PRODUCT
STAR.RU

Структуры данных в Python

ИВАН ГРОМОВ

Старший разработчик в одной из FAANG-компаний

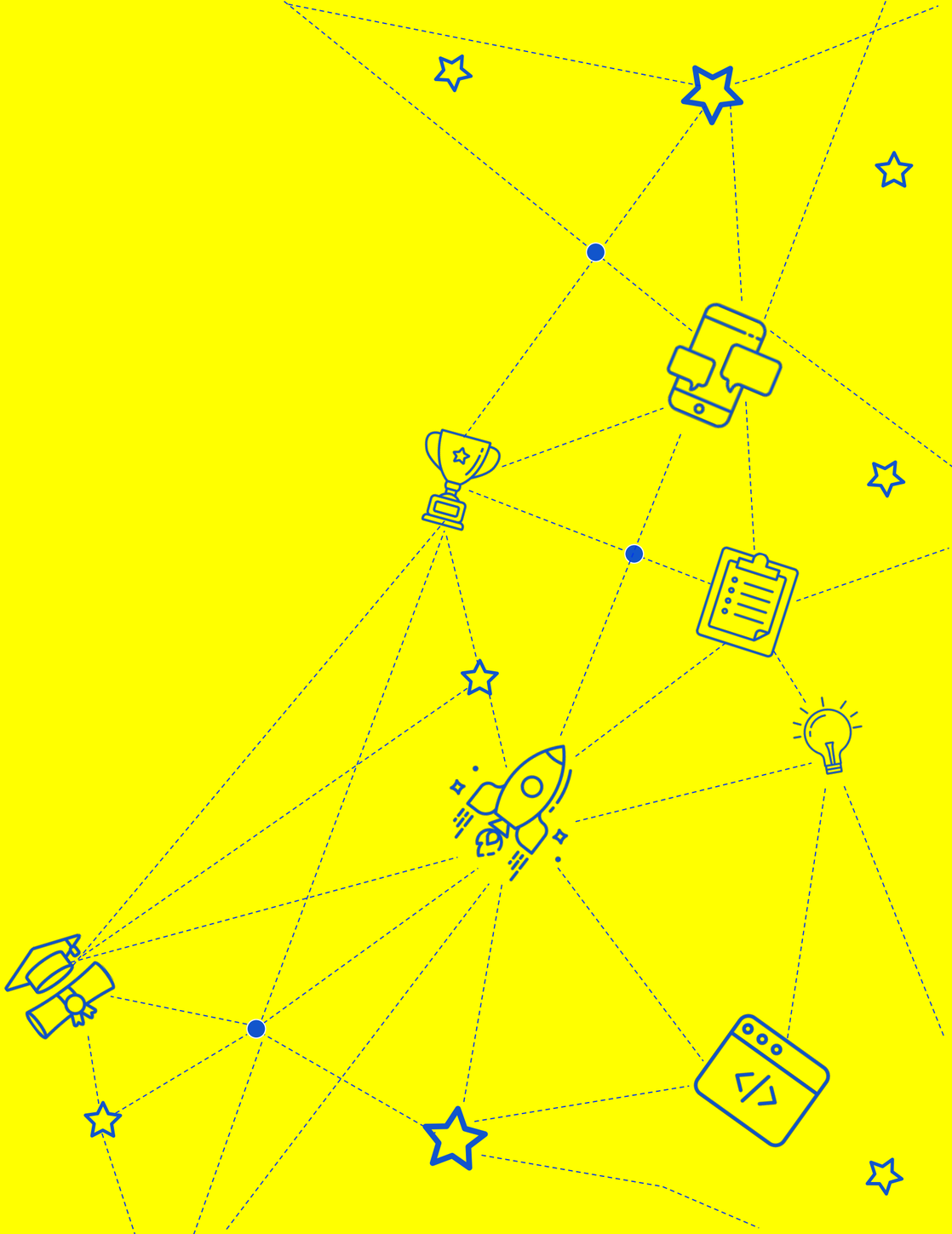
ex-Yandex, Cian, Lamoda

- Развиваю операционную систему для шлемов виртуальной реальности
- Делал Алису и Яндекс.Диалоги
- Занимался аналитикой и ML в Cian

Содержание урока

- Линейные структуры данных
- Хеш-таблицы
- Бинарный поиск

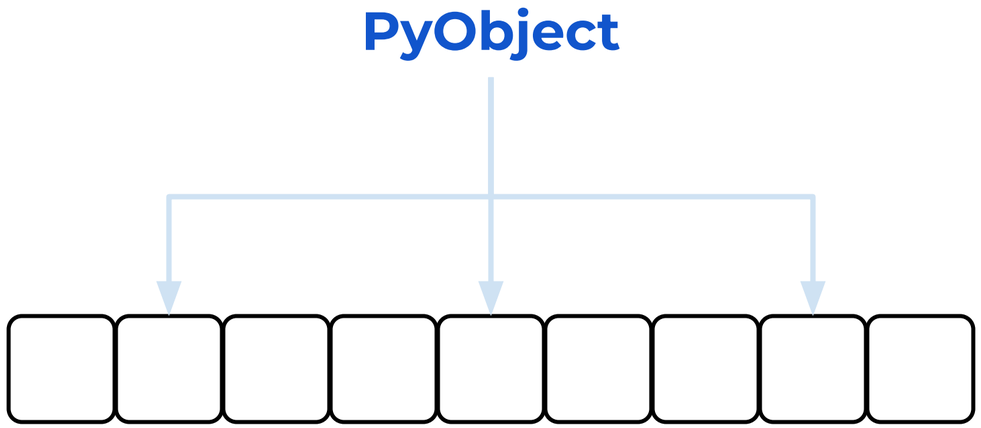
Массивы, стеки и очереди



Как устроен массив?

Массив – это набор объектов, идентифицируемых по номеру (индексу)

