MATA54 Estrutura de Dados e Algoritmos II - 2024.2

Count-Min Sketch

Alunos
Gustavo de Oliveira Ferreira

CONTEXTO

Tenho uma aplicação que está recebendo uma **stream** de eventos e precisamos contar fazer algumas análises

- Quais hashtags estão sendo mais usadas em uma rede social em um certo momento?
- Quais dados devo dar preferência para fazer cache?

Abordagem Simples

Hash Tables?
Sempre que tiver a ocorrência de um evento posso incrementar o valor dele na tabela

Abordagem Simples

Hash Tables?
Sempre que tiver a ocorrência de um evento posso incrementar o valor dele na tabela

Árvores Binárias de Busca?
Armazenar as chaves, com os valores de frequência maneira ordenada

Abordagem Simples

Mas e se estiver analisando uma quantidade massiva de dados?

Hash Tables
Complexidade espacial O(n)

Árvores Binárias de Busca?
Complexidade espacial O(n)

Abordagem Simples

Mas e se estiver analisando uma quantidade massiva de dados?

Hash Tables
Complexidade espacial O(n)

Árvores Binárias de Busca?
Complexidade espacial O(n)

Complexidade espacial linear

Abordagem Simples

Mas e se estiver analisando uma quantidade massiva de dados?

Hash Tables
Complexidade espacial O(n)

Árvores Binárias de Busca?
Complexidade espacial O(n)

Mas e se o espaço disponível for restrito?

MOTIVAÇÃO

Motivação

Analisar stream de eventos **potencialmente** infinitos

Quantidade massiva de dados

Limitação de espaço para guardar tais dados

Análise rápida

Motivação

Analisar stream de eventos **potencialmente** infinitos

Quantidade massiva de dados

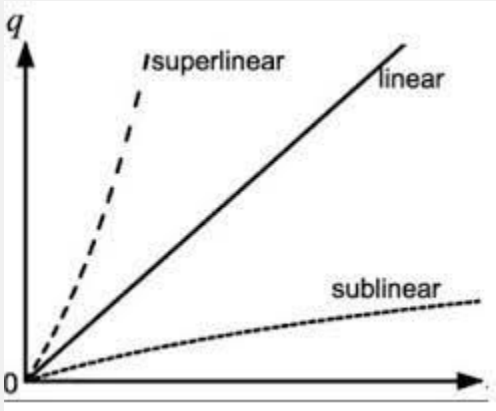
Limitação de espaço para guardar tais dados

Análise rápida

Precisamos de um estrutura de dados que possua espaço sub-linear.

Motivação

Estrutura de dados que possua **espaço sub-linear.**



COUNT-MIN SKETCH

Count-min Sketch

Algoritmo Probabilístico para **estimativa** de valores

Pode ser usado para:

- Calcular frequência simples de um evento
- Identificar elementos frequentes
- Computar quantidades

Count-min Sketch

Realiza operações usando uma matriz de contadores e uma série de funções hash.

hash 1	Mtr 11	Mtr 12	Mtr 13	…	Mtr 1M
hash 1	Mtr 21	Mtr 22	Mtr 23	…	Mtr 2M
…	…	…	…	…	…
hash n	Mtr M1	Mtr M2	Mtr M3	…	Mtr MN

Tabela CMS

	0	1	2	3	4	5	6
H1	0	0	0	0	0	0	0
H2	0	0	0	0	0	0	0
H3	0	0	0	0	0	0	0
H4	0	0	0	0	0	0	0

Valores

Stream: [A, B, K, A, A, K, S...]

H1(A) = 1, H2(A) = 6, H3(A) = 3, H4(A) = 1

H1(B) = 1, H2(B) = 2, H3(B) = 4, H4(B) = 6

H1(K) = 3, H2(K) = 4, H3(K) = 1, H4(K) = 6

H1(S) = 6, H2(S) = 2, H3(S) = 4, H4(S) = 1

Tabela CMS

	0	1	2	3	4	5	6
H1	0	**1**	0	0	0	0	0
H2	0	0	0	0	0	0	0
H3	0	0	0	0	0	0	0
H4	0	0	0	0	0	0	0

Valores

Stream: [**A**, B, K, A, A, K, S...]

