится частью объекта наблюдения...
Р. Хайнлайн
ПИТЕР

Нужная математика

Нужная математика

Разделить

$$\log_a x = \log_b x \log_a b$$

$$1 + 2 + 3 + \ldots + n = (n+1)\frac{n}{2}$$

$$q^0 + q^1 + q^2 + \ldots + q^{n-1} = \frac{1 - q^n}{1 - q} \quad (q \neq 1)$$

Умножить

(👨🏾‍🏫 на доске)

Нужная математика

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

Меньше

$$\lim_{x\to\infty}\frac{f(x)}{g(x)}=c\ \ (c\neq 0)$$

Одинаково

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

Больше

(👨🏾‍🏫 на доске)

Нужная математика

$$n! = n(n-1)!$$

$$F_i = F_{i-1} + F_{i-2},\ \ F_0 = 0,\ \ F_1 = 1$$

$$T(n) = T(n/2) + 1,\ \ T(1) = 1$$

Рекуррентные формулы

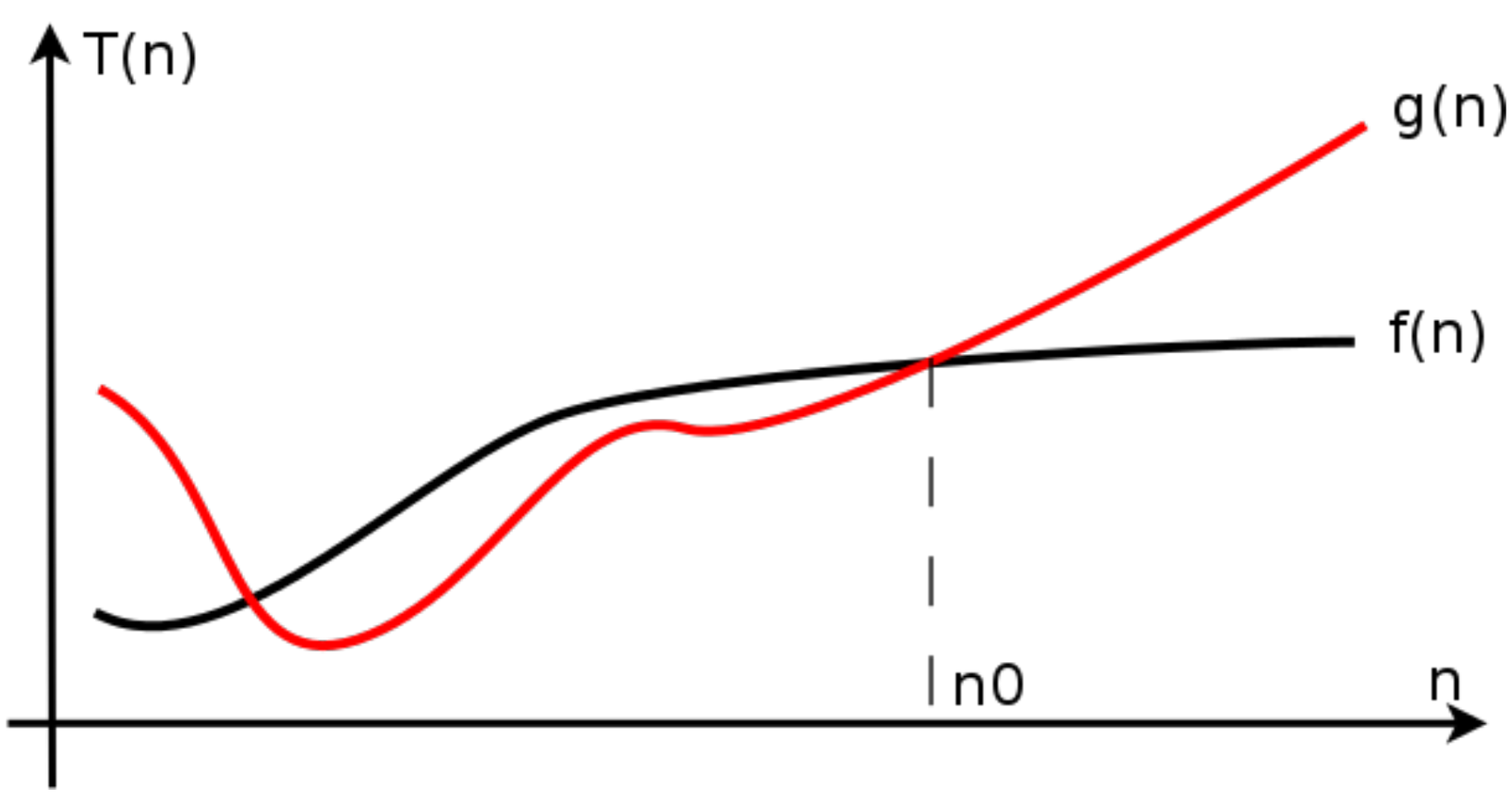
(👩🏾‍🏫 на доске)

