Problemas NP-Completos

Mário César San Felice

Departamento de Computação da Universidade Federal de São Carlos

felice@ufscar.br

23 de novembro de 2018

Introdução

Vamos apresentar uma classe de problemas muito importante

Relacionada com questões centrais da computação

Etapas:

- Tratabilidade
- Classes P e NP
- Redução
- Completude
- Problemas NP-Completos

Tratabilidade

Intuição:

- Problema é tratável se existem algoritmos eficientes para ele
- Problema é intratável se tais algoritmos não existem

Até este ponto do curso praticamente todos os problemas que encaramos eram tratáveis

Mas nada garante que seja sempre este o caso

Tratabilidade

Definição:

- Problema é "**tratável**" se pode ser resolvido em tempo polinomial, i.e., $\mathrm{O}(n^k)$
- Problema é "**intratável**" se não é conhecido algoritmo polinomial que o resolva

Note que, para k grande resolver o problema pode ser inviável

E existem algoritmos de tempo exponencial eficientes na prática

Mas, em geral, definição captura bem a dificuldade dos problemas

Classes P e NP

Classes de complexidade relacionadas à definição de tratabilidade

P é a classe dos **problemas resolvidos em tempo polinomial**

Exemplos de problemas em P:

- Caminho mais curto
- Ordenação
- Árvore geradora mínima
- Fluxo em redes
- Multiplicação de matrizes

Classes P e NP

Nem todo problema está em P :(

Existem problemas para os quais não existe qualquer algoritmo

- Ex.: problema da Parada

Existem outros para os quais, por mais que se busque, não são conhecidos algoritmos polinomiais

- Ex.: problema do Caixeiro Viajante (TSP)

Mas, a princípio, isso não garante que tais algoritmos não existem

Como descrever uma classe que englobe problemas como o TSP?

Classes P e NP

NP é a classe dos problemas que:

- Tem **solução com tamanho polinomial** na entrada
- **Podemos verificar** se **uma solução** é correta também **em tempo polinomial**

Vale citar que estas definições são algorítmicas

Definições formais destas classes (e explicações para seus nomes) vem da área de computabilidade

Em particular, NP não significa "Não Polinomial"

E sim "Nondeterministic Polynomial"

Classes P e NP

Todo problema em NP é resolvível usando busca por força bruta:

- Soluções com tamanho polinomial implicam espaço de soluções com tamanho no máximo exponencial
- Cada solução pode ser testada em tempo polinomial

Exemplos:

- Satisfatibilidade (SAT)
- Cobertura por Vértices
- Problema do Caixeiro Viajante (TSP)

Classes P e NP

Situação dos problemas:

- Temos problemas em P, que também estão em NP (por que?)
- Temos problemas em NP, que não sabemos se estão em P