H1(A) = 1, H2(A) = 6, H3(A) = 3, H4(A) = 1

H1(B) = 1, H2(B) = 2, H3(B) = 4, H4(B) = 6

H1(K) = 3, H2(K) = 4, H3(K) = 1, H4(K) = 6

H1(S) = 6, H2(S) = 2, H3(S) = 4, H4(S) = 1

Tabela CMS

	0	1	2	3	4	5	6
H1	0	1	0	0	0	0	0
H2	0	0	0	0	0	0	**1**
H3	0	0	0	0	0	0	0
H4	0	0	0	0	0	0	0

Valores

Stream: [**A**, B, K, A, A, K, S...]

H1(A) = 1, **H2(A) = 6**, H3(A) = 3, H4(A) = 1

H1(B) = 1, H2(B) = 2, H3(B) = 4, H4(B) = 6

H1(K) = 3, H2(K) = 4, H3(K) = 1, H4(K) = 6

H1(S) = 6, H2(S) = 2, H3(S) = 4, H4(S) = 1

Tabela CMS

	0	1	2	3	4	5	6
H1	0	1	0	0	0	0	0
H2	0	0	0	0	0	0	1
H3	0	0	0	**1**	0	0	0
H4	0	0	0	0	0	0	0

Valores

Stream: [**A**, B, K, A, A, K, S...]

H1(A) = 1, H2(A) = 6, **H3(A) = 3**, H4(A) = 1

H1(B) = 1, H2(B) = 2, H3(B) = 4, H4(B) = 6

H1(K) = 3, H2(K) = 4, H3(K) = 1, H4(K) = 6

H1(S) = 6, H2(S) = 2, H3(S) = 4, H4(S) = 1

Tabela CMS

	0	1	2	3	4	5	6
H1	0	1	0	0	0	0	0
H2	0	0	0	0	0	0	1
H3	0	0	0	1	0	0	0
H4	0	**1**	0	0	0	0	0

Valores

Stream: [**A**, B, K, A, A, K, S...]

H1(A) = 1, H2(A) = 6, H3(A) = 3, **H4(A) = 1**

H1(B) = 1, H2(B) = 2, H3(B) = 4, H4(B) = 6

H1(K) = 3, H2(K) = 4, H3(K) = 1, H4(K) = 6

H1(S) = 6, H2(S) = 2, H3(S) = 4, H4(S) = 1

Tabela CMS

	0	1	2	3	4	5	6
H1	0	**2**	0	0	0	0	0
H2	0	0	0	0	0	0	1
H3	0	0	0	1	0	0	0
H4	0	1	0	0	0	0	0

Valores

Stream: [A, **B**, K, A, A, K, S...]

H1(A) = 1, H2(A) = 6, H3(A) = 3, H4(A) = 1

H1(B) = 1, H2(B) = 2, H3(B) = 4, H4(B) = 6

H1(K) = 3, H2(K) = 4, H3(K) = 1, H4(K) = 6

H1(S) = 6, H2(S) = 2, H3(S) = 4, H4(S) = 1

Tabela CMS

	0	1	2	3	4	5	6
H1	0	2	0	0	0	0	0
H2	0	0	**1**	0	0	0	1
H3	0	0	0	1	0	0	0
H4	0	1	0	0	0	0	0

Valores

Stream: [A, **B**, K, A, A, K, S...]

H1(A) = 1, H2(A) = 6, H3(A) = 3, H4(A) = 1

H1(B) = 1, **H2(B) = 2**, H3(B) = 4, H4(B) = 6

H1(K) = 3, H2(K) = 4, H3(K) = 1, H4(K) = 6

H1(S) = 6, H2(S) = 2, H3(S) = 4, H4(S) = 1

Tabela CMS

	0	1	2	3	4	5	6
H1	0	2	0	0	0	0	0
H2	0	0	1	0	0	0	1
H3	0	0	0	1	**1**	0	0
H4	0	1	0	0	0	0	0

Valores

Stream: [A, **B**, K, A, A, K, S...]

