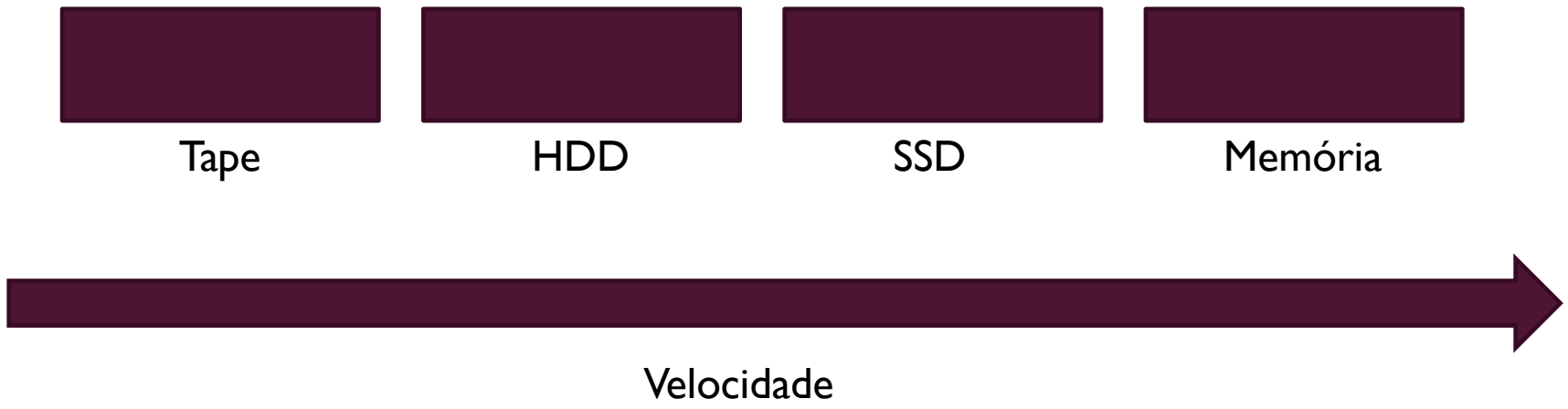
ESTRUTURAS DE DADOS PROBABILISTICAS

POR ARTUR BARUCHI – GROUPY-SP

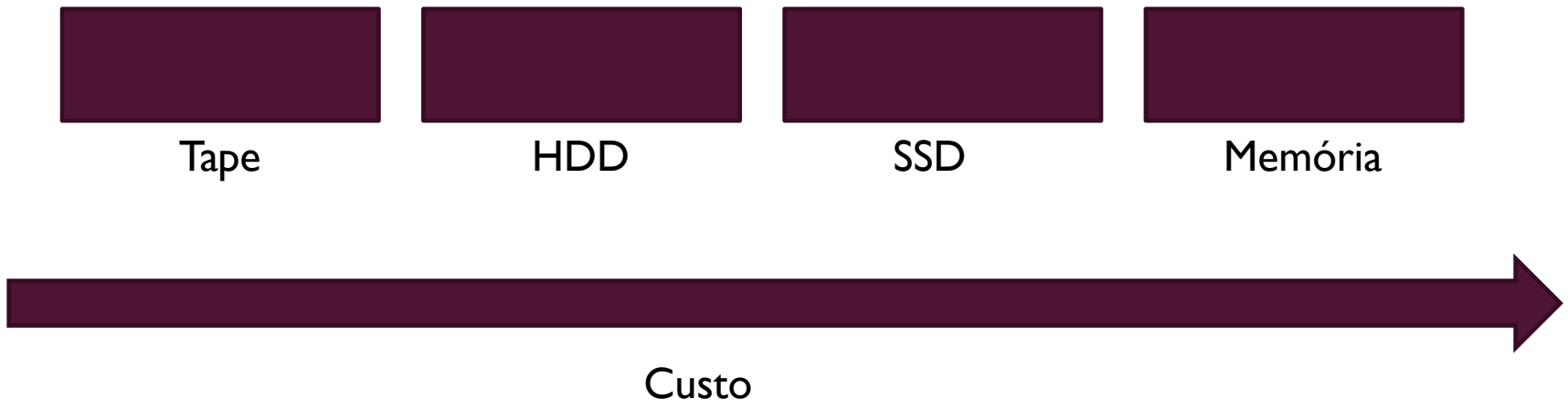
AGENDA

- Motivação
- Bloom Filters
- Count-Min Sketch
- Hyper LogLog

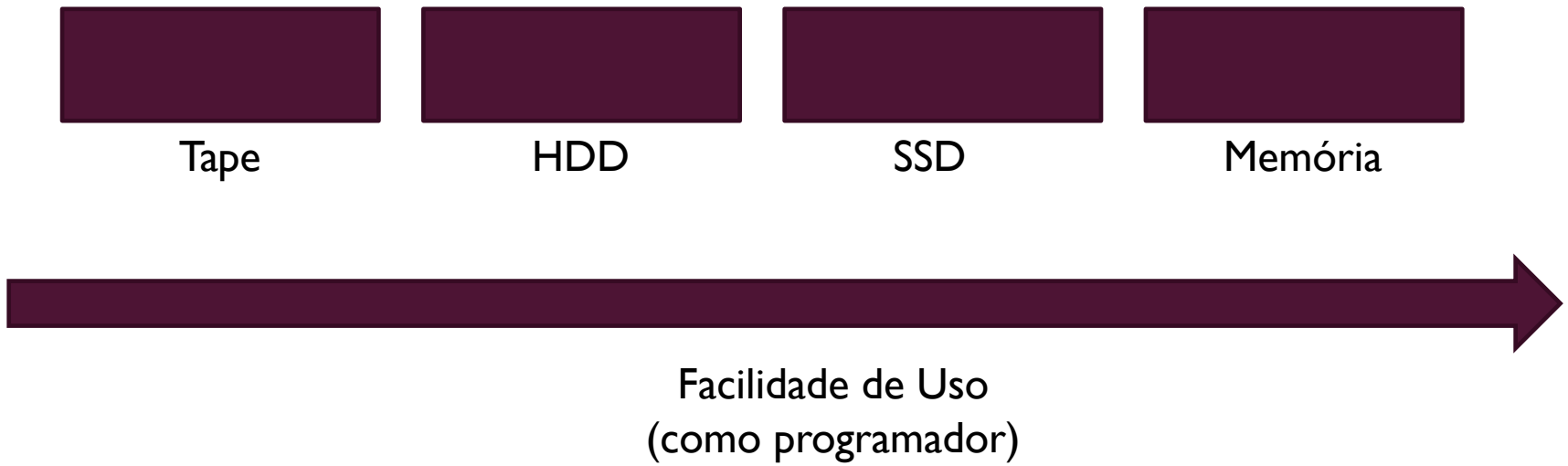
MOTIVAÇÃO



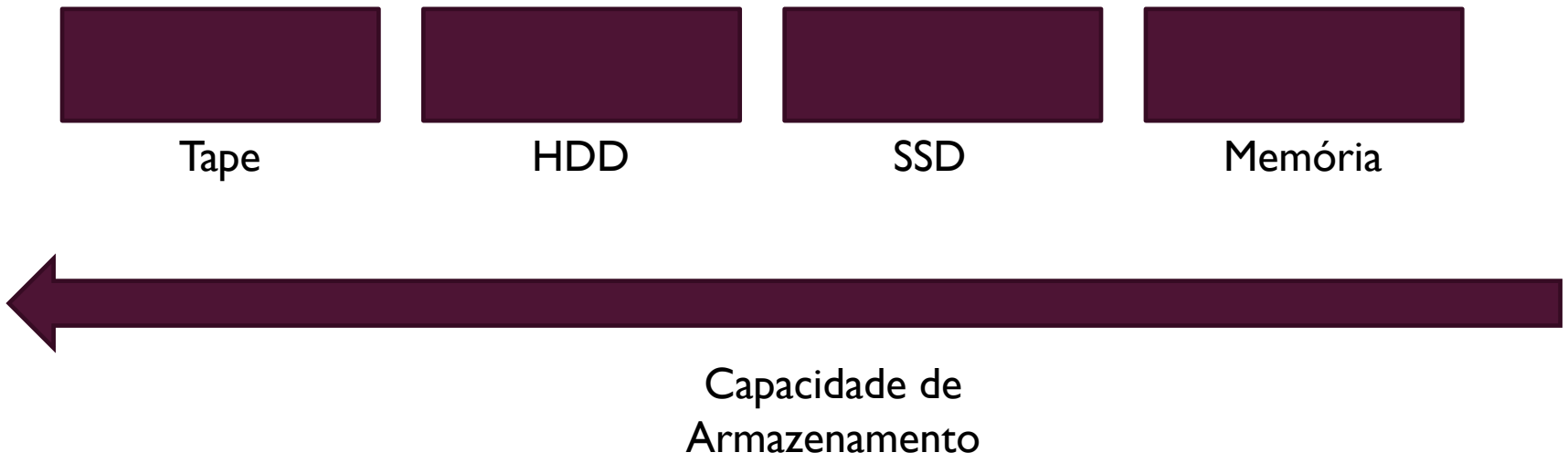
MOTIVAÇÃO



MOTIVAÇÃO



MOTIVAÇÃO

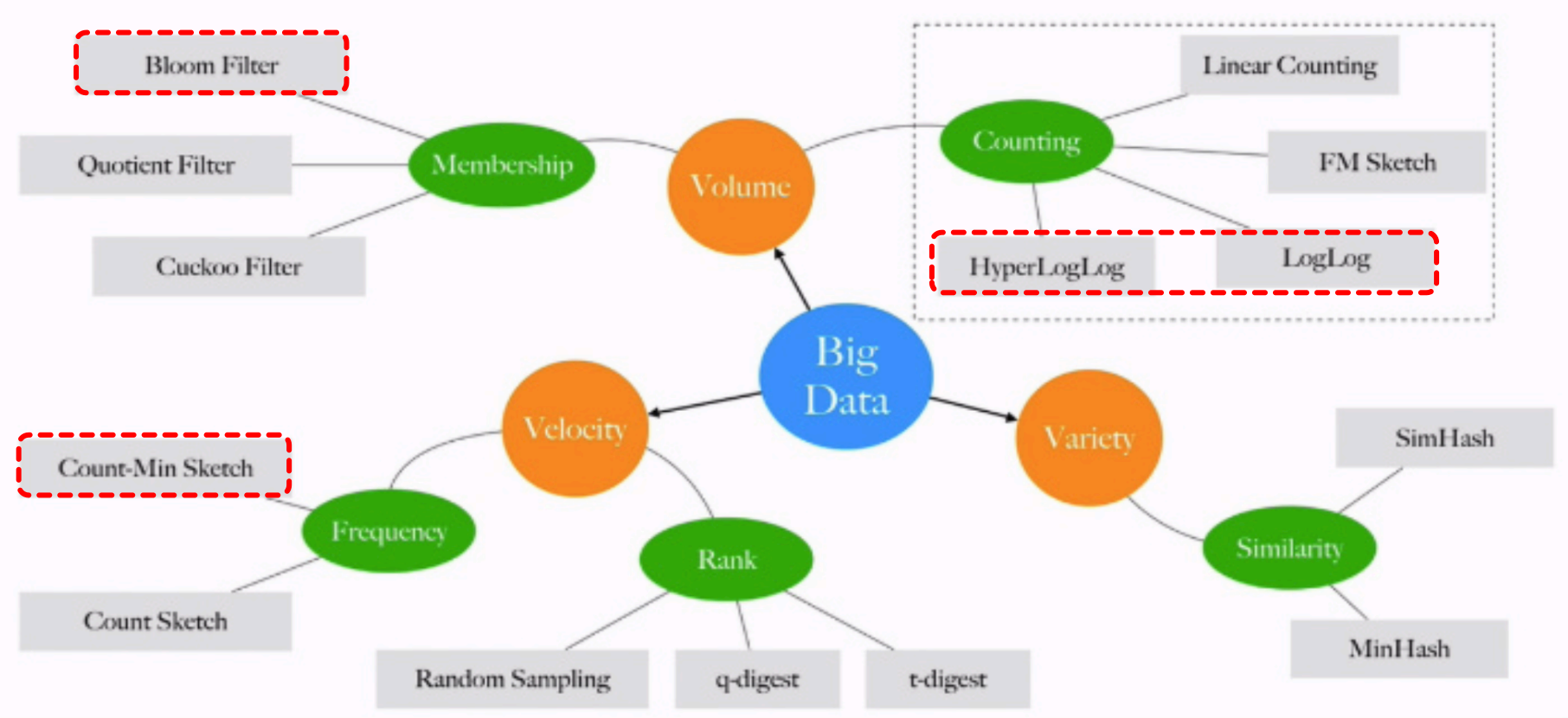


MOTIVAÇÃO

ESTRUTURAS DE DADOS PROBABILISTICAS

- Principal Premissa:
  - *"Como desenvolvedor, aceito que haja algum nível de imprecisão"*

ESTRUTURAS DE DADOS PROBABILISTICAS



Fonte: https://www.kdnuggets.com/2019/08/count-big-data-probabilistic-data-structures-algorithms.html

BLOOM FILTERS

"SPACE/TIME TRADE-OFFS IN HASH CODING WITH ALLOWABLE ERRORS.", BURTON H. BLOOM. 1970

BLOOM FILTERS

- Bloom Filters dizem respeito a presença ou não um element em um conjunto
  - O endereço IP x.x.x.x acessou a minha página?

BLOOM FILTERS

```
mySet = set()
mySet.add('192.168.10.1')
mySet.add('10.10.1.5')
mySet.add('192.168.10.20')
'192.168.10.1' in mySet  #True
'192.168.10.2' in mySet  #False
```

