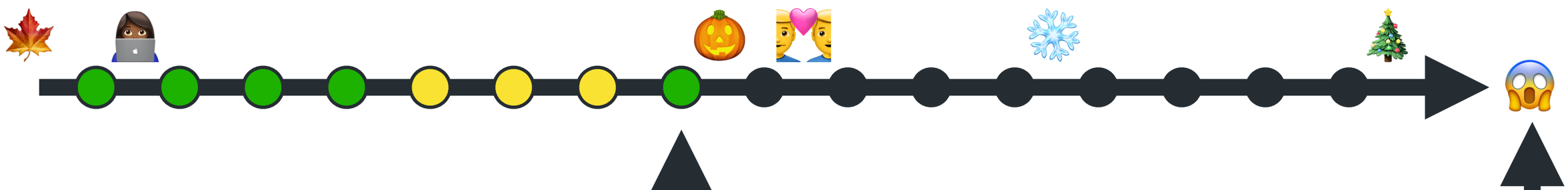
ИНФОРМАТИКА

```
~/polytech $ date
осень 2019
~/polytech $ name
константин кориков
~/polytech $ topic
атд: словарь и множество
```

ГДЕ МЫ?



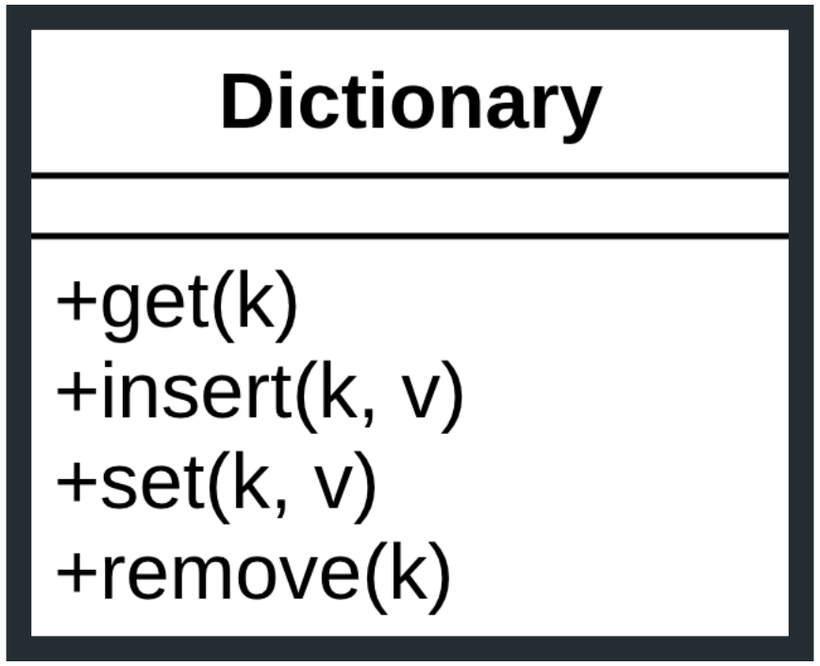
Мы тут

Сессия

О ЧЁМ ПОГОВОРИМ?

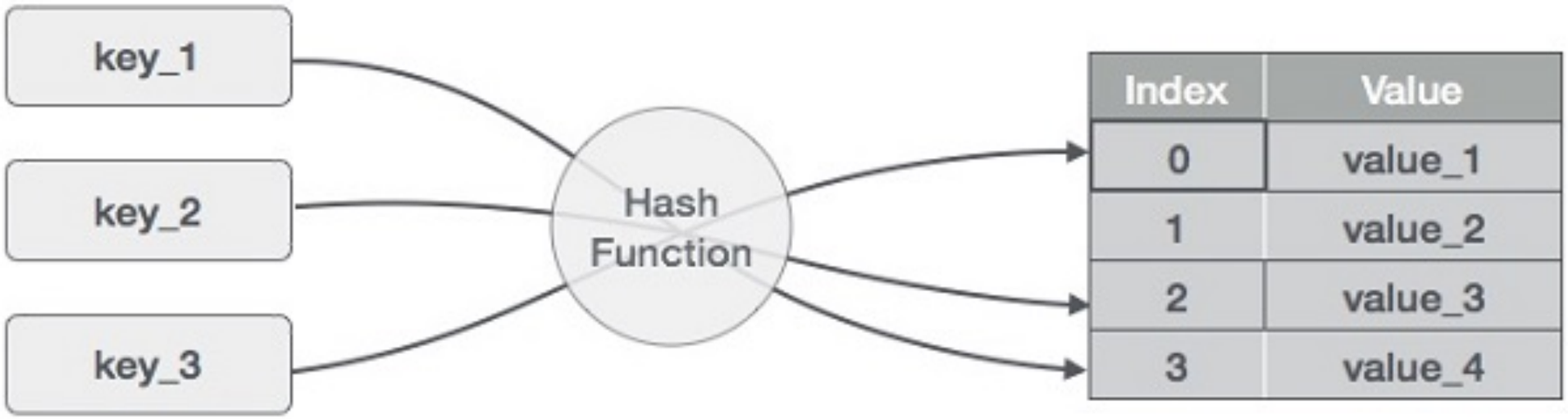
1. АТД словарь
2. Словарь на хэш-таблице
3. Алгоритм Рабина-Карпа
4. АТД множество
5. Словарь и множество в Python

