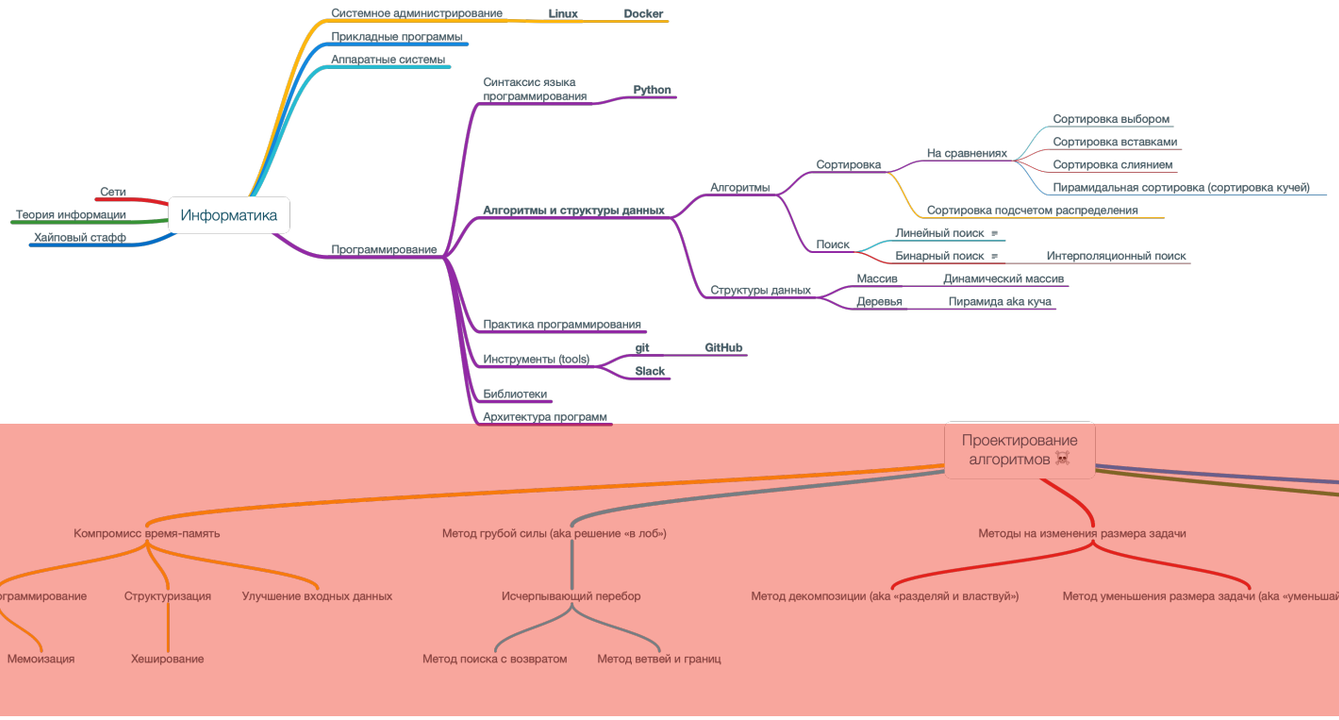
ИНФОРМАТИКА

```
~/polytech $ date
осень 2019
~/polytech $ name
константин кориков
~/polytech $ topic
атд с линейным порядком
```

ГДЕ МЫ?



Методы с прошлого занятия

О ЧЁМ ПОГОВОРИМ?

1. Еще раз об АТД
2. АТД с линейным порядком:
    A. Списки
    B. Строки
    C. Очереди
    D. Деки и стеки
3. ~~Реализация в Python~~