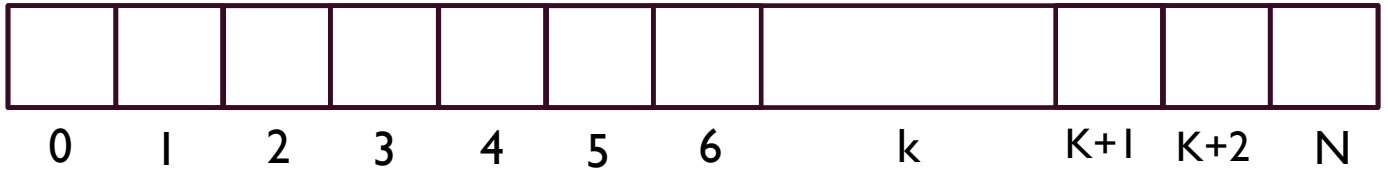
BLOOM FILTERS

# Itens	Uso Memória (MB)*
1	< 1
10	< 1
100	< 1
1.000	< 1
10.000	< 1
100.000	4
1.000.000	33

* Memória dos objetos obtidos por `sys.getsizeof`

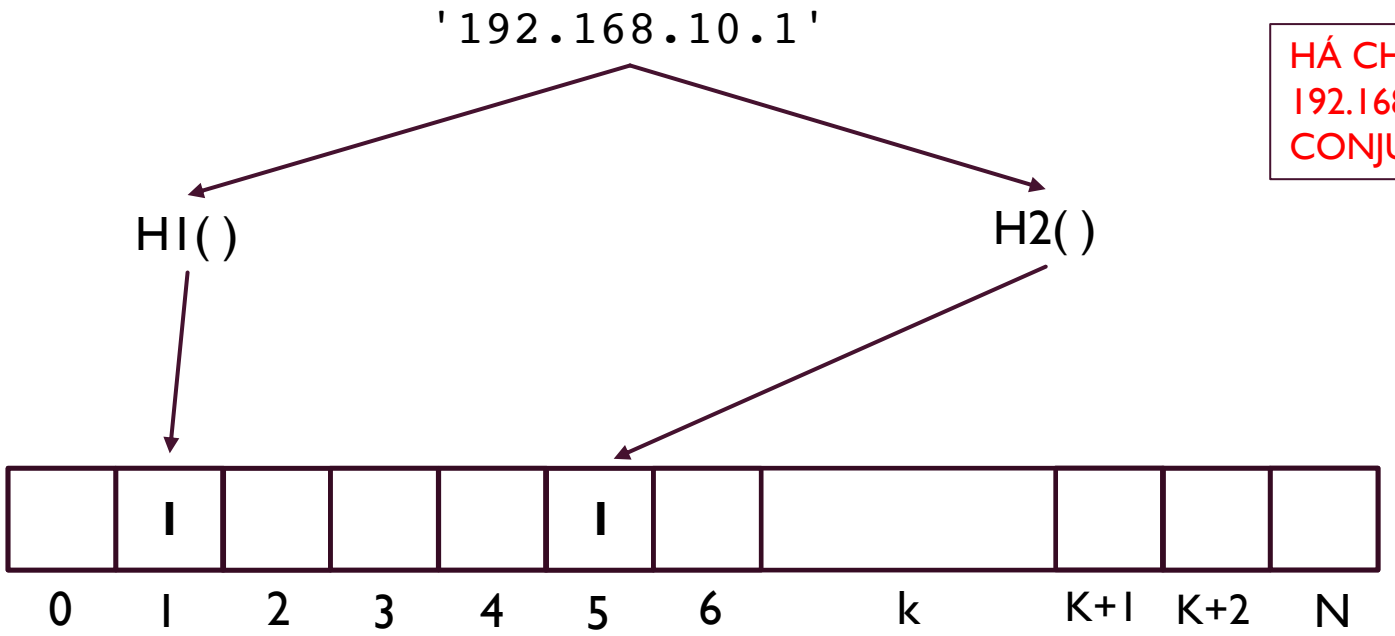
BLOOM FILTERS

- Array de Tamanho N



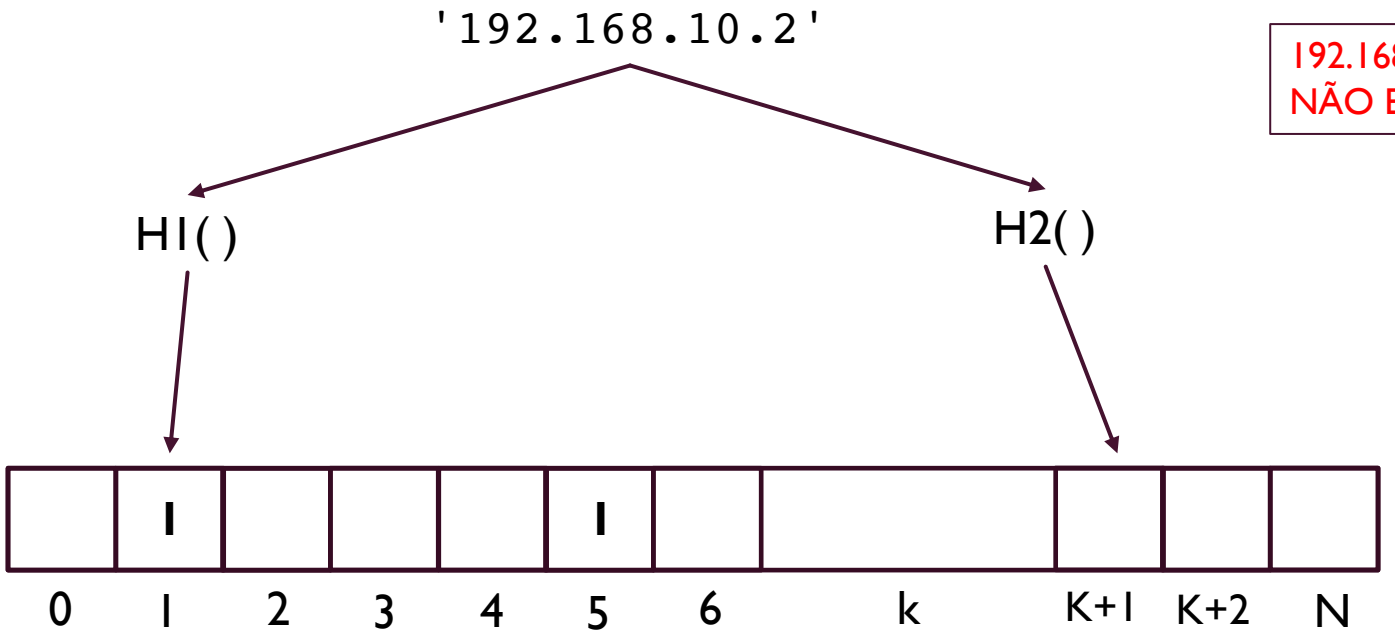
- K funções Hash
  - H1( ) e H2 ( )

BLOOM FILTERS



HÁ CHANCES DE QUE 192.168.10.1 ESTEJA NO CONJUNTO

BLOOM FILTERS



192.168.10.2 COM CERTEZA
NÃO ESTÁ NO CONJUNTO

BLOOM FILTERS

```
from pybloom import BloomFilter
f = BloomFilter(capacity=1000)
f.add('192.168.1.1')
f.add('192.168.1.1')
f.add('10.10.1.5')
'10.10.1.5' in f  #True
'192.168.1.100' in f #False
```

GitHub Link

BLOOM FILTERS

# Itens	Uso Memória (MB)*
1	< 1
10	< 1
100	< 1
1.000	< 1
10.000	< 1
100.000	< 1
1.000.000	< 1

* Memória dos objetos obtidos por `sys.getsizeof`

BLOOM FILTERS

- Casos de Uso
  - Evitar Lookups (Cassandra e Hbase)
  - Match em Tempo Real
  - Outros…

COUNT-MIN SKETCH

"*APPROXIMATING DATA WITH THE COUNT-MIN DATA STRUCTURE.*", CORMODE, GRAHAM, AND S. MUTHUKRISHNAN. 2012

COUNT-MIN SKETCH

- Diz respeito a contagem de elementos
  - Quantas vezes o endereço IP x.x.x.x acessou a minha página?

COUNT-MIN SKETCH

```
myDict = dict()
ipList = ['192.168.1.10', '192.168.1.11', '10.10.1.5', '192.168.1.10', '192.168.1.10']
for ip in ipList:
    if myDict.get(ip):
        myDict[ip] += 1
    else:
        myDict[ip] = 1
myDict
Out: {'192.168.1.10': 3, '192.168.1.11': 1, '10.10.1.5': 1}
```

