Estrutura de Dados e Algoritmos
Projeto 06: Senha fraca
Ciência da Computação

Prof. Daniel Saad Nogueira Nunes

1 Contextualização

O objeto de estudo desse projeto são os filtros de Bloom aplicados à cibersegurança. Para contextualizá-lo melhor, precisamos de alguns conceitos.

1.1 Filtros de Bloom

Filtros de Blooms são estruturas de dados probabilísticas capazes de responder consultas de pertinência sobre um conjunto S, isto é, elas permitem dizer se, para um dado x:

- $x \notin S$;
- ou se $x \in S$ com *a*lta probabilidade.

Repare que os filtros de Bloom não respondem com 100% de acerto de um elemento x está no conjunto, isto é, ele permite a ocorrência de falsos positivos. Isto acontece por conta dos mecanismos que envolvem o Filtro de Bloom: **funções de hashing**.

Um filtro de Bloom possui os seguintes elementos:

- Um vetor de bits $B[0, m-1]$, de tamanho m, inicializados com 0s;
- e uma coleção de funções de hashing $(f_1, f_2, \ldots, f_k)$.

Para inserir um elemento x em um filtro de Bloom:

- Computamos $y_1 = f_1(x), y_2 = f_2(x), \ldots, y_k = f_k(x)$;
- Atribuímos 1 nas posições, $B[y_1], B[y_2], \ldots B[y_k]$.

Para consultar a pertinência de um elemento x no filtro:

- Computamos $y_1 = f_1(x), y_2 = f_2(x), \ldots, y_k = f_k(x)$;
- Se todos os valores $B[y_1], B[y_2], \ldots, B[y_k]$ são 1, então x **provavelmente** está no conjunto S, caso contrário, x **certamente** não está em S.

Considere a Figura 1.1. Na inserção do elemento x, os bits 1, 5 e 13 são ligados. Em seguida, após o elemento y ser inserido, ligam-se os bits 4, 11 e 16. Por fim, após a inserção do elemento z, os bits 3, 5 e 11 estão ligados. Suponha agora que queremos verificar se w está no conjunto. Após aplicar as funções de hash sobre w, retornam-se as posições 4, 14 e 15, e, como $B[15] = 0$, podemos concluir que **certamente** $w \notin S$.

Contudo, não podemos afirmar que um dado elemento **certamente** está em S utilizando um filtro de Bloom por conta das colisões das funções de hash. Se as funções de hash sobre w na 1.1 retornassem as posições 4, 11 e 16, o filtro, incorretamente, responderia que $w \in S$, um caso de **falso positivo**, oriundo das inserções de x, y e z no filtro.

Quanto maior o filtro de Bloom, menos colisões existem e, consequentemente, menor a ocorrência de falsos positivos.

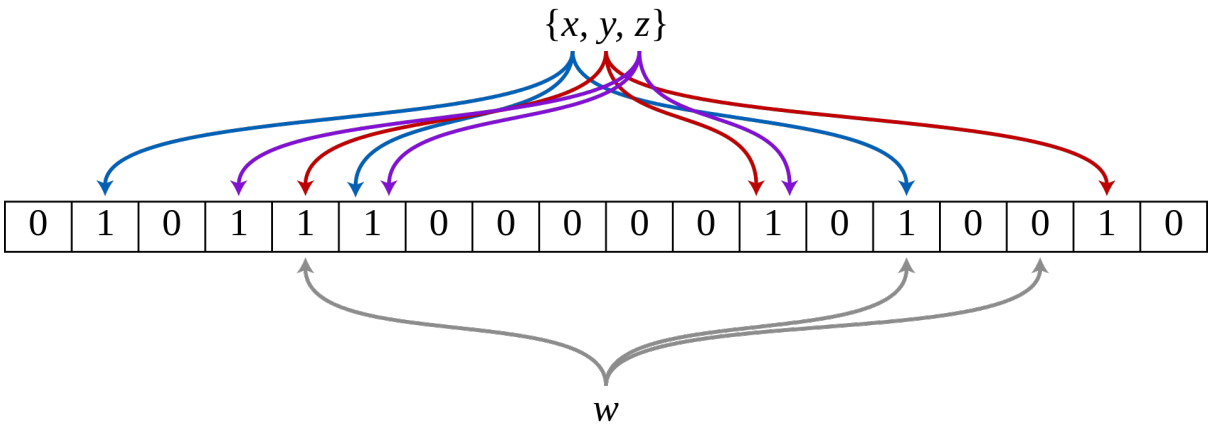

Figura 1: Filtro de Bloom. Após a inserção dos elementos x, y e z no filtro, temos uma negativa ao perguntar para o filtro se w está no conjunto de itens, pois as funções de hashing sobre w retorna uma posição em que o valor é 0. Figura retirada de [Wik22].