Массив – это не связный список, хоть и называется list

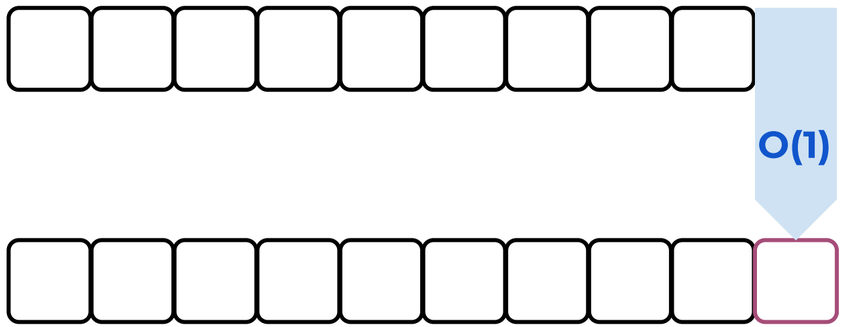


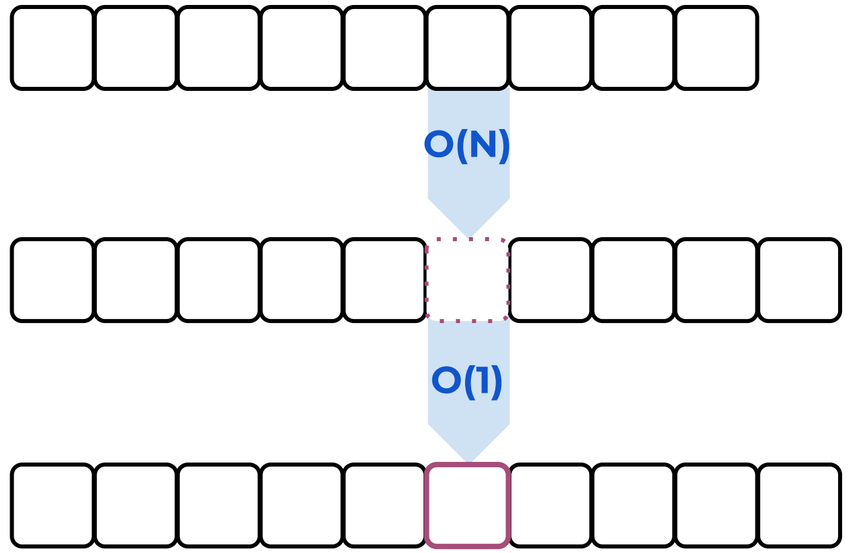
Вставка элементов

list.append(e) добавляет элемент **e** в конец массива

list.insert(i, e) вставляет элемент **e** на позицию **i**

Вставка в конец и вставка в середину имеют разную алгоритмическую сложность



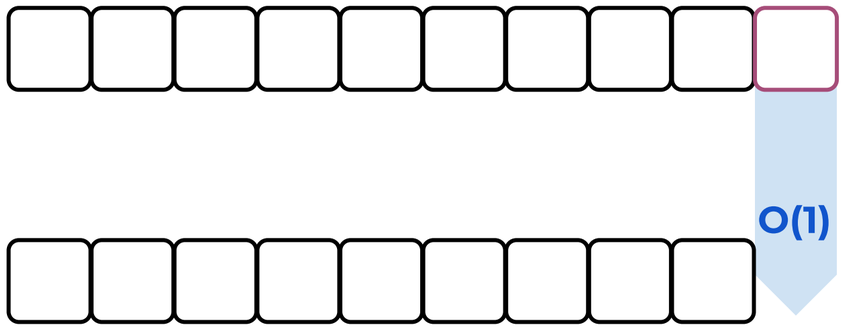


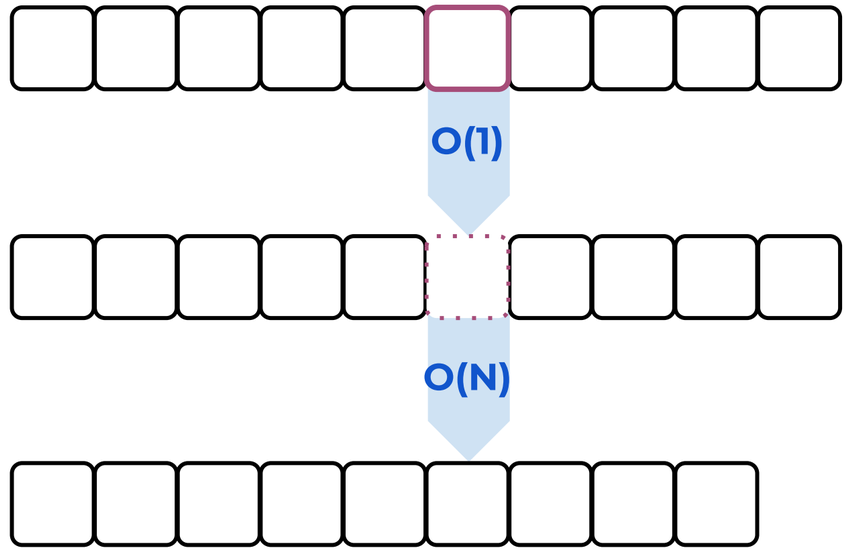
Удаление элементов

list.pop() удаляет последний элемент массива и возвращает его значение

list.remove(e) удаляет элемент со значением **e**

del list[i] удаляет элемент с индексом **i**



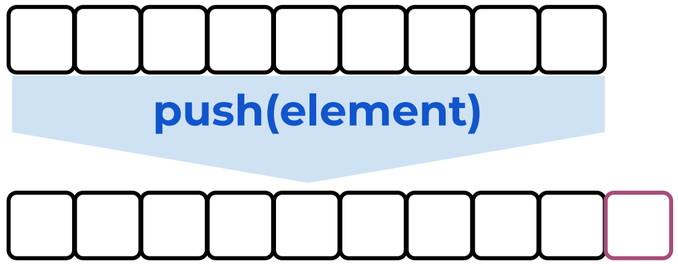


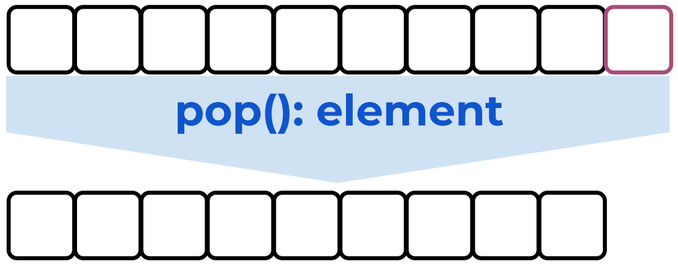
Стек

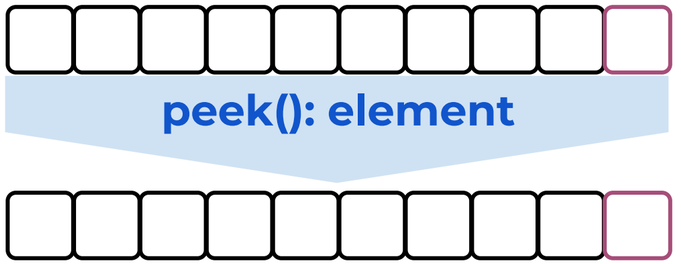
Операции (LIFO, last in first out):

- ★ Добавить в конец **(push)**
- ★ Удалить последний элемент **(pop)**
- ★ Вернуть последний элемент, не удаляя его **(peek)**

Все операции имеют сложность **O(1)**



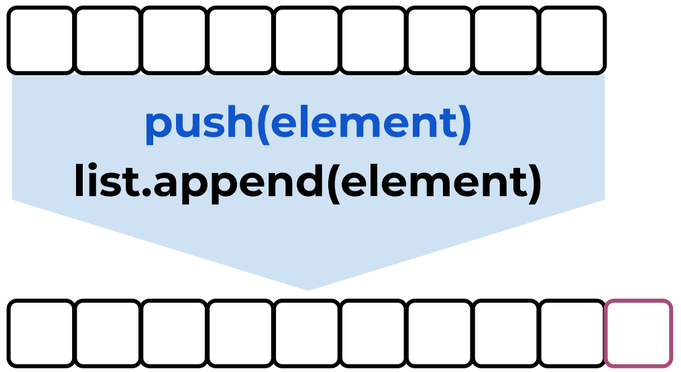


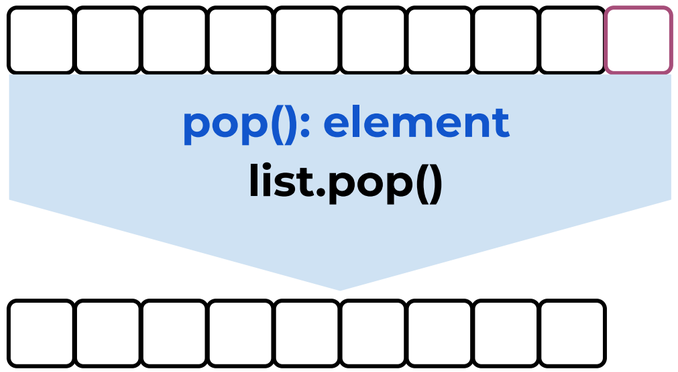


Стек

Операции:

- ★ Добавить в конец **(push)**
- ★ Удалить последний элемент **(pop)**
- ★ Вернуть последний элемент, не удаляя его **(peek)**





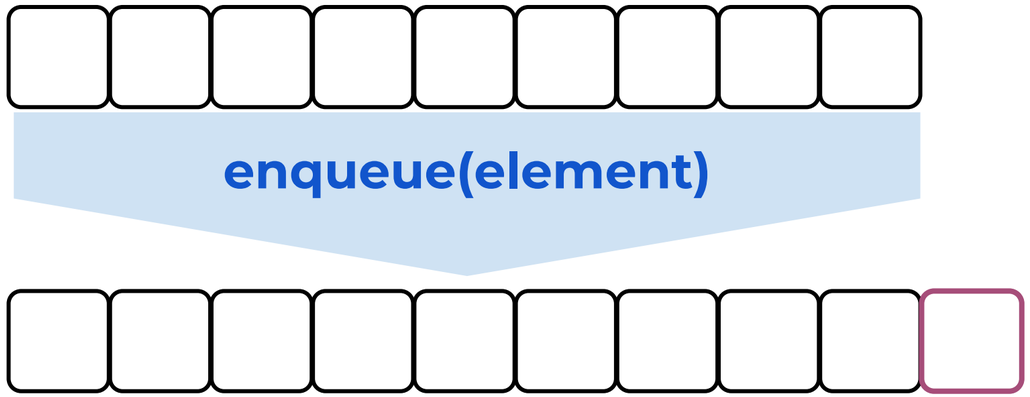
Очередь

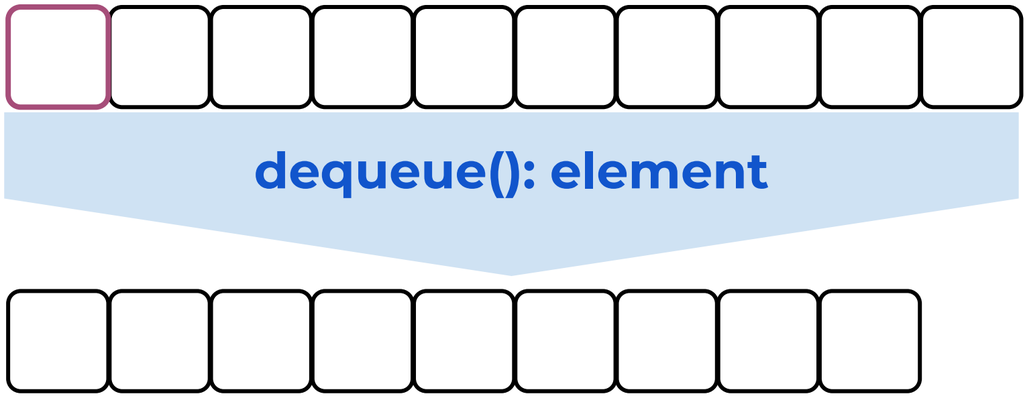
Операции (FIFO, first in first out):

★ Добавить в конец **(enqueue)**

★ Удалить первый элемент **(dequeue)**

Все операции имеют сложность **O(1)**





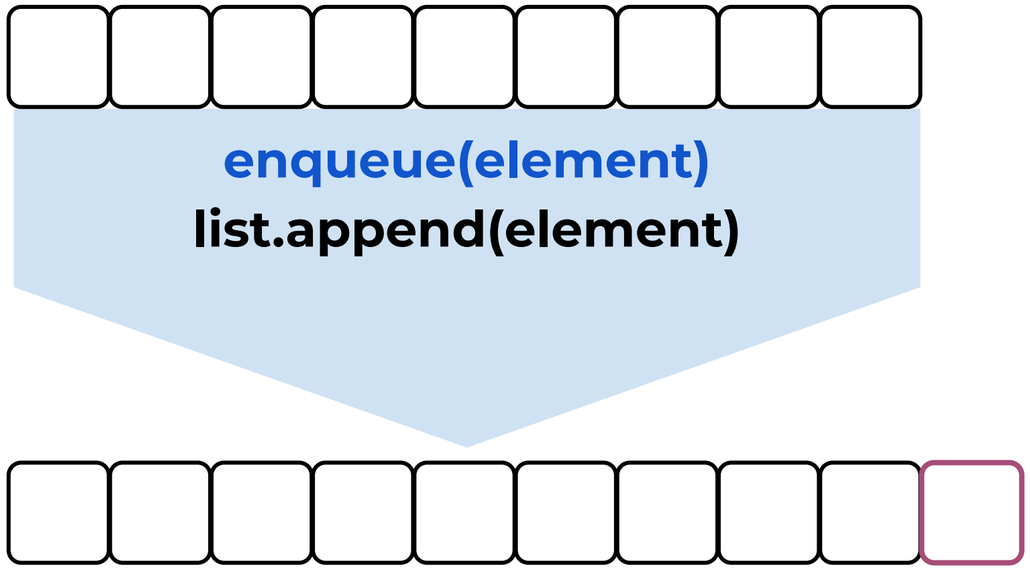
Очередь

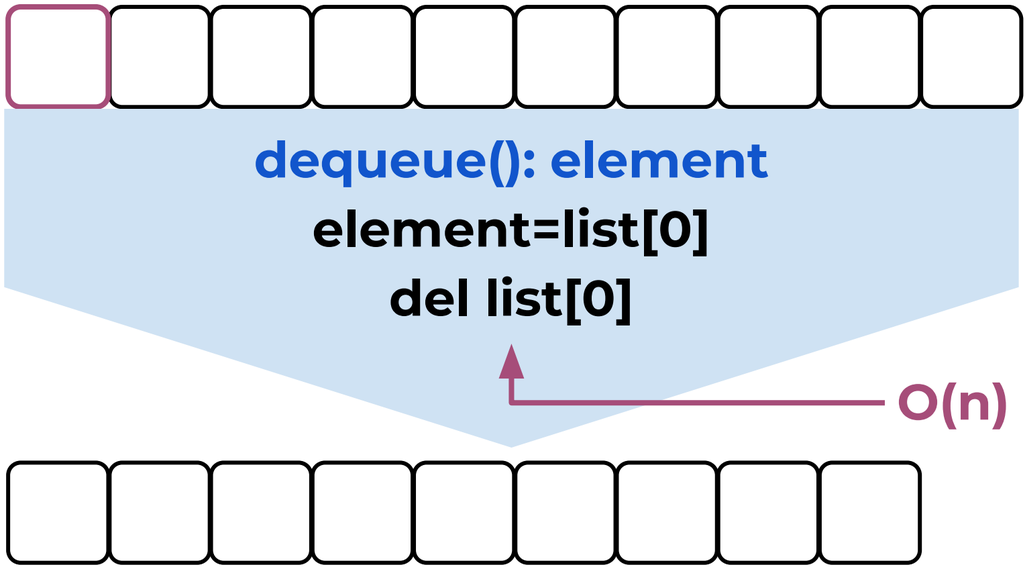
Операции (LIFO, first in first out):

