Complexidade computacional

Marina Andretta

ICMC-USP

15 de setembro de 2015

Baseado no livro Uma introdução sucinta a Algoritmos de Aproximação, de M. H. Carvalho, M. R. Cerioli, R. Dahab, P. Feofiloff, C. G. Fernandes, C. E. Ferreira, K. S. Guimarães, F. K. Miyazawa, J. C. Piña Jr., J. A. R. Soares e Y. Wakabayashi.

Palavras

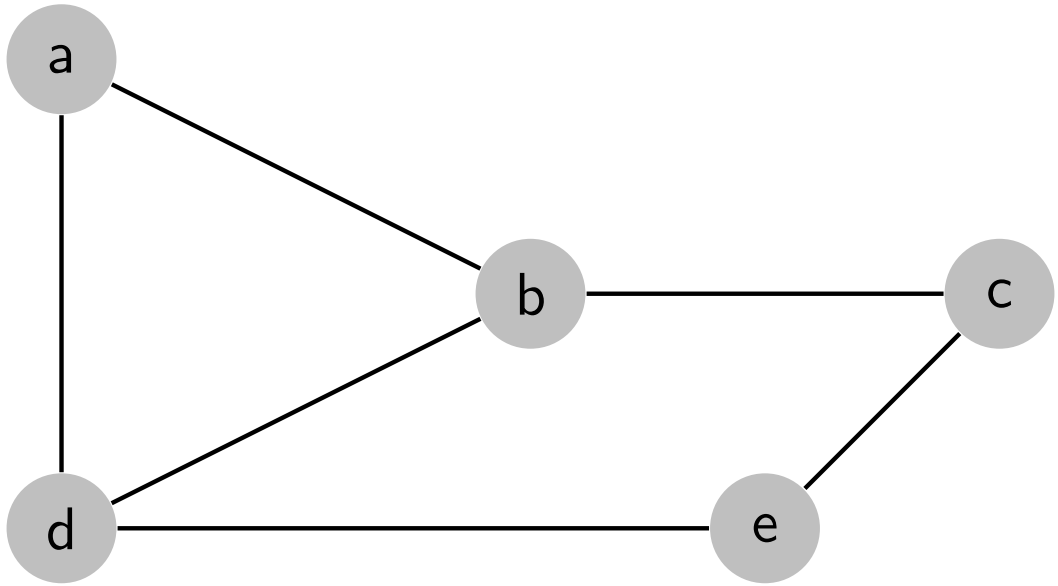
Para resolver um problema usando um computador é necessário descrever os dados do problema por meio de uma sequência de caracteres.

Qualquer sequência de caracteres será chamada de uma palavra.

Não é difícil representar números inteiros, números racionais, vetores, matrizes, grafos, etc. por meio de palavras.

Palavras

Por exemplo, o grafo com vértices $\{a, b, c, d, e\}$ e arestas $\{\{a, b\}, \{b, c\}, \{a, d\}, \{b, d\}, \{c, e\}, \{d, e\}\}$



pode ser representado pela palavra
$(\{a, b, c, d, e\}, \{\{a, b\}, \{b, c\}, \{a, d\}, \{b, d\}, \{c, e\}, \{d, e\}\})$.

Palavras

Em geral, não fazemos distinção entre um objeto e uma palavra que o representa.

Um vetor racional c indexado por um conjunto finito E pode ser representado por uma sequência de pares (e, c_e), onde e é um elemento de E.

Por exemplo, a palavra

$$((\{a,b\},2),(\{b,c\},-1),(\{a,d\},3/2),(\{b,d\},-7),(\{c,e\},0),(\{d,e\},1)\})$$

codifica um vetor indexado pelo conjunto das arestas do grafo anterior.

Palavras

O tamanho de uma palavra w, denotado por $\langle w \rangle$, é o número de caracteres usados em w (contando-se multiplicidades).

Como faremos estimativas "a menos de uma constante", não é necessário contar rigorosamente os caracteres $\{$, $\}$, $($, $)$ e $,$ dos exemplos anteriores.

Assim, o tamanho de um vetor c reduz-se à soma dos tamanhos de seus componentes, ou seja,

$$\langle c \rangle = \sum_e \langle c_e \rangle.$$

Palavras

O tamanho dos números inteiros e racionais merece especial atenção.

O tamanho de um inteiro α é essencialmente $log|\alpha|$.

Analogamente, se α é um número inteiro e β é um número inteiro não nulo então o tamanho do racional α/β é, essencialmente, $log|\alpha| + log|\beta|$.