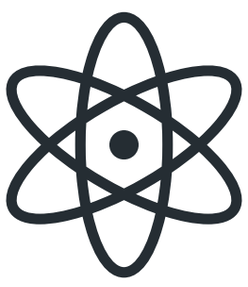
Еще раз про АТД

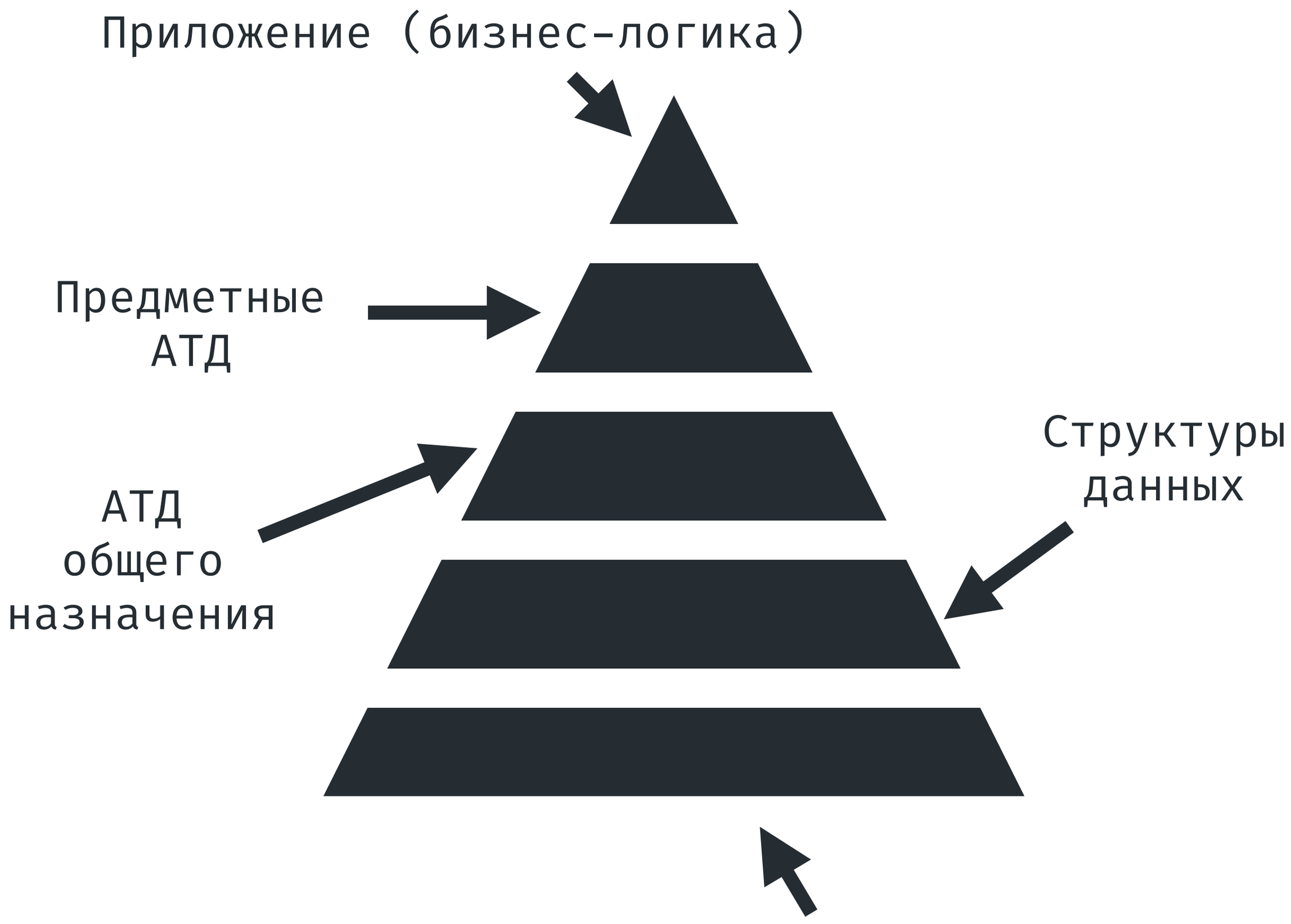
Jan Chipchase

Jan Chipchase

Понятный интерфейс

Скрытая реализация

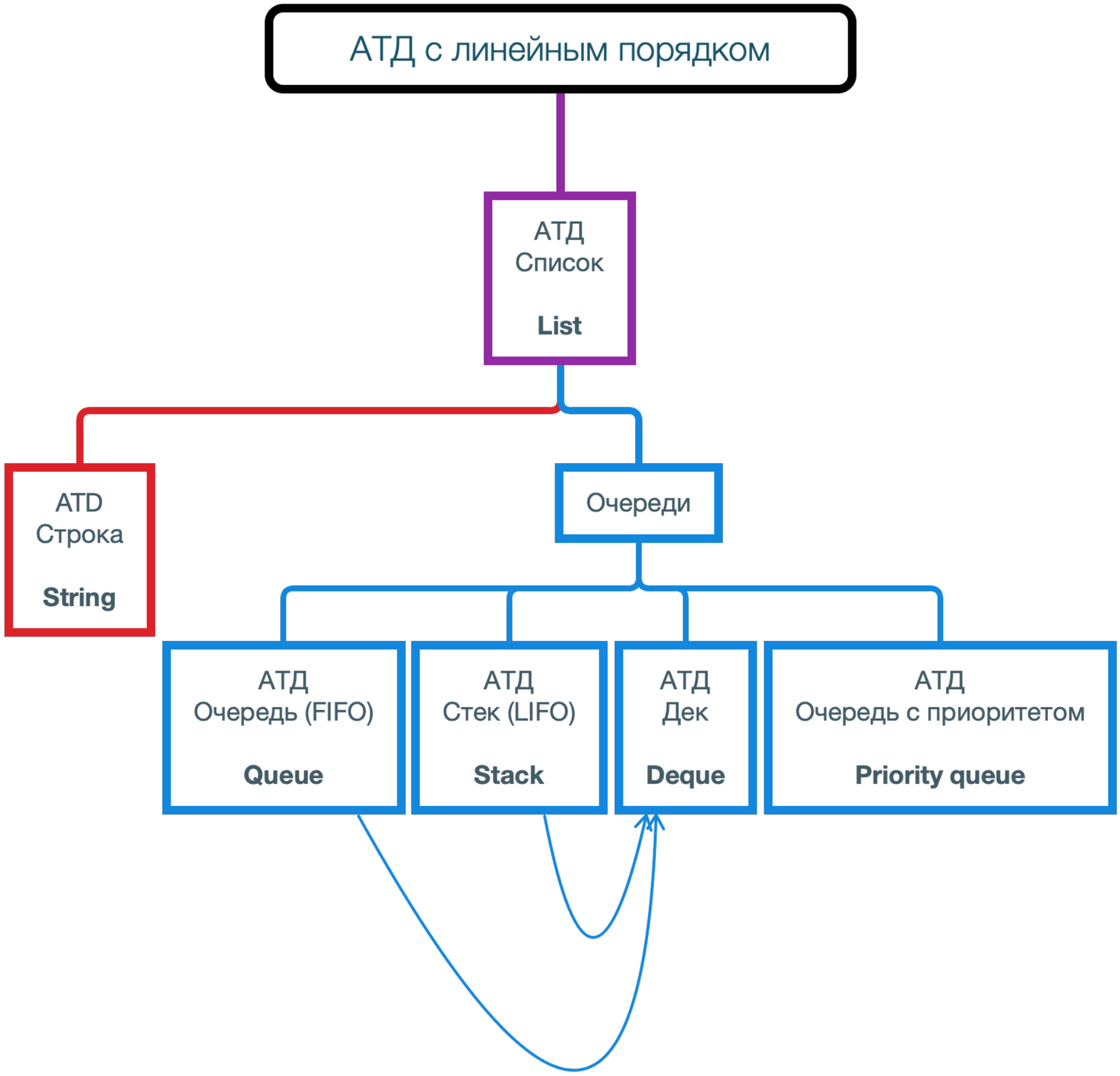



Приложение (бизнес-логика)
Предметные АТД
АТД общего назначения
Структуры данных
Примитивы языка (типы данных)

NB: АТД ≈ интерфейс

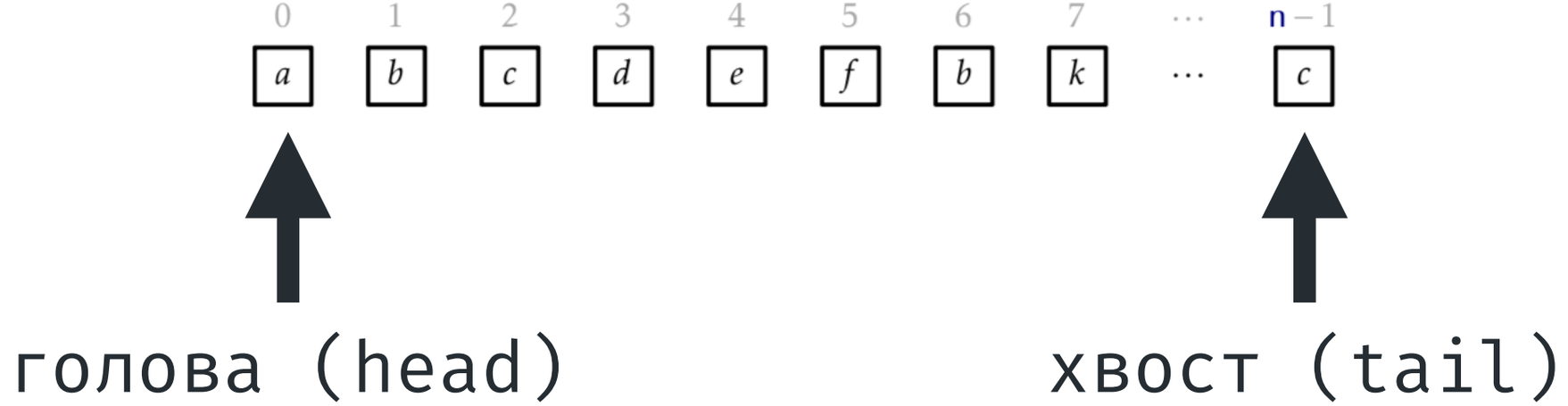
NB: Структура данных ≈ реализация

АТД с линейным порядком


АТД с линейным порядком
АТД
Список
List
ATD
Строка
String
Очереди
АТД
Очередь (FIFO)
Queue
АТД
Стек (LIFO)
Stack
АТД
Дек
Deque
АТД
Очередь с приоритетом
Priority queue

