P, NP e NP-Completo

André Vignatti

DINF- UFPR

Problemas Difíceis, Problemas Fáceis

O mundo está **cheio** de problemas de busca.

- Alguns podem ser resolvidos eficientemente, outros parecem ser muito difíceis.

Isto é descrito na tabela a seguir:

Problemas difíceis	Problemas fáceis
3SAT	2SAT, HORN SAT
CAIXEIRO VIAJANTE	ÁRVORE GERADORA MÍNIMA
LONGEST PATH	SHORTEST PATH
3D MATCHING	BIPARTITE MATCHING
MOCHILA	MOCHILA fracionária
INDEPENDENT SET	INDEPENDENT SET em árvores
ZERO-ONE EQUATIONS	ZOE fracionário
RUDRATA PATH	EULER PATH
MAXIMUM CUT	MINIMUM CUT

Problemas Difíceis, Problemas Fáceis

Na direita temos problemas que são resolvidos eficientemente.

À esquerda, temos "nozes duras" que escaparam da solução eficiente durante décadas ou séculos.

Os problemas à direita são resolvidos por algoritmos especializados e diversos: programação dinâmica, fluxo de rede, busca em grafo, gulosos.

- Estes problemas são fáceis por várias razões diferentes.

Problemas Difíceis, Problemas Fáceis

Em contraste, os problemas da esquerda **são todos difíceis pela mesma e única razão**:

- No fundo, eles são todos o mesmo problema, apenas com **diferentes disfarces**!

Em outras palavras...

Todos são **equivalentes**: cada um pode ser reduzido a qualquer outro e vice-versa.

NP (a.k.a Problemas de Busca)

É hora de introduzir alguns **conceitos importantes**.

Lembrando: um problema de busca é aquele onde uma solução proposta pode ser **rapidamente verificado** quanto à correção.

Denotamos a classe de todos os problemas de busca como **NP**.

Definição: Classe NP

A classe NP é a classe de **TODOS** os problemas de busca.

Muitos problemas **NP** (i.e, de busca) podem ser resolvidos em tempo polinomial.

Nesses casos, existe algoritmo que recebe uma instância I e tem tempo de execução polinomial em função de $|I|$.

- Se I tem solução, o algoritmo a retorna, caso contrário o algoritmo diz que não há solução.

Definição: Classe P

A classe **P** é a classe de **TODOS** os problemas de busca que são resolvidos em tempo polinomial.

Os problemas de busca no lado direito da tabela estão em **P**.

$P \subseteq NP$

A definição de **P** é igual a **NP**, mas **P** tem uma restrição a mais (deve ser de tempo polinomial)

- Isso faz com que **P** seja uma sub-classe (talvez igual) de **NP**.
- Ou seja, **P** $\subseteq$ **NP**.

Por que P e NP?

P vem de “polynomial time”

NP significa “nondeterministic polynomial time”, um termo que remonta às raízes da teoria da complexidade

Abre-parênteses: Algoritmo Não Determinístico

Um algoritmo não determinístico é um tipo especial (e bastante irreal) de algoritmo que “adivinha corretamente” em todos os passos.

Assim, **NP** são os problemas cuja solução pode ser **encontrada e verificada** em tempo polinomial por um algoritmo não determinístico.

Por que P e NP?

Aliás, a definição original de **NP** (e seu uso mais comum até hoje) não foi para problemas de busca, mas para problemas de decisão:

Problemas de Decisão:

São perguntas algorítmicas que podem ser respondidas por **SIM** ou **NÃO**.

Exemplo:

"Existe atribuição que satisfaz esta fórmula booleana?"

Isso reflete uma realidade histórica: na época, a teoria da NP-completude era desenvolvida por pesquisadores interessados em linguagens formais e lógica, onde problemas de decisão são de importância central.

P $\neq$ NP?

Existem problemas de busca que **NÃO** podem ser resolvido em tempo polinomial?

Em outras palavras...

P $\neq$ NP?

A maioria dos pesquisadores de algoritmos acha que sim:

- É difícil acreditar que uma busca de tempo exponencial sempre possa ser evitada,
- ou que um simples truque vai desfazer a dificuldade de todos estes problemas difíceis

P $\neq$ NP?

Os matemáticos também acreditam que P $\neq$ NP:

- Encontrar uma prova para uma proposição matemática é um problema de busca e, portanto, está em **NP**.
- Quando uma prova é escrita em detalhes, ela pode ser verificada de forma mecânica, linha por linha, por um algoritmo eficiente.

Então, se P = NP, haveria um método eficiente para provar qualquer teorema, portanto, **eliminando a necessidade de matemáticos**!

P $\neq$ NP?

Somando tudo, há várias razões pelas quais acredita-se a P $\neq$ NP.

No entanto, provar isso se mostrou extremamente difícil

- Ninguém conseguiu **até hoje**, a cada 6 meses sai uma prova (errada) disso.

É um dos enigmas **mais profundos e importantes** não resolvidos da matemática.

Reduções, novamente

A maioria das pessoas acredita que P $\neq$ NP

Em quais evidências elas se baseiam? (Além da discussão informal de antes)

Tal evidência é fornecida por **reduções**.