Muitos problemas importantes na segunda situação

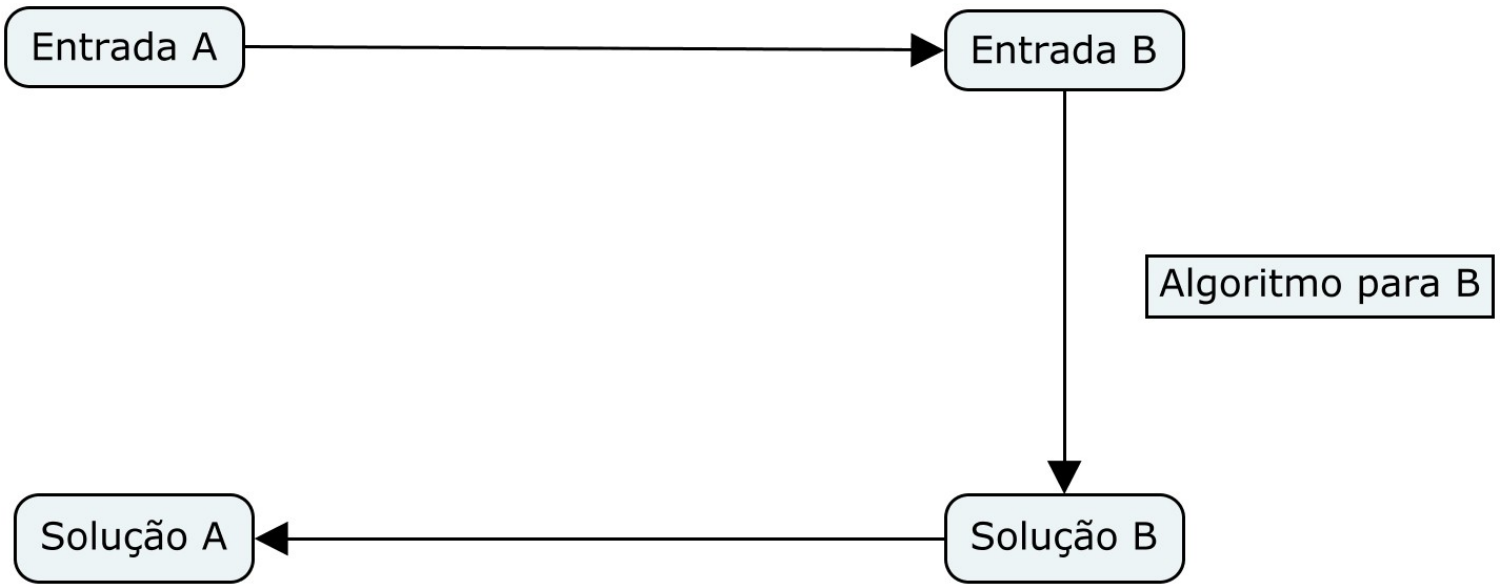
Como corroborar a dificuldade destes problemas?

Vamos utilizar evidências relativas, usando:

- Redução
- Completude

Redução

Problema A se reduz a B se conseguimos resolver qualquer instância de A usando um algoritmo que resolve B