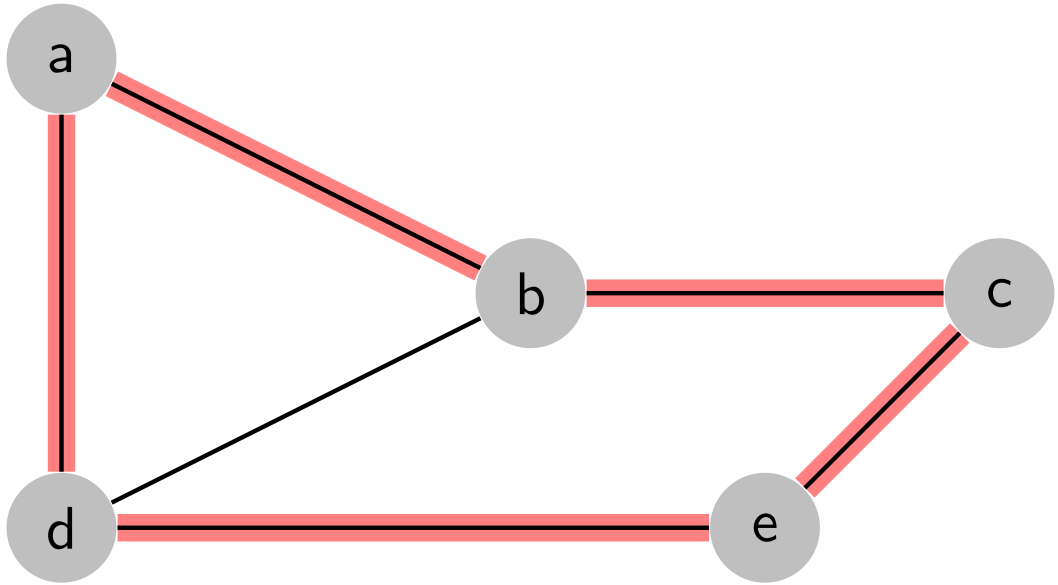
Problemas

Informalmente, um problema é uma questão ou tarefa.

Exemplos de problemas são: o problema do circuito hamiltoniano (*Hamiltonian circuit problem*), denotado pela sigla HCP e o problema da árvore geradora mínima (*minimum spanning tree problem*), denotado pela sigla MST.

Problema do circuito hamiltoniano - $\mathrm{HCP}(G)$

Dado um grafo G, ele possui um circuito hamiltoniano? Ou seja, existe um circuito que passe por todos os vértices de G exatamente uma vez?

