PRODUCT
STAR.RU

Структуры данных и алгоритмы

Содержание урока

- Алгоритмы
- Структуры данных

НИКОЛАЙ АНТОНОВ

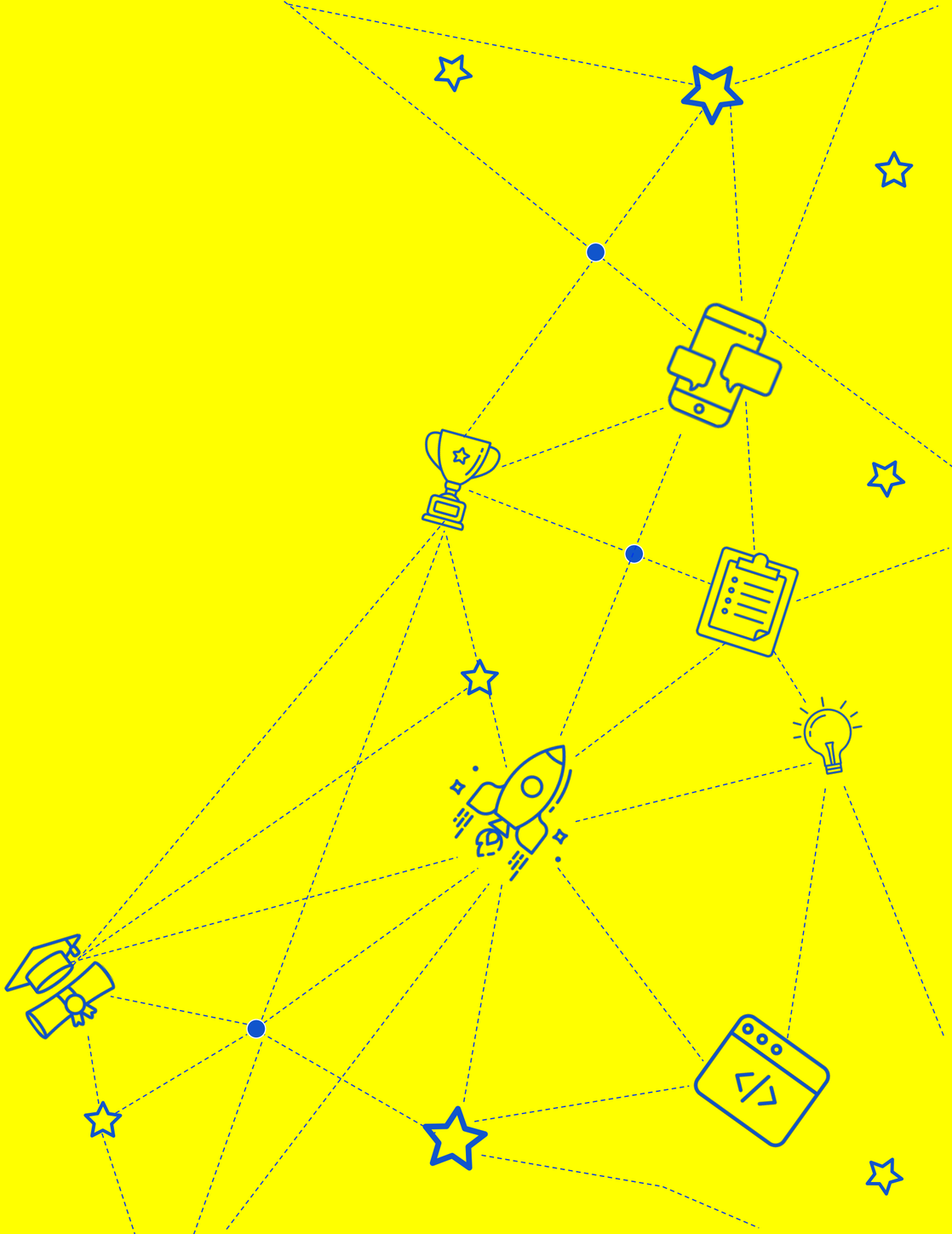
Software Engineer, vektor.ai

ex-Yandex.Cloud

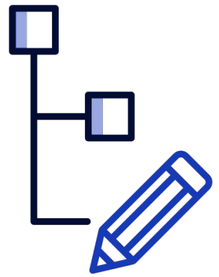
ex-exactpro

- Профессионально программирую больше 10 лет
- Участник четвертьфинала студенческой олимпиады ACM ICPC

