**Resumo de NP-Completude, por Ermano Arruda(eaa3)**

**Definições básicas e conclusões importantes**

* **Problemas Polinomiais**

Um problema é dito polinomial, pertencente à classe P, se existir um algoritmo que o resolve cujo tempo de execução é O( P( n ) ) onde P ( n ) é uma função polinomial do tamanho, n, do input.

* **Problemas Não-Deterministicamente Polinomiais ( NP )**

Um problema é NP se existe um **algoritmo não determinístico** que o resolve em tempo polinomial, ou, em outras palavras, dada uma possível solução, a verificação se essa solução está certa ou não é feita em tempo polinomial. Um algoritmo não determinístico é um algoritmo que é capaz de executar uma instrução à mais, quando está resolvendo o problema, além daquelas possíveis para algoritmos determinísticos tradicionais, essa instrução nova que ele usa(**se necessário**) é chamada de “saldo-nd” ou “salto/escolha **n**ão **d**eterminística”. Basicamente o que o salto-nd faz é: dado uma instância de um problema X, se houver um conjunto de escolhas que levem à solução desse problema, então o salto-nd fornecerá esse conjunto que será a resolução do problema, em um problema de decisão teríamos a resposta “SIM”. Mas, caso não exista um conjunto de escolhas que levem a solução desse problema, o salto-nd retornará uma conjunto de escolhas que não irá resolver o problema, num problema de decisão teríamos a resposta “NÃO”, pois não foi possível encontrar um conjunto de escolhas que resolvam o problema, então ele retorna um conjunto de escolhas qualquer que gerará a resposta “NÃO” do problema de decisão.

Por exemplo, dado o problema : (x v y) é satisfatível?

Instância do problema -> (x v y)

Resposta -> “SIM”

Chamaríamos o salto-nd duas vezes para escolher os valores verdade de x e de y, e depois verificaríamos( **em tempo polinomial** ) se (x v y) é satisfatível testando a expressão para os valores verdade escolhidos pelo salto-nd. Como ( x v y ) é satisfatível, as escolhas de valores que o salto-nd iria fazer para x e para y seriam de tal forma que satisfariam à expressão, como o salto-nd escolheria a resposta ou uma das respostas certas, não se sabe, o salto-nd é uma abstração, mas com esse artifício resolveríamos o problema SAT em tempo polinomial. Se a expressão fosse ( x^¬x), insatisfatível, o salto-nd iria atribuir qualquer valor pra x, e quando nós fossemos verificar daríamos a resposta “NÃO”, pois não foi possível encontrar uma valoração que satisfaça a expressão, se tal escolha existisse o salto-nd teria encontrado.

Segue da definição que **todo problema P é NP**, e que, portanto, $P ⊆ NP$, pois todo problema P pode ter uma candidata à resposta verificada em tempo polinomial.

* **Problemas NP-Hard**

Um problema X é dito NP-Hard se todo problema NP é polinomialmente redutível à X. Significa que X é tão ou mais difícil que qualquer problema NP.

* **Problemas NP-Completos**

Um problema X é NP-Completo se (1) pertence à NP e (2) X é NP-Hard.

Segue da definição que se algum problema X, NP-Hard, for provado que pertence à P, então teremos que, já que todo problema NP é polinomialmente redutível à X, todo problema NP seria P, i.e., teríamos $NP ⊆ P$, e como $P ⊆ NP$, consequentemente P = NP.

Da definição, segue que, uma vez que conhecermos apenas um problema NP-Completo, provar que algum outro problema é NP-Completo é mais fácil.

Um problema X é NP-Completo se (1) for NP e (2’) Y é polinomialmente redutível à X, onde Y é NP-Completo.

Isso é verdade, pois, como Y é NP-Completo, então todo problema NP é redutível à Y, e sendo(2’) Y polinomialmente redutível à X, teremos que todo problema NP também será polinomialmente redutível à X, pois a relação de redutibilidade é transitiva, portanto X será NP-Hard, e como (1) X é NP também, então X é NP-Completo.

Então o que foi feito foi achar um problema canônico que seja NP-Completo, e sair descobrindo novos problemas NP-Completos à partir dele. O problema padrão que escolheram foi o problema SAT. É fácil de ver que o problema SAT é NP, pois nós podemos chutar, uma valoração e verificar se ela satisfaz ou não uma certa expressão booleana, isso em tempo polinomial. Falta provar então, que SAT é NP-Hard, teríamos que provar que todo Problema NP é redutível polinomialmente à SAT. E isso foi provado por um cara chamado Stephen Cook que formolou então o seguinte teorema:

“O problema SAT é NP-Completo.”

Com base nisso, sabendo que o problema SAT é NP-Completo, podemos sair procurando novos problemas NP-Completos, provando que um problema X é NP-Completo se for NP e se SAT for redutível polinomialmente à X.

[De acordo com todas as definições acima]

Uma implicação interessante é que se nós provarmos que SAT pode ser resolvido em tempo polinomial, então P = NP = NPC. Isso é porque, se SAT for resolvido em tempo polinomial, teremos que todos os problemas NP serão resolvidos em tempo polinomial, pois SAT é NP-Hard, e sendo assim, como todo NP é redutível polinomialmente à SAT, todo NP seria resolvido em tempo polinomial, portanto teríamos que $NP ⊆ P$, e como $P ⊆ NP$, então P = NP. Ainda mais, seja X um problema NPC, isso significa que X é NP, mas como P = NP, então X é P, portanto todo problema NPC, pela sua definição, seria P também, ou seja, P = NPC.

-Fontes: <http://en.wikipedia.org/wiki/P_%3D_NP_problem#Formal_definition_for_NP-completeness>

<http://www.ime.usp.br/~pf/algoritmos/aulas/footnotes/instance.html>

<http://docs.google.com/viewer?a=v&q=cache:sm04B37mtiIJ:moreno.cin.ufpe.br/~if672cc/2011.1/monitoria/NP-Completude2.ppt+%22portanto,+P+est%C3%A1+contido+em+NP%22&hl=pt-BR&gl=br&pid=bl&srcid=ADGEESgmglsfPosaMUjfogx6O3lPPg1znqOpas-GRCISmn2NHDUk1x2iBMSWdGB5Duthgc2MHPEGeaxRCaxlXgr6tiAMSL5Ojgk5s_Jx0WhqcPlYQftkpMBD9SbT67D6SF1_D3GGRkuI&sig=AHIEtbSlC7PgtBpn7PY91ec8ZA9tslLDCQ>

[http://pt.wikipedia.org/wiki/Problema\_de\_decisão](http://pt.wikipedia.org/wiki/Problema_de_decis%C3%A3o)

-UDI-Manber

Um link pra baixar UDI Manber está no meu public, na área de arquivos:

<http://www.cin.ufpe.br/~eaa3/?page=arquivos>