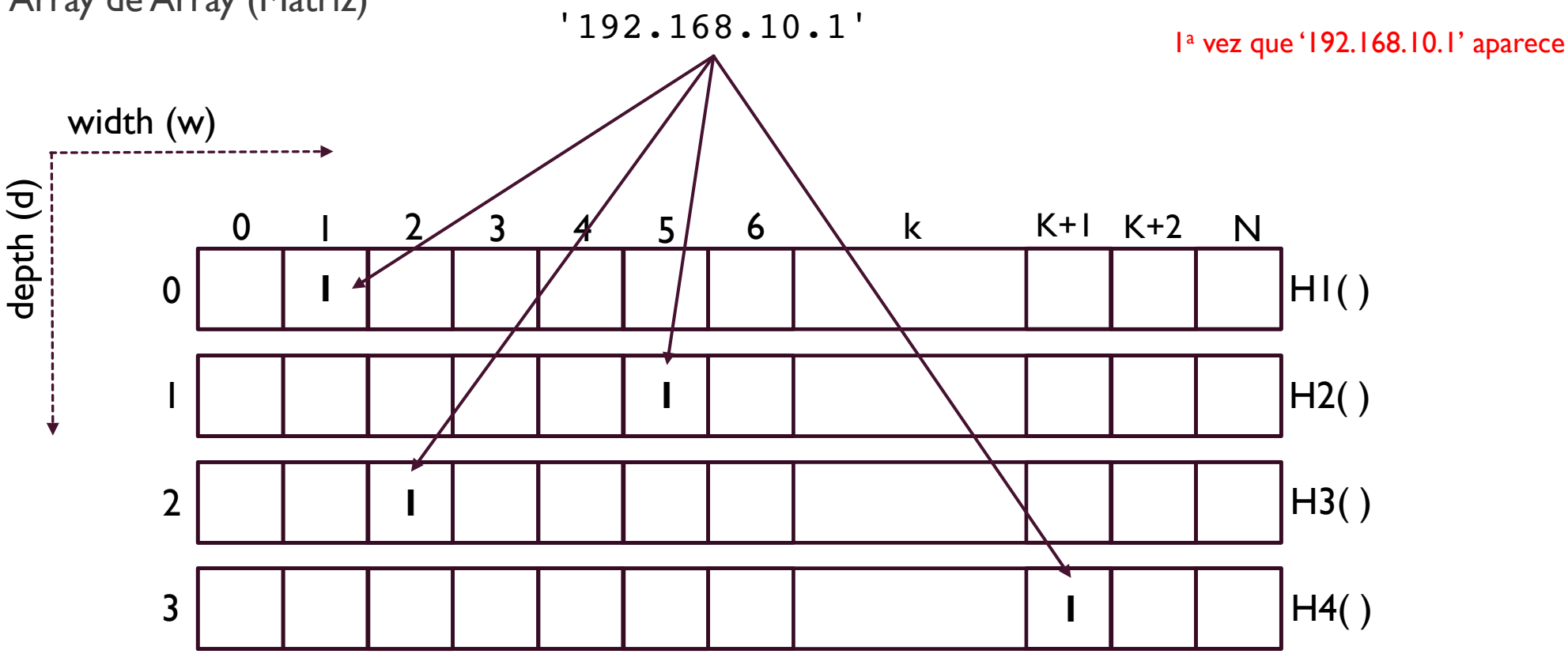
COUNT-MIN SKETCH

# Itens	Uso Memória (MB)*
1	< 1
10	< 1
100	< 1
1.000	< 1
10.000	< 1
100.000	5
1.000.000	41

* Memória dos objetos obtidos por `sys.getsizeof`

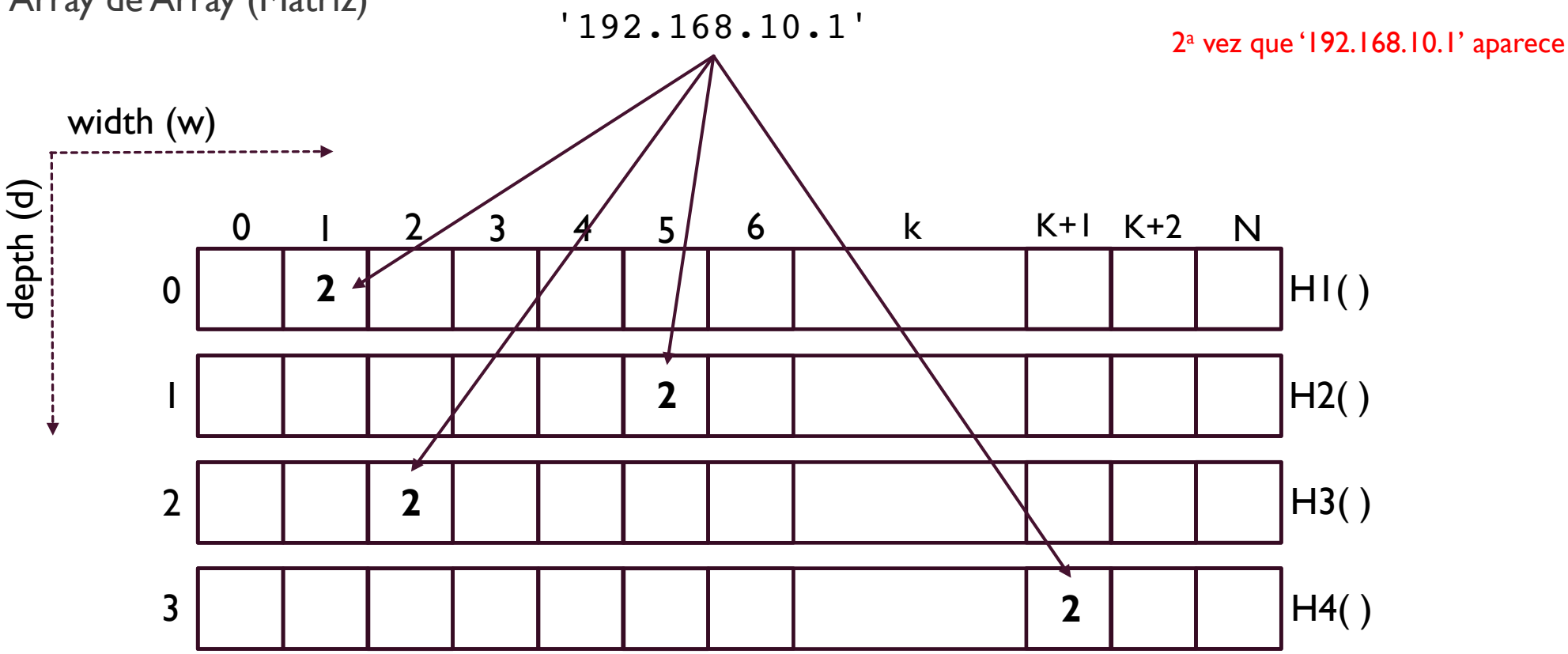
COUNT-MIN SKETCH

- Array de Array (Matriz)



