Centro Universitário SENAC

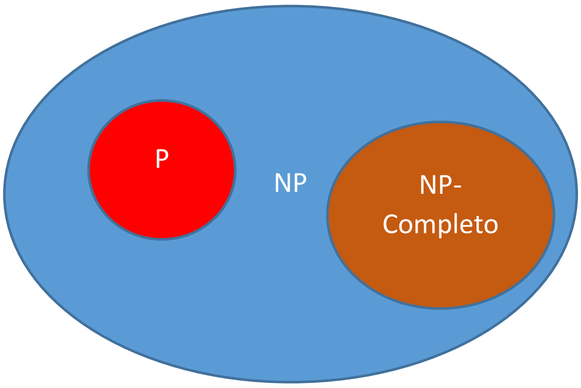
Aula de Teoria da Computação – 16 Outubro de 2014

A Classe NP-Completo.

Atividade 1: ler no livro texto (Sipser) o subcapítulo 7.4 sobre a classe NP-Completo e responda as seguintes questões:

1) O que é uma redução?
2) De um exemplo de uma redução?
3) Porque uma redução deve ser feita em tempo polinomial?

Note que a classe NP-Completa é uma classe especial de problemas que está na Classe NP, conforme indicado na figura abaixo:



Portanto, um problema para ser NP-Completo ele precisa também ser um problema NP, isto é, é preciso que o problema possua um verificador em tempo polinomial.

A Classe NP-Completo possui uma característica muito peculiar onde qualquer problema dentro da classe NP pode ser reduzido em tempo polinomial a um problema dentro da classe NP-Completo. Isto quer dizer que se um problema dentro da classe NP-Completo for resolvido em tempo polinomial todos os problemas em NP serão resolvidos em tempo polinomial.

Portanto, para estar na classe NP-Completo um problema L precisa:

1) L precisa estar em NP, ou seja, L precisa de um verificador em tempo polinomial.
2) Para todo problema T em NP, existe uma redução em tempo polinomial de forma que $T \leq_P L$.

A classe de problemas em NP-Completo são mais difíceis que os problemas em NP, e possui o tipo de problema menos provável de star na classe P.

Questão 4: como é possível demonstrar que todo problema T em NP é redutível em tempo polinomial a um outro problema se não sabemos ainda se existem ou não novos problemas na classe NP?

Questão 5: Se a classe NP-Completo é uma sub-classe de NP, porque se um problema em NP for resolvido em tempo polinomial toda a classe NP também será resolvida em tempo polinomial e P = NP?

Questão 6: Na mesma linha da questão 4, como foi encontrado o primeiro problema NP-Completo?

O problema SAT: Considere uma fórmula em lógica booleana formada por variáveis booleanas (variáveis que aceitam apenas como atribuição os valores de VERDADEIRO (1) ou FALSO (0)) geralmente com os operadores E (∧), OU (∨) e NÃO (~), por exemplo:

$$f_1 = {\sim}X \vee ((Y \vee {\sim}Z) \wedge (X \vee Z)) \vee (({\sim}R \wedge Y) \wedge {\sim}Y)$$

Uma atribuição para um conjunto de variáveis booleanas é um conjunto de valores 0s e 1s de forma que cada uma das variáveis tenha seu respectivo valor, por exemplo, uma atribuição para as variáveis da fórmula f_1 acima pode ser:

X = 1; Y = 1; Z = 1; R =0;

Note que dada uma atribuição é possível determinar o valor final da fórmula booleana, no exemplo acima para a atribuição dada a fórmula possui o valor VERDADEIRO, isto é, o valor de saída da fórmula é igual a 1. Neste caso dizemos que a atribuição dada SATISFAZ a fórmula booleana.

O problema SAT de uma maneira geral é o seguinte: dada uma fórmula booleana qualquer existe alguma atribuição para as variáveis desta fórmula que a SATISFAÇA, isto é, que a torne VERDADEIRA.

Questão 7: existe alguma outra atribuição para as variáveis de f_1 que a satisfaça?

Questão 8: existe alguma atribuição para as variáveis de f_1 que não satisfaça a fórmula?

Questão 9: considere a fórmula abaixo e encontre uma atribuição que satisfaça a fórmula:

$f_2 = Z \wedge (({\sim}X \vee {\sim}Z) \wedge (X \vee Y)) \vee (({\sim}R \wedge {\sim}Y) \wedge R)$

O teorema de Cook Levin descrito no livro mostra de uma forma relativamente simples como foi feita a prova do primeiro problema na classe NP-Completo, ao invés de reduzir todos os problemas da classe NP para um problema que se acreditava estar em NP-Completo, foi feita uma redução de uma sequência de computação de uma máquina de Turing M para SAT (ou 3-SAT, o que é equivalente). A fórmula foi construída de forma que, se existir uma sequência de computação que leve a um estado de aceite de M então a fórmula é satisfatível e se a sequência de computação não levar para um estado de aceite então não existe atribuição que satisfaça a fórmula.

Note que ao utilizar uma sequência de computação de uma máquina de Turing genérica M, esta máquina poderia estar computando qualquer problema da classe NP e, portanto, a sequência de computação de aceitação seria aquela onde o problema da classe NP seria resolvido. Como podemos ter uma sequência para qualquer problema em NP, para cada problema em particular é possível ter uma fórmula correspondente e, desta forma, prova-se que todo problema em NP pode ser reduzido a SAT (ou 3-SAT) e a partir daí chegou-se ao primeiro problema na classe NP-Completo.

Questão 10: para provar agora que um outro problema D está na classe NP-Completo basta mostrar que D está na classe NP e mostrar que existe uma redução em tempo polinomial de SAT (ou 3-SAT ou qualquer outro problema em NP-Completo) para D (SAT $\leq_P$ D). Porque isto é verdade?