Grafos em Criptografia

Arthur Bridi Guazzelli
Gustavo Figueira Olegário
João Paulo T. I. Zanette

23 de Novembro de 2016

Abordagem

- Uso de hash para senhas, identificador de arquivos, etc;
- Garantir irreversibilidade das informações criptografadas:
  - Desenvolver métodos para avaliar algoritmos de criptografia;
  - Explorar algoritmos alternativos que sejam eficientes.

Contextualização

- **Grafo esparso:** Um grafo que possui poucas arestas para muitos vértices[1];
- **Grafo aleatório:** Um grafo gerado a partir de um processo aleatório[2];
- **Grafo pseudo-aleatório:** Se comporta como um grafo aleatório, com a mesma densidade de vértices, porém é gerado por um processo pseudo-aleatório[3].

Função de Hash

Mapeia um valor de qualquer tamanho para um valor de tamanho pré-estabelecido[4].

- $D \subseteq \mathbb{N}$
- $H \subset \mathbb{N}$
- N é fixo

$$f : D[1..x] \mapsto H[1..N] \mid x, N \in \mathbb{N}_{\geq 1} \tag{1}$$

- Python: Utilizado na indexação de objetos como chave em um *set*.

Cuidados

- Pré-imagem:
  - Valor que gera uma determinada hash.
    - Ataque por pré-imagem: Descobrir qual valor gerou determinada hash;
    - Algoritmo de criptografia deve se proteger na pré-imagem.
- Colisão:
  - Ocorre quando valores distintos geram a mesma hash (eq. 2).
  - Indica fraqueza na função de hash: receptor acha que recebeu a mensagem original.

$$\exists x_1, x_2 \mid f(x_1) = f(x_2) \tag{2}$$

Grafo Pseudo-Aleatório

- Seja $M = \{x \mid 0 \leq x \leq N\}$;
- Seja f uma função aleatória $\mid f : M \mapsto M$;
- Seja um grafo G(V, A):
  - $V : \{x \mid x \in M\}$
  - $A : \{(x_1, x_2) \mid x_1, x_2 \in M, f(x_1) = x_2\}$

Grafo Pseudo-Aleatório

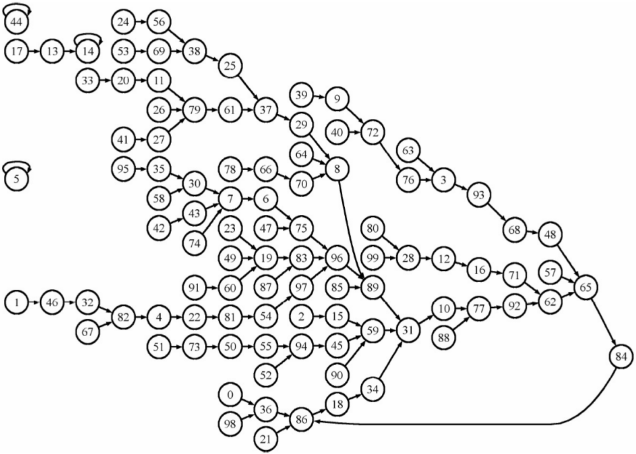

Figura 1: Exemplo de grafo pseudo-aleatório baseado em uma função de hash hipotética.

Floyd (Tartaruga e a Lebre)

Entrada: Um grafo G.
Saída : Flag indicando se há ciclos em G.
seja *T um ponteiro para um vértice*;
seja *L um ponteiro para o mesmo vértice que C*;
enquanto $L \neq$ **nulo faça**
    $T.caminhar(1)$;
    $L.caminhar(2)$;
    se $L \neq$ **nulo e** $L = T$ **então**
        retorna verdadeiro;
    fim
fim
retorna falso;

Floyd (Tartaruga e a Lebre) - Demonstração

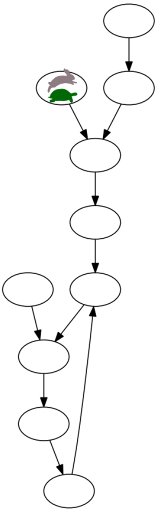

Figura 2: Demonstração do algoritmo: passo 1

Floyd (Tartaruga e a Lebre) - Demonstração

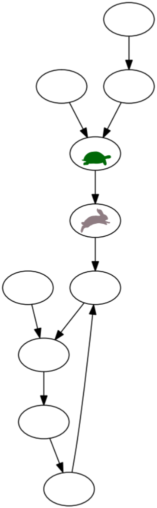

Figura 3: Demonstração do algoritmo: passo 2

Floyd (Tartaruga e a Lebre) - Demonstração

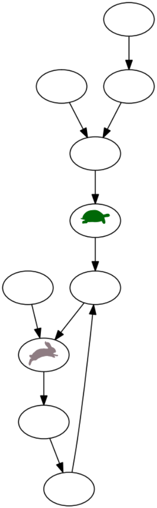

Figura 4: Demonstração do algoritmo: passo 3

Floyd (Tartaruga e a Lebre) - Demonstração

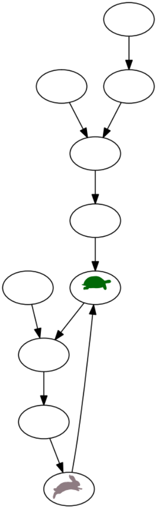

Figura 5: Demonstração do algoritmo: passo 4

Floyd (Tartaruga e a Lebre) - Demonstração

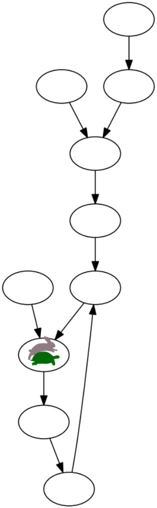

Figura 6: Demonstração do algoritmo: passo 5

Complexidade

- Temporal: $O(n) \mid n$ é número de vértices no grafo;
- Espacial: $O(1)$.

Em um caminho com n vértices, o algoritmo leva cerca de $3n$ passos para identificar um ciclo[4]. Mais ponteiros permitem maior eficiência na checagem.

C. T. Pozzer, "Material didático," Jun 2010.

Wikipédia, "Grafo aleatório," 2016.

M. Krivelevich and B. Sudakov, *Pseudo-random Graphs*, pp. 199–262.
Berlin, Heidelberg: Springer Berlin Heidelberg, 2006.

N. Tokareva, "Connections between graph theory and cryptography," *G2C2: Graphs and Groups, Cycles and Coverings*, Sep 2014.