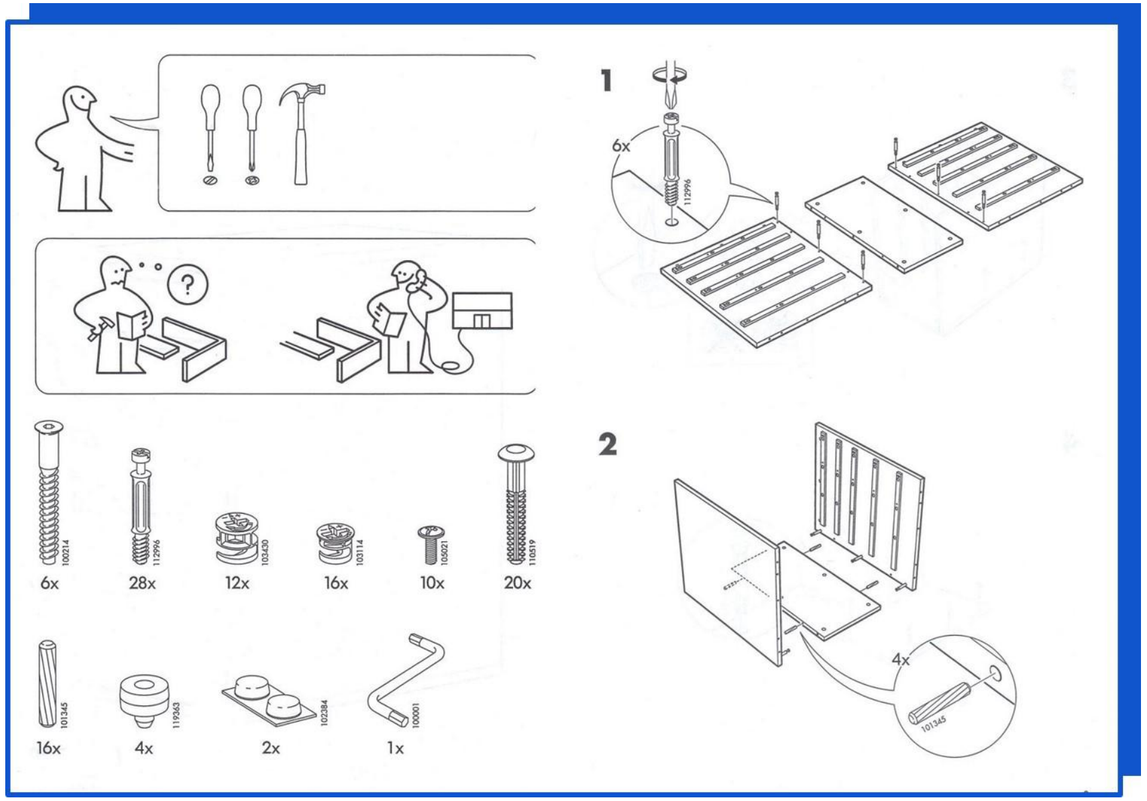
Алгоритмы



Алгоритмы

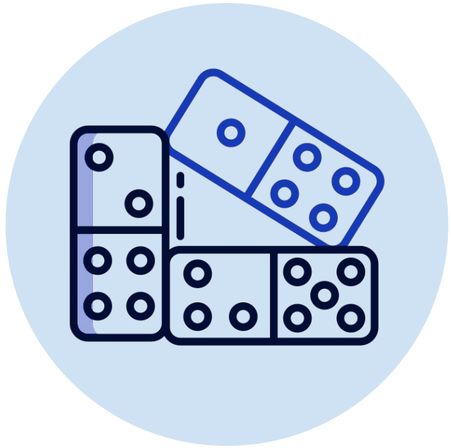


Алгоритм — конечная совокупность точно заданных правил решения некоторого класса задач или набор инструкций, описывающих порядок действий исполнителя для решения определённой задачи.