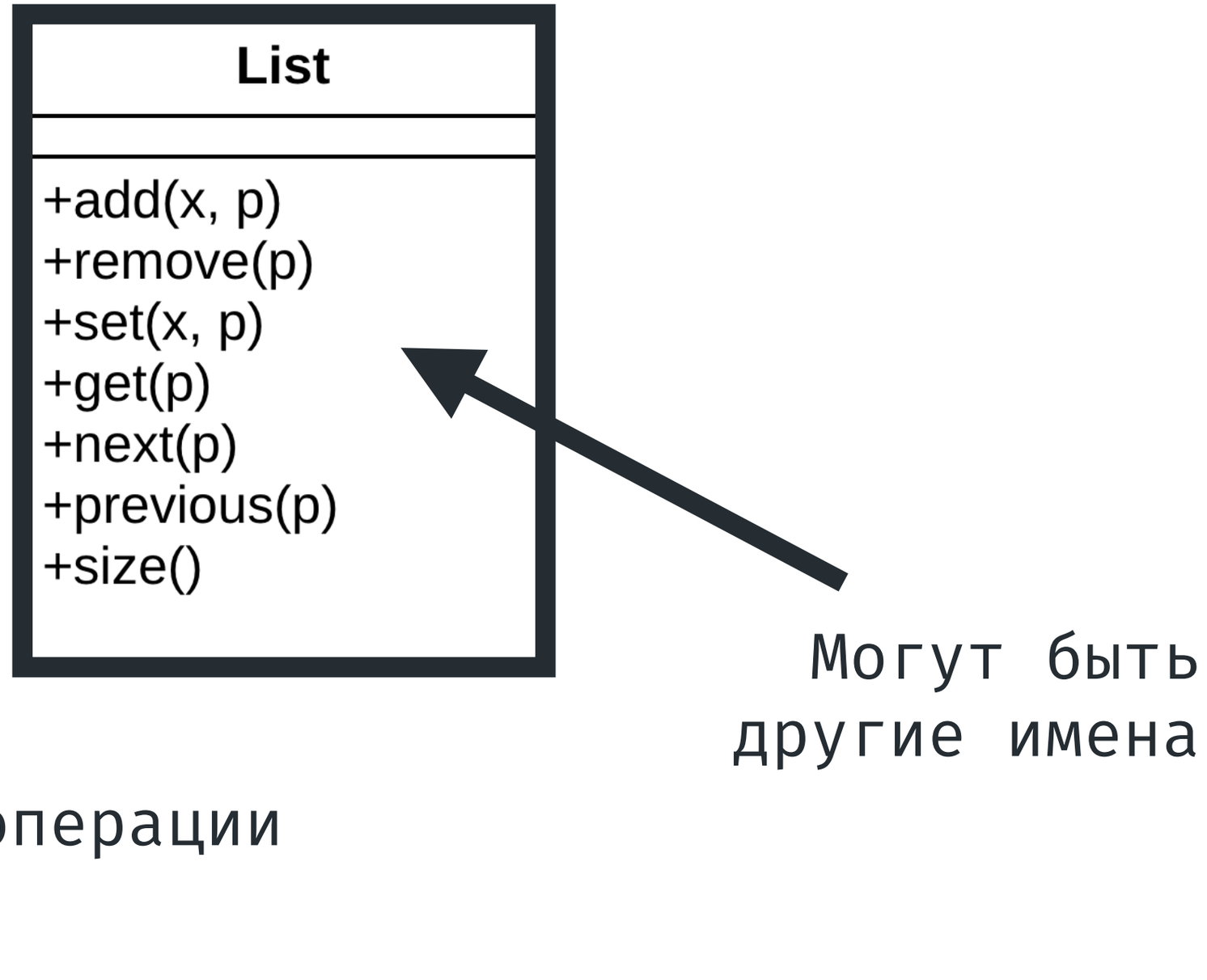
Список (list)

Строка (string)



Список (list)

Строка (string)

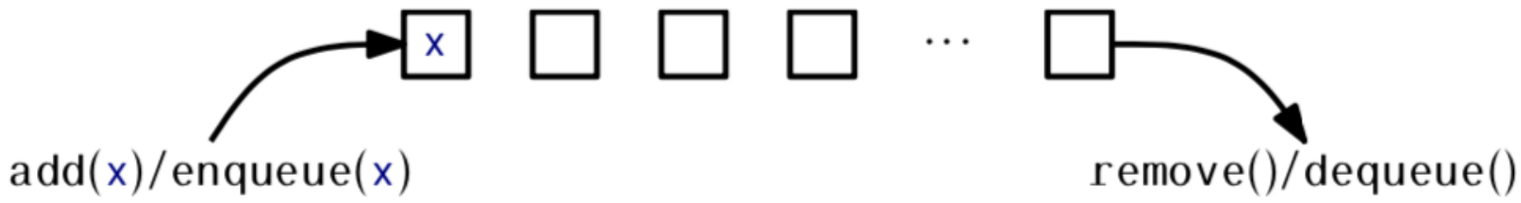


Могут быть
другие имена

Бывают и другие операции

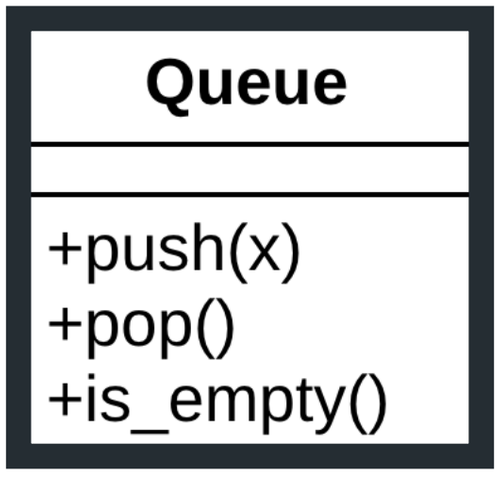
Очередь (queue)

разновидность очереди: FIFO



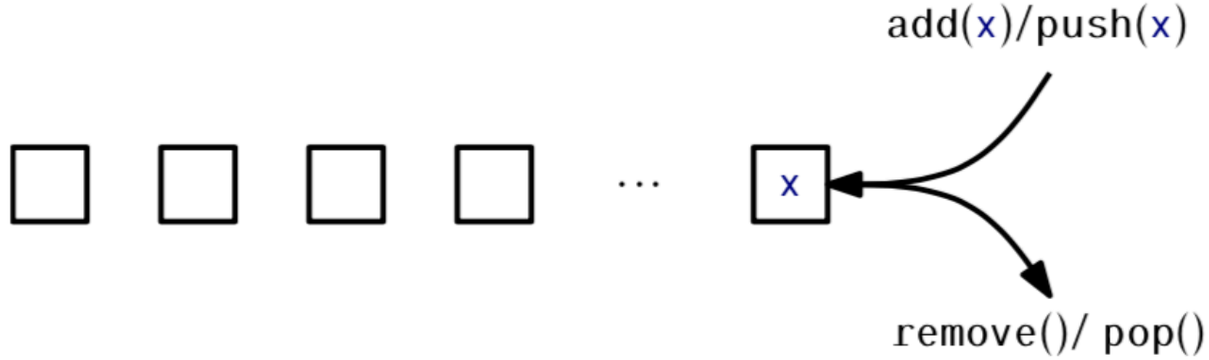
Очередь (queue)

разновидность очереди: FIFO



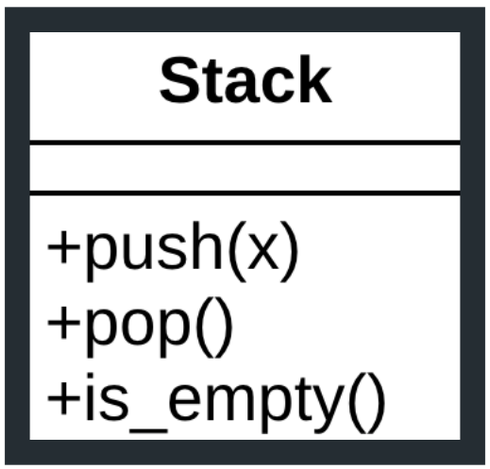
Стек (stack)

разновидность очереди: LIFO



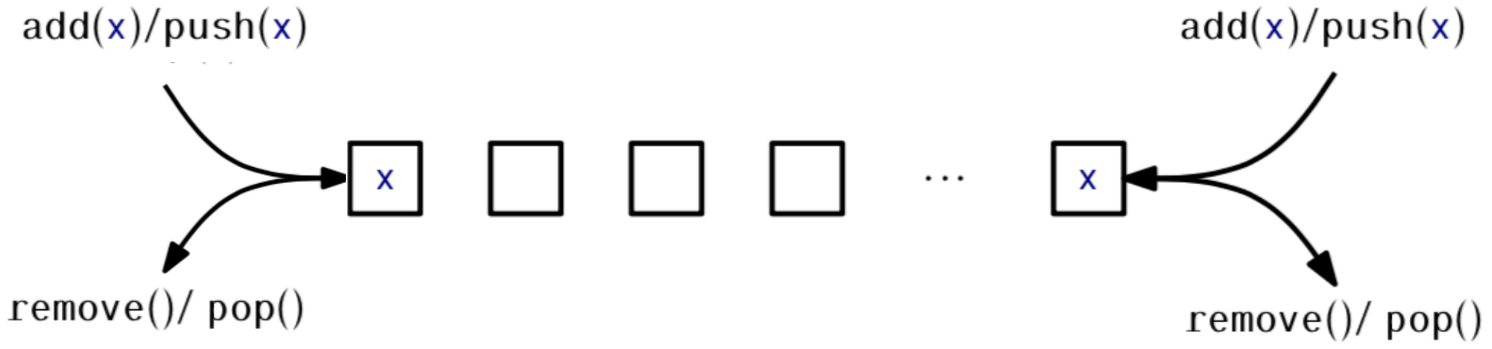
Стек (stack)