1.2 O método de Rabin para obtenção de fingerprints de strings

Fingerprints (impressões digitais) são representações de um objeto grande em um padrão de bits menor. O método de Rabin para obtenção de uma fingerprint sob uma string consiste em transformar uma strings em um inteiro sem sinal de 32 ou 64 bits. Suponha que Σ seja o alfabeto sob a qual a string é composta e tome $\sigma = |\Sigma|$. Por exemplo, se consideramos o alfabeto $\Sigma = \{a, b, c, \ldots, z\}$, temos $\sigma = 26$ e podemos dizer que a palavra $W =$ abracadabra é uma palavra composta por símbolos de Σ. Para obter a fingerperint de uma palavra $W[0, n-1]$ qualquer sobre o alfabeto Σ, aplicamos a seguinte transformação polinomial:

$$h = \sum_{i=0}^{n-1} \left(\sigma^{n-i-1} \cdot W[i] \mod k\right) \mod k$$

em que o parâmetro k é escolhido de modo que h caiba em um inteiro sem sinal.

Para a string $W = abra$, $\Sigma = \{a, \ldots, z\}$ e o parâmetro $k = 50$, temos que:

$$\begin{aligned}
h &= \sum_{i=0}^{3} \left(26^{n-i-1} \cdot W[i] \mod 50\right) \mod 50 \\
h &= (26^3 \cdot 0 \mod 50) + (26^2 \cdot 1 \mod 50) + (26^1 \cdot 17 \mod 50) + (26^0 \cdot 0 \mod 50) \mod 50 \\
h &= (0 + 26 + 42 + 0) \mod 50 \\
h &= 68 \mod 50 \\
h &= 18
\end{aligned}$$

Repare que o caractere a, por ser o primeiro símbolo do alfabeto, possui valor 0, b o valor 1 e r o valor 17.

Para evitar overflows, é importante notar algumas propriedades da aritmética modular:

- $a + b \mod k = (a \mod k + b \mod k) \mod k$
- $a \cdot b \mod k = ((a \mod k) \cdot (b \mod k)) \mod k$
- $x^a \mod k = ((x^{a-1} \mod k) \cdot x) \mod k$

em que a, b, x e k são inteiros.

1.3 Funções de hash

As duas funções de hash a serem aplicadas no filtro de Bloom a ser projetado, atuam sobre strings sobre o alfabeto das letras minúsculas $\Sigma = \{a, \ldots, z\}$, e são calculadas como:

$$f(x) = h(x) \mod m$$

e

$$g(x) = \lfloor m \cdot (h(x) \cdot A \mod 1) \rfloor$$

em que x é uma string sobre o alfabeto Σ, m é o tamanho do filtro de Bloom, h é a função de fingerprinting da Seção 1.2 e A uma constante no intervalo $0 < A < 1$. A função $f(x)$ corresponde é baseada no método de divisão e a função $g(x)$ é baseada no método de multiplicação de funções de hash.

1.4 Dicionário de senhas fracas

Uma forma de garantir que usuários não criem senhas fracas é construir um filtro de Bloom para uma coleção de senhas fracas e, quando o usuário propõe a sua senha, o sistema verifica se aquela senha está no filtro de Bloom e, em caso positivo (ou falso positivo), o sistema recomenda que o usuário crie outra senha, pois a proposta é considerada fraca. A vantagem de utilizar filtros de Bloom nesta aplicação é pela sua rapidez na resposta às consultas e pelo baixo consumo de espaço.

1.5 Objetivos

O objetivo do projeto consiste em implementar o filtro de Bloom, para verificação de senhas fracas. Serão fornecidos os parâmetros:

- m: o tamanho do filtro de Bloom (Seção 1.1).
- k, o parâmetro de resto da função $h(x)$ (Seção 1.2).
- A: a constante envolvida no método de multiplicação (Seção 1.3).

Para a função $h(x)$, inteiros sem sinal de 32 bits devem ser utilizados (`uint32_t`, biblioteca `<stdint.h>`), isto é, tanto o parâmetro k quanto o retorno da função são inteiros, sem sinal, de 32 bits.

2 Especificação

O projeto deverá ser executado através da linguagem `C`.

A entrada deve ser lida da entrada padrão (`stdin`), enquanto a saída deverá ser impressa na saída padrão (`stdout`).