Свойства алгоритма

Конечность



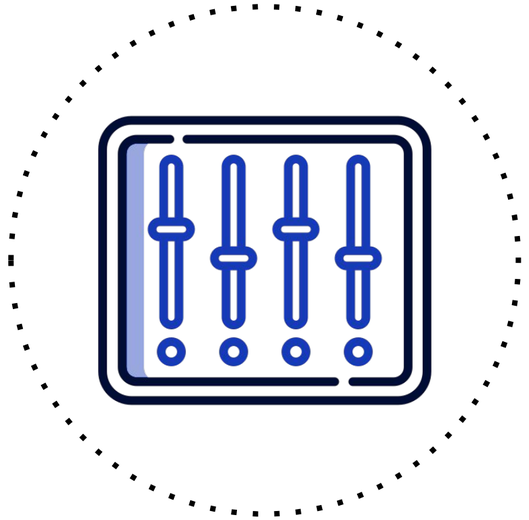
Массовость

Детерминированность, понятность

Результативность

Сортировка выбором

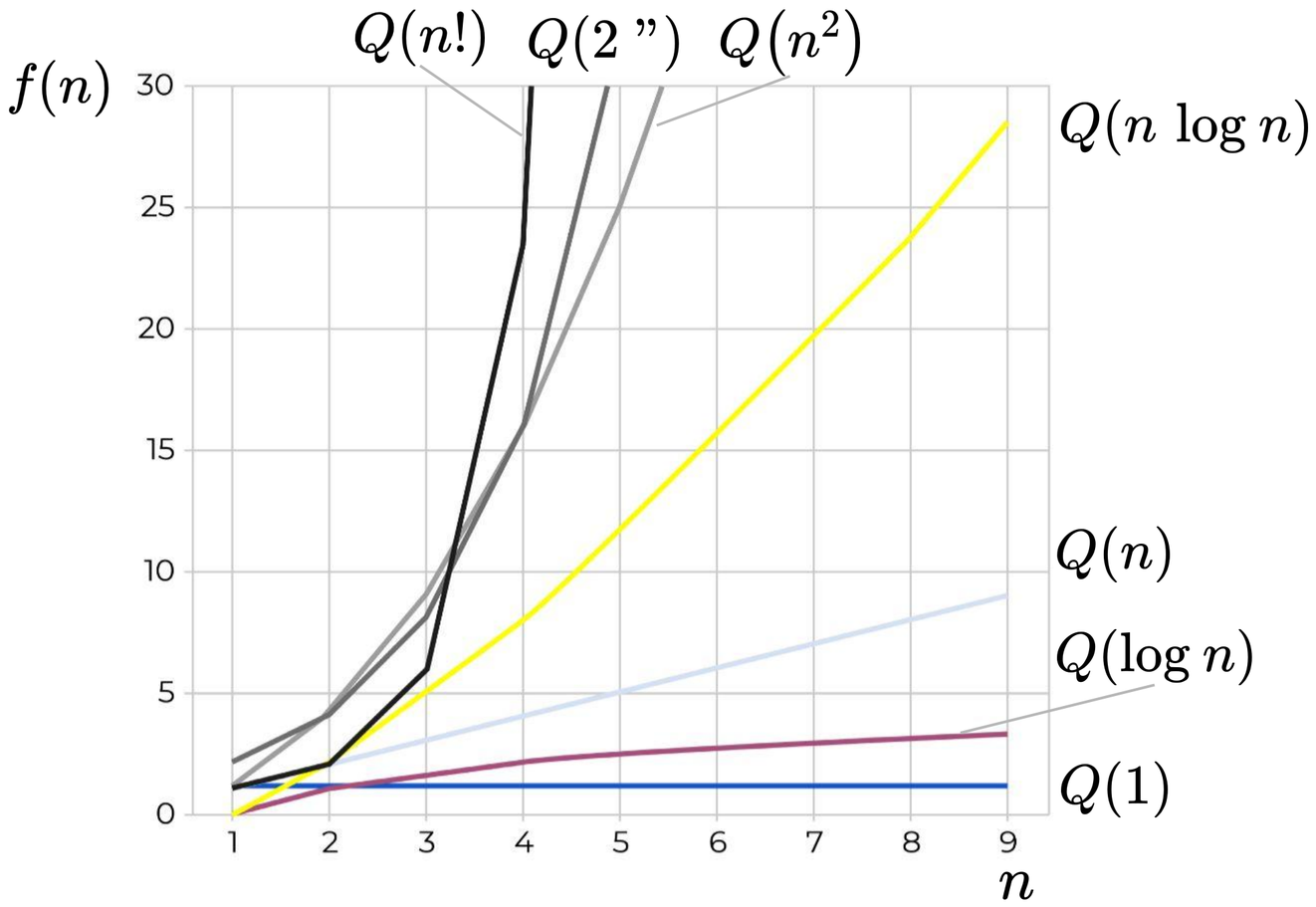
```
def select_sort(A):
①   for i in range(len(A) - 1):
②       min_index = i
        for k in range(i + 1, len(A)):
            if A[k] < A[min_index]:
                min_index = k
③       tmp = A[min_index]
        A[min_index] = A[i]
        A[i] = tmp
    return A
```



Сортировка выбором

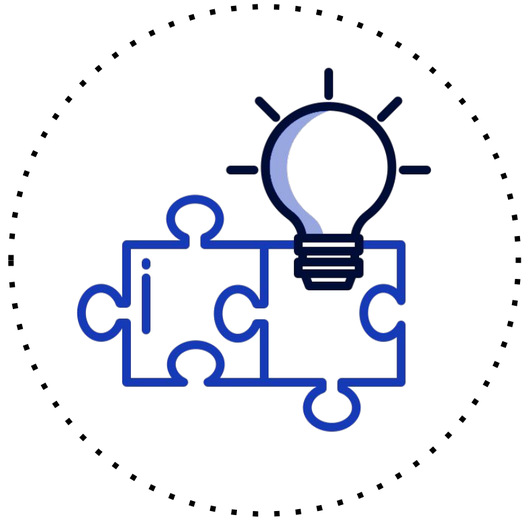
```
def select_sort(A):
①   for i in range(len(A) - 1):
②       min_index = i
            for k in range(i + 1, len(A)):
                if A[k] < A[min_index]:
                    min_index = k
③       A[i], A[min_index] = A[min_index], A[i]
    return A
```

Вычислительная сложность



Сортировка вставками

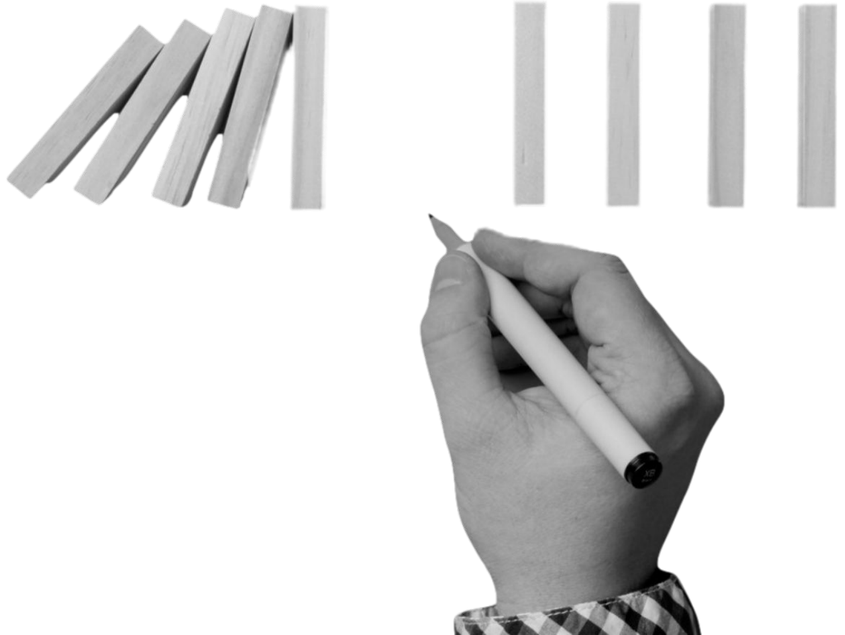
```
def insertion_sort(A):
①   for i in range(1, len(A)):
②       j = i - 1
        tmp = A[i]
③       while (j >= 0 and tmp < A[j]):
            A[j + 1] = A[j]
            j = j - 1
④       A[j + 1] = tmp
```