разновидность очереди: LIFO



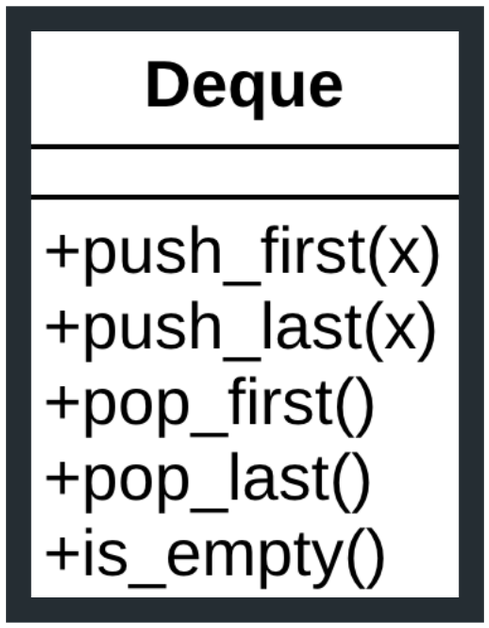
Дек (deque)

разновидность очереди: FIFO + LIFO



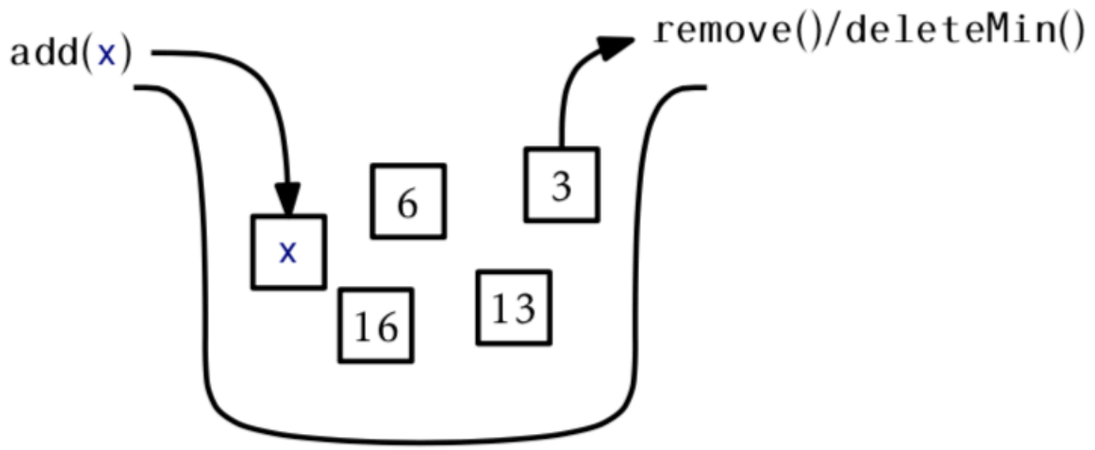
Дек (deque)

разновидность очереди: FIFO + LIFO



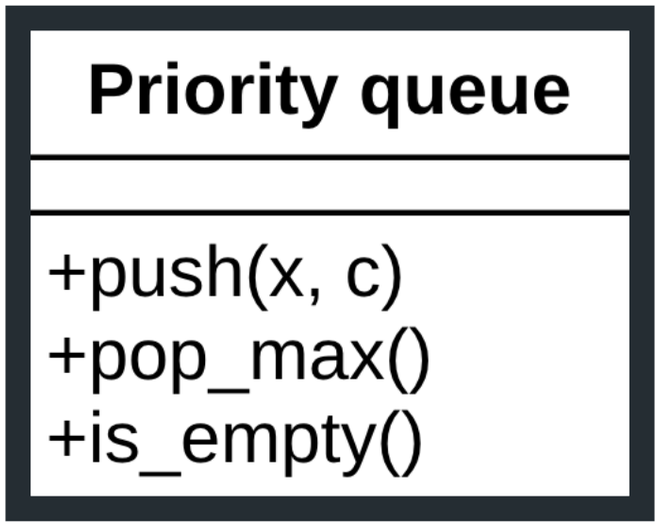
Очередь с приоритетом (priority queue)

разновидность очереди



Очередь с приоритетом (priority queue)