H1(A) = 1, H2(A) = 6, H3(A) = 3, H4(A) = 1

H1(B) = 1, H2(B) = 2, **H3(B) = 4**, H4(B) = 6

H1(K) = 3, H2(K) = 4, H3(K) = 1, H4(K) = 6

H1(S) = 6, H2(S) = 2, H3(S) = 4, H4(S) = 1

Tabela CMS

	0	1	2	3	4	5	6
H1	0	2	0	0	0	0	0
H2	0	0	1	0	0	0	1
H3	0	0	0	1	1	0	0
H4	0	1	0	0	0	0	**1**

Valores

Stream: [A, **B**, K, A, A, K, S...]

H1(A) = 1, H2(A) = 6, H3(A) = 3, H4(A) = 1

H1(B) = 1, H2(B) = 2, H3(B) = 4, **H4(B) = 6**

H1(K) = 3, H2(K) = 4, H3(K) = 1, H4(K) = 6

H1(S) = 6, H2(S) = 2, H3(S) = 4, H4(S) = 1

Tabela CMS

	0	1	2	3	4	5	6
H1	0	2	0	**1**	0	0	0
H2	0	0	1	0	**1**	0	1
H3	0	**1**	0	1	1	0	0
H4	0	1	0	0	0	0	**2**

Valores

Stream: [A, B, **K**, A, A, K, S...]

H1(A) = 1, H2(A) = 6, H3(A) = 3, H4(A) = 1

H1(B) = 1, H2(A) = 2, H3(A) = 4, H4(A) = 6

H1(K) = 3, H2(K) = 4, H3(K) = 1, H4(K) = 6

H1(S) = 6, H2(S) = 2, H3(S) = 4, H4(S) = 1

Tabela CMS

	0	1	2	3	4	5	6
H1	0	**3**	0	1	0	0	0
H2	0	0	1	0	1	0	**2**
H3	0	1	0	**2**	1	0	0
H4	0	**2**	0	0	0	0	2

Valores

Stream: [A, B, K, **A**, A, K, S...]

H1(A) = 1, H2(A) = 6, H3(A) = 3, H4(A) = 1

H1(B) = 1, H2(B) = 2, H3(B) = 4, H4(B) = 6

H1(K) = 3, H2(K) = 4, H3(K) = 1, H4(K) = 6

H1(S) = 6, H2(S) = 2, H3(S) = 4, H4(S) = 1

Tabela CMS

	0	1	2	3	4	5	6
H1	0	**4**	0	1	0	0	0
H2	0	0	1	0	1	0	**3**
H3	0	1	0	**3**	1	0	0
H4	0	**3**	0	0	0	0	2

Valores

Stream: [A, B, K, A, **A**, K, S...]

H1(A) = 1, H2(A) = 6, H3(A) = 3, H4(A) = 1

H1(B) = 1, H2(B) = 2, H3(B) = 4, H4(B) = 6

H1(K) = 3, H2(K) = 4, H3(K) = 1, H4(K) = 6

H1(S) = 6, H2(S) = 2, H3(S) = 4, H4(S) = 1

Tabela CMS

	0	1	2	3	4	5	6
H1	0	4	0	**2**	0	0	0
H2	0	0	1	0	**2**	0	3
H3	0	**2**	0	3	1	0	0
H4	0	3	0	0	0	0	**3**

Valores

Stream: [A, B, K, A, A, **K**, S...]

H1(A) = 1, H2(A) = 6, H3(A) = 3, H4(A) = 1

H1(B) = 1, H2(B) = 2, H3(B) = 4, H4(B) = 6

H1(K) = 3, H2(K) = 4, H3(K) = 1, H4(K) = 6

H1(S) = 6, H2(S) = 2, H3(S) = 4, H4(S) = 1

Tabela CMS

	0	1	2	3	4	5	6
H1	0	4	0	2	0	0	**1**
H2	0	0	**2**	0	2	0	3
H3	0	2	0	3	**2**	0	0
H4	0	**4**	0	0	0	0	3

Valores

Stream: [A, B, K, A, A, K, **S**...]