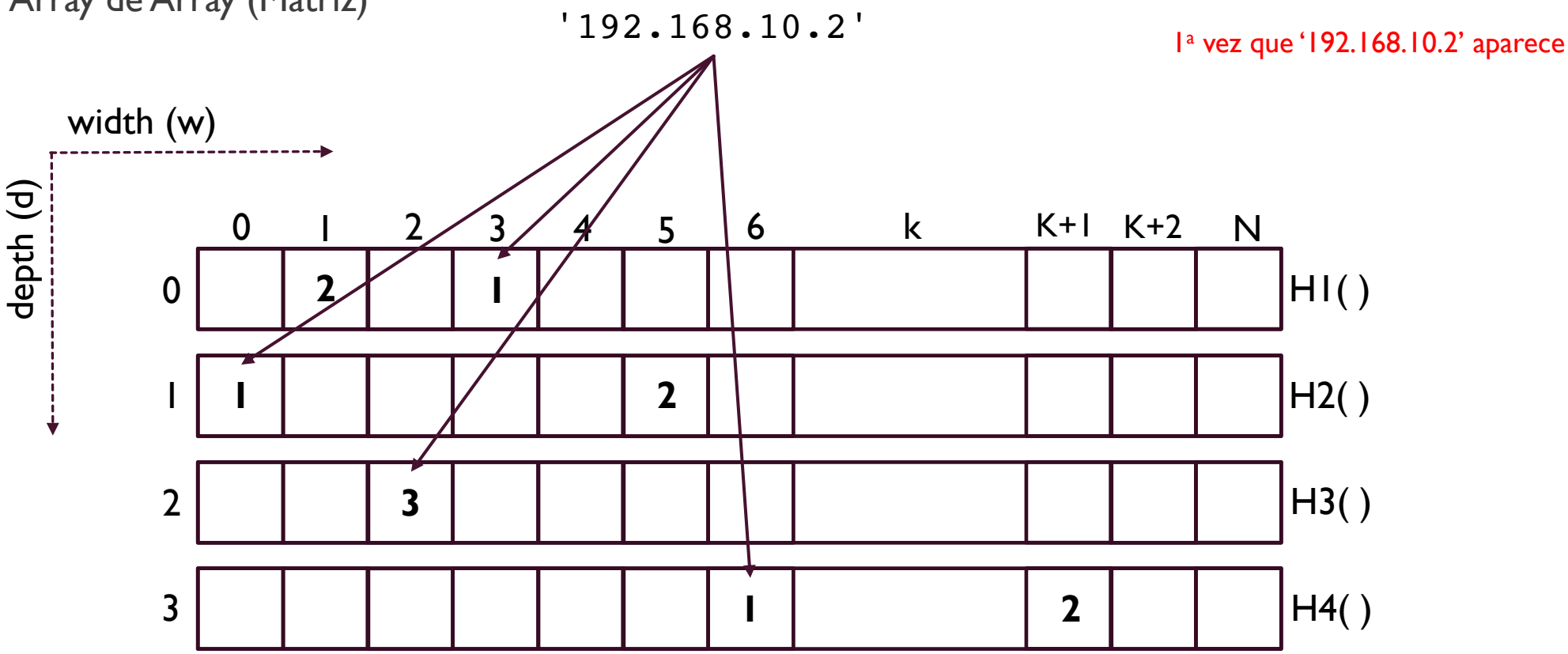
COUNT-MIN SKETCH

- Array de Array (Matriz)



COUNT-MIN SKETCH

- Array de Array (Matriz)



COUNT-MIN SKETCH

- Como saber a quantidade de ocorrências de cada um dos elementos:
  - 192.168.10.1: `min(2,2,3,2) = 2`
  - 192.168.10.2: `min(1,1,3,1) = 1`

COUNT-MIN SKETCH

```
from bounter import bounter, CountMinSketch
counts = CountMinSketch(width=8388608, depth=8)
counts.update(['192.168.1.1', '10.10.1.5', '192.168.1.1'])
counts['192.168.1.1']  #2
counts['192.168.1.10'] #0
```

GitHub Link

COUNT-MIN SKETCH

# Itens	Uso Memória (MB)*
1	< 1
10	< 1
100	< 1
1.000	< 1
10.000	< 1
100.000	< 1
1.000.000	< 1

* Memória dos objetos obtidos por `sys.getsizeof`

COUNT-MIN SKETCH

- Casos de Uso
  - Casos de contagem de frequencia
  - Processamento de Linguagem Natural (NLP)
  - Outros…

HYPER LOGLOG

"LOGLOG COUNTING OF LARGE CARDINALITIES", DI BATTISTA G., ZWICK U. 2003

HYPER LOGLOG

- Diz respeito a cardinalidade dos elementos
  - Quantos *acessos únicos* foram feitos no meu site?

HYPER LOGLOG

```
myDict = dict()
ipList = ['192.168.1.10', '192.168.1.11', '10.10.1.5', '192.168.1.10', '192.168.1.10']
for ip in ipList:
    if myDict.get(ip):
        myDict[ip] += 1
    else:
        myDict[ip] = 1
len(myDict) #3
```