разновидность очереди

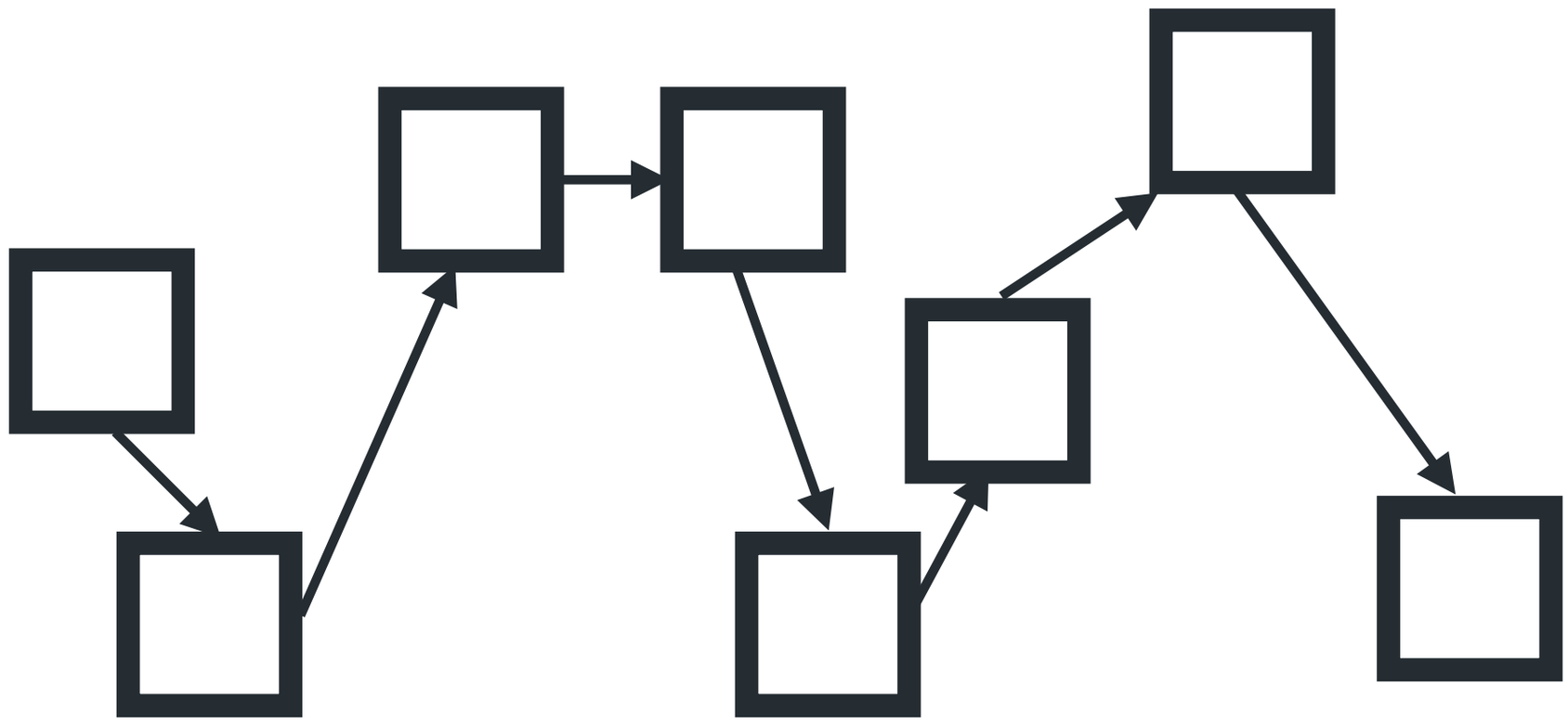


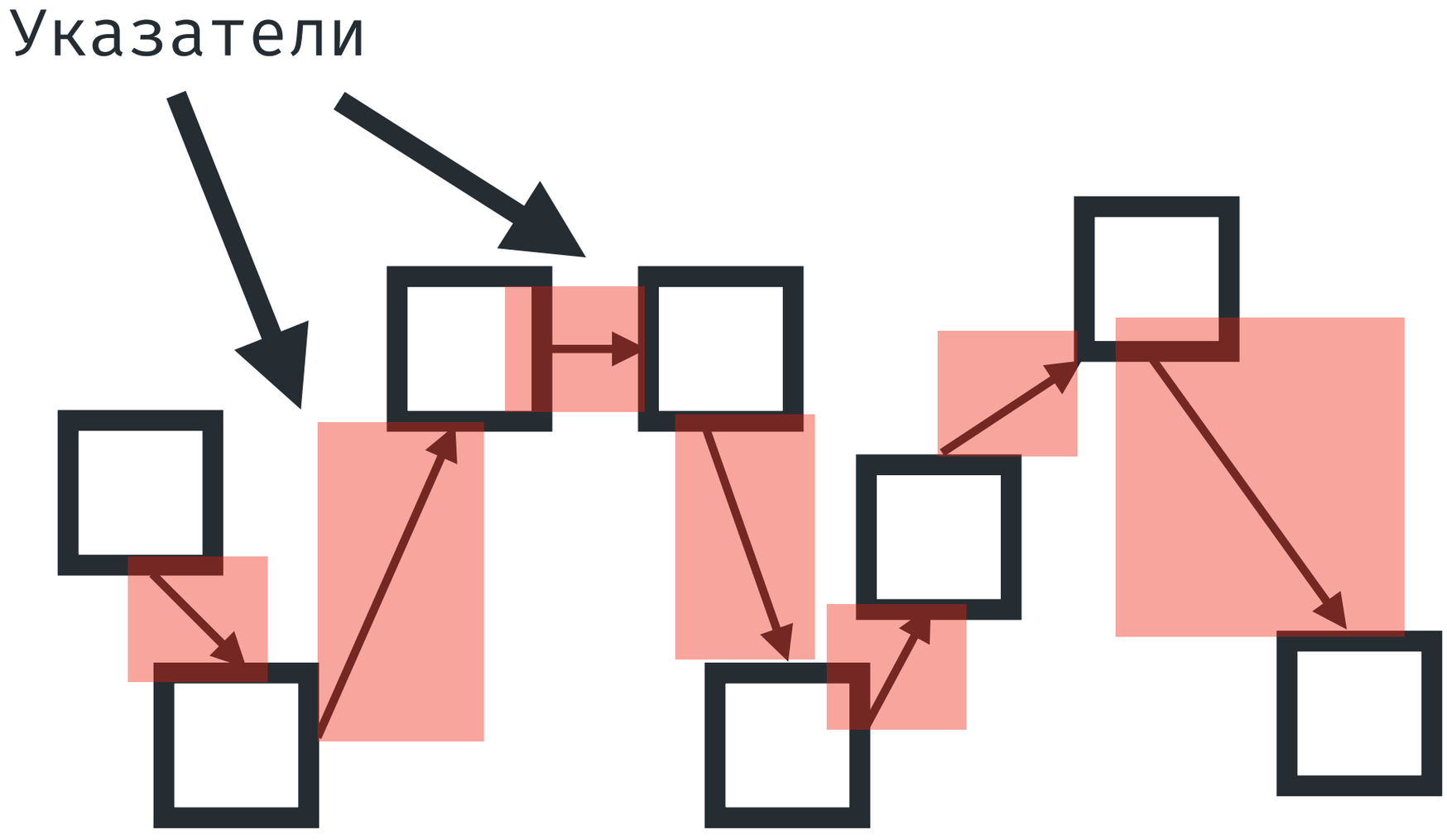
Реализация


Структуры данных
Массив
Array
Сязный список
Linked List
Пирамида (куча)
Heap
Односвязный список
Singly-Linked List
Двусвязный список
Doubly-Linked list
Список с пропусками
Skip List