АТД словарь


Dictionary
+get(k)
+insert(k, v)
+set(k, v)
+remove(k)

Реализации: массив, связный список, дерево, список с пропусками, хэш-таблица

Словарь на хэш-таблица



Хэш-функция

Хорошая хэш-функция — быстрая и равномерно распределяющая хэш-функция

Хэш-функция

пример для положительных целых чисел
(модульное хэширование)

$$h = k \% M$$

размер массива
(просто число)

Хэш-функция

пример для строк

$$h = \sum (k[i] \times R) \,\%\, M$$

i

символ строки

maciejczyzewski / retter

Watch 4 ★ Star 47 Fork 14

Code Issues 1 Pull requests 0 Projects 0 Security Insights

Branch: master ▾ retter / algorithms / Create new file Upload files Find file History

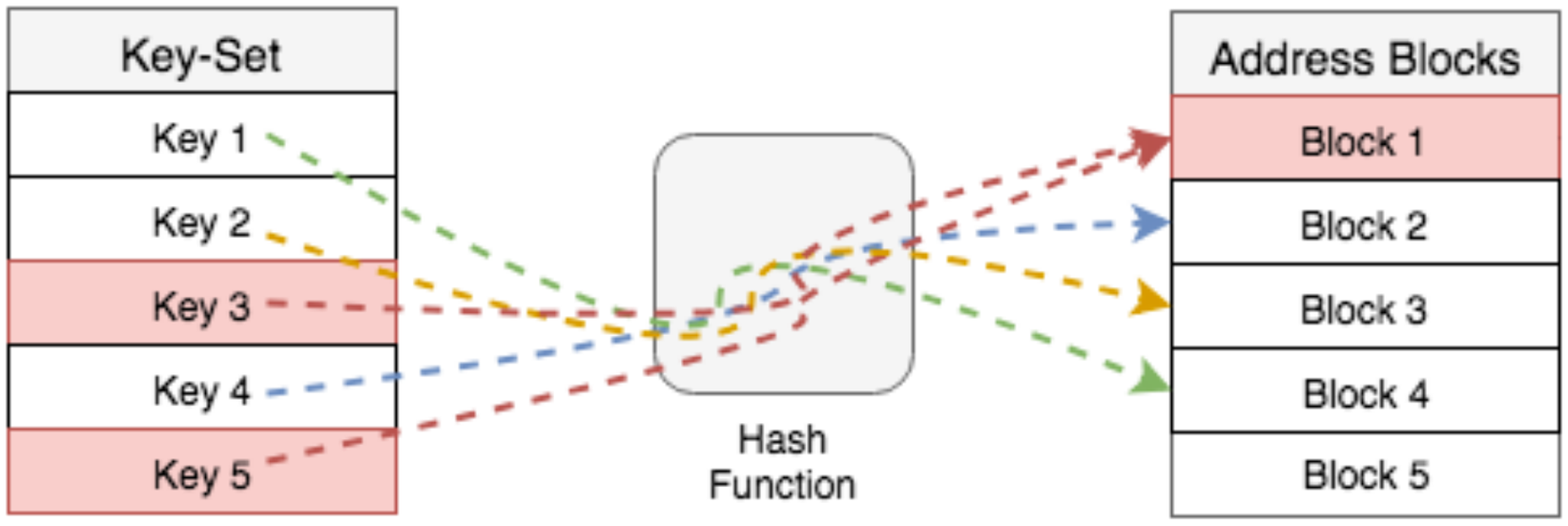
Maciej A. Czyzewski meta: re-initialized repository without old history — Latest commit 8e97be1 on 27 Dec 2016

..