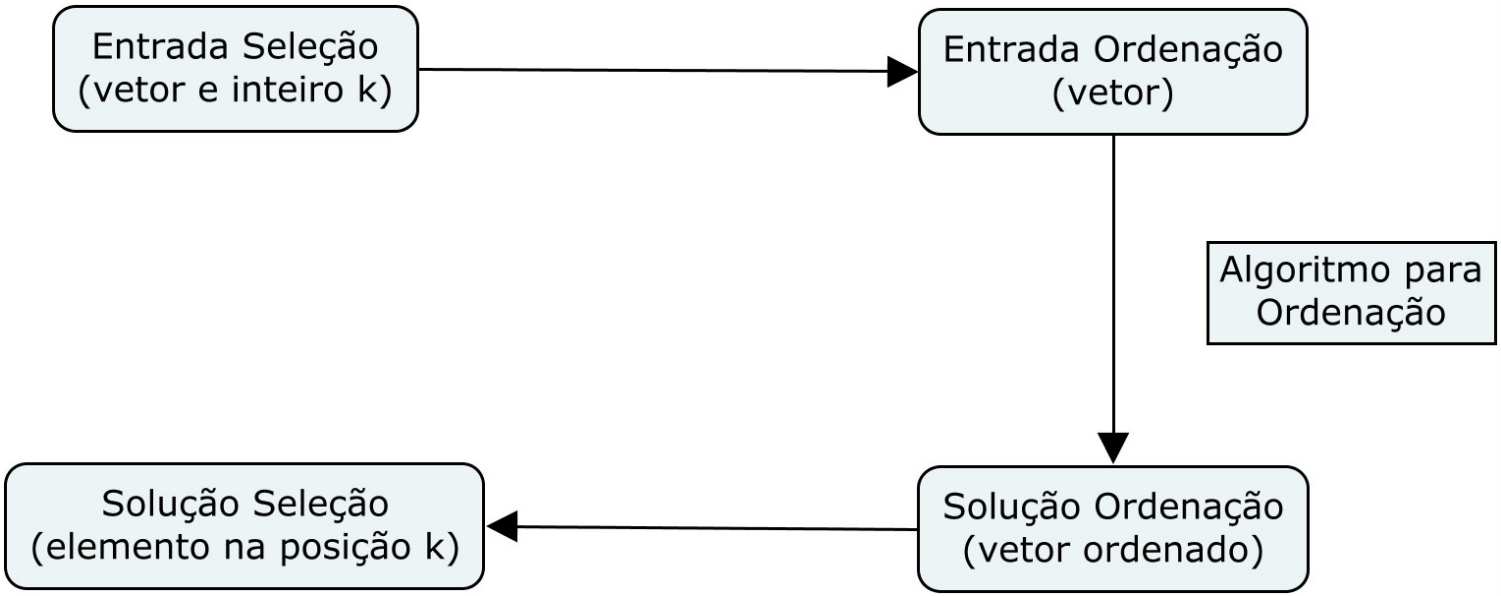


Transformações entrada A em entrada B e solução B em solução A

Se transformações são polinomiais, A é polinomialmente redutível a B

Redução

Exemplo: Seleção em Ordenação



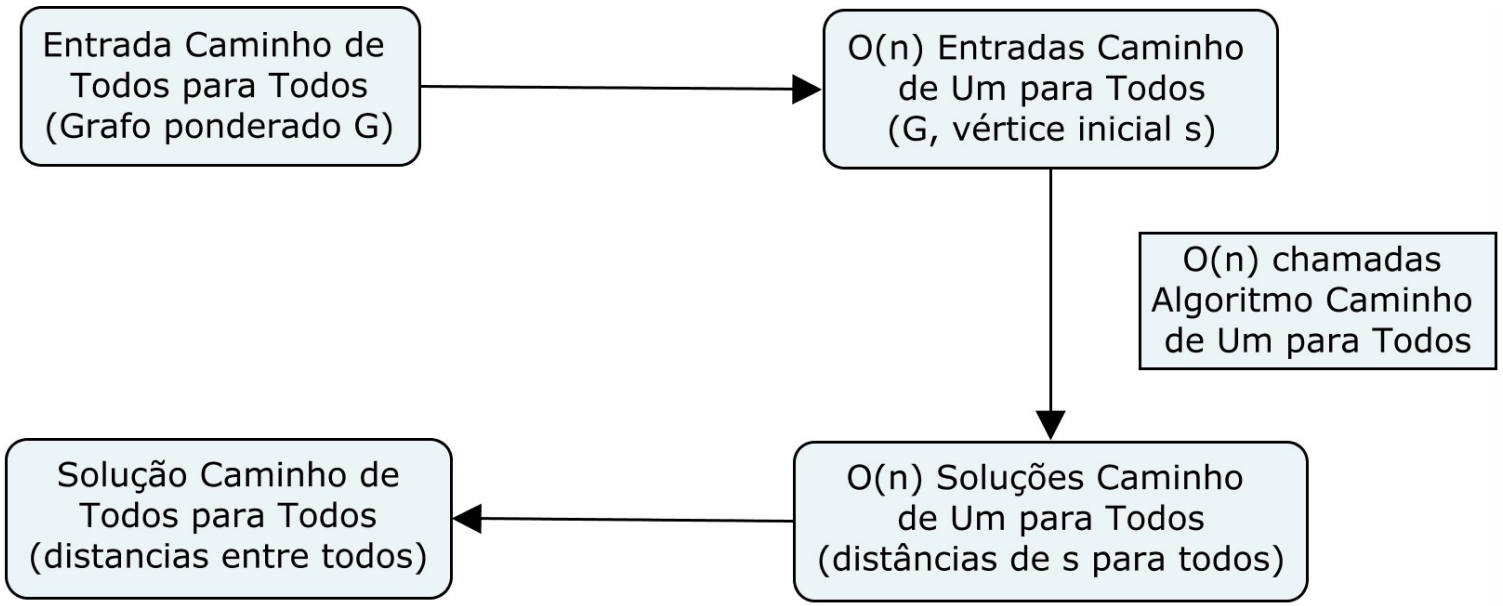
Esta redução resolve o problema da seleção em tempo $\mathrm{O}(n \log n)$