★ Добавить в конец **(enqueue)**

★ Удалить первый элемент **(dequeue)**

Все операции имеют сложность **O(1)**

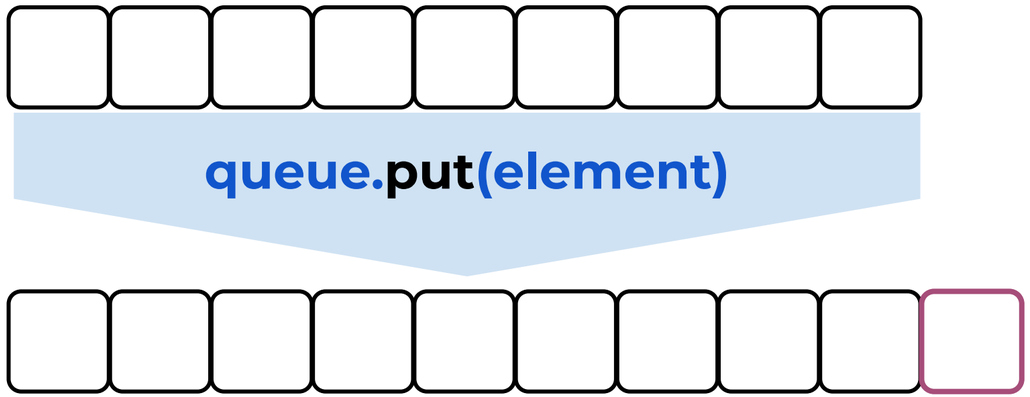


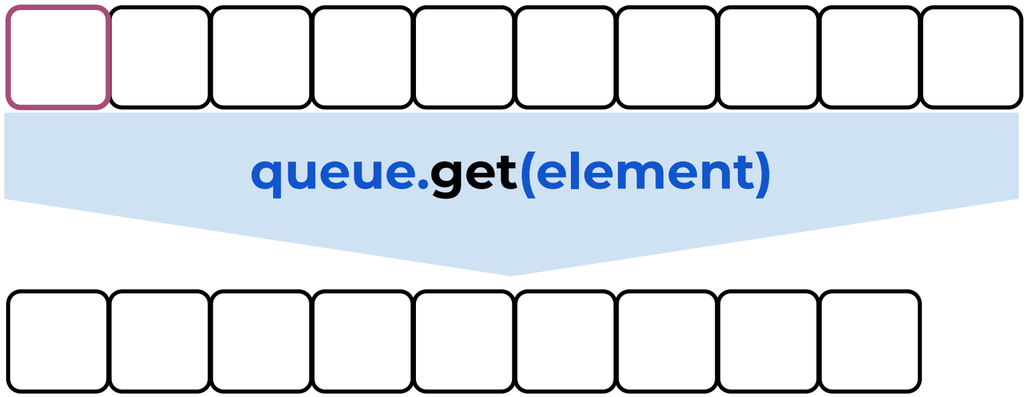


Очередь

В стандартной библиотеке Python есть реализация очереди в модуле **queue**

```
from queue import Queue
queue = Queue()
queue.put(1)
```

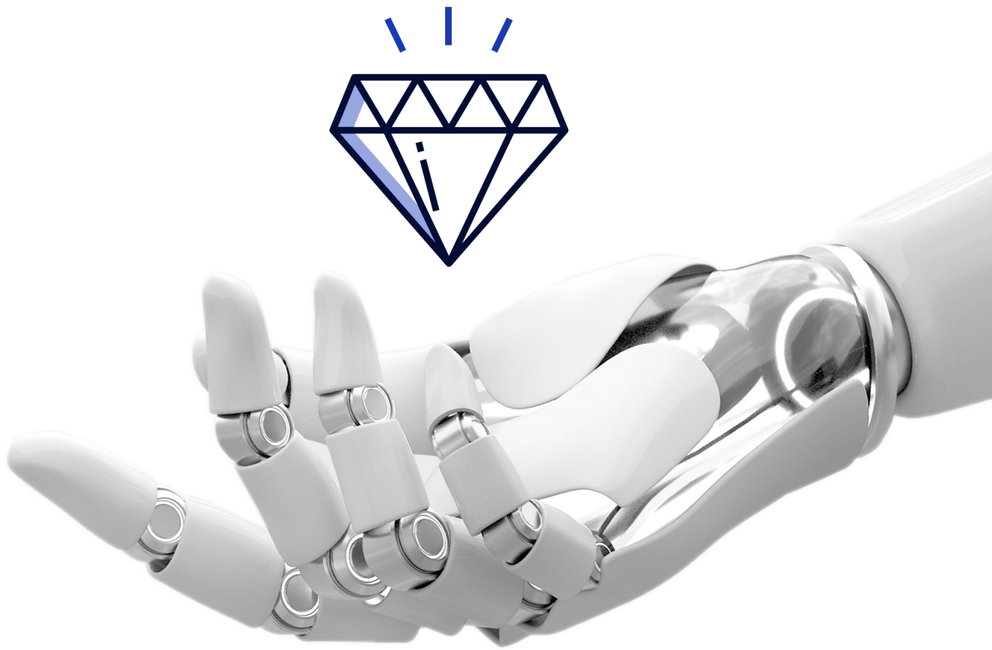




Deque – очередь + стек

from collections import deque

- my_deque = deque()
- my_deque.append(1)
- my_deque.pop()
- my_deque.appendleft(1)
- my_deque.popleft()



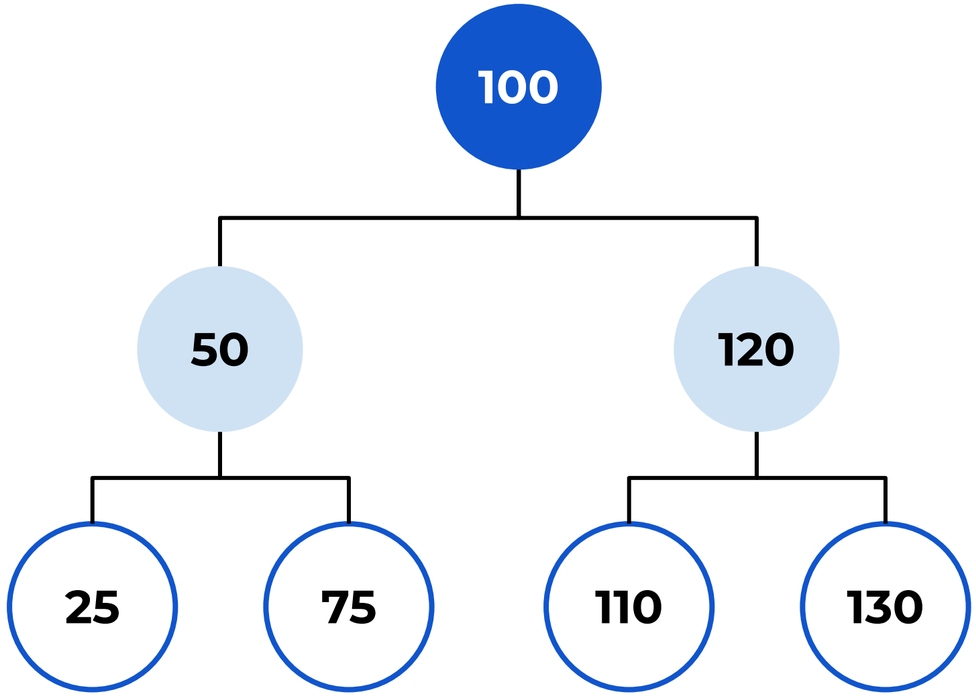
Куча
(приоритетная очередь)

Куча – это очередь, всегда отсортированная по возрастанию (min-heap) или убыванию (max-heap) элементов.

