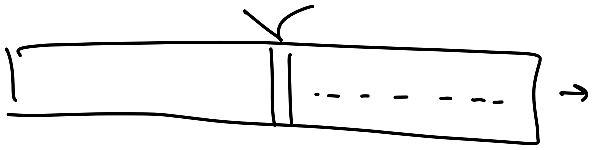
# Хеш - таблица

## I Мотивация

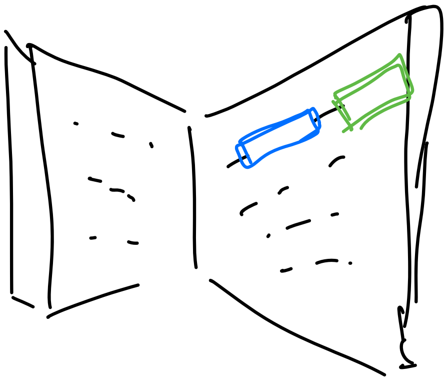
Время

| | Push | Pop | Find |
|---|---|---|---|
| Static Array | $\underline{O}(N)$ | $\underline{O}(N)$ | $\underline{O}(N)$ |
| List | $\underline{O}(1)$ | $\underline{O}(1)$ | $\underline{O}(N)$ |
| Sorted Array | $\underline{O}(N)$ | $\underline{O}(N)$ | $\underline{O}(\log N)$ |
| Dynamic Array | $\underline{O}^*(1)$ | $\underline{O}^*(1)$ | $\underline{O}(N)$ |
| Хеш-таблица | $\underline{O}^*(1)$ | $\underline{O}^*(1)$ | $\underline{O}^*(1)$ |



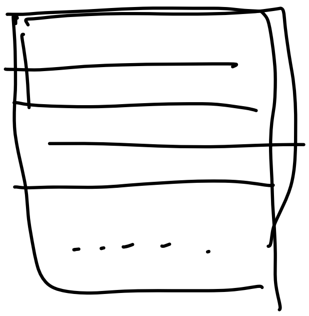
Память

## II Хеш-таблица с прямой адресацией.



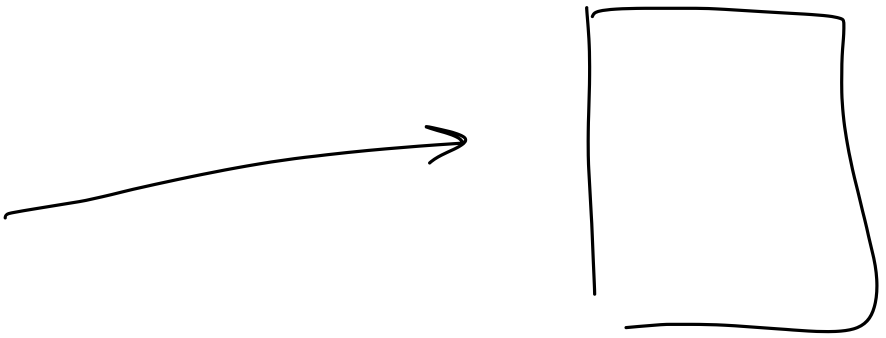
Номер телефона → владелец.

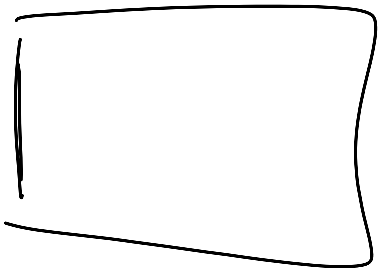
$\underbrace{8\,800\ldots\ldots}_{11\text{ цифр}}$ →