Note que é possível resolver a seleção em tempo linear

Redução

Numa redução o algoritmo para B pode ser utilizado várias vezes

Exemplo: Caminho Todos para Todos em Caminho Um para Todos



Notem que o algoritmo obtido ainda é polinomial

Redução

Redução entre problemas é uma técnica valiosa para o projeto de algoritmos

Permite utilizar soluções já conhecidas para resolver novos problemas

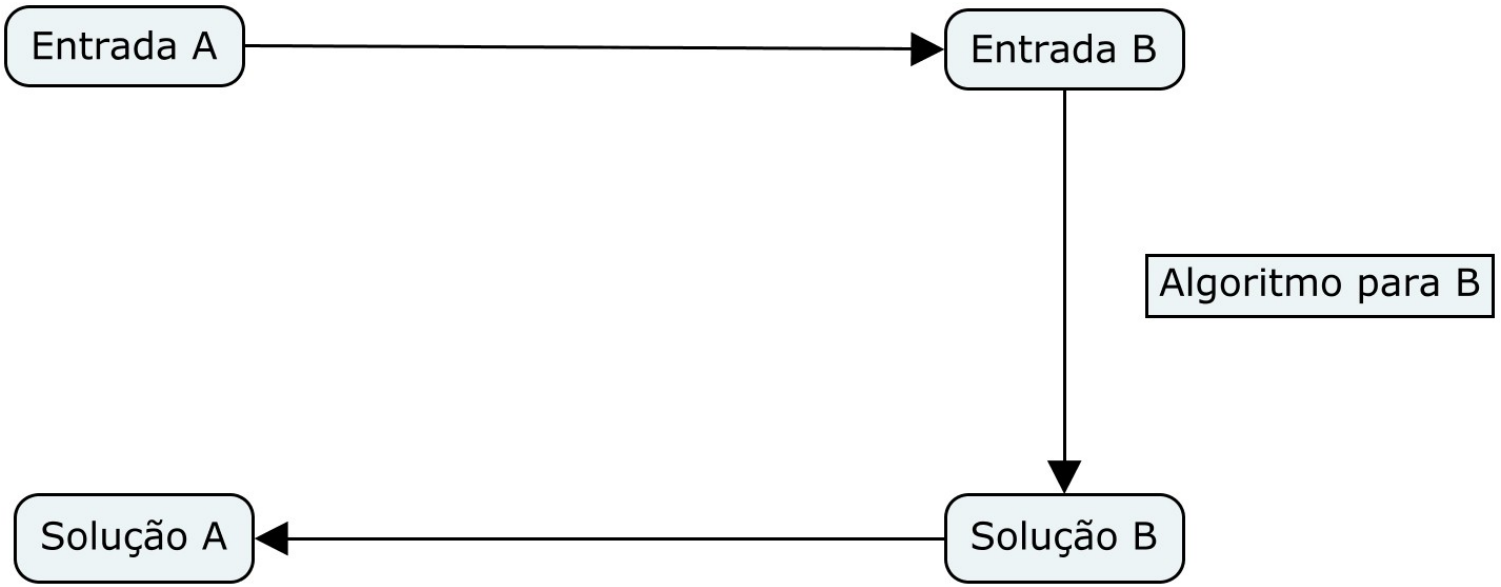
Um projetista de algoritmos sempre deve se perguntar:

"Será que este problema não se parece com algum que eu já conheço?"