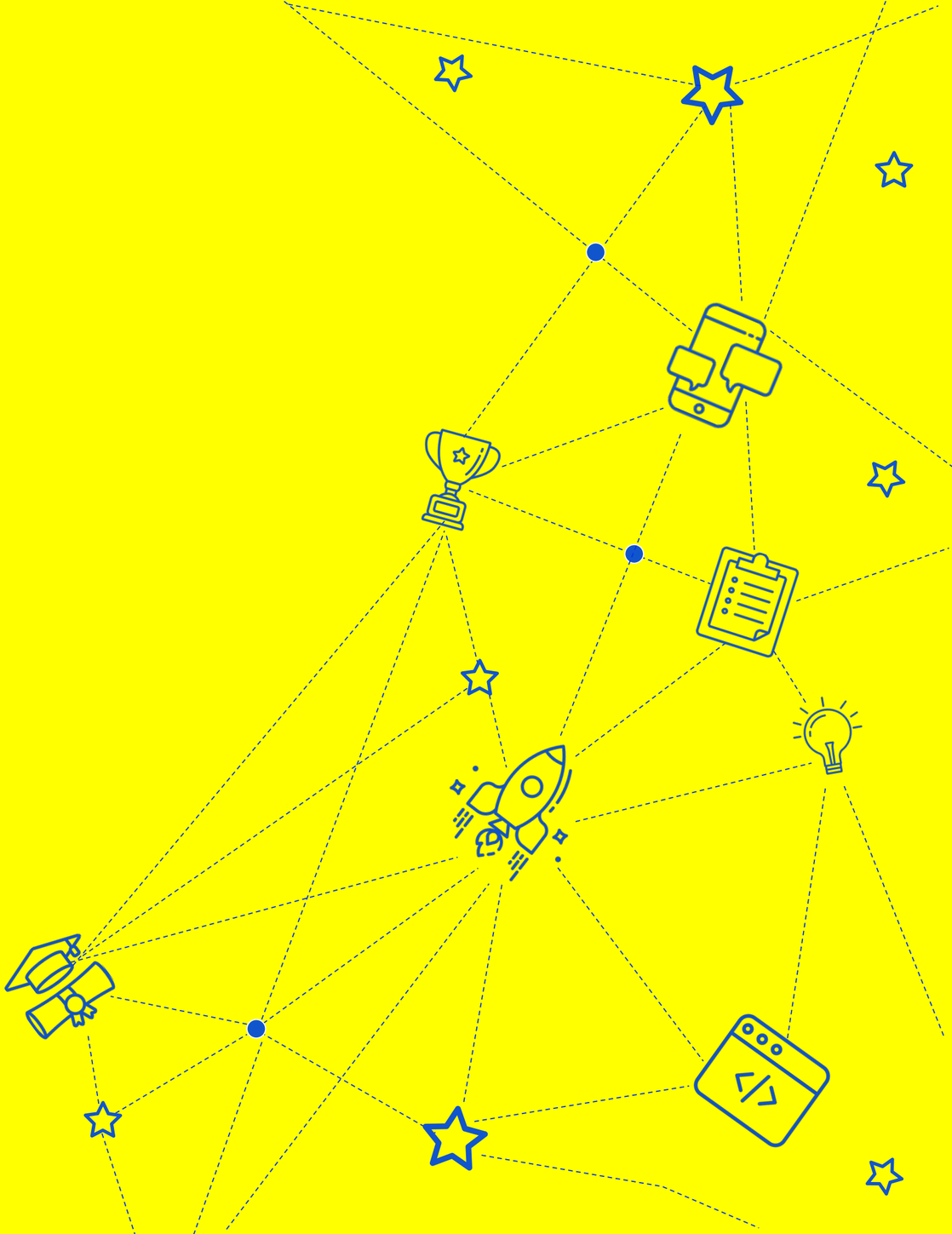
В стандартной библиотеке куча доступна в модуле heapq

Основные операции:

- ★ **heappush() –** добавить элемент
- ★ **heappop() –** забрать элемент с минимальным значением
- ★ **heappushpop() –** добавить элемент и потом забрать элемент с минимальным значением



Хеш-таблицы



Словарь – это хеш-таблица

Словарь – это структура данных, реализующая доступ по произвольному ключу (key-value структура или ассоциативный массив).

Основные операции работают за O(1)*:

вставка

удаление

поиск элемента

Что такое хеш?

Хеш-функция превращает любой объект в число.

Коллизией называют два объекта **X и Y**, для которых **hash(X) == hash(Y)**

«Хорошая» хеш-функция обладает следующими свойствами:

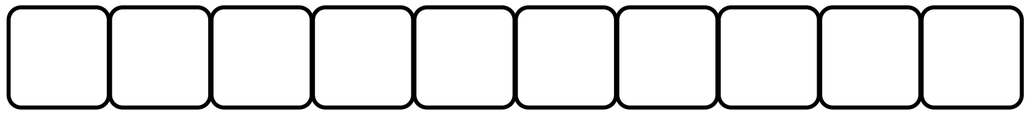
- ★ быстрое вычисление
- ★ минимальное количество коллизий

Как работают словари в Python?

Словарь – это хеш-таблица с открытой адресацией.

my_dict[‘hello’] = ‘world’

hash(hello) % 10 = 3



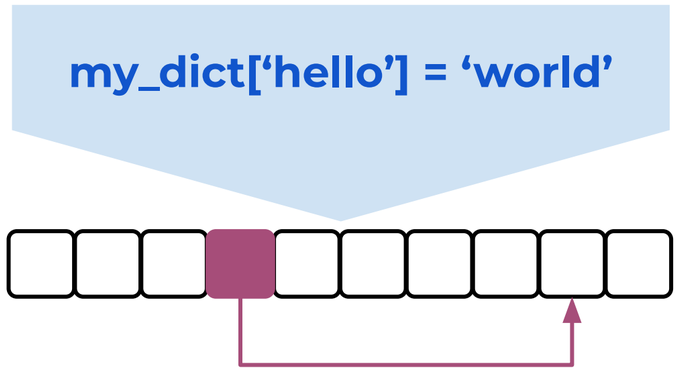
key = ‘hello’