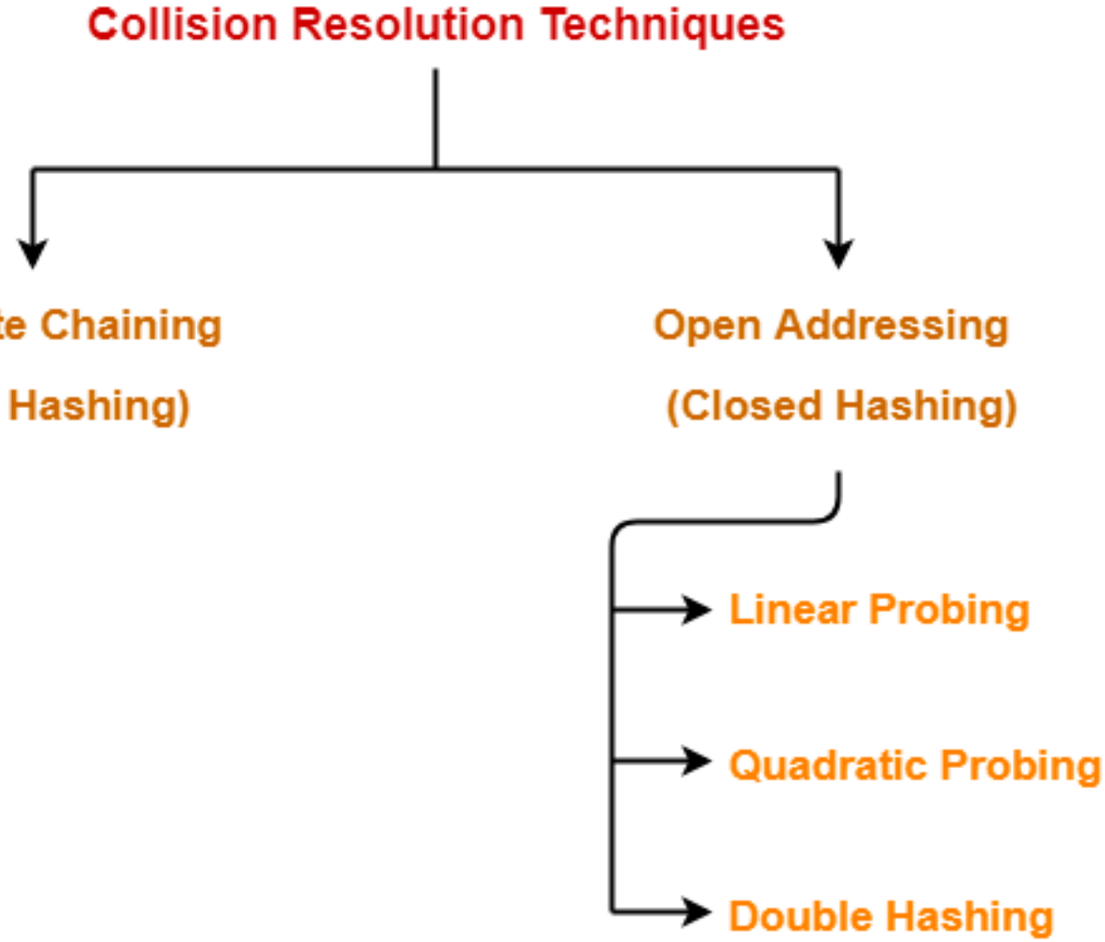
AES	meta: re-initialized repository without old history	3 years ago
Adler-32	meta: re-initialized repository without old history	3 years ago
B.B.S.	meta: re-initialized repository without old history	3 years ago
BLAKE	meta: re-initialized repository without old history	3 years ago
BSD	meta: re-initialized repository without old history	3 years ago
CRC	meta: re-initialized repository without old history	3 years ago
CityHash	meta: re-initialized repository without old history	3 years ago
Damm	meta: re-initialized repository without old history	3 years ago
ECOH	meta: re-initialized repository without old history	3 years ago
FNV	meta: re-initialized repository without old history	3 years ago
FSB	meta: re-initialized repository without old history	3 years ago
Fletcher	meta: re-initialized repository without old history	3 years ago
GOST	meta: re-initialized repository without old history	3 years ago
Grostl	meta: re-initialized repository without old history	3 years ago
HAS-160	meta: re-initialized repository without old history	3 years ago
HAVAL	meta: re-initialized repository without old history	3 years ago
ISAAC	meta: re-initialized repository without old history	3 years ago
JH	meta: re-initialized repository without old history	3 years ago
Jenkins	meta: re-initialized repository without old history	3 years ago
Luhn	meta: re-initialized repository without old history	3 years ago
MD2	meta: re-initialized repository without old history	3 years ago
MD4	meta: re-initialized repository without old history	3 years ago
MD5	meta: re-initialized repository without old history	3 years ago
MD6	meta: re-initialized repository without old history	3 years ago
MT19937	meta: re-initialized repository without old history	3 years ago
MurmurHash	meta: re-initialized repository without old history	3 years ago
Pearson	meta: re-initialized repository without old history	3 years ago
RC4	meta: re-initialized repository without old history	3 years ago
RIPEMD	meta: re-initialized repository without old history	3 years ago
RSA	meta: re-initialized repository without old history	3 years ago
RadioGatun	meta: re-initialized repository without old history	3 years ago
SHA-1	meta: re-initialized repository without old history	3 years ago
SHA-2	meta: re-initialized repository without old history	3 years ago
SHA-3	meta: re-initialized repository without old history	3 years ago
SWIFFT	meta: re-initialized repository without old history	3 years ago
SYSV	meta: re-initialized repository without old history	3 years ago
SipHash	meta: re-initialized repository without old history	3 years ago
Skein	meta: re-initialized repository without old history	3 years ago
Snefru	meta: re-initialized repository without old history	3 years ago
Spectral	meta: re-initialized repository without old history	3 years ago
Tiger	meta: re-initialized repository without old history	3 years ago
Verhoeff	meta: re-initialized repository without old history	3 years ago
Whirlpool	meta: re-initialized repository without old history	3 years ago
Wichmann-Hill	meta: re-initialized repository without old history	3 years ago
Xorshift	meta: re-initialized repository without old history	3 years ago
Zobrist	meta: re-initialized repository without old history	3 years ago
xxHash	meta: re-initialized repository without old history	3 years ago