Redução

Redução usada para atestar a dificuldade relativa de um problema

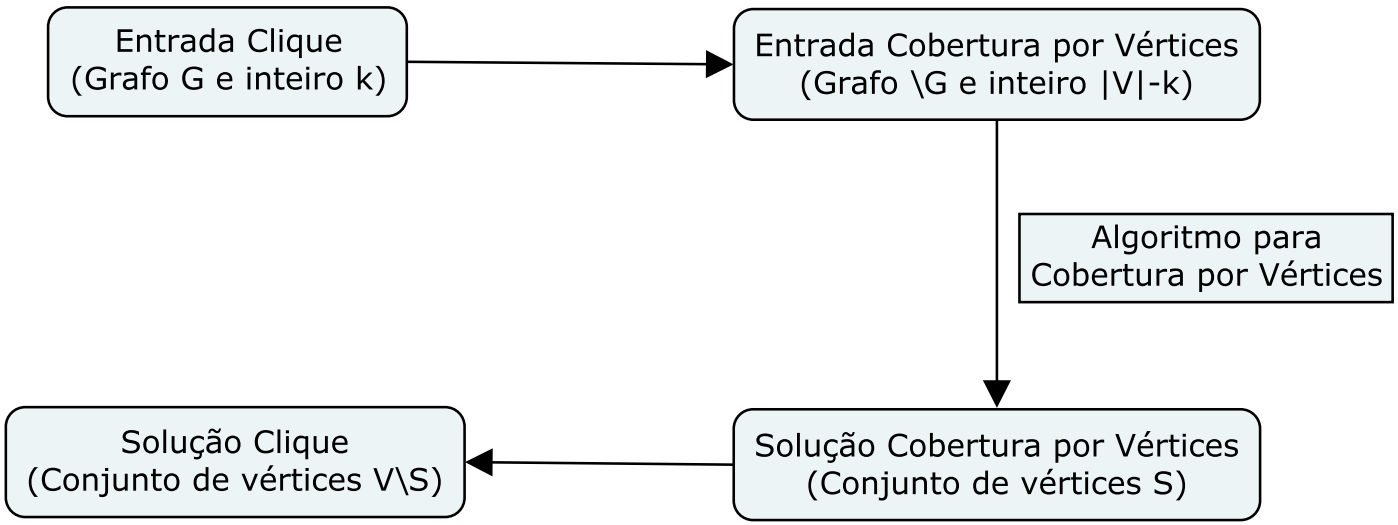
Como provar que um problema é tão difícil quanto outro?



B não pode ser mais fácil que A

Redução

Exemplo: Clique em Cobertura por Vértices



Se Clique não tem algoritmo polinomial

Então Cobertura por Vértices também não tem

Redução

Interpretações para "A é redutível a B":

- (Positiva) A é no máximo tão difícil quanto B
- (Negativa) B é pelo menos tão difícil quanto A

Primeira expande o conjunto de problemas tratáveis

Segunda expande o conjunto dos intratáveis

Completude

Problema π é C-Completo se:

- π **está em C**
- **Todo problema em C é redutível a** π

Podemos pensar que π é o problema mais difícil em C

Notem quão forte é esta definição

E que não é óbvio que algum problema em C a satisfaça

Completude

Apesar disso, vamos supor que existam problemas C-Completos

Sendo π um problema em C

Para mostrar que π é C-Completo temos que:

- Escolher um problema C-Completo π'
- Reduzir o problema π' para π

Isso funciona pois:

- Todo problema em C é redutível a π'
- E por consequência é redutível a π

Problemas NP-Completos

Voltamos à questão:

"Como mostrar a dificuldade dos problemas em NP (que não sabemos se estão em P)?"

Um forte atestado de dificuldade seria mostrar que todo problema de NP reduz pra um desses problemas

Já que isso mostraria que esse problema é o mais difícil dentre todos de NP e que bastaria resolver ele pra resolver todos os outros

Mas, será que a classe dos problemas NP-Completos é não vazia?