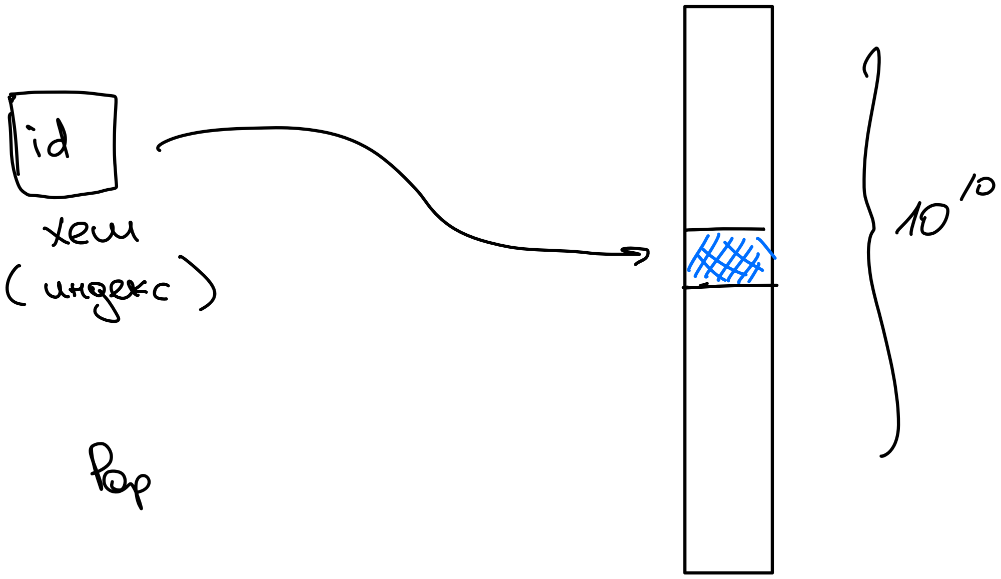
1 Гб

телефон





Find

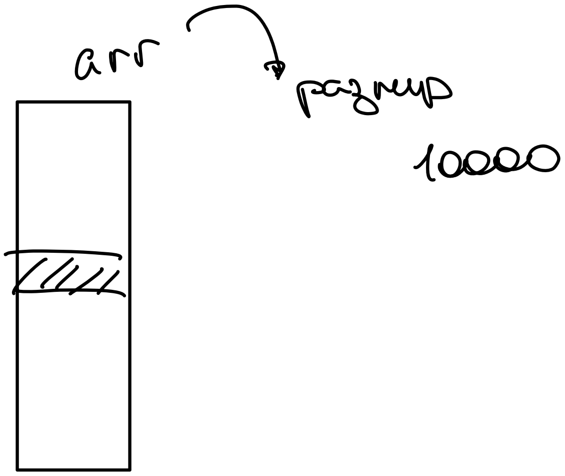


Push Pop

Ⓥ Прямая Адресация

1. Мн значения id
2. Не повторяются данные

$0 \le$ число $\le 10000$



Push

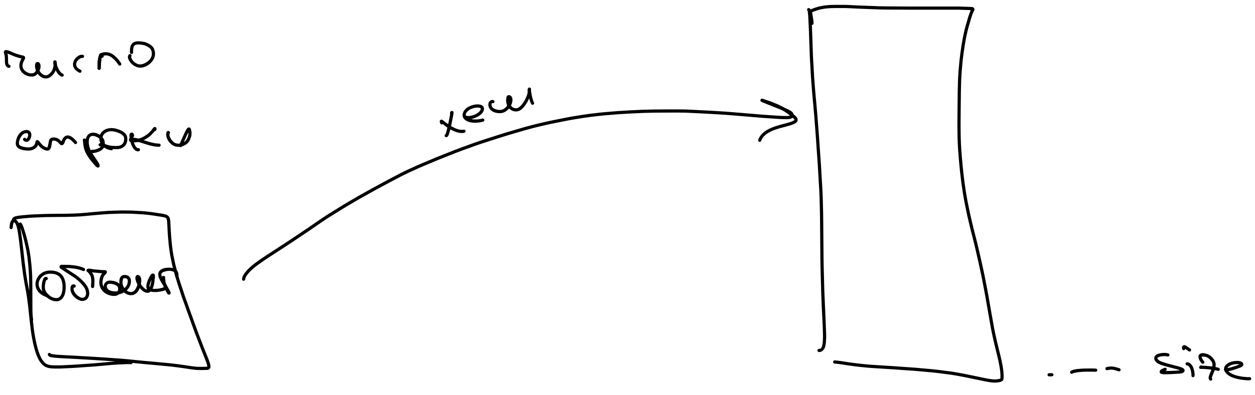
```
num = ввести данные
arr[num] = True        O(1)
```