HYPER LOGLOG

# Itens	Uso Memória (MB)*
1	< 1
10	< 1
100	< 1
1.000	< 1
10.000	< 1
100.000	5
1.000.000	41

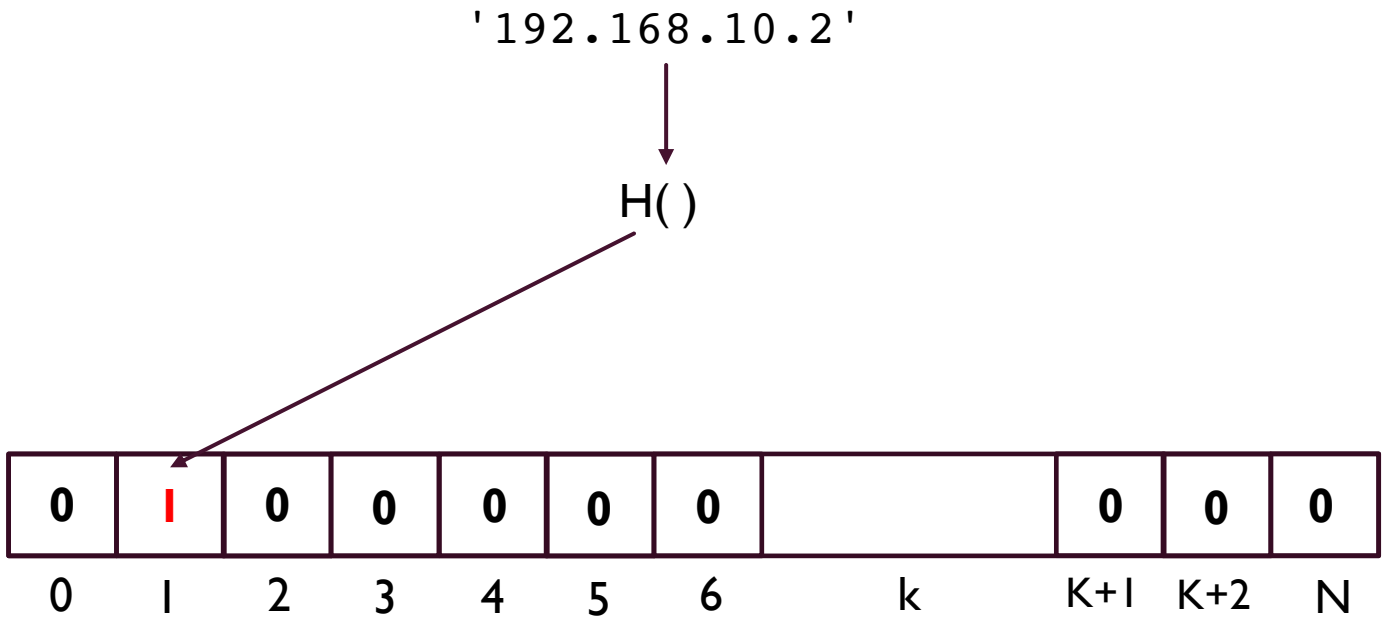
* Memória dos objetos obtidos por `sys.getsizeof`

HYPER LOGLOG

- Linear Counting
  - *"A linear-time probabilistic counting algorithm for database applications."*, Kyu-Young Whang, Brad T. Vander-Zanden, and Howard M. Taylor., 1990

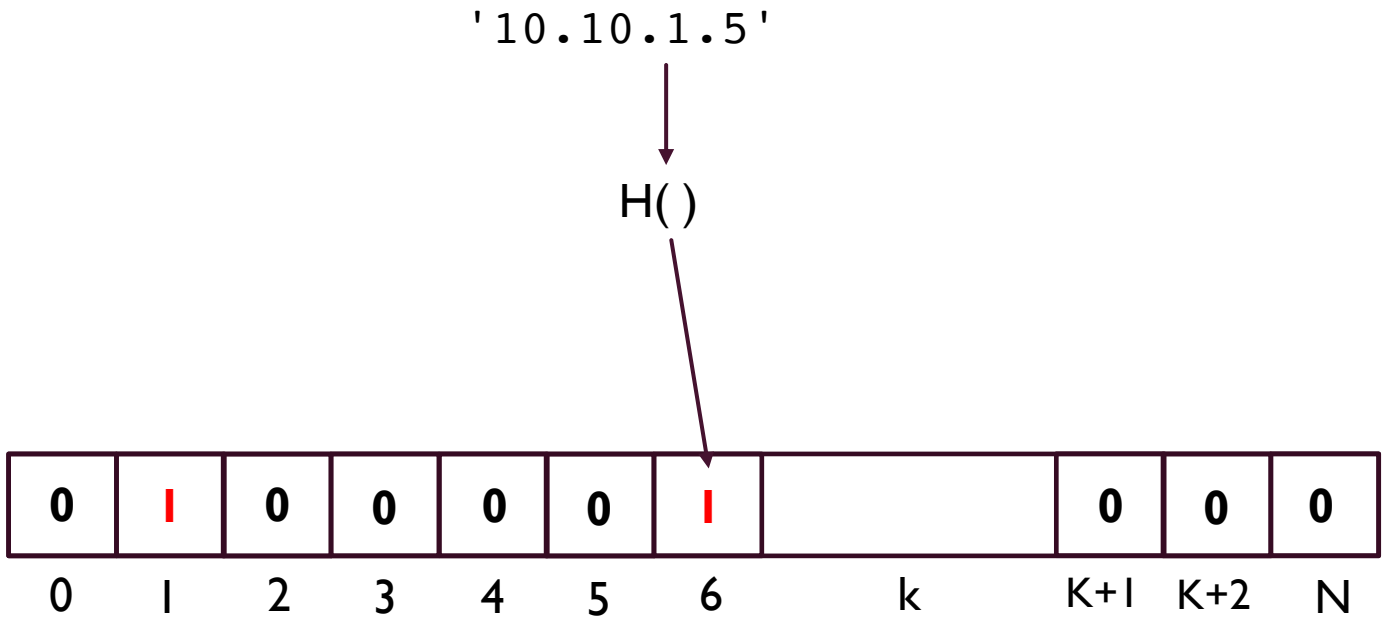
HYPER LOGLOG

- Bit Array de Tamanho N e uma fução Hash `H()`



HYPER LOGLOG

- Bit Array de Tamanho N e uma fução Hash `H()`



HYPER LOGLOG

- Estimativa da Cardinalidade
- `Cardinalidade = -m * ln(m-w/m)`
  - m: tamanho do array
  - w: quantidade de 1's no array

