Bacharelado em Ciência da Computação
Universidade de São Paulo
Instituto de Matemática e Estatística

# Estudo do efeito de variações de *Bloom filters* no desempenho de algoritmos de hifenização de palavras

Matheus Barbosa Silva
Departamento de Ciência da Computação, IME/USP
matheus_barbosa137@usp.br

## Resumo

A técnica de *cuckoo hashing* é utilizada pelo *cuckoo filter*, estrutura de dados capaz de responder a testes de membresia. Essa é uma estrutura alternativa aos *Bloom filters* que usualmente apresenta melhor desempenho de consultas e menor consumo de espaço em várias aplicações práticas. Tais vantagens são demonstradas em um algoritmo hifenizador de palavras.

## Principais objetivos

1. Descrever o funcionamento de *Bloom filters* e *Cuckoo filters*;
2. Comparar o desempenho dessas estruturas em algoritmos de hifenização.

## 1 *Bloom filters*

O *Bloom filter*, estrutura de dados aleatorizada concebida por Bloom [1], tem o objetivo de responder a testes de membresia com **eficiência no consumo de espaço**. Essa estrutura permite representar os elementos de um conjunto de forma compacta com o custo de resultados **falsos positivos** – quando a estrutura responde que um elemento é membro do conjunto, quando de fato não é.

Um *Bloom filter* que representa um dado conjunto $S$ funciona a partir de um vetor booleano de $m$ bits (inicialmente zerado) e $k$ funções de *hashing* universais independentes $(h_1, \ldots, h_k)$ utilizadas pelo filtro, $h_i : \mathcal{U} \to [0, m-1], \forall i \in [1, k]$. Para a **inserção** de um dado elemento $x$ na estrutura, todos os bits de índices $h_i(x), \forall i \in [1, k]$ recebem o valor 1. Para verificar se um dado elemento pertence ao conjunto, basta verificar se os $k$ bits indicados pelas funções de *hashing* contêm o valor 1, caso contrário o resultado é negativo – tais processos são ilustrados na Figura 1.

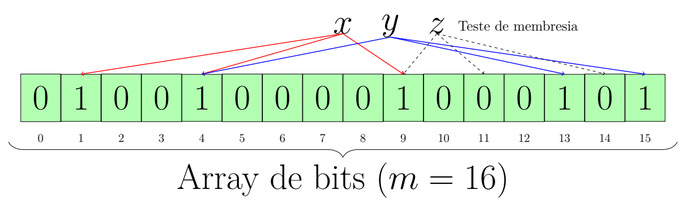


**Figura 1:** Esquema do funcionamento de um *Bloom filter* com parâmetros $m = 16, k = 3, n = 2, S = \{x, y\}$ com teste de membresia do elemento $z$ que não pertence à estrutura

## 2 *Cuckoo filters*

O *cuckoo filter*, descrito por Fan *et al.* [2], é proposto como uma alternativa ao *Bloom filter* tradicional para os cenários em que a **remoção de elementos** da estrutura é necessária. Para isso, utiliza-se o ***cuckoo hashing*** como um modo de construção do filtro. Essa estrutura é composta por um vetor de *buckets* e duas funções de *hashing* $h_1(x)$ e $h_2(x)$ que mapeiam um dado elemento $x$ para dois *buckets* da estrutura.

Para a inserção, caso algum dos dois *buckets* de índices dados pelas funções de *hashing* tenha espaço para um novo elemento, $x$ é inserido no *bucket* disponível. Caso contrário, seleciona-se um dos dois *buckets* e $x$ toma o lugar de um elemento anteriormente inserido. Nesse cenário, o elemento removido é **realocado** (esse processo é ilustrado na Figura 2).

Essa estrutura é especialmente vantajosa com relação aos Bloom filters tradicionais pois permite que elementos sejam removidos da estrutura dinamicamente, tem melhor desempenho em consultas e usualmente consome menos espaço se a razão de falsos positivos $\epsilon$ alvo é menor que 3%.