H1(A) = 1, H2(A) = 6, H3(A) = 3, H4(A) = 1

H1(B) = 1, H2(B) = 2, H3(B) = 4, H4(B) = 6

H1(K) = 3, H2(K) = 4, H3(K) = 1, H4(K) = 6

H1(S) = 6, H2(S) = 2, H3(S) = 4, H4(S) = 1

Tabela CMS

	0	1	2	3	4	5	6
H1	0	**4**	0	2	0	0	1
H2	0	0	2	0	2	0	**3**
H3	0	2	0	**3**	2	0	0
H4	0	**4**	0	0	0	0	3

Quantas vezes **A** foi recebido no stream?

H1(A) = 1, H2(A) = 6, H3(A) = 3, H4(A) = 1

min [4, **3, 3**, 4] => 3 vezes

Valores

Stream: [A, B, K, A, A, K, S...]

H1(A) = 1, H2(A) = 6, H3(A) = 3, H4(A) = 1

H1(B) = 1, H2(B) = 2, H3(B) = 4, H4(B) = 6

H1(K) = 3, H2(K) = 4, H3(K) = 1, H4(K) = 6

H1(S) = 6, H2(S) = 2, H3(S) = 4, H4(S) = 1

Tabela CMS

	0	1	2	3	4	5	6
H1	0	4	0	2	0	0	**1**
H2	0	0	**2**	0	2	0	3
H3	0	2	0	3	**2**	0	0
H4	0	**4**	0	0	0	0	3

Quantas vezes **S** foi recebido no stream?

H1(S) = 6, H2(S) = 2, H3(S) = 4, H4(S) = 1

min [**1**, 2, 2, 4] => 1 vez

Valores

Stream: [A, B, K, A, A, K, S...]

H1(A) = 1, H2(A) = 6, H3(A) = 3, H4(A) = 1

H1(B) = 1, H2(B) = 2, H3(B) = 4, H4(B) = 6

H1(K) = 3, H2(K) = 4, H3(K) = 1, H4(K) = 6

H1(S) = 6, H2(S) = 2, H3(S) = 4, H4(S) = 1

Tabela CMS - Colisão

	0	1	2	3	4	5	6
H1	0	4	0	2	0	0	**2**
H2	**1**	0	2	0	2	0	3
H3	**1**	2	0	3	2	0	0
H4	0	4	**1**	0	0	0	3

Mas e se nós adicionarmos um novo valor a stream, que a função hash incremente a posição do menor valor de outro elemento?

H1(S) = 6, H2(S) = 2, H3(S) = 4, H4(S) = 1

min [**2**, 2, 2, 4] => No máximo 2 vezes

Adicionando E ao stream

Stream: [A, B, K, A, A, K, S, **E**...]

H1(A) = 1, H2(A) = 6, H3(A) = 3, H4(A) = 1

H1(B) = 1, H2(B) = 2, H3(B) = 4, H4(B) = 6

H1(K) = 3, H2(K) = 4, H3(K) = 1, H4(K) = 6

H1(S) = 6, H2(S) = 2, H3(S) = 4, H4(S) = 1

H1(E) = 6, H2(E) = 0, H3(E) = 0, H4(E) = 2

Count-min Sketch Pseudocódigo

```
1:  procedure CountMinSketch(depth, width)
2:      function add(item)
3:          while counter < depth do
4:              buckets ←getHashBuckets(item,depth,width)
5:              M tr[counter][buckets[counter]] ← +1
6:              counter ← +1
7:          end while
8:      end function
```

Count-min Sketch Pseudocódigo

```
1:   procedure CountMinSketch(depth, width)
9:       function estimateCount(item)
10:          r ← MAXVALUE
11:          while counter < depth do
12:              val ← M tr[counter][buckets[counter]]
13:              r ←Min(r, val)
14:              counter ← +1
15:          end while
16:      return r
17:      end function
```

TEORIA

Probabilidade

O CMS é uma estrutura com parâmetros (ε, δ) representada por uma matriz bi-dimensional.