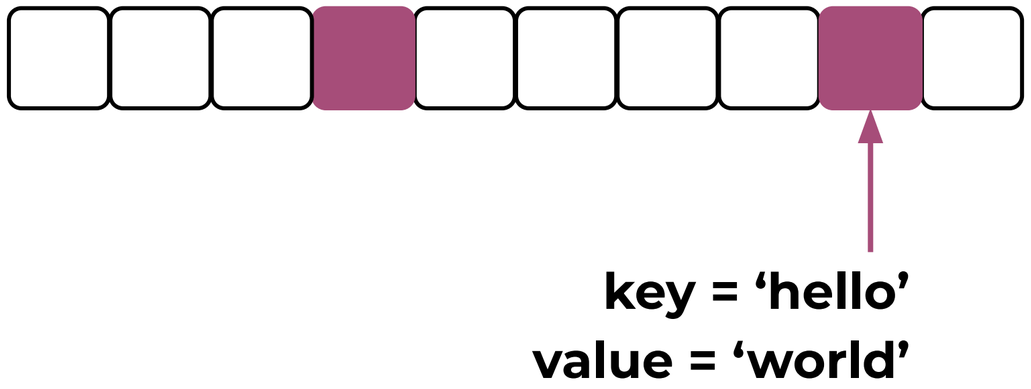
value = ‘world’

Разрешение коллизий

Если элемент массива уже занят (произошла коллизия),
то словарь сохранит его куда-то ещё:

- ★ в следующий элемент **(linear probing)**
- ★ в элемент с номером **i^2 (quadratic probing)**
- ★ в псевдослучайный элемент **(pseudorandom probing)**





Хеширование экземпляров классов

```
class HashExample:
    def __init__(self, value):
        self.value = value

    def __hash__(self):
        return 42

    def __eq__(self, other):
        return (
            isinstance(other, HashExample) and
            self.value == other.value
        )
```

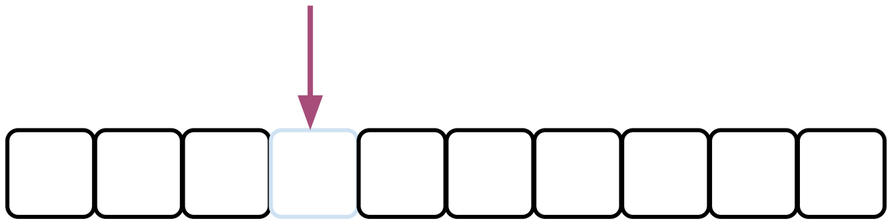
Нужно объявить оба метода

__hash__ без __eq__

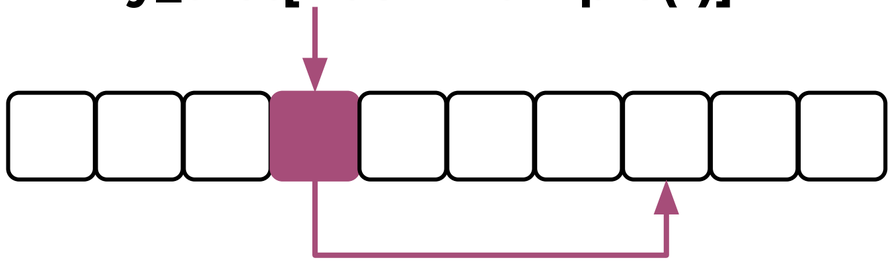
Если мы не реализуем метод **__eq__()**, Python будет использовать стандартную реализацию:

```
>>> HashExample(1) == HashExample(1)
False
```

my_dict[HashExample(1)] = 1



my_dict[HashExample(1)] = 1



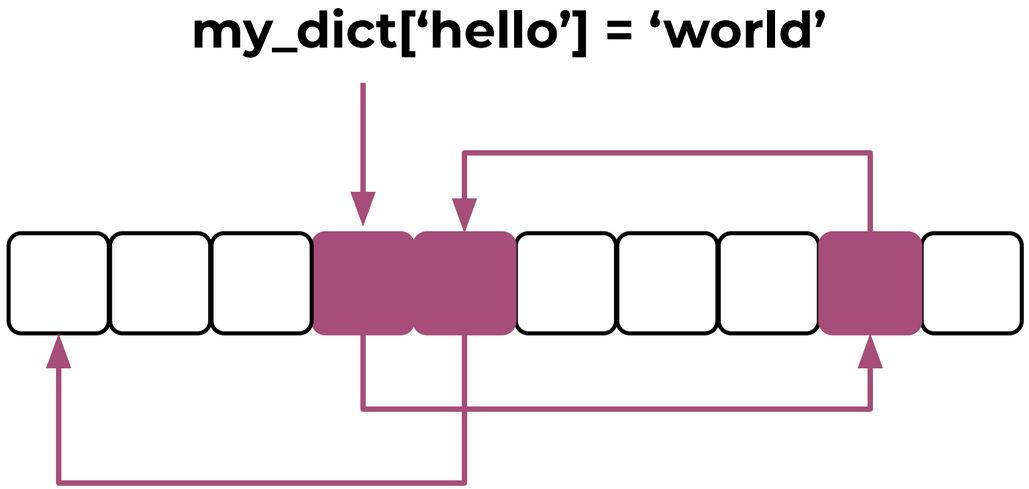
__eq__ без __hash__

Если мы не реализуем метод **__hash__()**, Python выбросит ошибку:

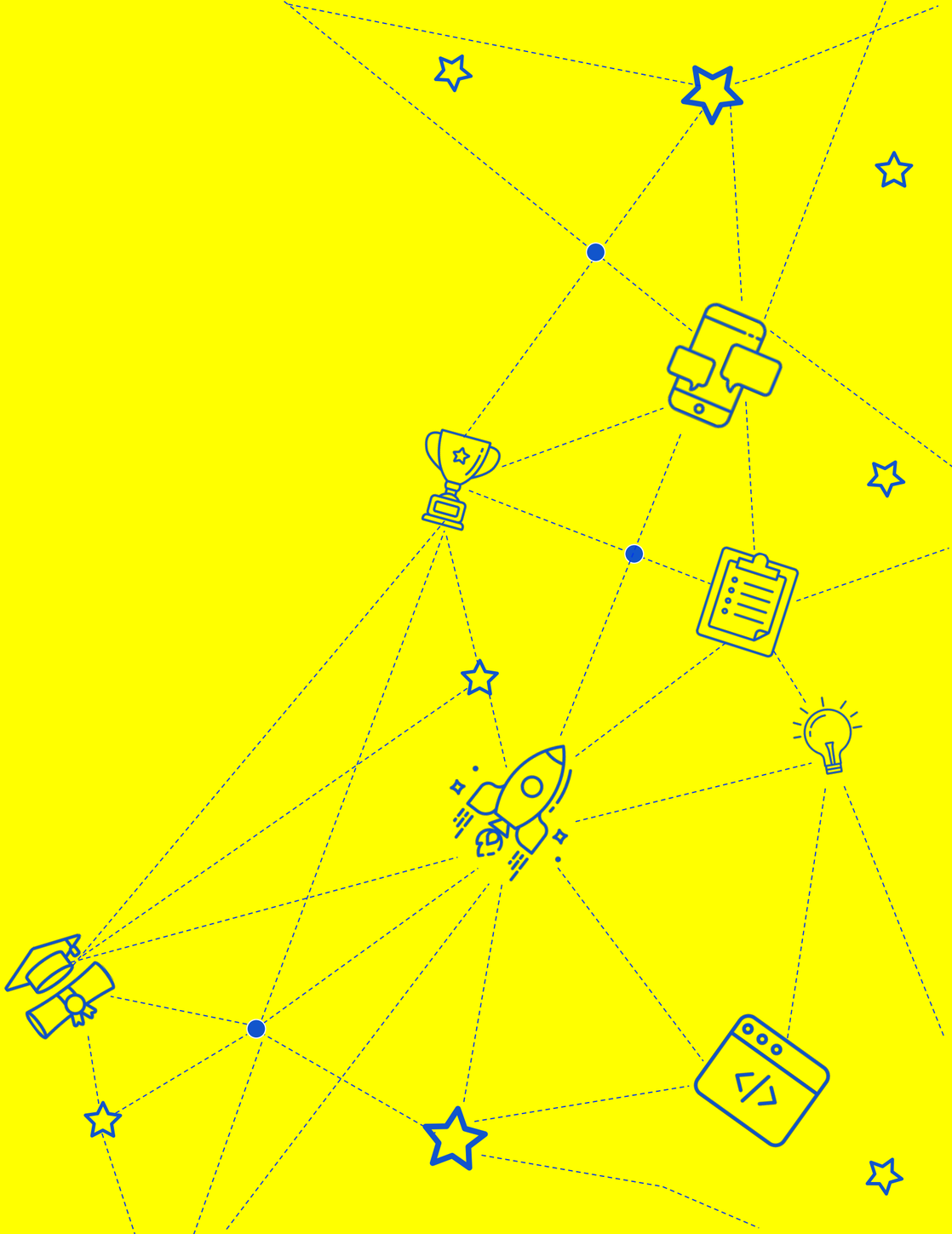
```
>>> h1 = HashExample(1)
>>> hash(h1)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: unhashable type: 'HashExample'
```