Нужная математика

О-большое →

$$f(n) = O(g(n))$$

$$|f(n)| \leq C|g(n)|, \ n \geq n_0$$



Нужная математика

$$f(n) = O(g(n))$$

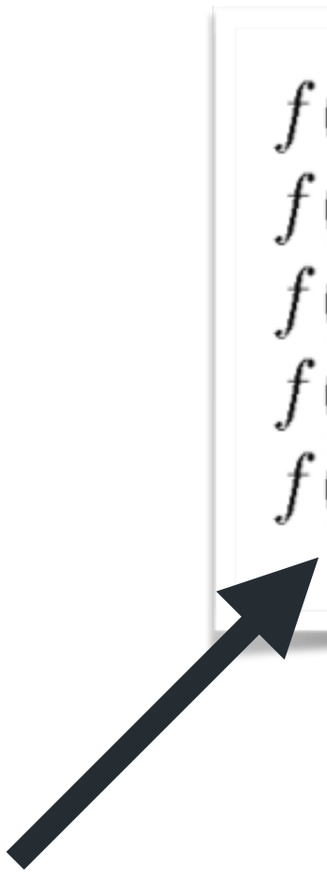
Свойства

$$\beta g(n) = O(g(n))$$

$$f(n) + g(n) = O(\max[f(n), g(n)])$$

Нужная математика

$$
\begin{array}{lcl}
f(n) = O(g(n)) & \approx & a \leqslant b \\
f(n) = \Omega(g(n)) & \approx & a \geqslant b \\
f(n) = \Theta(g(n)) & \approx & a = b \\
f(n) = o(g(n)) & \approx & a < b \\
f(n) = \omega(g(n)) & \approx & a > b
\end{array}
$$



Аналогия и связанные определения

Задача №1

Чтобы зарегистрироваться на сайте надо ввести свои логин и пароль. Не может быть двух пользователей с одинаковыми логинами. Поэтому сервер должен убедиться, что имя пользователя уникально.

Напишите программу, которая проверяет нового пользователя в списке зарегистрированных.

(👨🏾‍🏫 на доске)

```python
# Линейный поиск
def check(login, users):
    for x in users:
        if login == x:
            return True
    return False


users = ['megatron', 'superman', 'ira_dota', 'masha18', 'ivan']

result_1 = check('ira_dota', users)
result_2 = check('kirill', users)

print(f'ira_dota: {result_1}, kirill: {result_2}')
```

```python
# Бинарный поиск
def check(login, users):
    low = 0
    high = len(users)-1
    while low <= high:
        m = low + (high-low)//2 # эквивалентно (high+low)//2
        if users[m] == login:
            return True
        elif users[m] < login:
            low = m+1
        else:
            high = m-1
    return False


users = ['megatron', 'superman', 'ira_dota', 'masha18', 'ivan']

users_sorted = sorted(users)

result_1 = check('ira_dota', users_sorted)
result_2 = check('kirill', users_sorted)

print(f'ira_dota: {result_1}, kirill: {result_2}')
```

В общем случае можно возвращать позицию элемента, а не просто True или False. Для этого надо немного изменить код

Как сравнивать алгоритмы?