Сортировка Шелла

```
def shell_sort(A):
    n = len(A)
    interval = n // 2
①   while interval > 0:
②       for i in range(interval, n):
            temp = A[i]
            j = i
③           while j >= interval and A[j - interval] > temp:
                A[j] = A[j - interval]
                j -= interval

            A[j] = temp
        interval //= 2
```



А что на практике?

```
>>> x = [1,2,7,3,5,6,3]
>>> x.sort()
>>> print(x)
[1, 2, 3, 3, 5, 6, 7]
```

Динамическое программирование

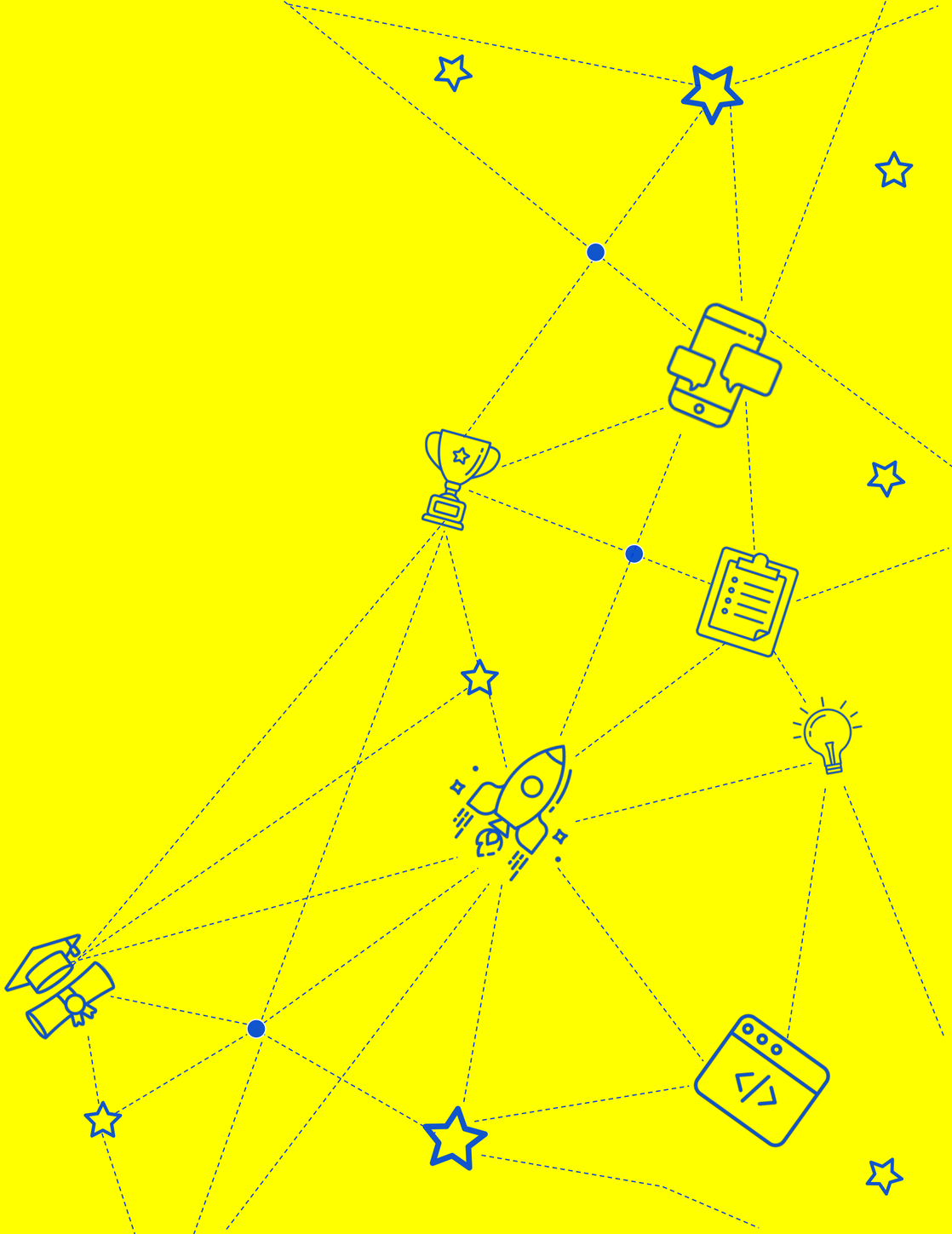
```
def fibonacci(n):
    if n < 2:
        return 1
    return fibonacci(n - 1) + fibonacci(n - 2)
```

```
def fibonacci(n):
    dp = [0, 1]
    for i in range(2, n+1):
        dp.append(dp[i-1] + dp[i-2])
    return dp[n]
```

Резюме

- **Алгоритмы –** набор инструкции для решения задач.
- Одна и та же задача может решаться различными алгоритмами. Алгоритмы можно сравнивать по вычислительной сложности, по количеству используемой памяти.
- Существуют методы построения алгоритмов: динамическое программирование, жадные алгоритмы, «разделяй и властвуй», специализированные алгоритмы.