Плохая хеш-функция

Плохая хеш-функция будет генерировать много коллизий, из-за которых **O(1)** превратится в **O(N)**



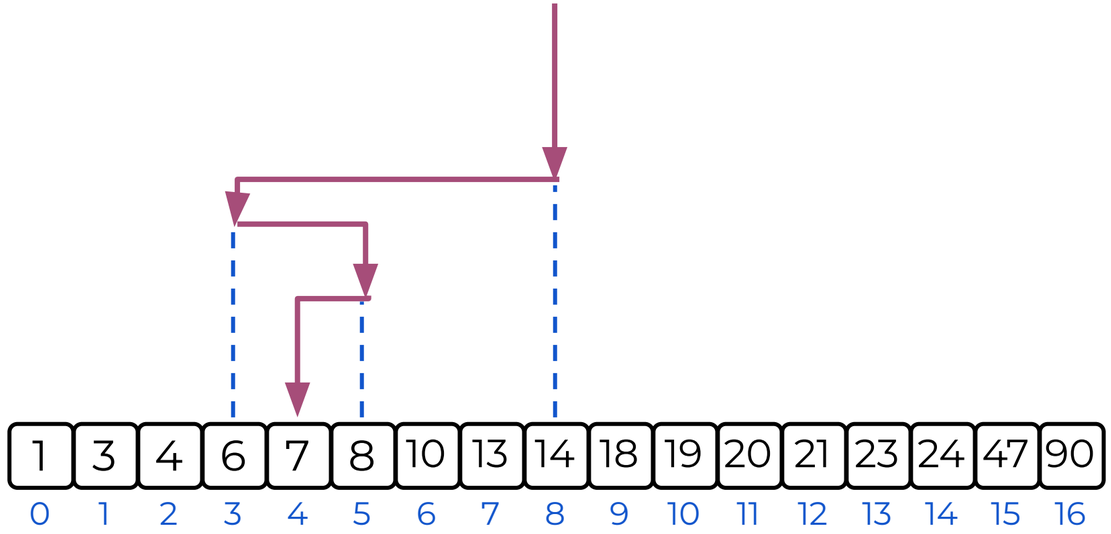
Бинарный поиск



Алгоритм бинарного поиска

Используется для поиска элемента в **отсортированном** массиве, сложность **$O(\log_2(N))$**

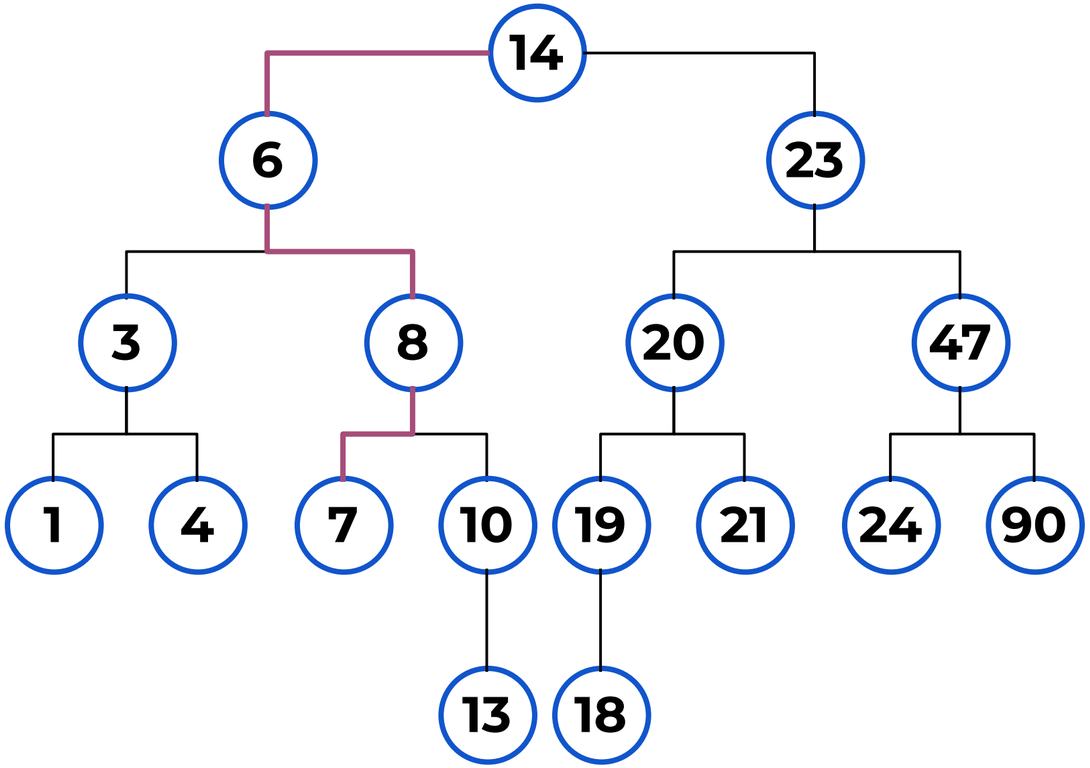
1. Выбрать центральный элемент
2. Если искомый элемент меньше центрального, то повторить процедуру в левом подмассиве, иначе – в правом
3. Если размер подмассива равен 1, то искомый элемент найден или не существует