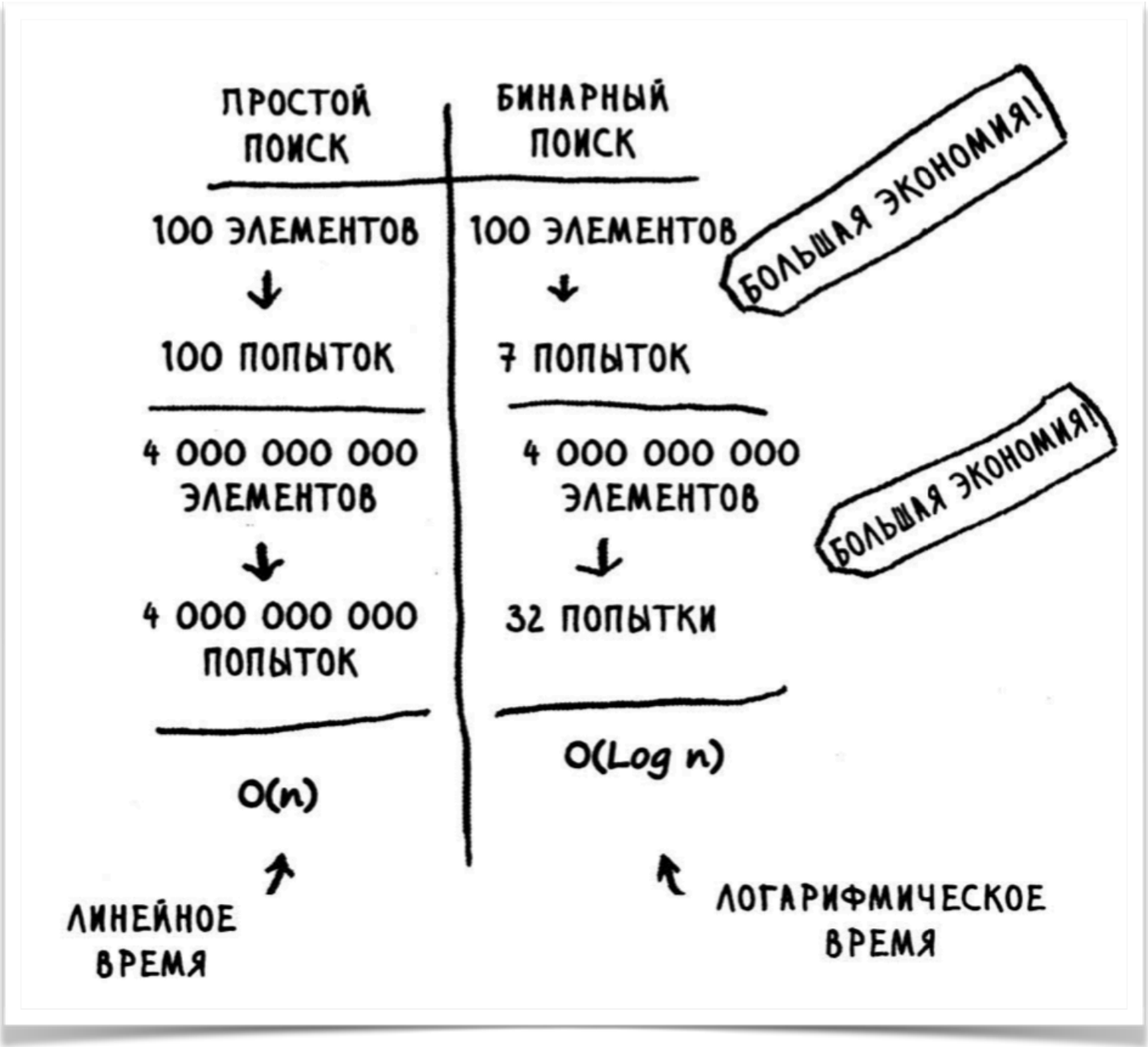
Время работы?

Память?

Количество инструкций?

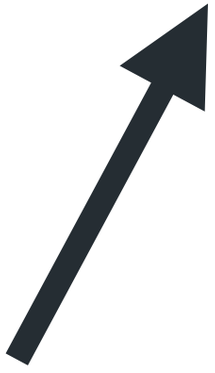
Скорость роста времени или памяти?