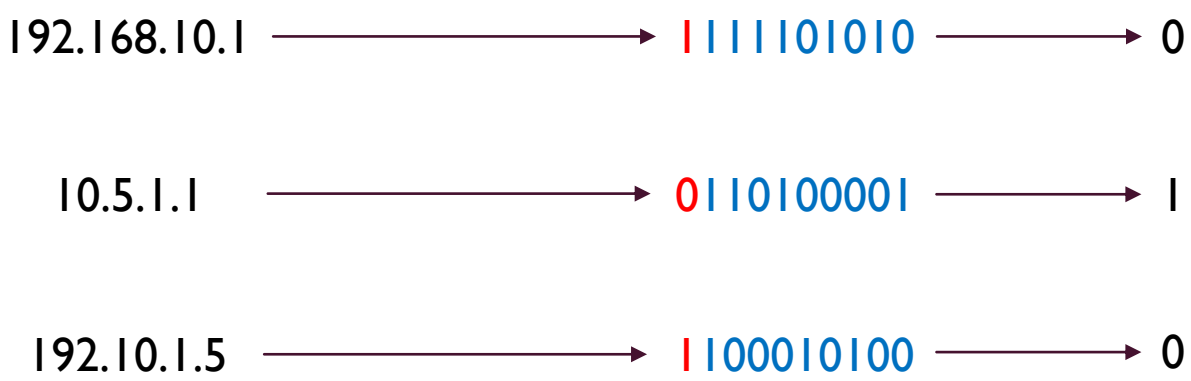
  - Ex. Array com 10 posições e duas posições em 1.
  - Cardinalidade = -10 * ln(10-2/10) = 2.2

HYPER LOGLOG

- LogLog
  - *"LogLog counting of large cardinalities"*, Durand, Flajolet (2003)
  - Lançamento de Moedas
    - 50% de chances de ser cara
    - 50% de chances de ser coroa
    - Após X lançamentos, obtive 10 caras. Consegue descobrir X?

HYPER LOGLOG

- Como fazer isso com Hash?

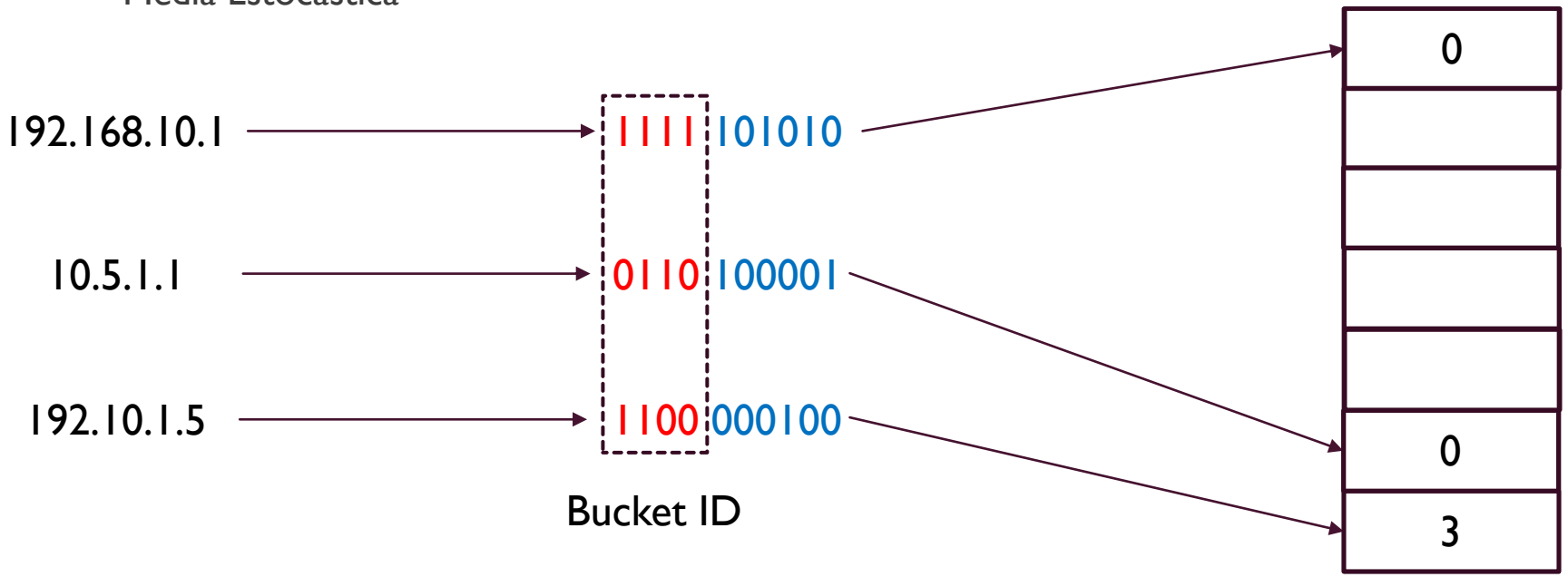


$2^n = 2^1 = 2$

(n = qtde de 1's)

HYPER LOGLOG

- Como melhorar?
  - Média Estocástica



HYPER LOGLOG

- $2^{(\sum(\text{máximo de zeros}) / \text{buckets})} * \text{buckets} * \text{Fator\_de\_Estimativa}$

HYPER LOGLOG

```
import hyperloglog
hll = hyperloglog.HyperLogLog(0.03)
hll.add('192.168.10.1')
len(hll) #1
hll.add('192.168.5.1')
len(hll) #2
hll.add('192.168.5.1')
len(hll) #2
hll.add('10.10.1.5')
len(hll) #3
```

GitHub Link

HYPER LOGLOG

# Itens	Uso Memória (MB)*
1	< 1
10	< 1
100	< 1
1.000	< 1
10.000	< 1
100.000	< 1
1.000.000	< 1

* Memória dos objetos obtidos por `sys.getsizeof`

HYPER LOGLOG

- Casos de Uso
  - Qualquer situação que precise descobrir a cardinalidade em O(n)
  - Visitas únicas em um site
  - Stream de dados
  - Estimativas em tabelas de dados massivas
  - Outros…

- Obrigado!
  - abaruchi.dev