Смежные структуры данных

Связные структуры данных




Указатели

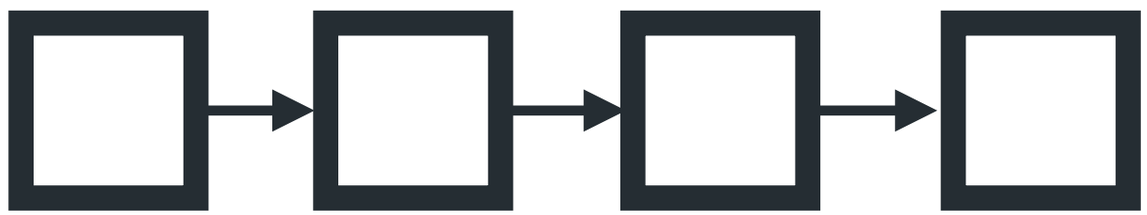
Массив

Массив

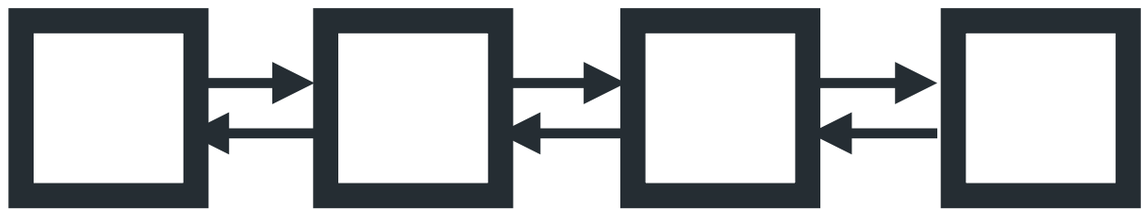
Common Data Structure Operations

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

Связный список



Singly-linked list



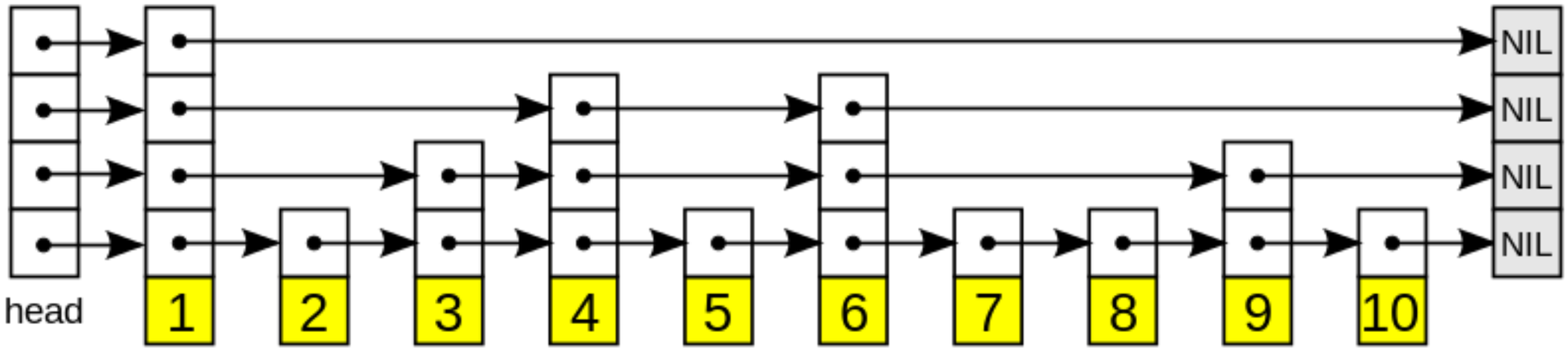
Doubly-linked list

Связный список

Common Data Structure Operations

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

Список с пропусками



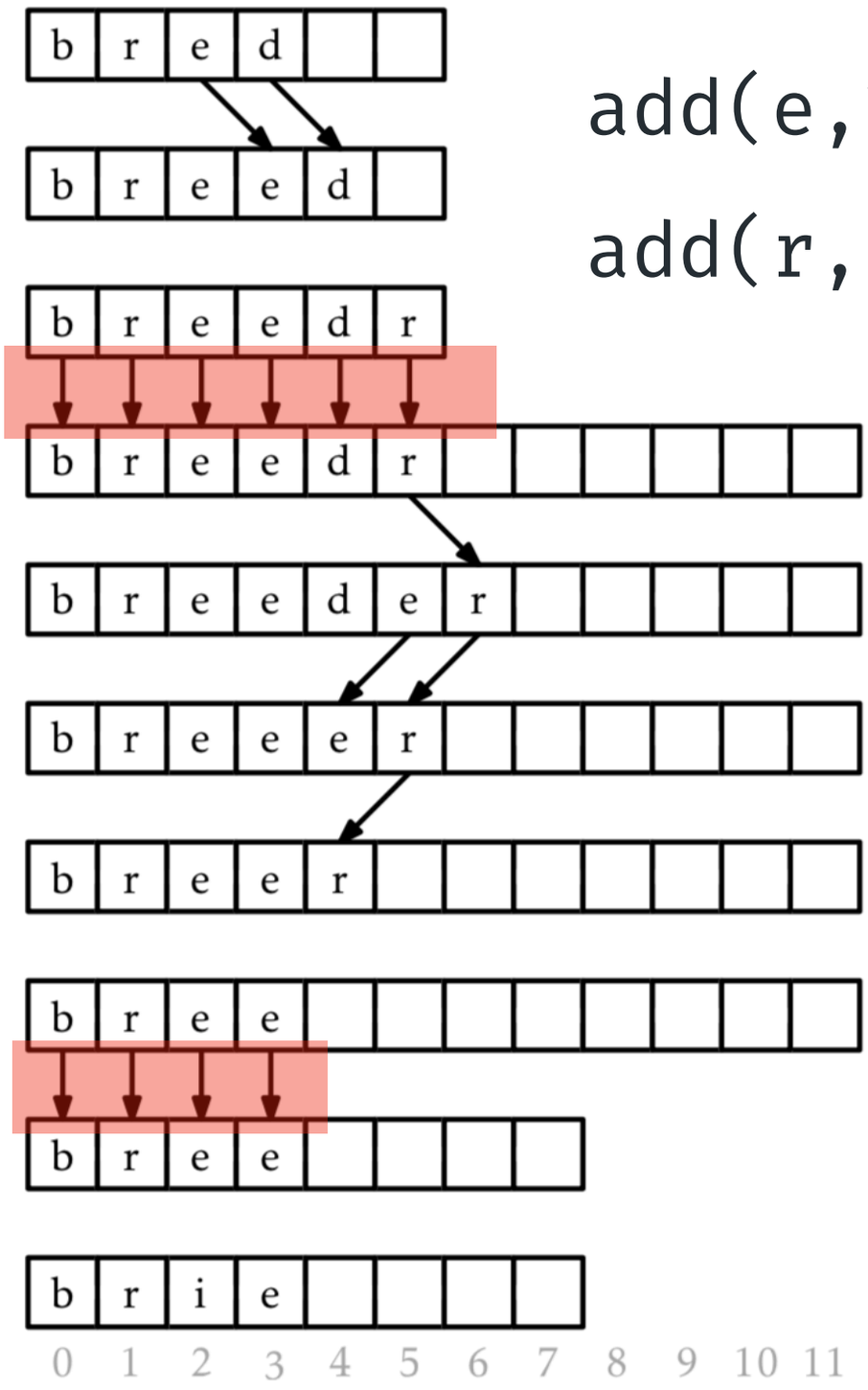
Список с пропусками

Common Data Structure Operations

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

Реализация списка

Изменение размера



add(e,2)

add(r,5)

} add(e,5)

remove(4)

remove(4)

} remove(4)

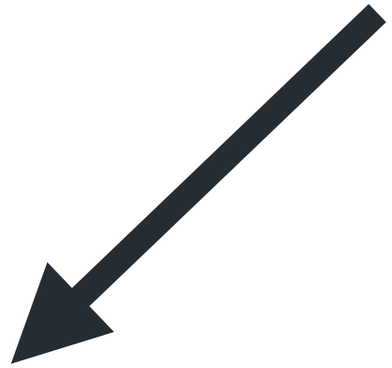
set(i,2)

Реализация очереди

Очередь (queue)

на массиве

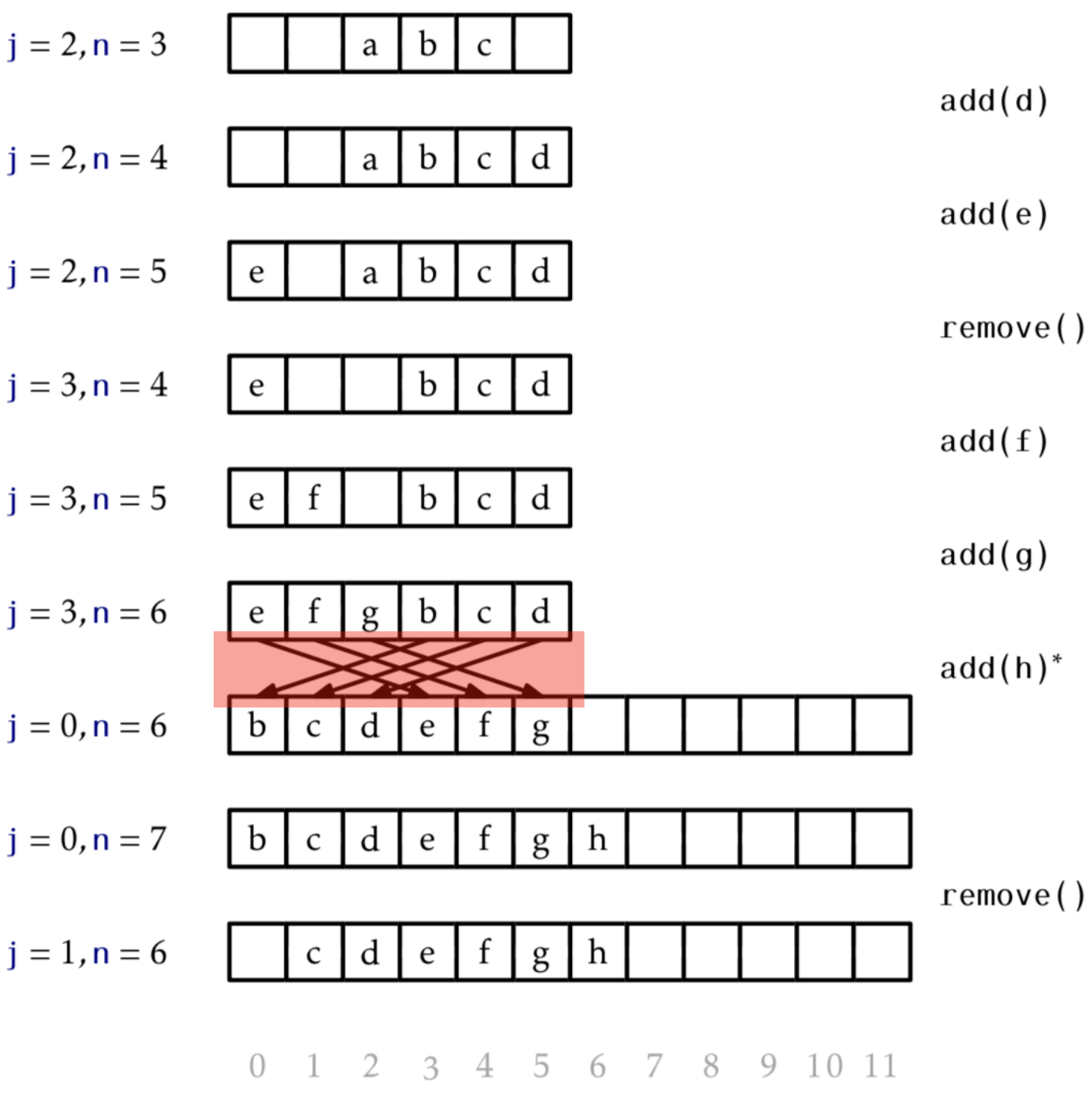
Дорого



```
push(x)     = add(x,0)
pop()       = remove(size()-1)
is_empty()  = size() == 0
```