https://github.com/maciejczyzewski/retter

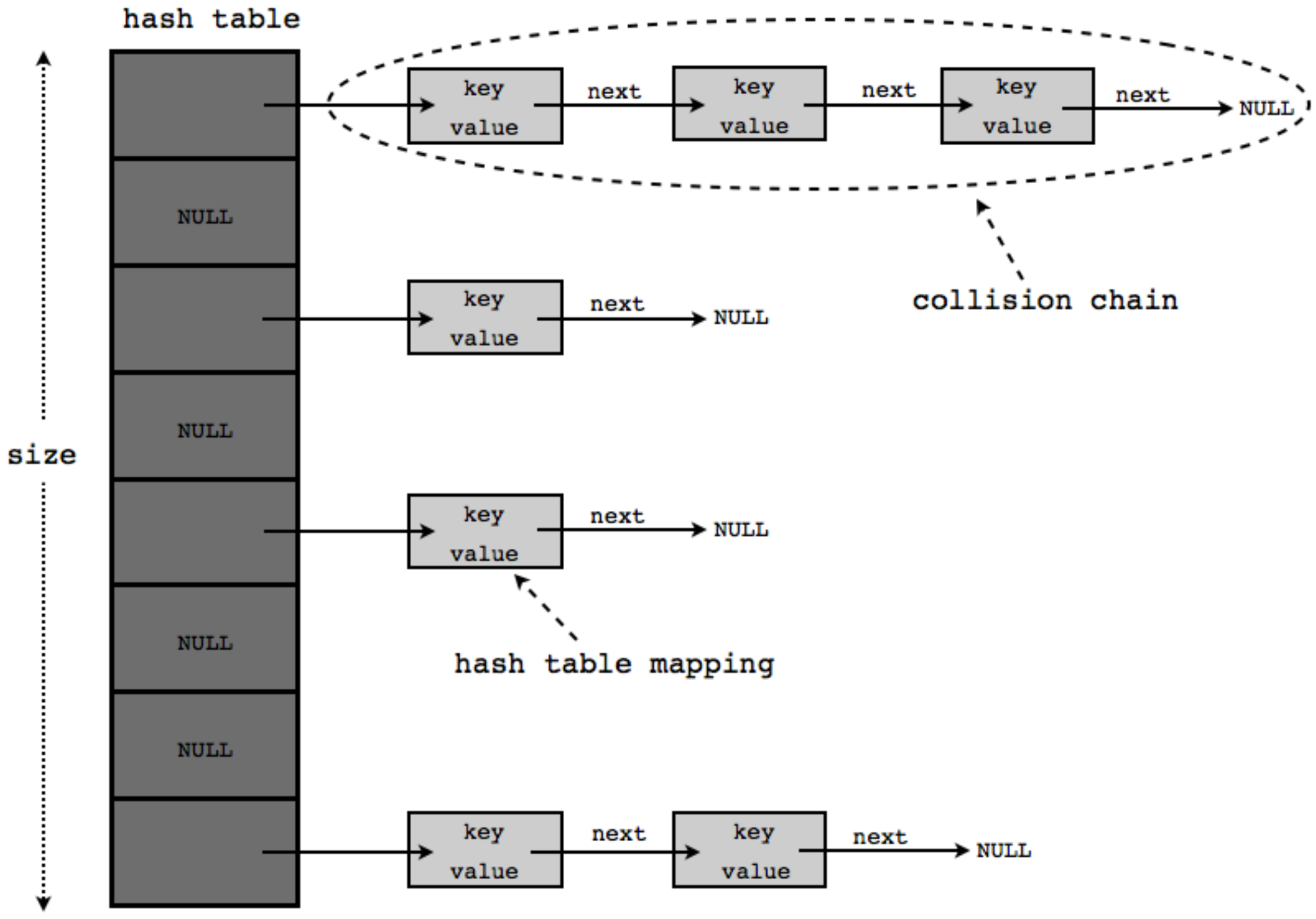
Коллизии


Key-Set
Key 1
Key 2
Key 3
Key 4
Key 5
Hash
Function
Address Blocks
Block 1
Block 2
Block 3
Block 4
Block 5