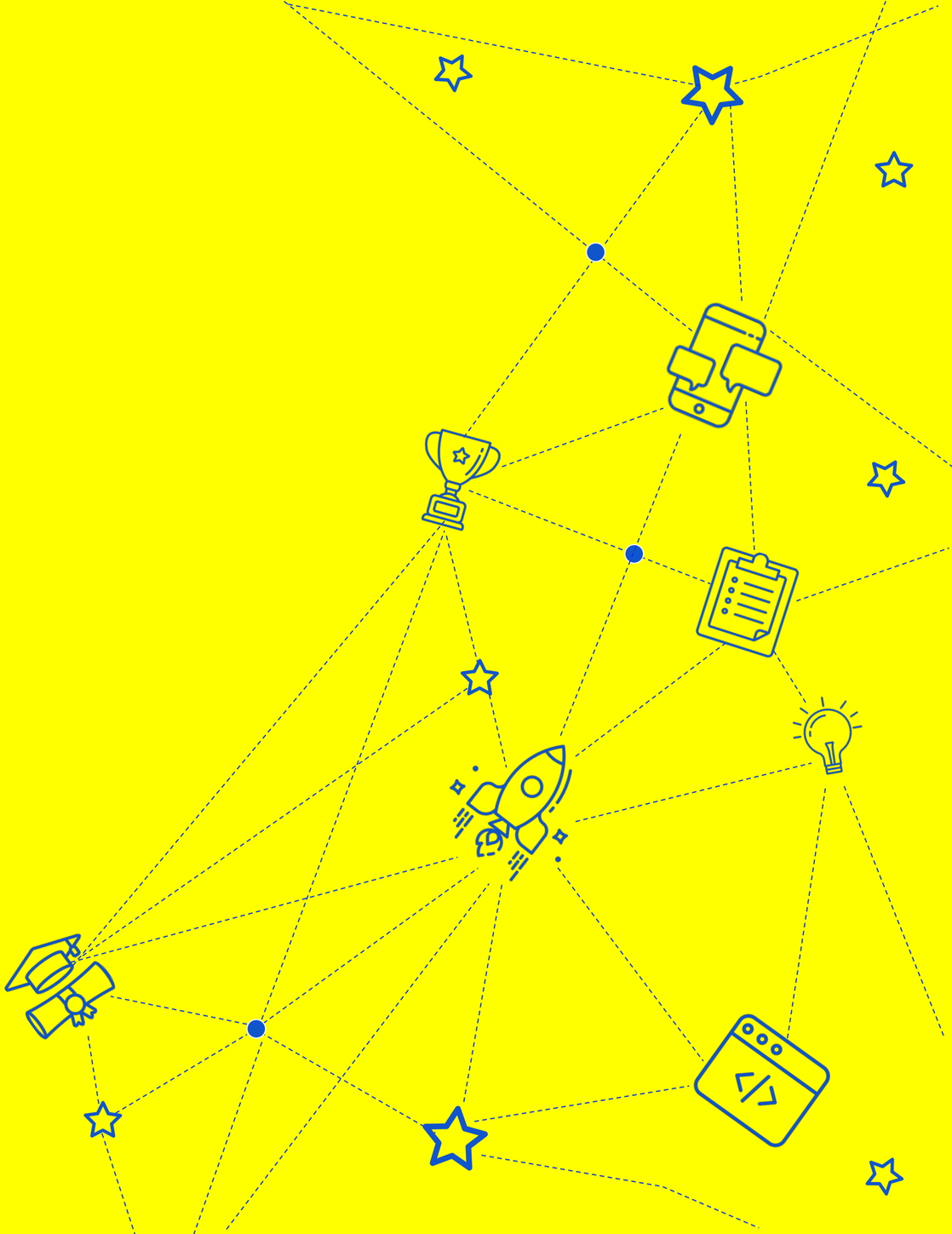
Дерево бинарного поиска

Это двоичное дерево, для которого выполняются следующие условия:

1. Левое и правое поддеревья – тоже деревья бинарного поиска
2. У всех узлов левого поддерева узла X значения элементов меньше или равны X
3. У всех узлов правого поддерева значения элементов больше X



Домашнее задание



Домашнее задание 1

1. Напишите функцию, которая принимает на вход массив **nums** и число **target** и возвращает массив из индексов двух элементов **nums**, сумма которых равна **target**

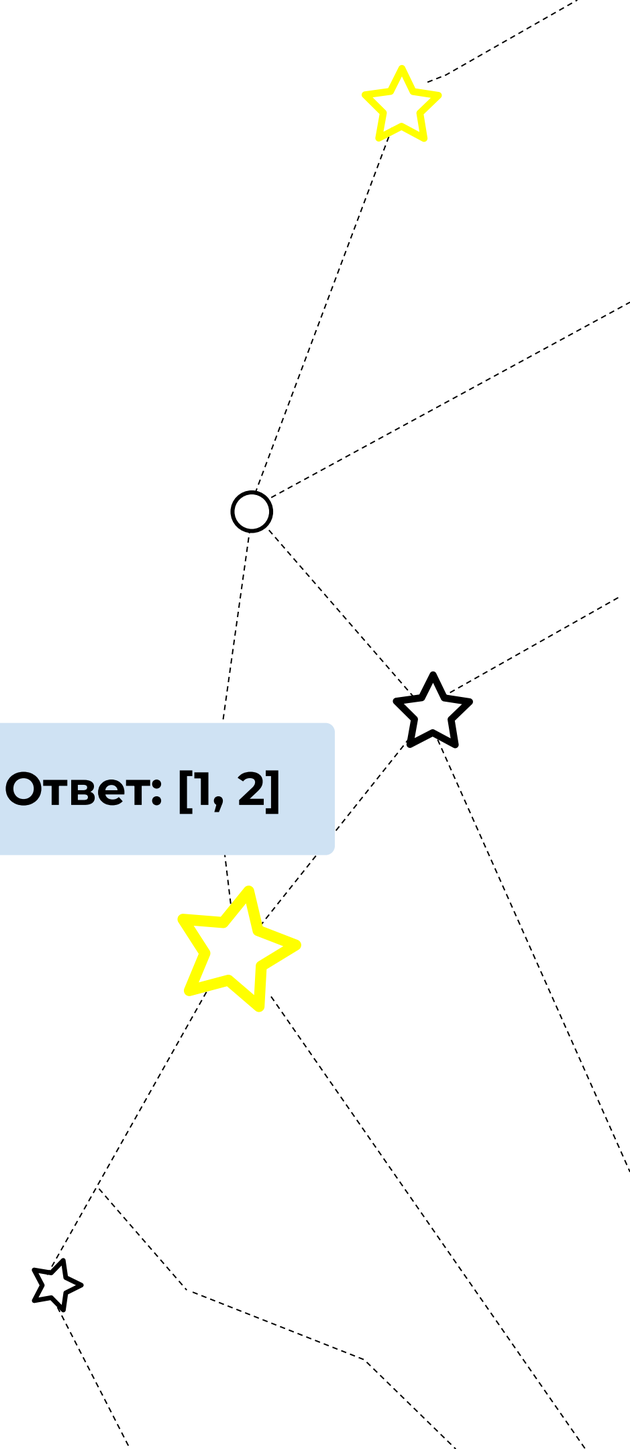
Примеры:

nums = [2,7,11,15], target = 9; Ответ: [0, 1]

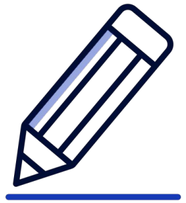
nums = [3,2,4], target = 6; Ответ: [1, 2]

2. Попробуйте решить эту задачу за один проход по массиву при помощи одной из структур данных, которые мы изучили.

3. Проверьте, что функция проходит все тесты, представленные в файле с задачей.



Домашнее задание 2



Панграмма – это фраза, содержащая все буквы алфавита, например «Съешь же ещё этих мягких французских булок да выпей чаю»

1. Напишите функцию, которая принимает на вход одну строку и проверяет, является ли она панграммой на русском языке.

2. Проверьте, что функция проходит все тесты, представленные в файле с задачей.

СПАСИБО ЗА ВНИМАНИЕ