INSTITUTO FEDERAL DE SÃO PAULO

BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

FERNANDA MARTINS DA SILVA

P vs. NP

São João da Boa Vista/SP

Um problema computacional é complexo dado o melhor consumo de tempo possível para um problema. Muitos dos problemas resolvidos nem sempre são as melhores soluções possíveis, apenas as mais conhecidas. Pode-se dizer que um algoritmo que resolva da melhor maneira um problema X, é chamado de polinomial, em contrapartida, um algoritmo que não resolva esse problema X da melhor maneira, é chamado de polinômio não determinado (do inglês *nondeterministic polynomial*).

Algoritmos polinomiais (P)

São polinomiais algoritmos que resolvem um problema recebendo uma instância, devolve uma solução ou informa que a mesma não é possível. Desta forma, um algoritmo capaz de resolver qualquer instância que ocorra, e até mesmo aquelas que não tem solução, é chamado de algoritmo polinomial. Isto se deve por conta de análises do tempo de consumo, que, no pior caso de instância que o algoritmo receber, será limitado a uma equação polinomial.

Tais algoritmos são considerados razoavelmente rápidos, além de seu consumo de tempo ser polinomial para todas as outras instâncias, sem exceções. Considerados problemas tratáveis, um exemplo de problema polinomial é a equação inteira de segundo grau (dados a, b e c números inteiros, x *é* um número inteiro tal que $ax^2 + bx + c = 0$ (ou retornar que x não existe)). Um problema é não polinomial se não existe algoritmo polinomial que resolva o problema.

Então, a classe P de problemas é o conjunto de todos os problemas de decisão que são polinomiais.

Algoritmos *nondeterministic polynomial* (NP)

Diferente dos polinomiais, os *nondeterministic polynomial* são considerados lentos (ainda que, em alguns casos, possam ser mais rápidos que algoritmos polinomiais para valores modestos). Um problema *nondeterministic polynomial* não se trata de problemas de impossível resolução, mas de problemas que nunca serão polinomiais, então, são chamados de problemas intratáveis. Não existe de fato uma melhor opção de tratamento dessas instâncias, logo, todos os algoritmos que tentam a resolver são candidatos a uma possível real solução. Problemas de classe NP na verdade são problemas de classe P, com características "não determinísticas", ou seja, ele pode ser provado em tempo polinomial. Um exemplo é a fatorização (Dados n e m , há um inteiro f na faixa $1 < f < m$ que seja um fator de n (divida com resultado inteiro)?).

Então, a classe NP é na verdade um conjunto de problemas de decisão que são polinomialmente verificáveis. A complexibilidade do código não importa de fato, mas sim se existe um verificador para o caso. Em problemas de classe P, o problema é mostrado como polinomialmente verificável se um verificador executa o algoritmo e recebe “correto” quando recebe um sim e “errado” quando recebe um não.

Uma teoria, entretanto, chegou a definição de uma subclasse NP, as NP-completas. Em geral, problemas de difícil resolução da classe NP são agrupados, de tal forma que, se qualquer algoritmo se encontrar polinomial, então todos os outros também seriam resolvidos polinomialmente, rebaixando as NP em P, chegando a P = NP.

P vs. NP

Nesse sentido, o que dizer sobre P = NP?. Ainda sem respostas ou grandes avanços, teoristas comentam sobre o que pode ocorrer: Se P for diferente de NP, logo, a situação atual dos algoritmos não mudaria muito, mas, caso contrário, muitas das coisas que conhecemos hoje mudariam (não necessariamente para melhor). A importância desses questionamentos vem da questão P vs. NP, e principalmente, pelo sucesso das NPs-Completas e o interesse em criptografia.

Se um problema NP é resolvido polinomialmente, mas não há provas de que outros problemas agrupados em NP também possam ser polinomiais, a teoria que toda NP é P é desmentida: o que conhecemos hoje não mudaria muito, simplesmente não haveria mais procura de algoritmos em P para uma série de problemas intratáveis.

Porém, se provado que toda NP pode ser resolvida em P, muitas coisas mudariam. Seria benéfico para diversas áreas, como as de comunicações, indústrias e áreas de desenvolvimento em geral, já que todas as centenas de problemas úteis até então resolvidos por algoritmos trabalhosos, seriam simplificados, e consequentemente, resolvidos rapidamente. A problemática vem quando quebrar criptografias fica bem mais fácil. A questão é: a criptografia depende de um problema em NP (decomposição de um número em fatores). Se realmente provado P = NP, então, a decomposição de um número em fatores passaria a ser resolvida de forma eficiente, logo, a criptografia hoje seria facilmente quebrada.

Dado contexto, desde 1971 programadores do mundo todo discutem a possibilidade de P ser realmente igual a NP (e vice versa), e, caso a teoria se mostre verdadeira, o mundo como é hoje passaria por diversos avanços tecnológicos e matemáticos, que por hora, são inimagináveis.

REFERÊNCIAS

Problemas NP-completos. Disponível em: <https://www.ime.usp.br/~pf/analise_de_algoritmos/aulas/NPcompleto.html>. Acesso em: 2 de mar. 2024.

O que é um problema NP completo? Disponível em: <https://pt.stackoverflow.com/questions/34104/o-que-%C3%A9-um-problema-np-completo>. Acesso em: 2 de mar. 2024.

POLTOSI, I. P X NP: um problema matemático para o terceiro milênio. lume.ufrgs.br, 2003. Acesso em: 3 mar. 2024.

MARÍ, R. P. O problema que os programadores não conseguiram resolver em 45 anos. Disponível em: <https://brasil.elpais.com/brasil/2017/05/19/tecnologia/1495202801_698394.html>. Acesso em: 3 mar. 2024.