Асимптотическая сложность



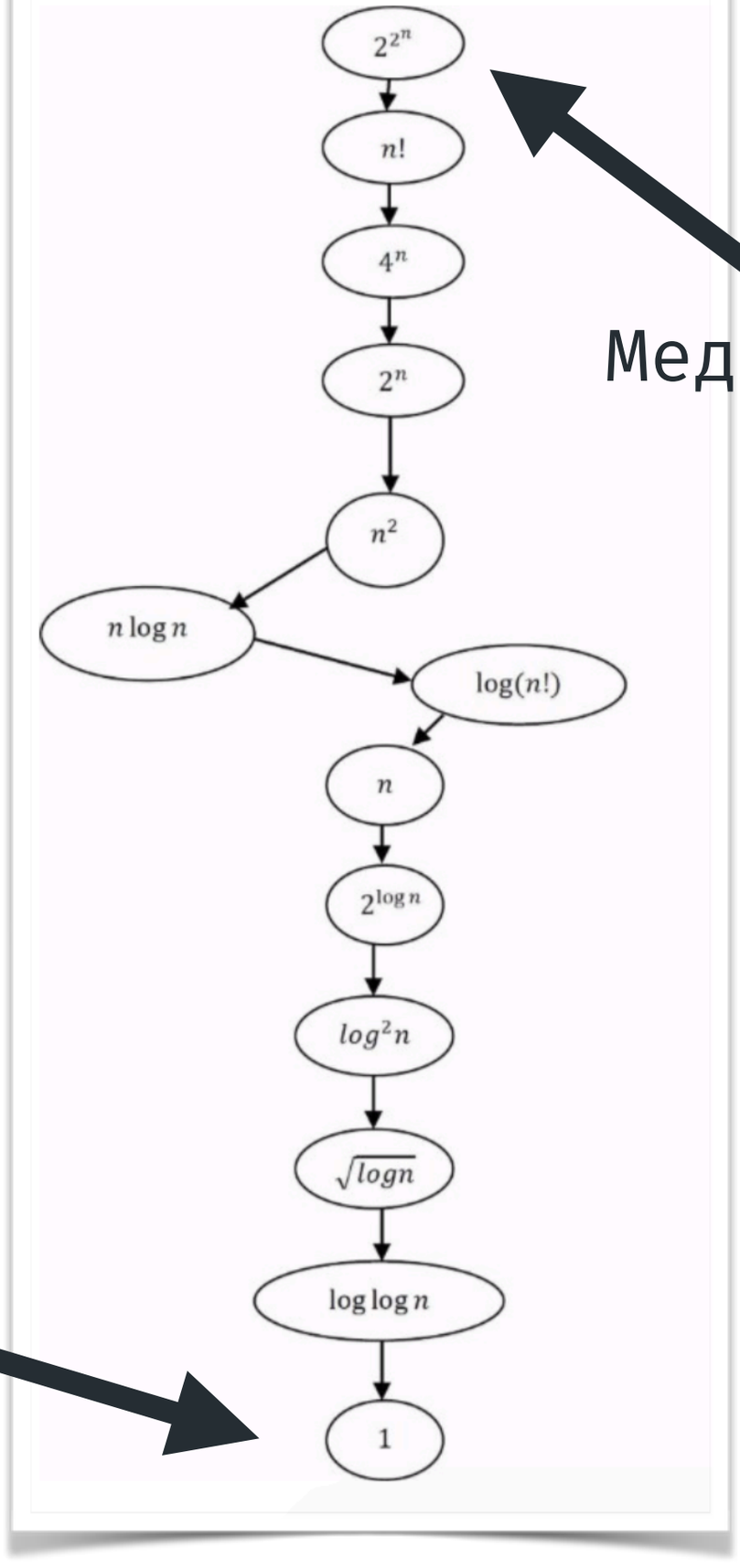
Амортизационная стоимость

$$\tilde{T}(n) = \frac{T(n)}{n}$$



Средняя стоимость одной из n операций в наихудшем случае

(👨🏾‍🏫 на доске)



Медленный алгоритм…

Быстрый алгоритм!