Problemas NP-Completos

Por incrível que pareça, existem muitos problemas NP-Completos

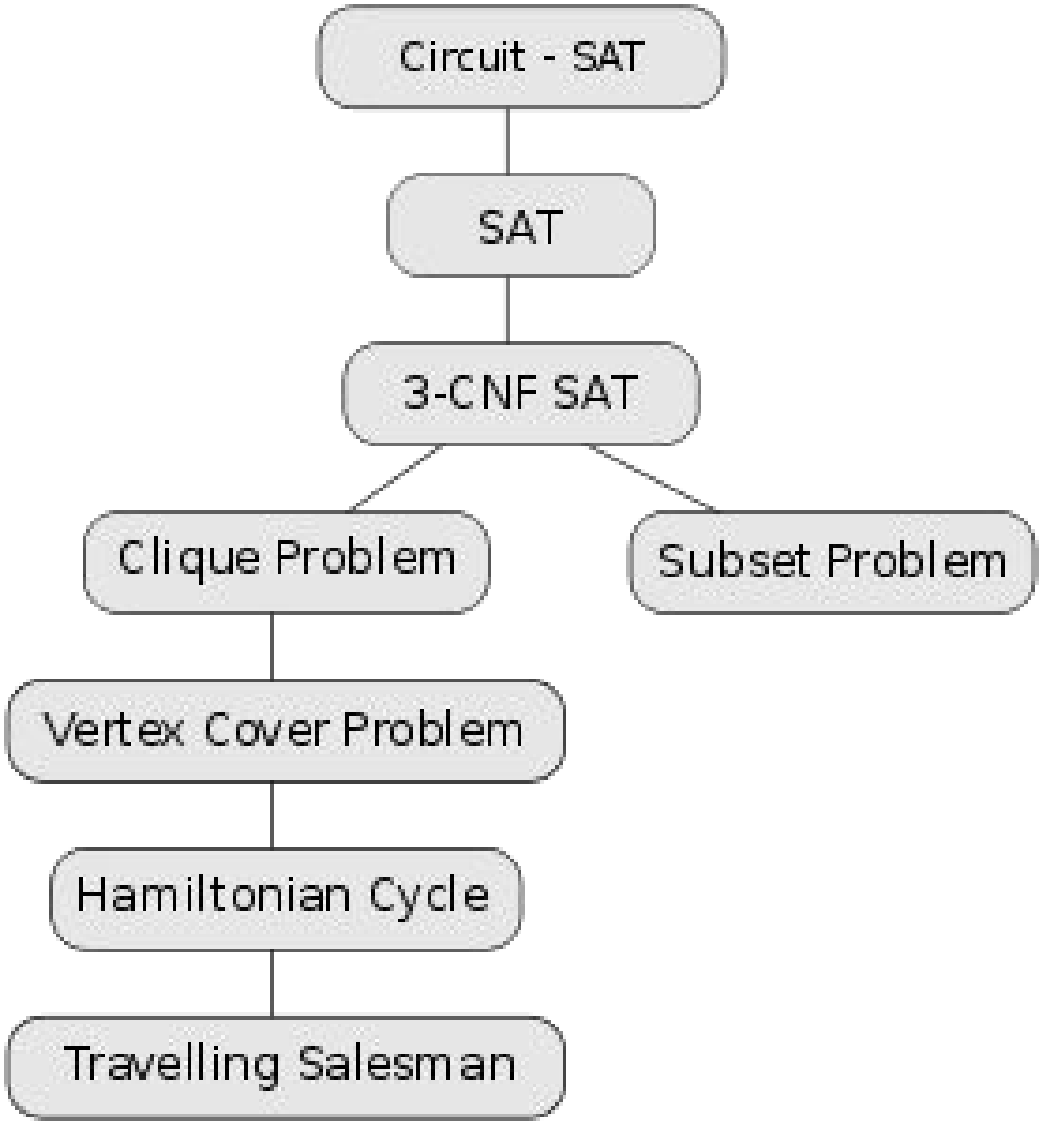
Sendo um exemplo notável o problema da satisfatibilidade (SAT)

Já sabemos que para mostrar que um problema é NP-Completo

Basta reduzir qualquer problema NP-Completo a ele

Assim, muitos outros problemas foram provados NP-Completos

Problemas NP-Completos



Problemas NP-Completos

Se existir algoritmo polinomial para um problema NP-Completo

Conseguimos resolver todos os problemas em NP em tempo polinomial

Provando que P=NP

Respondendo assim à maior questão em aberto da ciência da computação

Problemas NP-Completos

A ligação entre NP-Completude e a questão P vs. NP demonstra a importancia desta classe para a ciência da computação

Mas por que um projetista de algoritmos deveria se preocupar?

Porque problemas NP-Completos são extremamente comuns

E existe pouca esperança de obter um algoritmo polinomial para eles

Problemas NP-Completos

Assim, se voce se deparar com um problema difícil

É importante saber verificar se ele é NP-Completo

Para tratá-lo de modo adequado

Hoje não apresentamos opções para lidar com problemas NP-Completos

Mas mostramos como identificá-los

Problemas NP-Completos