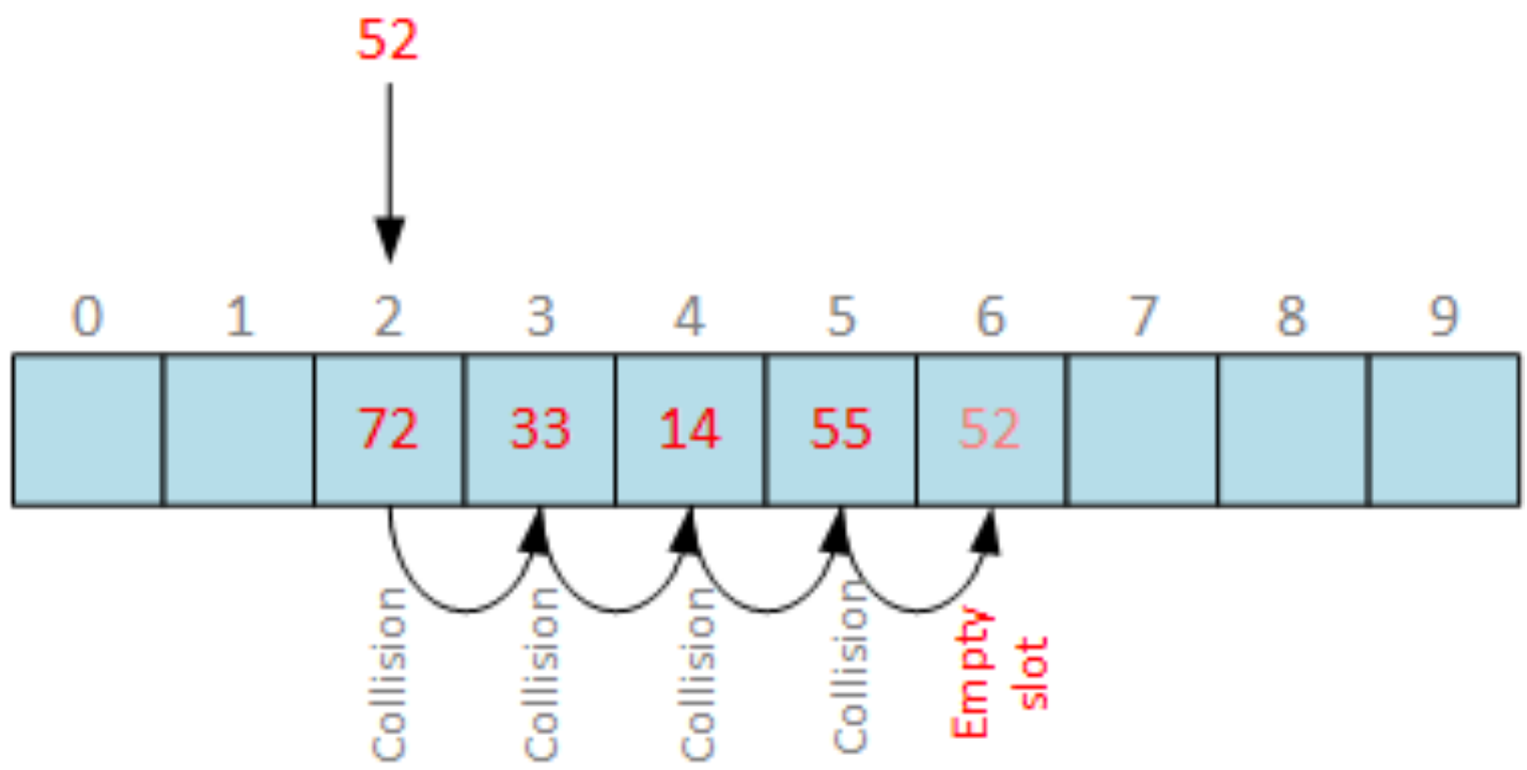
Коллизии


Collision Resolution Techniques
Separate Chaining
(Open Hashing)
Open Addressing
(Closed Hashing)
Linear Probing
Quadratic Probing
Double Hashing