Факториальная

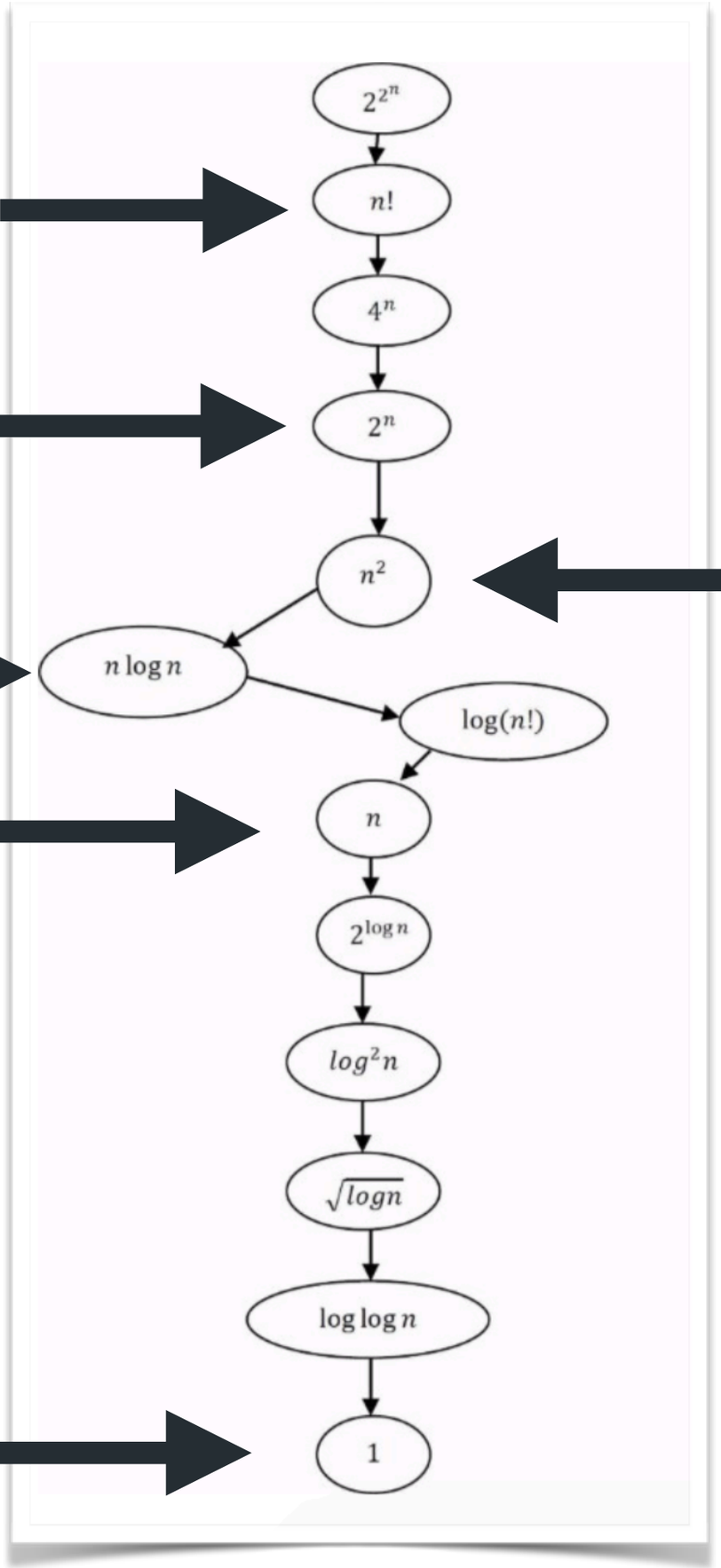
Экспоненциальная

Полиномиальная

Квазилинейная

Линейная

Константная



<BIG-O-CHEATSHEET>

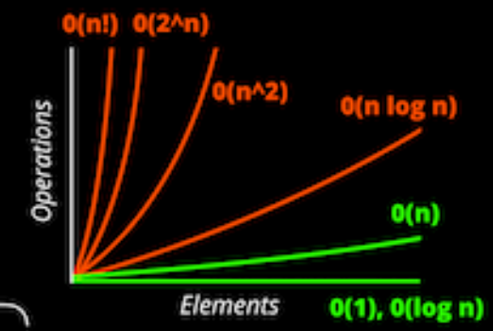
www.bigocheatsheet.com

LEGEND

TIME Complexity vs. **SPACE** Complexity

Good Fair Bad

Good Fair Bad



DATA STRUCTURE Operations

DATA Structure

TIME Complexity

SPACE Complexity

Average: Access, Search, Insertion, Deletion

Worst: Access, Search, Insertion, Deletion

	Average Access	Average Search	Average Insertion	Average Deletion	Worst Access	Worst Search	Worst Insertion	Worst Deletion	Worst Space
Array	Θ(1)	Θ(n)	Θ(n)	Θ(n)	0(1)	0(n)	0(n)	0(n)	0(n)
Stack	Θ(n)	Θ(n)	Θ(1)	Θ(1)	0(n)	0(n)	0(1)	0(1)	0(n)
Queue	Θ(n)	Θ(n)	Θ(1)	Θ(1)	0(n)	0(n)	0(1)	0(1)	0(n)
Singly-Linked List	Θ(n)	Θ(n)	Θ(1)	Θ(1)	0(n)	0(n)	0(1)	0(1)	0(n)
Doubly-Linked List	Θ(n)	Θ(n)	Θ(1)	Θ(1)	0(n)	0(n)	0(1)	0(1)	0(n)
Skip List	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	0(n)	0(n)	0(n)	0(n)	0(n log(n))
Hash Table	N/A	Θ(1)	Θ(1)	Θ(1)	N/A	0(n)	0(n)	0(n)	0(n)
Binary Search Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	0(n)	0(n)	0(n)	0(n)	0(n)
Cartesian Tree	N/A	Θ(log(n))	Θ(log(n))	Θ(log(n))	N/A	0(n)	0(n)	0(n)	0(n)
B-Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	0(log(n))	0(log(n))	0(log(n))	0(log(n))	0(n)
Red-Black Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	0(log(n))	0(log(n))	0(log(n))	0(log(n))	0(n)
Splay Tree	N/A	Θ(log(n))	Θ(log(n))	Θ(log(n))	N/A	0(log(n))	0(log(n))	0(log(n))	0(n)