Pop

```
arr[num] = false       O(1)
```

Find

```
O(1)  print( arr[num] )
```



хеш-функция (объект) → хеш.

Работать с числами

```
функция hash(num)        число        num ~ $10^{100}$
{
    return num mod size
                остаток от деления
}
```

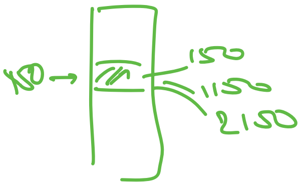
## III Коллизия.

size = 1000

| | |
|---|---|
| num < size | num >= size |
| hash=num | hash = num % size |

50 → hash=50

150 → hash=150



коллизия

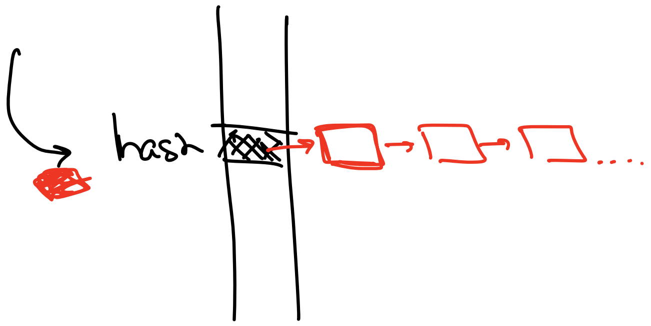
1500 → hash= 500

2505 → hash = 505

1150 → hash: 150

## IV Решение коллизий → двойной адресации

метод цепочек



```
Find (elem)   //односв.

runner= root
while   runner!= Null
        if  runner.vallue=elem
              return True
        runner= runne. next
return   False
```

V

```
Class        HashTable
{
    array =[]/10        // храним пустые
    size                     односв. списки
    capacity

метод     Find ( obj) {
            hash = HashFunction (obj, capacity)

            return  односв. список → $\underbrace{\text{array[hash]}}_{\text{односв. список}}$. Find (obj)
}
```

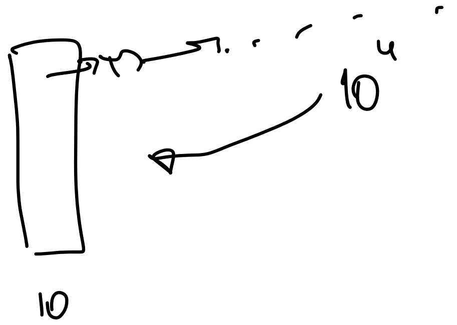
```
метод     Push (obj) {

            hash= HashFunction(obj, capacity)
            if   arr[hash]. find (obj) ==true
                   ↳ return
            arr[hash]. Push (obj)
            size =size+1
}
```

если
   size >, capacity/2
  ↳ resize()

```
метод     Pop (obj) {,

            hash= . . . .
            arr[hash]. Pop(obj)
}           size  -=1
```



$\to O$

$\underline{O}(n)$

```
Resize ( ) {

    new_arr = [ ]   // capacity * 2
    for   i=0 ...  capacity
        List = arr[i]
        while    List.isnotempty()
                element = List.Top()
                List.pop()
                h = HashFunction(elem, 2* capacity)
                new-arr[h]. push (element)

    capacity *= 2
}
```

инъекции ≡ нет коллизий