Решение на связном списке

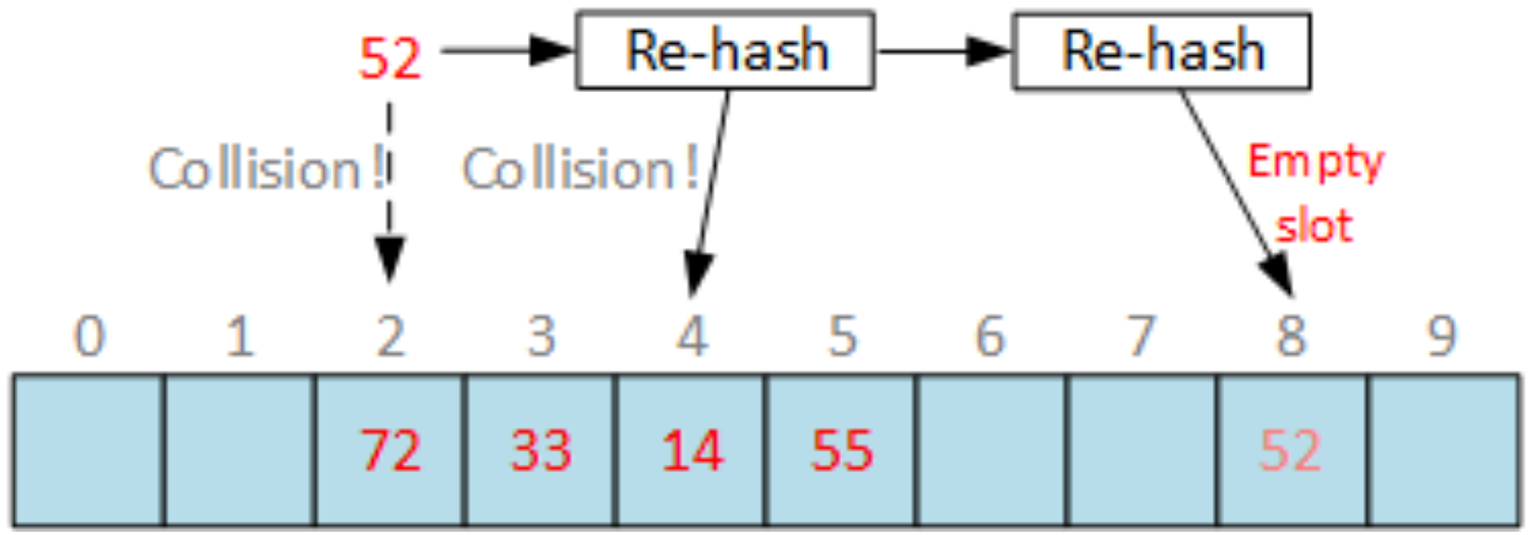


Решение с открытой адресацией



сложности с удалением

Решение с открытой адресацией



сложности с удалением

Hash table

Type	Unordered associative array
Invented	1953

Time complexity in big O notation

Algorithm	Average	Worst case
Space	O(*n*)[1]	O(*n*)
Search	O(1)	O(*n*)
Insert	O(1)	O(*n*)
Delete	O(1)	O(*n*)

чем хороши
хэш-таблицы

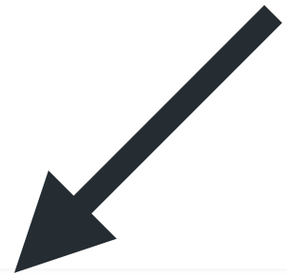
Common Data Structure Operations

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)

Алгоритм Рабина-Карпа

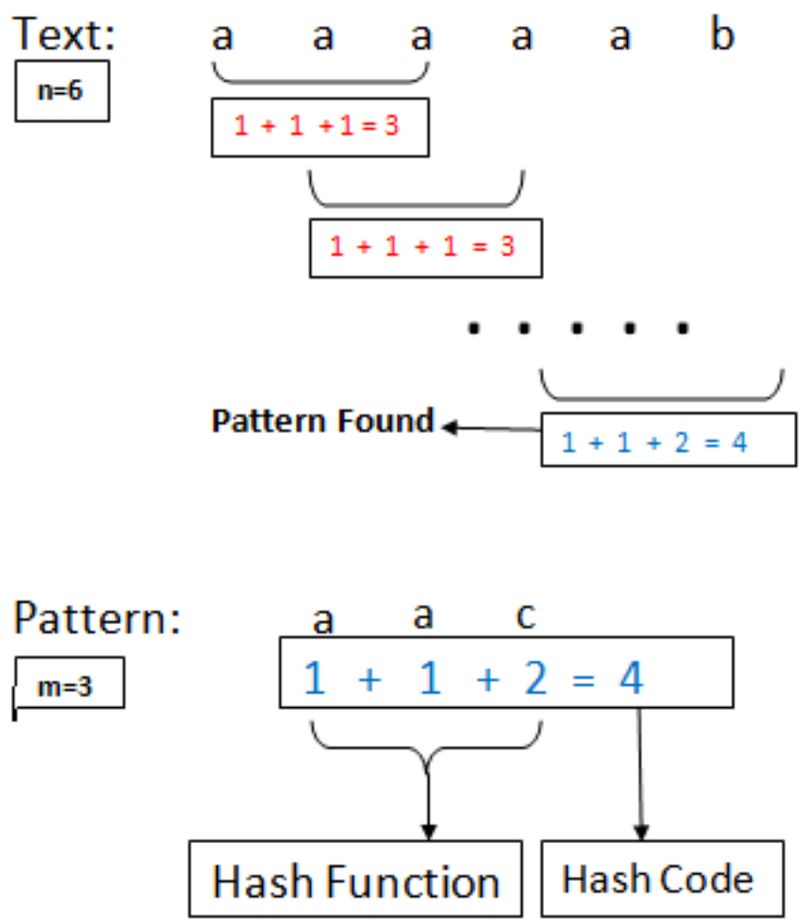
задача: найти подстроку в строке

наивный поиск



```
1 function NaiveSearch(string s[1..n], string sub[1..m])
2     for i from 1 to n-m+1
3         for j from 1 to m
4             if s[i+j-1] ≠ sub[j]
5                 перейти к следующей итерации внешнего цикла
6         return i
7     return not found
```

