Insper

Lógica da Computação - 2020/1

Aula 18/T09 - 29/Apr/2020

Raul Ikeda - rauligs@insper.edu.br

Objetivos

1. Introdução à Complexidade

Complexidade Parte 1

1. Dado $L = \{0^n 1^n | n \geq 0\}$, **descreva** uma MT que verifica se uma entrada $w \in L$.

`>> Ver Sipser Pag. 265.`

Definição: "Seja M uma MT que para sobre todas as entradas. O **tempo de execução** ou **complexidade de tempo** de M é a função $f : N \to N$ onde $f(n)$ é o número máximo de passos que M usa sobre entradas de comprimento n." Sipser Pag. 262.

2. Analise o algoritmo acima e determine qual é o tempo de execução da MT.

Definição: "Sejam f e g funções $f, g : N \to R^+$. Digamos que $f(n) = O(g(n))$ se existem inteiros positivos c e n_0 tais que para todo inteiro $n \geq n_0$

$$f(n) \leq c\ g(n)$$

Quando $f(n) = O(g(n))$ dizemos que $g(n)$ é um limitante superior para $f(n)$ ou, mais precisamente, que $g(n)$ é um limitante superior assintótico para $f(n)$" Sipser Pag 263.

Conforme já visto no curso Desafios de Programação, calcular o tempo exato de um algoritmo pode ser uma tarefa complexa. Para entender o tempo de execução de um algoritmo quando uma entrada n é muito muito grande, utiliza-se a **análise assintótica**. A notação Big-O é usada para ilustrar o **limitante superior** para uma $f(n)$.

Mais detalhes: Cormen Cap. 3.

2. Tentativa #2: Tente agora fazer uma MT $O(n\ log\ n)$ para L.

`>> Ver Sipser Pag. 266.`

Teorema: "Seja $t(n)$ uma função onde $t(n) \geq n$. Então toda MT multifita de tempo $t(n)$ tem uma MT de fita única equivalente de tempo $O(t^2(n))$." Sipser, Pag. 268.

Dados os dois modelos diferentes de MT que resolvem o mesmo problema, é possível observar que, enquanto o estudo da computabilidade se importava mais com a equivalência de modelos e a capacidade de solucionar problemas, na intratabilidade o tempo de execução de um algoritmo importa.
Ainda, algoritmos que possuem tempo $O(n^k)$ com $k > 0$, são considerados **polinomiais**.

3. Tentativa #3: Será que é possível fazer uma MT O(n) para L?

`>> Ver Sipser Pag. 267.`

Teorema: "Seja $t(n)$ uma função, onde $t(n) \geq n$. Então para toda MT não determinística de uma única fita de tempo $t(n)$ existe uma MT determinística de única fita equivalente de tempo $2^{O(t(n))}$" Sipser, Pag. 270.

Ver também: MT não determinística e Teorema 3.16. Sipser Pag. 157.

Pelo teorema acima, algoritmos $2^{O(n^{\delta})}$, que são considerados **exponenciais**, podem ser decididos em tempo **polinomial** por uma MT não-determinística.

3. Como ficam os algoritmos $O(log\ n)$, $O(n\ log\ n)$ e $2^{O(log\ n)}$?

Classes P e NP

Definição: "**P** é a classe de linguagens que são **decidíveis** em tempo polinomial sobre uma máquina de Turing **determinística** de única fita."

Exemplo: Algoritmos de ordenação, busca, etc.

Definição: "**NP** é a classe de linguagens que são **decidíveis** em tempo polinomial sobre uma máquina de Turing **não-determinística**."

Exemplo: Problema do Caixeiro Viajante (TSP), SAT, etc.

Exercícios (Sipser):

1. Estudar os teoremas 7.15 e 7.16.
2. Problemas 7.1, 7.2, 7.10, 7.12 e 7.18.

Quem quer ser um milionário? Parte II

Problema: "Suppose that you are organizing housing accommodations for a group of four hundred university students. Space is limited and only one hundred of the students will receive places in the dormitory. To complicate matters, the Dean has provided you with a list of pairs of incompatible students, and requested that no pair from this list appear in your final choice."
Fonte: http://www.claymath.org/millennium-problems/p-vs-np-problem

1. Crie um algoritmo para solucionar o problema acima. O problema é P ou NP?

O prêmio para quem conseguir construir uma MT determinística que decide em tempo polinomial é de US$ 1.000.000,00.
Mais detalhes: http://www.claymath.org/sites/default/files/pvsnp.pdf

Próxima aula:

- Geração de código

Referências:

- Aho et al. Cap. 8
- J. J. Neto - Cap. 4.3