Resumo - Teoria da Complexidade

Geovane Fonseca de Sousa Santos

[1] Ciência da Computação - Pontíficia Universidade Católica de Minas Gerais
Belo Horizonte, MG - Brasil

{geovane}geovanefss@gmail.com

A Teoria da complexidade: É o ramo da computação teórica e da matemática que classifica problemas computacionais de acordo com sua dificuldade, e os relaciona entre si. Para isso, entende-se como problema computacional uma tarefa que é, em princípio, passível de ser resolvida por um computador. Isso quer dizer que esse problema pode ser definido através de um conjunto de expressões matemáticas.

Um problema é considerado inerentemente difícil se sua solução requer bastante recursos, independendo de seus algoritmos. A teoria é formalizada através de modelos matemáticos de computação que estuda e quantifica os recursos necessários para resolver o problema. Uma das abordagens da Teoria da Complexidade é definir através de sua lógica matemática o que se pode ou não alcançar com os recursos disponíveis da tecnologia atual.

Melhor, pior e caso médio de complexidade: Esses três casos de complexidade referem-se a diferentes maneiras de medir a uma medida de complexidade, como por exemplo o tempo, de valores de entradas diferentes mas do mesmo tamanho. Dependendo dos valores de entrada, elas podem ser mais ou menos rápidas de resolver.

Pior caso: É quando um algoritmo resolve um problema utilizando o maior tempo possível ou o maior custo de recursos. Um exemplo desse tipo é o bubblesort que tem o pior caso para ordenação e um custo na ordem de $O(n^2)$.

Melhor caso: É quando um algoritmo resolve um problema utilizando o menor tempo possível ou o menor custo de recursos. Um exemplo desse tipo é o quicksort que tem o melhor caso para ordenação e um custo na ordem de $O(n * lg(n))$.

Caso médio: Essa complexidade só é definida com relação a uma distribuição de probabilidade sobre as entradas. Por exemplo, se todas as entradas do mesmo tamanho são consideradas terem a mesma probabilidade de aparecer, a complexidade do caso médio pode ser definida com relação à distribuição uniforme sobre todas as entradas de tamanho n.

Problemas ainda não resolvidos: Existem problemas computacionais onde se usa muito processamento de máquina e tempo para resolver uma pequena parcela de código ou nenhuma. Eles são classificados como problemas em aberto, pois ainda não foi possível definir com base na teoria da complexidade formas de torná-los viáveis.

Um dos mais famosos problemas em aberto é o do "cacheiro viajante", onde se busca o menor caminho para se visitar vários locais passando por eles uma única vez.

Problemas P: São um conjunto de problemas que podem ser executados em um

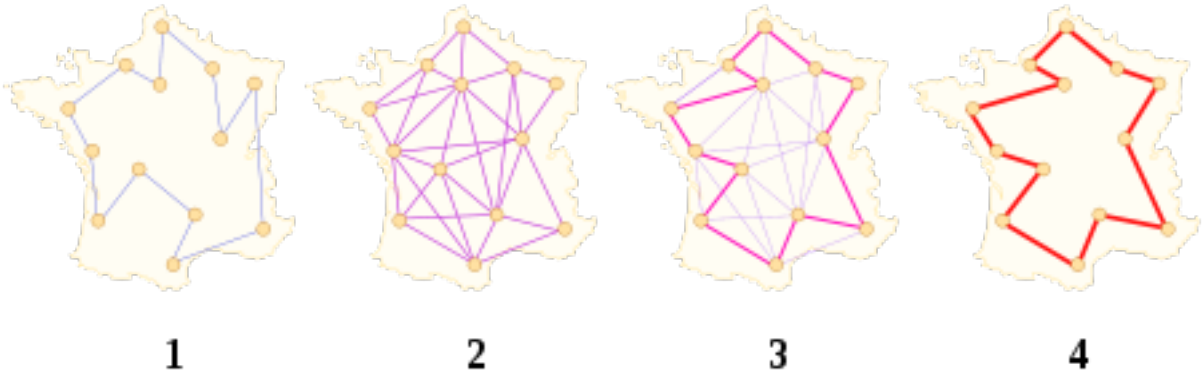

Figure 1. Cacheiro Viajante

tempo polinomial determinístico. Algoritmos usados para resolver esse tipo de problema, o fazem na proporção de tempo de $O(n^k)$, sendo que k é uma constante. Os exemplos mais comuns são cálculo de MMC, MDC e somatório.

Problemas NP: São um conjunto de problemas que podem ser executados em um tempo polinomial não determinístico. Nesse tipo de problema, o exemplo mais famoso é o do cacheiro viajante.

A maior dificuldade envolvendo essa classe de problemas é encontrar algoritmos que solucionam seus códigos de forma polinomial ao tamanho de entrada. Desse modo, a questão da teoria da complexidade é se os problemas NP realmente não podem ser resolvidos em tempo polinomial.

Problemas NP-Completo: É um subconjunto dos problemas NP, onde retirando da classe NP os problemas que podem reduzir a um tempo polinomial, sobra os chamados NP-completo. Dessa forma, eles são os problemas mais complexos de serem resolvidos e dificilmente fazem parte da classe P.

Exemplo de um problema NP: Um computador que precisa calcular a menor rota entre 20 cidades, supondo que ele faça 1 bilhão de adições por segundo, faria $10^9/19 = 53$ milhões de rotas por segundo. Sendo assim demoraria $1,2 * 1017/(53 * 106) = 2,3 * 109$ segundos para completar sua tarefa de executar as $19!$ operações necessárias para o problema, o que equivale a cerca de 73 anos. O que podemos perceber através da tabela:

n	rotas por segundo	(n-1)!	cálculo total
5	250 milhões	24	insignificante
10	110 milhões	362.880	0.003 seg
15	71 milhões	87 bilhões	20 minutos
20	53 milhões	$1,2 * 10^{17}$	73 anos
25	42 milhões	$6,2 * 10^{23}$	470 milhões de anos

References

[Bueno 2011] Bueno, L. (2011). Teoria da complexidade Computacional. http://professor.ufabc.edu.br/ leticia.bueno/classes/aa/materiais/complexidade2.pdf.