As reduções demonstram que os problemas no lado esquerdo da tabela anterior são exatamente o mesmo problema.

Obviamente, eles são apresentados em diferentes formas, mas no fundo são todos iguais.

Reduções, novamente

Então, se resolvermos um deles em tempo polinomial, resolvemos **todos os outros da tabela**.

Mas não se trata somente dos problemas da tabela...

Além disso, usaremos reduções que mostram que tais problemas são os problemas de busca mais difíceis em **NP**

Pare para pensar: eles são os problemas MAIS DIFÍCEIS, englobam todos os outros. E são **equivalentes** entre si.

Os problemas mais difíceis de ser resolver:

Se eles são "os mais difíceis" de NP e resolvermos eficientemente somente um deles, então resolvemos eficientemente todos os problemas em NP.

Reduções, novamente

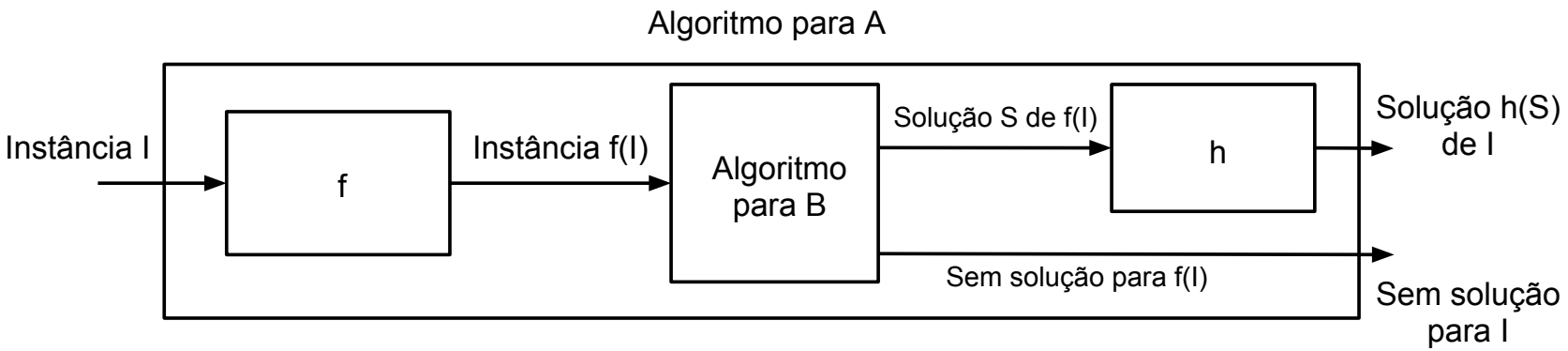
Para tentar entender essas **coisas esquisitas**, vamos especializar a definição de redução para problemas de busca.

Reduções (para problemas **NP**)

Uma **redução** de um problema de busca A para um problema de busca B são **dois algoritmos** de tempo polinomial f e h que:

- f transforma qualquer instância I de A em uma instância $f(I)$ de B,
- h transforma qualquer solução S de $f(I)$ de volta em uma solução $h(S)$ de I,
- Se $f(I)$ não tem solução, então I também não tem.

Reduções, novamente



NP-Completo

Agora podemos finalmente definir a classe dos problemas mais difíceis da busca:

Classe NP-Completo

Um problema de busca é **NP**-completo se todos os problemas de busca se reduzem a ele.

Esta é uma exigência muito forte!

- Para um problema ser **NP**-completo, ele deve poder resolver todos os problemas de busca do mundo!

Seria muito notável e muito impressionante se tais problemas existissem!

DE FATO ELES EXISTEM: os problemas da coluna esquerda da tabela que vimos anteriormente são os exemplos mais famosos.

Objetivos da disciplina a partir de agora

Nas próximas aulas, nosso objetivo será:

1. Ver as **reduções** entre esses problemas difíceis.
2. Entender como **todos** os problemas de busca se reduzem a eles.

As Duas Maneiras de Usar Reduções

Até agora o propósito de uma redução de um problema A para B foi **simples e honrosa**:

- Sabemos como resolver B de forma eficiente, e queremos usar esse conhecimento para resolver A.

De certa forma, B é mais genérico (mais difícil) que A.

No entanto, as reduções de A para B servem também para um objetivo mais **perverso**:

- Sabemos que A é difícil, e nós usamos a redução para provar que B é difícil também!

Algumas Exceções

FATORAÇÃO:

Encontre todos os fatores primos de um número inteiro dado.

A dificuldade de FATORAÇÃO é de uma natureza diferente dos outros problemas de busca difíceis.

Por exemplo, ninguém acredita que FATORAÇÃO seja NP-Completo.

Uma diferença importante é que neste problema não temos a frase (agora já familiar) **"ou diga que não existe"**.

Um número SEMPRE pode ser fatorado em números primos.

Algumas Exceções

Outra diferença (possivelmente relacionada): FATORAÇÃO sucumbe ao poder da computação quântica, enquanto SAT, TSP e as outros problemas NP-Completos até agora não...

Na Prática:

A grande maioria dos problema em **NP** que se conhece estão em **P** ou **NP-Completo**.

Notáveis exceções (além da fatoração): **Isomorfismo de Grafos**, **Equilíbrio de Nash**.