Структуры данных

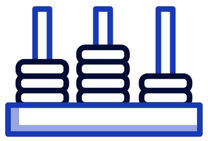


Структуры данных

Структуры данных — программная единица, позволяющая хранить и обрабатывать однотипные и/или логически связанные данные.

Структуры данных

Массивы

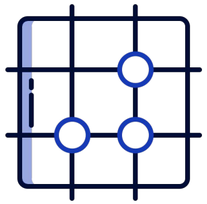


Стек и Очередь

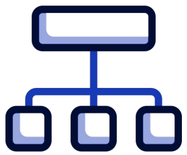
Связный список

Словарь

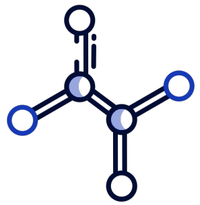
Деревья



Хеш-Таблица



Префиксное дерево

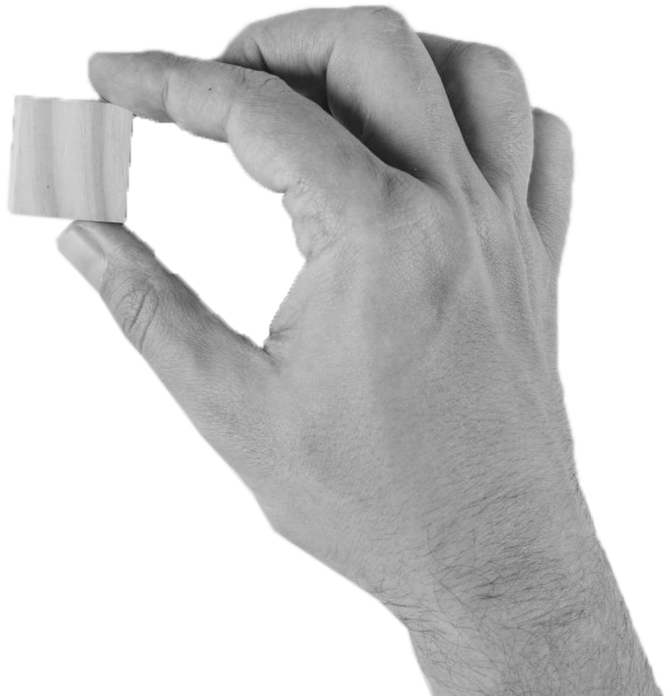


Графы

Массив (Array)

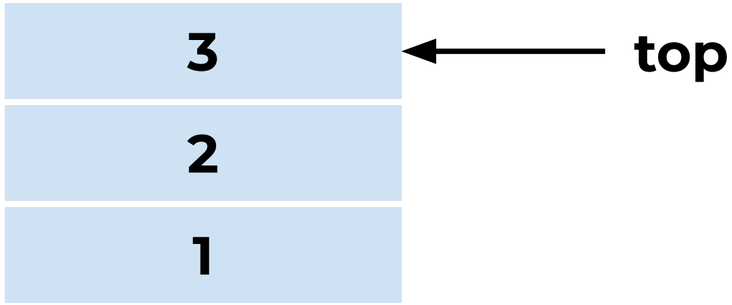
Массив — структура данных, представляющая собой непрерывную последовательность значений.

1	2	3	4



Стек (Stack)

Стек — структура данных с доступом «Последний пришел, Первый ушел».

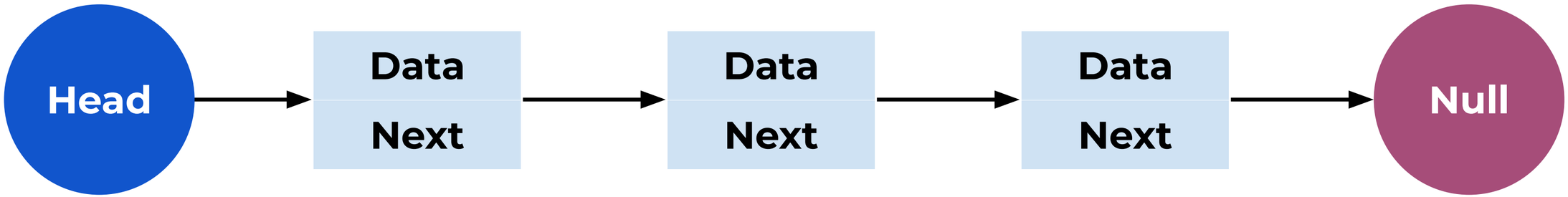


Операции:

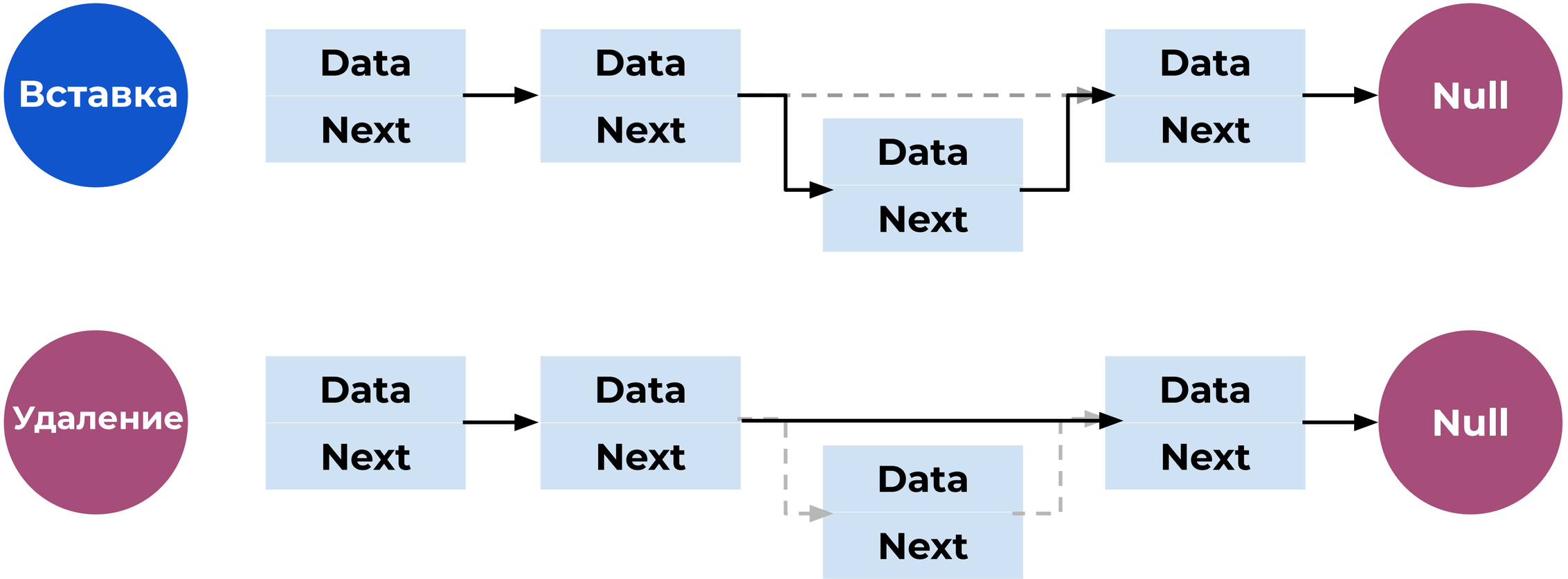
- **push –** положить на вершину стека
- **pop –** взять с вершины стека
- **peek –** посмотреть что на вершине стека
- **size –** узнать сколько элементов лежит в стеке

Связный список (Linked List)

Связный список – структура данных с последовательным доступом.

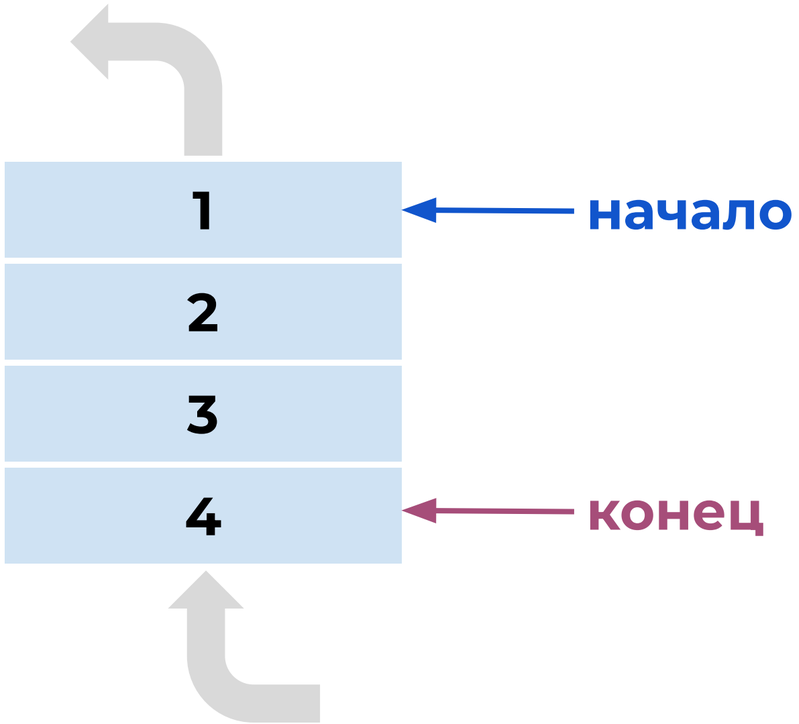


Связный список



Очередь (Queue)

Очередь — структура данных с доступом «Первый пришел, Первый ушел».



Операции:

- ★ **offer –** добавить элемент в конец очереди
- ★ **pull –** достать элемент из начала очереди
- ★ **peek –** посмотреть что лежит в начале очереди
- ★ **size –** узнать количество элементов в очереди

Словарь / Карта (Map)

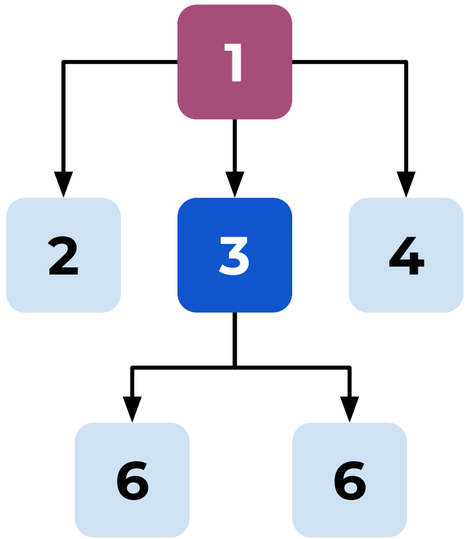
Словарь — ассоциативный массив, структура данных которая по индексу (ключу) возвращает значение. Ключём может быть любой тип данных.