Трюк для очереди на массиве

эмуляция бесконечного массива



Трюк для очереди на массиве

эмуляция бесконечного массива

Push

```
add((j+n)%size())
n += 1
```

Pop

```
get(j)
j = (j+1)%size()
n -= 1
```

Реализация стека

Стек (stack)

на массиве

```
push(x)    = add(x,size())
pop()      = remove(size()-1)
is_empty() = size() == 0
```

Реализация деки

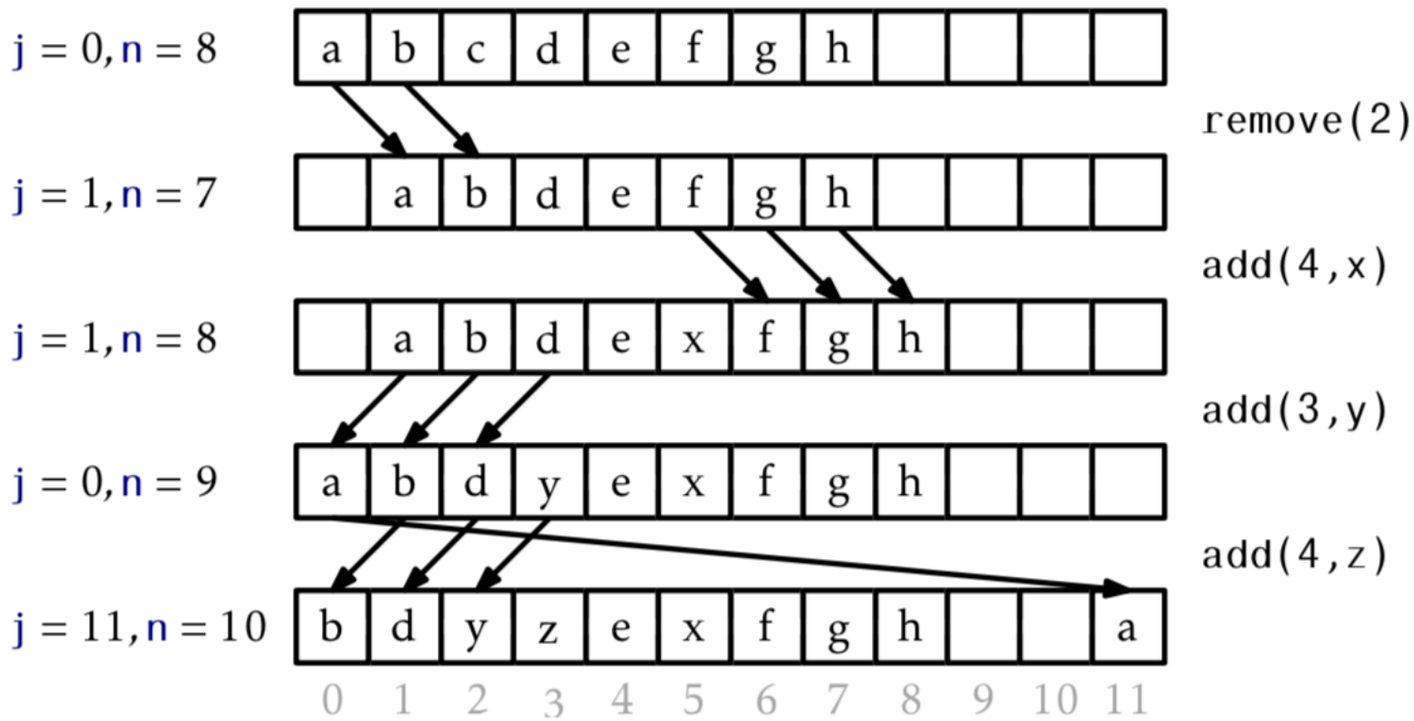
Дек (deque)

на массиве

```
push_first(x) = add(x,0)
push_last(x)  = add(x,size())
pop_first()   = remove(0)
pop_last()    = remove(size()-1)
is_empty()    = size() == 0
```

Трюк 1 для деки на массиве

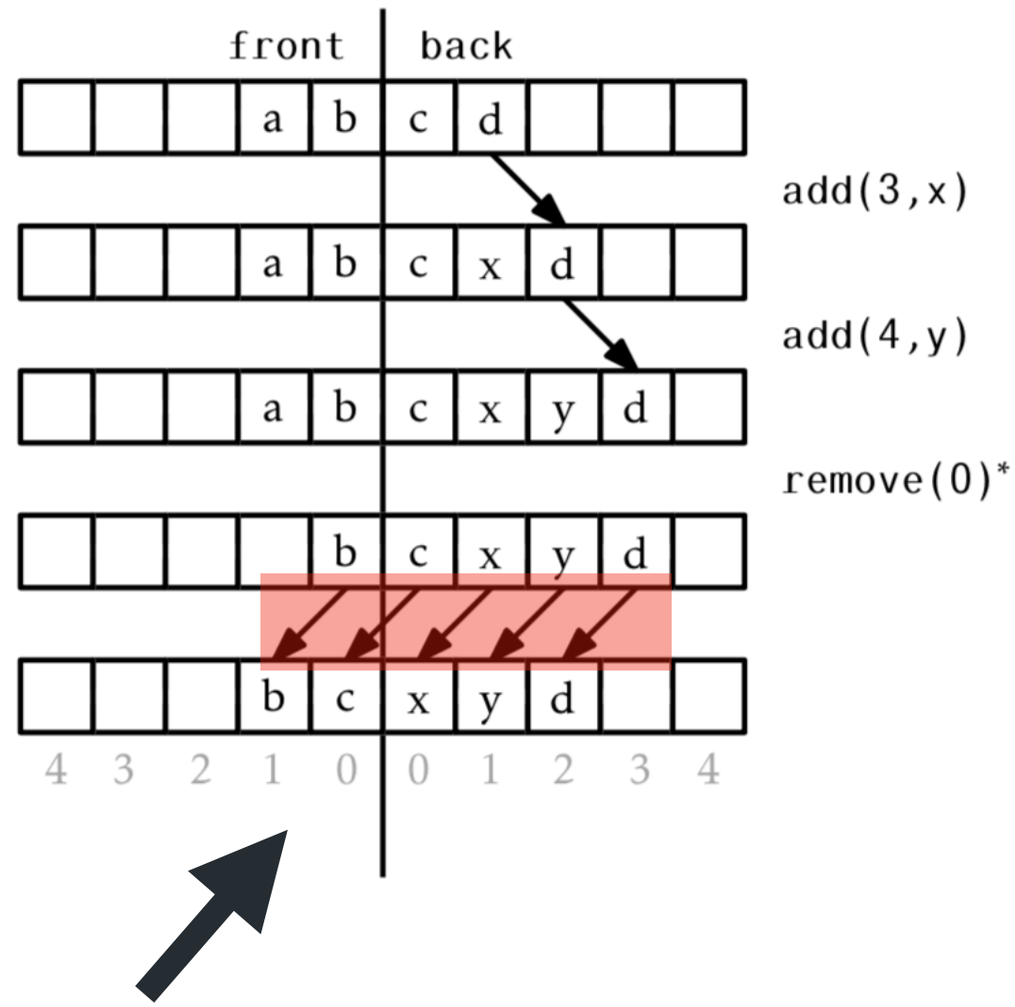
эмуляция бесконечного массива



сдвигаем в меньшую сторону

Трюк 2 для деки на массиве

два массива



балансировка

Реализация очереди с приоритетом

Очередь с приоритетом (priority queue)