ε representa o fator de erro na estimativa de frequência (precisão da estrutura de dados)

δ representa a probabilidade de erro (a chance de que a estimativa exceda o erro especificado por ε)

Probabilidade

A matriz é formada por duas listas com largura **w** e profundidade **d**.

Dados os parâmetros (ε, δ), temos que:

Probabilidade

A matriz é formada por duas listas com largura **w** e profundidade **d**.

Dados os parâmetros (ε, δ), temos que:

largura $w = \lceil e/\varepsilon \rceil$ profundidade $w = \lceil \ln 1/\delta \rceil$

Probabilidade

Dado:

- **m** elementos adicionados
- **â** estimativa de ocorrência de um elemento x
- **a** quantidade real de ocorrência de x

Temos que:

$$Pr(\hat{a} <= a + \varepsilon * m) >= 1 - \delta$$

Probabilidade

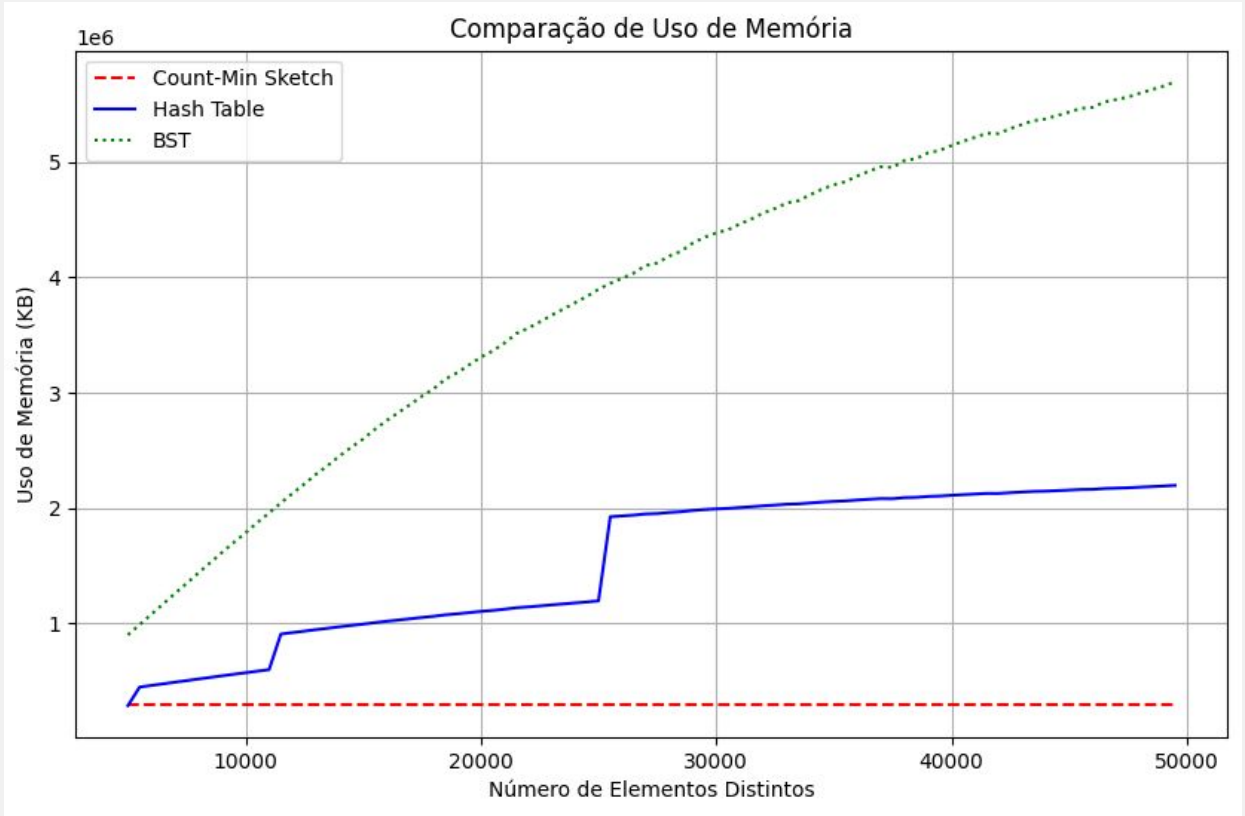
Dado:

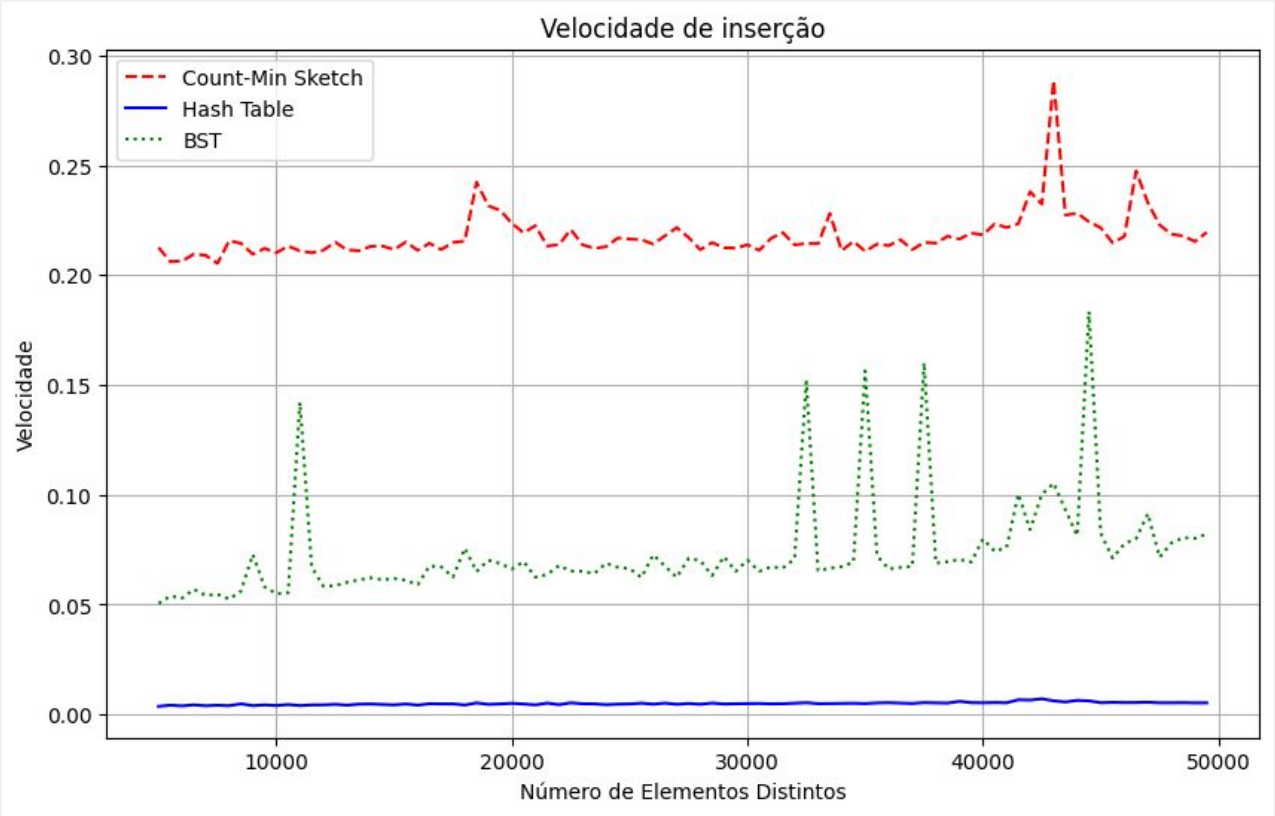
- **m** elementos adicionados
- **â** estimativa de ocorrência de um elemento x
- **a** quantidade real de ocorrência de x