$$O(nm)$$


Text:
a a a a a b
n=6
1 + 1 +1 = 3
1 + 1 + 1 = 3
Pattern Found
1 + 1 + 2 = 4
Pattern:
a a c
m=3
1 + 1 + 2 = 4
Hash Function
Hash Code

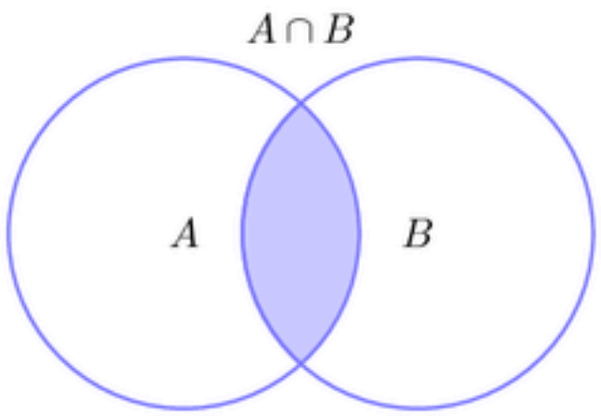
```
1 function RabinKarp(string s[1..n], string sub[1..m])
2     hsub := hash(sub[1..m])
3     hs := hash(s[1..m])
4     for i from 1 to (n-m+1)
5         if hs = hsub
6             if s[i..i+m-1] = sub
7                 return i
8         hs := hash(s[i+1..i+m])
9     return not found
```