на List с min/max или sort операциями

```
push(x)     = add(x, size())
pop_max()   = remove(max())
is_empty() = size() == 0
```

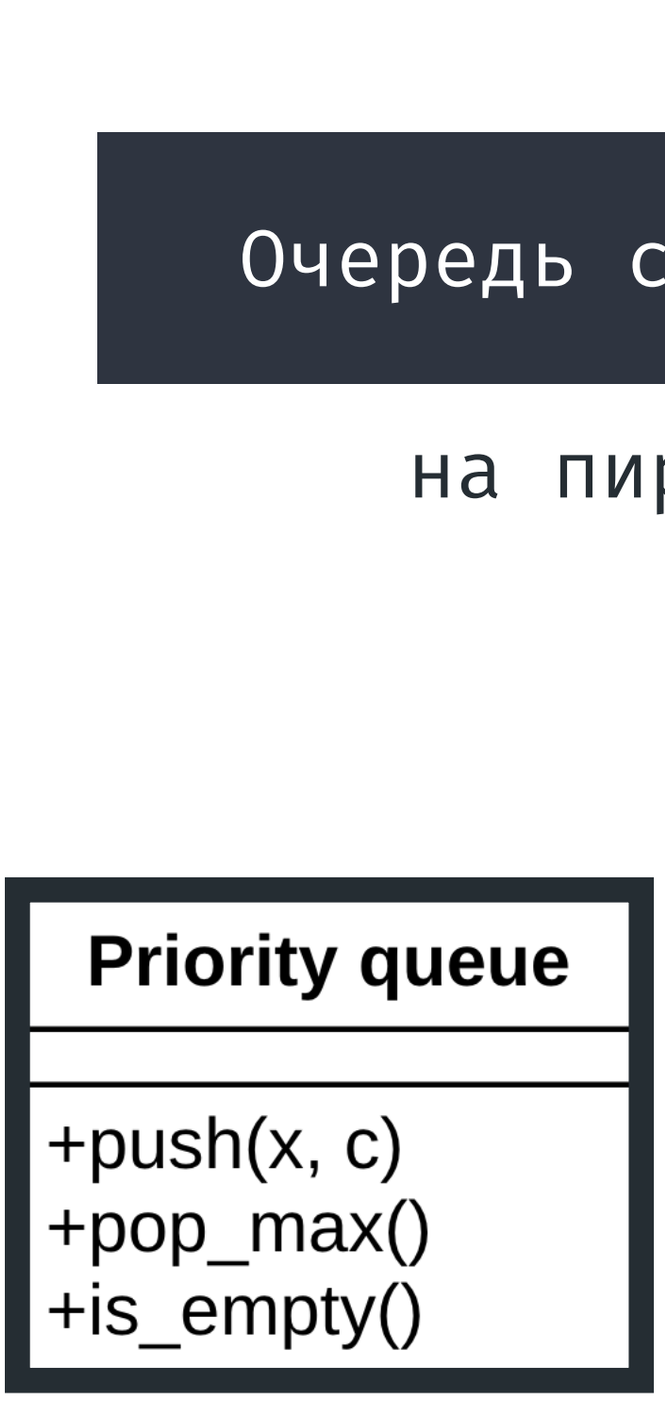
Очередь с приоритетом (priority queue)

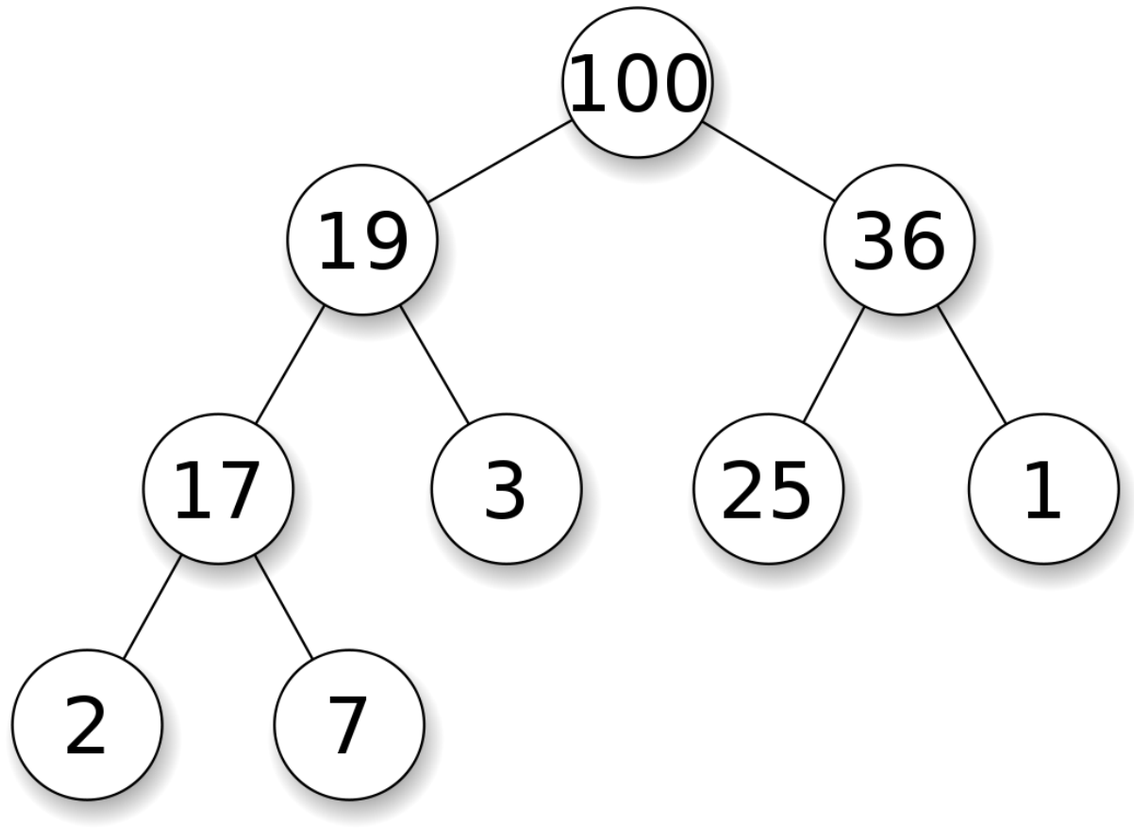
на List с min/max или sort операциями

```
push(x)     = add(x, size())
              sort()
pop_max()   = remove(size()-1)
is_empty()  = size() == 0
```

Очередь с приоритетом (priority queue)

на пирамиде (aka куча aka heap)





VISUALGO.NET/EN

visualising data structures and algorithms through animation

visualgo.net

Data Structure Visualizations

About
Algorithms
F.A.Q
Known Bugs / Feature Requests
Java Version
Flash Version
Create Your Own / Source Code
Contact

Currently, we have visualizations for the following data structures and algorithms:

- Basics
  - Stack: Array Implementation
  - Stack: Linked List Implementation
  - Queues: Array Implementation
  - Queues: Linked List Implementation
  - Lists: Array Implementation (available in java version)
  - Lists: Linked List Implementation (available in java version)
- Recursion
  - Factorial
  - Reversing a String

www.cs.usfca.edu/~galles/visualization/Algorithms.html

Реализация в Python


KOZLIK'S
DIJON BY ANTON
KOZLIK'S
DIJON BY ANTON