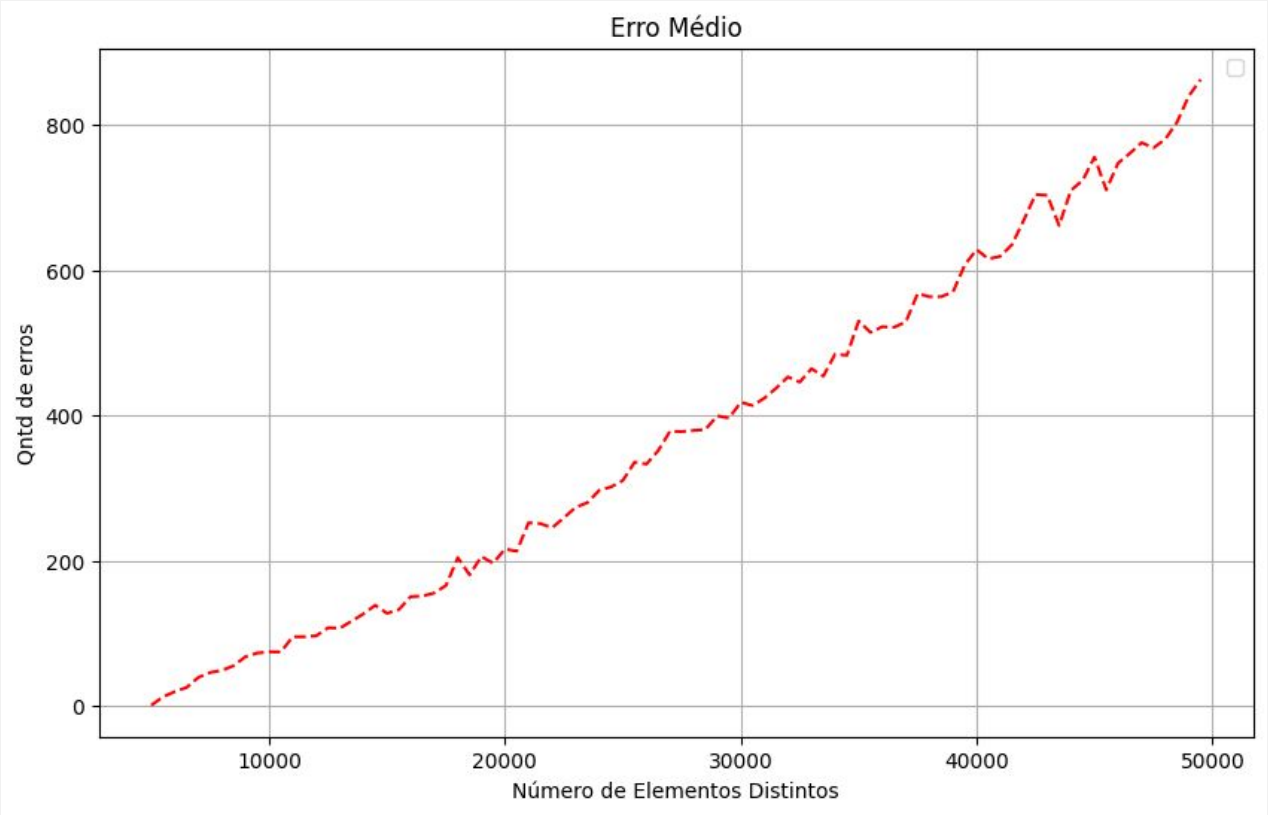
Temos que:

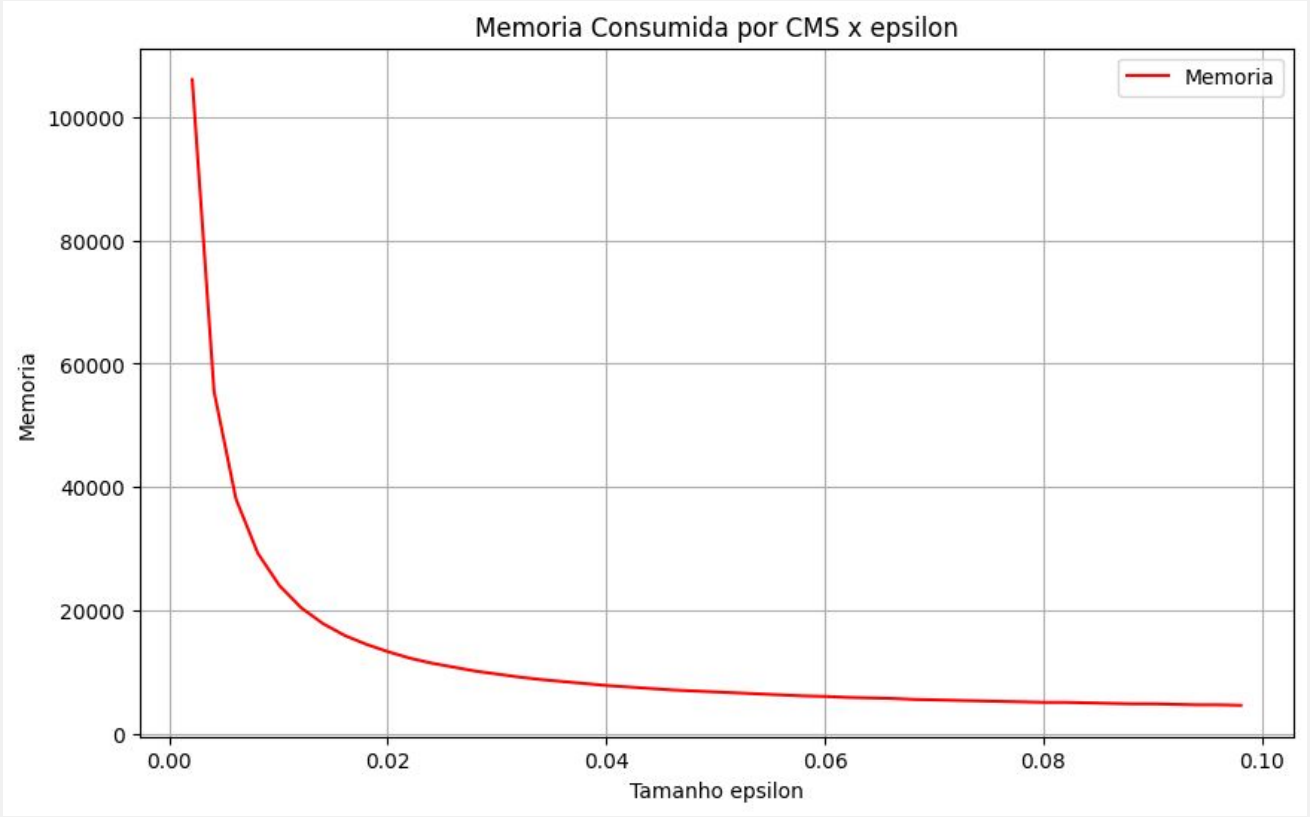
$$Pr(\hat{a} <= a + \varepsilon * m) >= 1 - \delta$$

ANÁLISE PRÁTICA


Comparação de Uso de Memória
1e6
Count-Min Sketch
Hash Table
BST
Uso de Memória (KB)
1
2
3
4
5
10000
20000
30000
40000
50000
Número de Elementos Distintos


Velocidade de inserção
Count-Min Sketch
Hash Table
BST
Velocidade
0.30
0.25
0.20
0.15
0.10
0.05
0.00
10000
20000
30000
40000
50000
Número de Elementos Distintos


Erro Médio
Qntd de erros
0
200
400
600
800
10000
20000
30000
40000
50000
Número de Elementos Distintos


Memoria Consumida por CMS x epsilon
Memoria
100000
80000
60000
40000
20000
0
0.00
0.02
0.04
0.06
0.08
0.10
Tamanho epsilon
Memoria

Referências

CORMODE, G.; MUTHUKRISHNAN, S. **An improved data stream summary: the count-min sketch and its applications**. Journal of Algorithms, v. 55, n. 1, p. 58–75, abr. 2005.

CORMODE, G.; MUTHUKRISHNAN, S. **What's hot and what's not: tracking most frequent items dynamically**. ACM Transactions on Database Systems, v. 30, n. 1, p. 249–278, mar. 2005.

SCHWARZ, K. **Count-Min Sketches**. Disponível em: <https://web.stanford.edu/class/archive/cs/cs166/cs166.1206/lectures/10/Slides10.pdf>. Acesso em: 20 jan. 2025.

Obrigado!