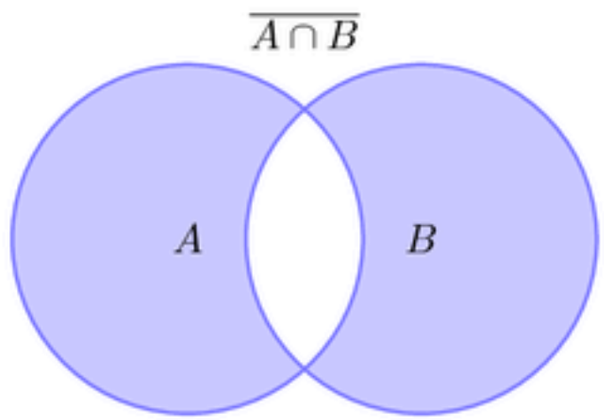
$$O(n + m)$$

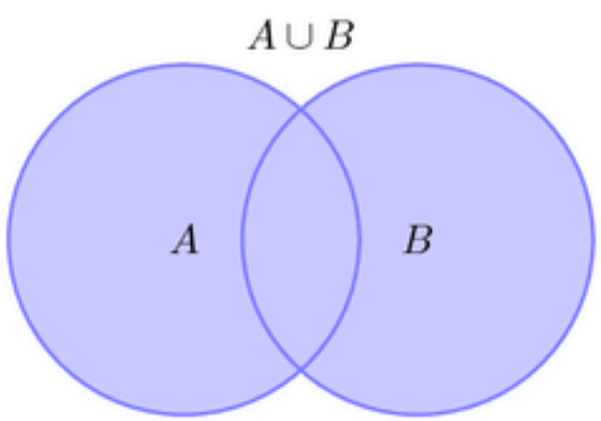
сокращаем число
вызовов этой части

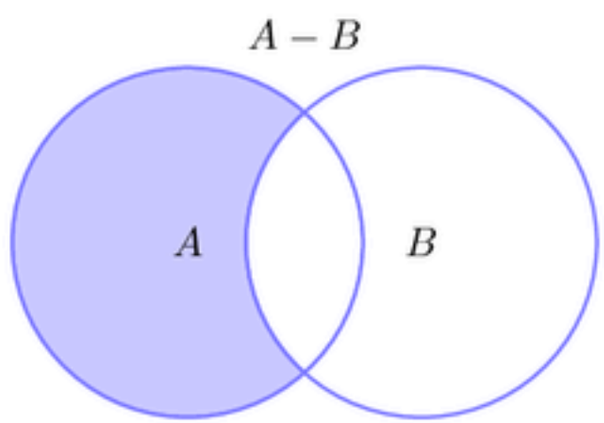
АТД множество

Диаграммы Эйлера-Венна

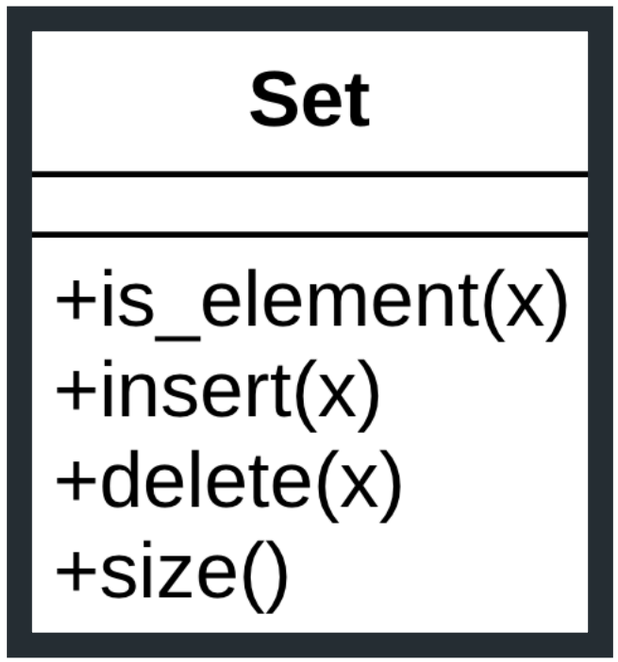








АТД множество



union, intersection, difference, subset

Мультимножество допускает дубли

Словарь в Python

```python
# Новый пустой словарь
d = dict()
d = {}

# Новый непустой словарь
d = dict([('lenta', 28), ('meduza', 42), ('rt', 34)])
d = {'lenta': 28, 'meduza': 42, 'rt': 34}
d = dict(lenta=28, meduza=52, rt=34)

# Поиск элемента (чтение)
d['meduza'] # вернет 42
d['furfur'] # вызовет исключение KeyError, потому что нет такого ключа
d.get('furfur', -1) # вернет -1, если нет ключа furfur

# Переопределение значения по ключу
d['rt'] = 15
d['adme'] = 21 # если ключа нет, то это вставка

# Ключ в словаре может быть любого хэшируемого типа (очевидно)
d[1] = 'значение' # целое число
d[3.14] = 'значение' # вещественное число
d[(1, 'строка')] = 'значение' # кортеж элементов хэшируемого типа
d[bool] = 'значение' # класс

# NB: значения могут быть любые
```

```python
d = {3: 'Да', 3.14: 'Нет'}
a = {3: 'Нет', 2: 'Нет'}

# Операции над словарем
3.14 in d # возвращает True, если есть в словаре ключ 3.14
3.15 in d and d[3.15] # трюк с предварительной проверкой значения в словаре
len(d) # количесвто пар ключ-значение в словаре
d.update(a) # обновить словарь d значениями из словаря a
d.update(one='Да') # обновить значение в словаре d (только для строк)
d.clear() # очистить словарь
d.pop(2, 'опциональное значение на случай отсуствия ключа') # удалить ключ из словаря и вернуть значение


# Преобразования словаря
d.items() # в список кортежей (ключ, значение)
d.keys() # в список ключей
d.values() # в список значений
```

Множество в Python

```python
# Новое пустое множество
s = set()
s = {*()} # литерала нет, это хак

# Новое непустое множество (может содержать значения любого хэшируемого типа)
s = set([1,2,3,4]) # из списка
s = set((1, 'a', True)) # из кортежа
s = {1, 'a', True}
# Осторожно! Это разные множества:
s = set('foo')
s = {'foo'}
```

```python
a = {1,2,3,4,5}
b = {2,4}
c = {1,3,5}

# Операции
len(a) # количество элементов
4 in b # проверка вхождения в множество
a | b # объединение через оператор
a.union(b) # объединение через метод
a & b # пересечение через оператор
a.intersection(b) # пересечение через метод
a - b # разность через оператор
a.difference(b) # разность через метод
a ^ b # симметрическая разность через оператор
a.symmetric_difference(b) # симметрическая разность через метод

# Проверка на подмножество
b ⩽ a
b.issubset(a)
b < a
a > b
a ⩾ b
a.issuperset(b)

# Модификации
a.update(b)
a |= b
a.add(6) # добавить элемент
a.remove(6) # удалить элемент
a.discard(6) # удалить элемент без ошибок
a.clear() # очистить множество
```

```python
fs = frozenset(['foo', 'bar', 'baz']) # множество только для чтения

from collections import Counter # мультимножество
a = Counter('gallahad')  # вернет Counter({'a': 3, 'l': 2, 'g': 1, 'h': 1, 'd': 1})
a.elements() # вернет итератор по всем элементам
a.most_common(4) # вернет top 4 самых частых элемента

b = Counter(('a', 'b'))
a + b # объединение
a - b # объединение (с неотрицательными счетчиками)
a | b # max-объединение
a & b # min-пересечение

+a # вернет все мультимножество с счетчиками ⩾0
-a # вернет все мультимножество с счетчиками <0
```