O programa deverá obedecer rigorosamente o formato de saída especificada, pois parte da correção será automatizada.

Obrigatoriamente o filtro de Bloom deve ser implementado neste trabalho.

2.1 Entrada

A primeira linha da entrada possui um inteiro m, uma constante real A e um inteiro sem sinal de 32-bits k, separados por um espaço. A próxima linha possui um inteiro n, indicando o número de palavras no dicionário de senhas fracas. As próximas n linhas descrevem, cada uma, uma palavra do dicionário de senhas fracas. Em seguida, temos uma linha com um inteiro q, indicando o número de consultas. As próximas q linhas indicam uma palavra a ser consultada no dicionário de senhas fracas.

Restrições:

- $1 \leq n \leq 10^4$;
- $1 \leq m \leq 10^4$;
- $1 \leq q \leq 10^4$;
- $0 < A < 1$;
- $0 \leq k \leq 2^{32} - 1$;
- As palavras a serem inseridas no dicionário ou serem consultadas possuem no máximo 30 caracteres sobre o alfabeto $\Sigma = \{a, \ldots, z\}$ das letras minúsculas.

2.2 Saída

Para cada consulta, seu programa deverá responder uma linha com "`provavelmente fraca`", caso o filtro de Bloom diga que a palavra está no dicionário de senhas fracas e uma linha com "`forte`", caso contrário.

3 Exemplos

Entrada:	**Saída:**

`5 0.141569 50`
`2`
`abra`
`cadabra`
`5`
`teste`
`inconstitucionalissimamente`
`pe`
`de`
`cabra`
	`forte`
	`provavelmente fraca`
	`forte`
	`provavelmente fraca`
	`forte`

Neste exemplo, as fingerprints de `abra` e `cadabra` são, respectivamente, 18 e 48. Assim, temos $f(18) = 3$, $g(18) = 2$, $f(48) = 3$ e $g(48) = 3$. Ficam ligadas as posições 2 e 3 do filtro de Bloom B.

- A palavra `teste` tem fingerprint 14, $f(14) = 4$ e $g(14) = 4$. Como $B[4] = 0$, `teste` é considerada forte.
- A palavra `inconstitucionalissimamente` tem fingerprint 48, $f(48) = 3$ e $g(48) = 3$. Como $B[3] = 1$, `inconstitucionalissimamente` é considerada provavelmente fraca.
- A palavra `pe` tem fingerprint 44, $f(44) = 4$ e $g(44) = 1$. Como $B[4] = 0$ e $B[1] = 0$, `pe` é considerada forte.
- A palavra `de` tem fingerprint 32, $f(32) = 2$ e $g(32) = 2$. Como $B[2] = 1$, `de` é considerada provavelmente fraca.
- A palavra `cabra` tem fingerprint 20, $f(20) = 0$ e $g(20) = 4$. Como $B[0] = 0$ e $B[4] = 0$, `cabra` é considerada forte.

4 Compilação

Um arquivo `Makefile` deve ser disponibilizado para compilação do projeto.

5 Limites de Tempo e Memória

Para cada caso de teste, será permitido a execução do programa por apenas 1 segundo com utilização máxima de 256 MB de memória. Caso o programa leve mais tempo ou memória do que isso, será considerado que o algoritmo empregado foi ineficiente.

6 Documentação

Junto do(s) código(s) necessário(s) para resolver o problema, deverá ser disponibilizado um arquivo README, identificando o autor do trabalho e especificando as instruções para compilação e execução do(s) código(s).

7 Critérios de Correção

Fazem partes dos critérios de correção:

- Eficiência do programa.
- Utilização de estruturas de dados adequadas.
- Documentação: além do arquivo README, o código deve estar bem documentado.
- Legibilidade.

7.1 Ambiente de Correção

Os projetos serão corrigidos em uma máquina com sistema `GNU/Linux` e compilador `gcc 10.2.0`.

Trabalhos que não compilarem não serão avaliados.

8 Considerações

- Este projeto deve ser executado individualmente.
- Os trabalhos que incidirem plágio serão avaliados automaticamente com nota 0 para os envolvidos. Medidas disciplinares também serão tomadas.
- O trabalho deve ser entregue dentro de uma pasta zipada com a devida identificação do(s) aluno(s) através da sala de aula virtual da disciplina na data estipulada no ambiente.

Referências

[Wik22] Wikipedia, *Bloom Filter*, `https://en.wikipedia.org/wiki/Bloom_filter`, 2022, Acessado em 26 de julho de 2022.