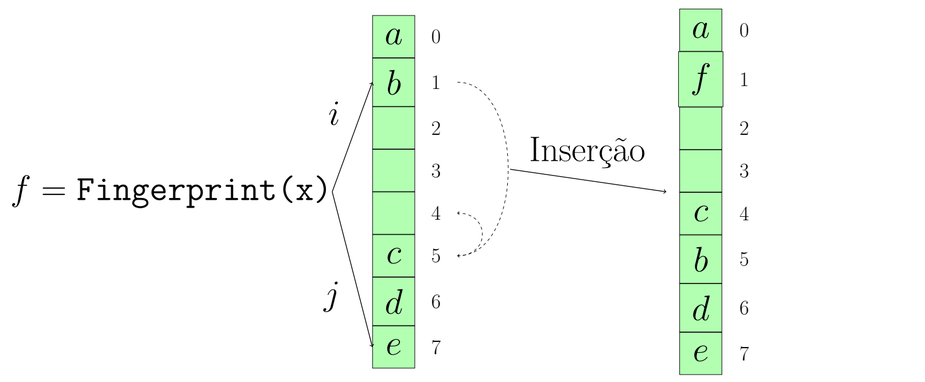


**Figura 2:** Esquema do funcionamento da inserção de um elemento $x$ em um *Cuckoo filter* com a técnica de *partial-key cuckoo hashing*

## 3 Experimentos

De acordo com Bloom [1], a aplicação de *Bloom filters* é vantajosa em algoritmos hifenizadores de palavras. Modifica-se o algoritmo `hypher`, um hifenizador de palavras popular, de modo a criar duas versões: uma que utiliza *Bloom filters* e outra que utiliza *Cuckoo filters*. O tempo médio de cada versão para hifenizar um mesmo dicionário é exibido na Figura 3.

Nota-se que as estruturas apresentam, nos experimentos realizados, tempos médios de hifenização relativamente próximos, ainda que a versão com *cuckoo filter* consuma menos espaço.

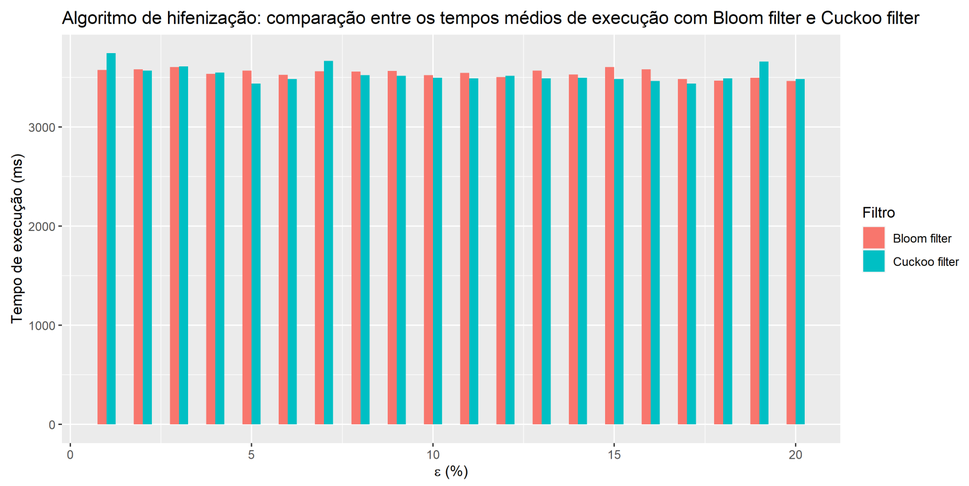


**Figura 3:** Comparação entre os tempos médios de execução de um mesmo algoritmo hifenizador de palavras implementado com *Bloom filter* e com *cuckoo filter*

## 4 Conclusões

- *Cuckoo filters* podem substituir *Bloom filters* em algoritmos hifenizadores de palavras, de modo que o consumo de tempo se mantenha similar para qualquer $\epsilon$, mas com menor consumo de espaço;
- O tempo de construção dessas estruturas é, também, semelhante.

## Referências

[1] Burton H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Commun. ACM*, 13(7):422–426, jul 1970.

[2] Bin Fan, Dave G. Andersen, Michael Kaminsky, and Michael D. Mitzenmacher. Cuckoo filter: Practically better than bloom. In *Proceedings of the 10th ACM International on Conference on Emerging Networking Experiments and Technologies*, CoNEXT '14, page 75–88, New York, NY, USA, 2014. Association for Computing Machinery.