Операции:

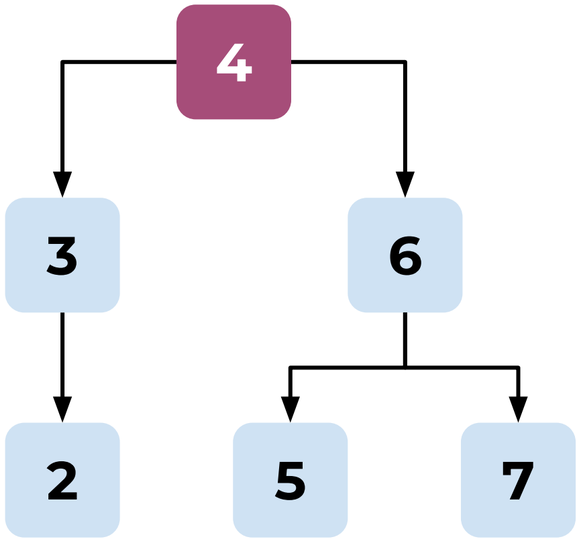
- ★ **put –** положить новое значение в словарь
- ★ **get –** искать по ключу в словаре
- ★ **remove –** удалить по ключу из словаря
- ★ **size –** узнать сколько элементов лежит в словаре

Деревья (Tree)

Деревья — иерархическая структура данных.

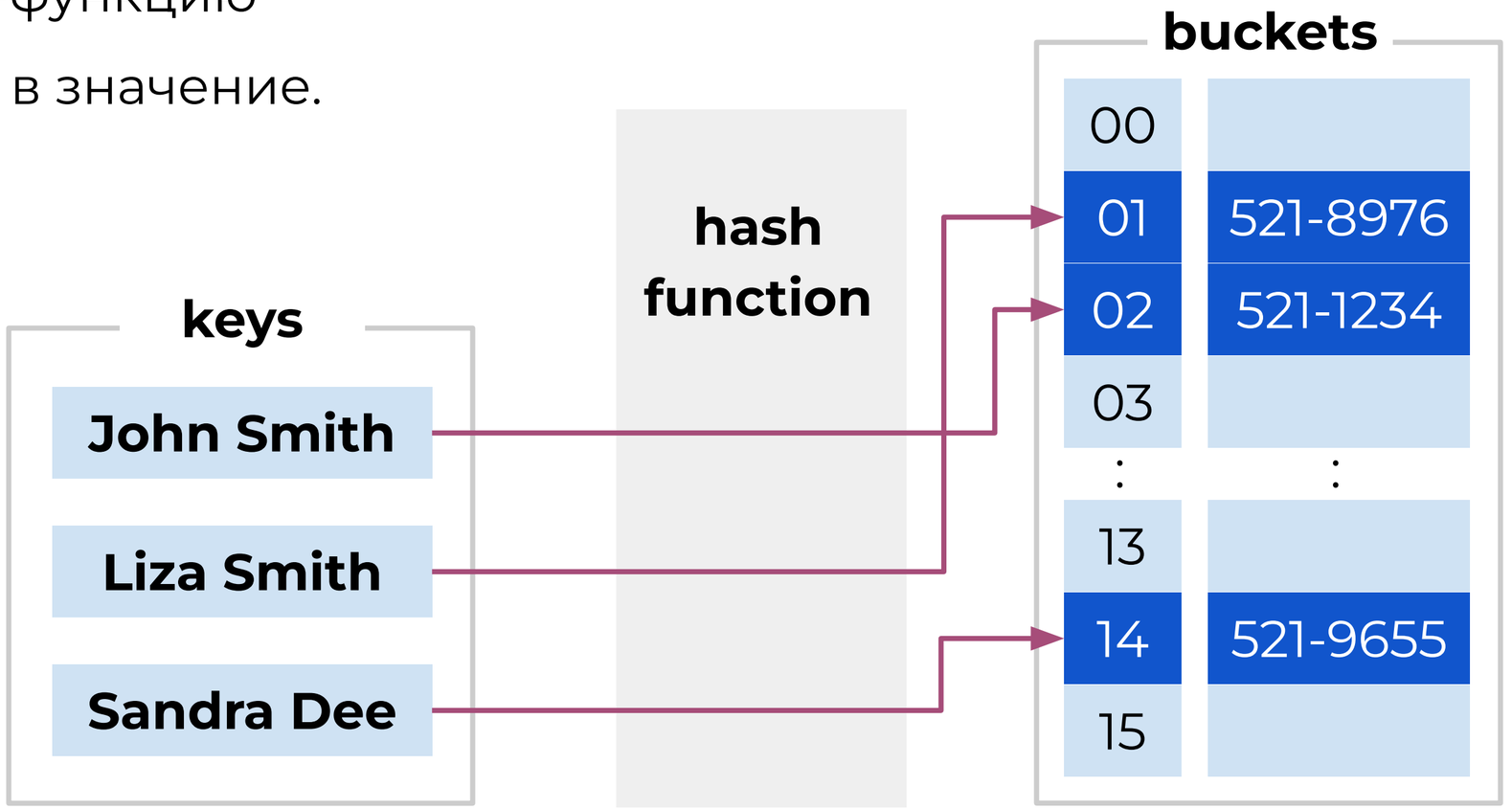


Бинарные деревья – деревья, в которых у каждого узла может быть не больше 2х потомков: левый и правый.

