Computational Complexity & Hard Problems

Teórica #7 de Criptografia Aplicada

Máquina de Turing

Uma máquina de Turing (TM) define uma "fita" (tape) infinita composta por operações read e write. As operações tanto podem seguir da esquerda para a direita, como da direita para a esquerda.

Uma linguagem *L* satisfaz a propriedade *Turing-Recognisable*, se uma TM reconhecer as palavras compostas pela linguagem, ou seja, a TM irá dar sempre uma resposta positiva e parar se a palavra *w* pertencer a *L*.

Uma linguagem *L* satisfaz a propriedade *Turing-Decidable*, se uma TM souber reconhecer palavras que pertençam à linguagem e souber rejeitar aquelas que não pertençam.

Landau Notation (Big O)

A notação Big O expressa que uma função:

```
f(n) = O(g(n))
```

Sendo que `g(n)` expressa o limite superior de `f(n)` (g(n) é um limite superior asintótico para f(n)), ignorando qualquer factor constante associado a `f(n)` (e.g., f(3n) = O(g(n)) da mesma forma que f(n) = O(g(n))).

Existe também a notação "small o", que expressa:

```
f(n) = o(g(n))
```

Sendo que `f(n)/g(n) = 0`, onde *n* tende o seu limite até ao infinito positivo.

Classes Complexionais

A complexidade computacional pode ser dividida em duas categorias de classes: temporal (time) e espacial (space). Estas classificam a complexidade em torno do tempo e espaço usado na fita da TM, para executar a função.

A classe complexional mais importante é *P*, que define a classe complexidade das funções que podem ser decididas em tempo polinomial por uma TM.

$$\mathrm{P} = \bigcup_{k \in \mathbb{N}} \mathrm{TIME}(n^k)$$

$$\mathrm{PSPACE} = \bigcup_{k \in \mathbb{N}} \mathrm{SPACE}(n^k)$$

A segunda classe complexional mais importante é *NP* (Nondeterministic Polynomial), que define que linguagens (problemas):

- Que possam ser decididas em tempo polinomial por uma TM não determinístico;
- Que possam ser verificadas em tempo polinomial por uma TM determinístico.

São não polinomiais.

$$\mathrm{NP} = \bigcup_{k \in \mathbb{N}} \mathrm{NTIME}(n^k)$$

O problema da factorização e a sua ligação com a criptografia

O problema da factorização consiste em encontrar números primos (p = primo se apenas p % 1 e p % p = 0), *p* e *q*, onde `N = p*q`, sendo N um número bastante grande. O problema aqui está em encontrar estes números, pois são expendiosos em tempo (TIME) e recursos (SPACE).

Uma das provas dos algoritmos RSA baseia-se neste problema, pois ao encontrar um numero bastante grande que comprove a formula, podemos dizer que a segurança da algoritmo é segura, dado a sua *rigidez* (hardness).

A factorização de um número *n* é dado pela divisão sequencial de n até o valor 2 {2, ..., n -1}. A complexidade computacional desta fatorização neste formato é `O(n)`. Contudo, há melhorias que podem ser feitas, como por exemplo dividir n a cada sqrt(n), o que reduz a complexidade em `O(n^(1/2))`. De seguida, pode-se dividir n pelos primos que são conhecidos e menor que sqrt(n), reduzindo a complexidade em `O(sqrt(n)/log(sqrt(n))`.

Problema do Logaritmo Discreto

Dado um intervalo multiplicativo *Zp*, composto por números primos *p*, *g* e *x*, o problema do logaritmo discreto (DLP) consiste em encontrar o valor *y*, tal que `g^y = x` pertence a Zp, ou seja:

```
g^y (=congruente) x (mod(p))
```