AVL Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	0(log(n))	0(log(n))	0(log(n))	0(log(n))	0(n)
KD Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	0(n)	0(n)	0(n)	0(n)	0(n)

ARRAY SORTING Algorithms

ARRAY Algorithms

TIME Complexity

SPACE Complexity

	Best	Average	Worst	Worst Space
Quicksort	Ω(n log(n))	Θ(n log(n))	0(n^2)	0(log(n))
Mergesort	Ω(n log(n))	Θ(n log(n))	0(n log(n))	0(n)
Timsort	Ω(n)	Θ(n log(n))	0(n log(n))	0(1)
Heapsort	Ω(n log(n))	Θ(n log(n))	0(n log(n))	0(1)
Bubble Sort	Ω(n)	Θ(n^2)	0(n^2)	0(1)
Insertion Sort	Ω(n)	Θ(n^2)	0(n^2)	0(1)
Selection Sort	Ω(n^2)	Θ(n^2)	0(n^2)	0(1)
Tree Sort	Ω(n log(n))	Θ(n log(n))	0(n^2)t	0(n)
Shell Sort	Ω(n log(n))	Θ(n(log(n))^2)	0(n(log(n))^2)	0(1)
Bucket Sort	Ω(n+k)	Θ(n+k)	0(n^2)	0(n)
Radix Sort	Ω(nk)	Θ(nk)	0(nk)	0(n+k)
Counting Sort	Ω(n+k)	Θ(n+k)	0(n+k)	0(k)
Cubesort	Ω(n)	Θ(n log(n))	0(n log(n))	0(n)

Задача №2

В системе регистрации был сбой и в списке идентификаторов пользователей есть повторяющиеся. Напишите программу, которая проверяет повторы.

(👨🏾‍🏫 на доске)

Шпаргалка

- Бинарный поиск работает намного быстрее простого.
- Время выполнения $O(\log n)$ быстрее $O(n)$, а с увеличением размера списка, в котором ищется значение, оно становится намного быстрее.
- Скорость алгоритмов не измеряется в секундах.
- Время выполнения алгоритма описывается *ростом* количества операций.
- Время выполнения алгоритмов выражается как «О-большое».