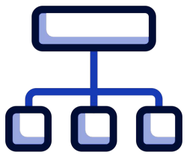


Хеш-таблица (Hash Table)

Хеш-таблица — структура данных, которая использует хэш-функцию для отображения ключа в значение.

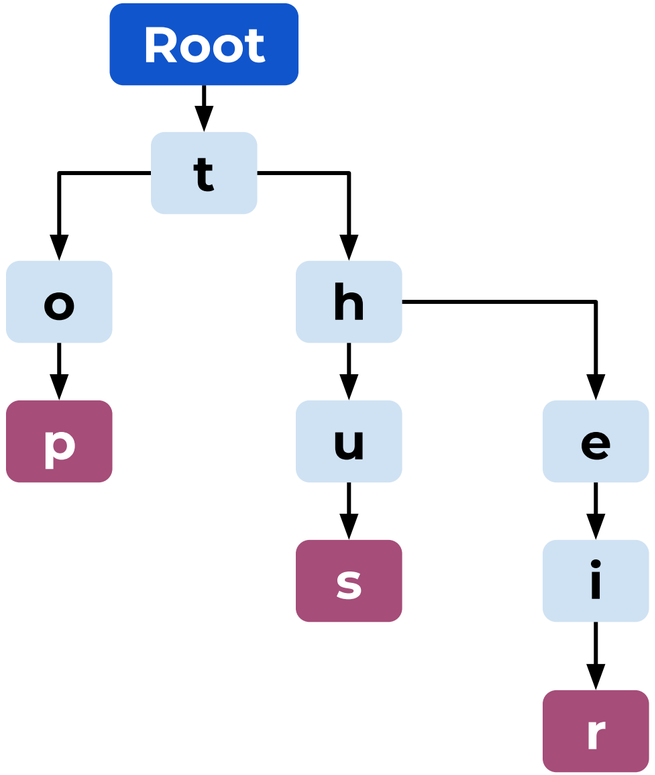


Префиксное дерево (Trie)



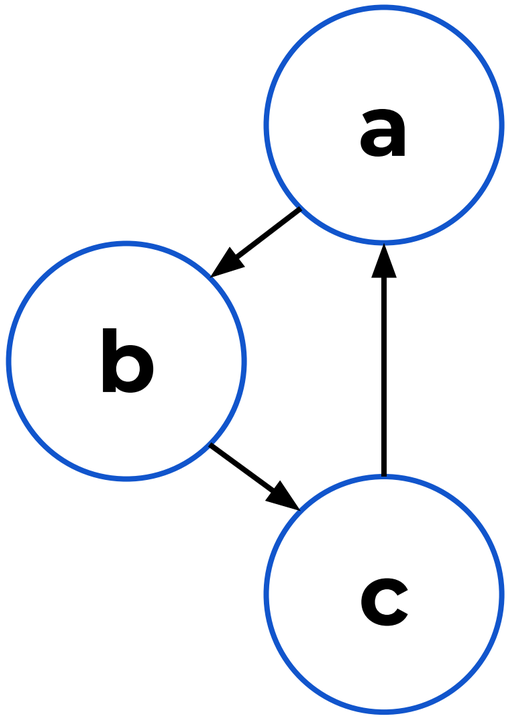
Префиксное дерево —

еще одно дерево для поиска строк.

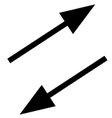


Графы (Graph)

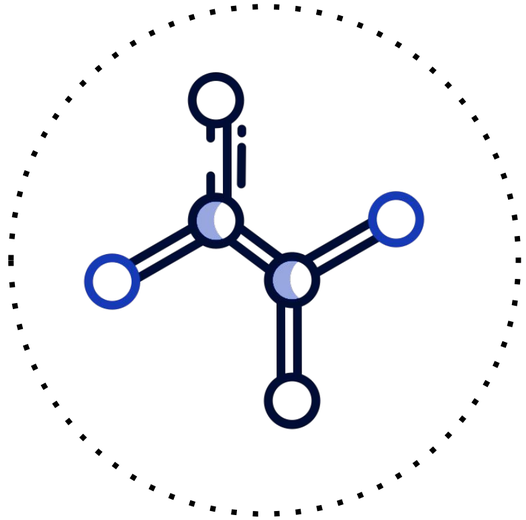
Графы — структура данных представляющая собой вершины, которые связаны ребрами.



вершина



ребро

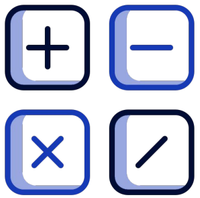


Резюме

Структуры данных с точки зрения
программиста-пользователя:

1. Список – структура данных для хранения однотипных данных.
2. Стек и Очередь – структуры с различным порядком доступа LIFO / FIFO.
3. Словарь – используется для отображения ключей в значения.
4. Префиксное дерево – используется для поиска ключа или части ключа.
5. Графы используются в задачах поиска путей, навигации и доставки товаров.

Резюме



Структуры данных с точки зрения
программиста-математика:

1. Массивы – это базовая структура данных для хранения однотипных объектов и построения других структур данных.
2. Связный список – структура данных с быстрой вставкой и удалением в произвольные места (только при условии, что у нас уже есть указатель на это самое место).
3. Хеш-Таблица – очень быстро работающий словарь.
4. Бинарное дерево поиска часто используется как словарь, который можно обойти в порядке возрастания ключей.

СПАСИБО ЗА ВНИМАНИЕ

ostinru