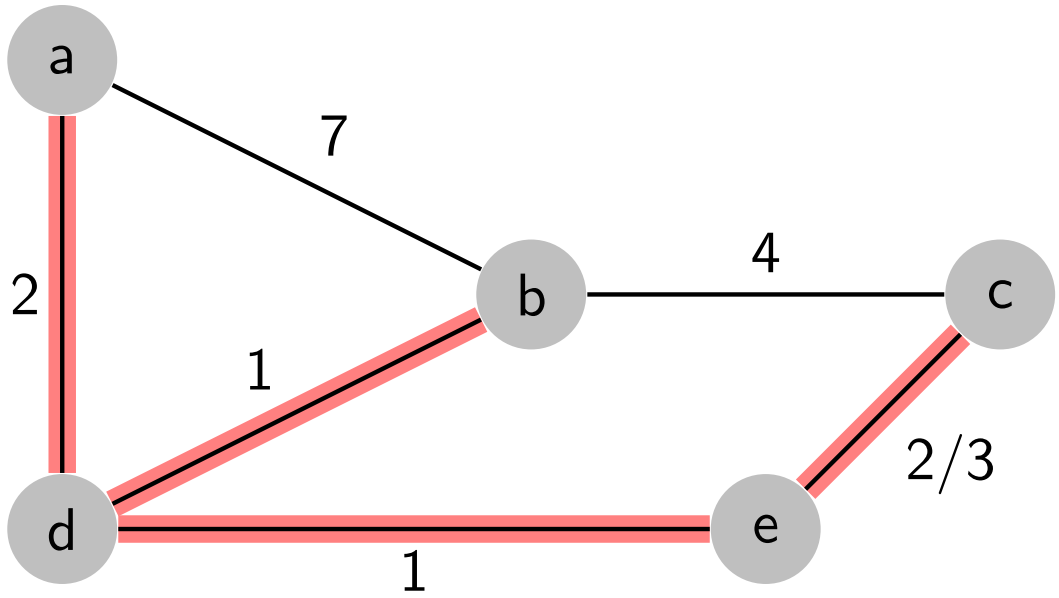


Problema da árvore geradora mínima - $\mathrm{MST}(G, c)$

- Um grafo G é conexo se para cada par de vértices u e v de G há um caminho de u para v.
- Uma árvore é um grafo conexo sem circuitos.
- Uma árvore geradora $T = (V_T, E_T)$ de um grafo $G = (V, E)$ é uma árvore com os mesmos vértices de G e que tem como arestas um sub-conjunto das arestas de G. Ou seja, $V_T = V$ e $E_T \subseteq E$.

Problema da árvore geradora mínima - $\mathrm{MST}(G, c)$

Dados um grafo G e um custo c_e em $\mathbb{Q}_{\geq}$ para cada aresta e, encontrar uma árvore geradora de custo mínimo.



Problemas

Cada conjunto específico de dados de um problema define uma instância do problema.

O tamanho de uma instância é o tamanho de uma palavra que representa a instância.

Problemas

Um problema que pede uma resposta do tipo SIM ou NÃO é chamado de problema de decisão.

Um problema que procura um elemento de um conjunto de soluções viáveis que seja melhor possível em relação a algum critério é um problema de otimização.

Os problemas HCP e MST são exemplos de problemas de decisão e otimização, respectivamente.

Algoritmos e modelo de computação

Um algoritmo é uma sequência finita de instruções ou operações que resolve um problema.

A formalização matemática clássica de um algoritmo é a maquina de Turing.

Um modelo de computação é uma descrição abstrata e conceitual (não necessariamente realista) de um computador que será usado para executar um algoritmo.

Um modelo de computação especifica as operações elementares que um algoritmo pode executar e o critério empregado para medir a quantidade de tempo que cada operação consome.

Modelo de computação

Exemplos de operações elementares típicas são operações aritméticas entre números e comparações.

O modelo de computação RAM (*random access machine*) baseia-se na manipulação de caracteres. Esse modelo é capaz de executar operações envolvendo números racionais.

Usaremos o RAM durante todo o curso.

Modelo de computação

Um dos critérios para medir o consumo de tempo de cada operação é o critério logarítmico (*logarithmic cost criterion*). Neste critério, o consumo de tempo de cada operação depende do tamanho dos operandos.

Por exemplo, a multiplicação de dois números α e β consome essencialmente $\langle\alpha\rangle\langle\beta\rangle$ unidades de tempo.

Modelo de computação

Um outro critério para medir o consumo de tempo, aparentemente menos realista, é o critério uniforme (*uniform cost criterion*), que supõe que cada operação elementar consome uma quantidade de tempo constante, ou seja, o consumo de tempo de cada operação elementar independe do tamanho dos operandos.

Este modelo é equivalente ao anterior se o tamanho dos operandos for limitado por uma função polinomial do tamanho da instância do problema.

Este critério de consumo de tempo constante por operação elementar, juntamente com operandos de tamanho limitado, será empregado em todo o curso.

Análise de algoritmos

Ao analisarmos um algoritmo em um determinado modelo de computação queremos estimar o seu consumo de tempo como uma função do tamanho da instância do problema.

Por exemplo, exprimimos o consumo de tempo de algoritmos para o $\mathrm{MST}(G, c)$ como uma função de $\langle (G, c) \rangle$.

Análise de algoritmos

Ao expressar a quantidade de tempo consumida por um algoritmo, ignoramos os fatores constantes.

Mais precisamente, dizemos que um algoritmo A resolve um problema em tempo $O(f(n))$, para uma função f de $\mathbb{Z}_{\geq}$ em $\mathbb{Z}_{\geq}$, se existe uma constante c tal que a quantidade de tempo consumida por A para resolver uma instância I do problema é limitada por $cf(\langle I \rangle)$.

Isto corresponde à chamada análise do pior caso do algoritmo.

Algoritmos polinomiais

O algoritmo A é dito polinomial se resolve o problema em tempo $O(p(n))$ para alguma função polinomial p.

É claro que o conceito de algoritmo polinomial depende não só do algoritmo, mas também do modelo de computação.

Critério de eficiência: classes P, NP e $co\text{-}NP$

Dizemos que um algoritmo é eficiente se ele é um algoritmo polinomial.

A classe de todos os problemas de decisão que podem ser resolvidos por algoritmos polinomiais é denotada por P.

Critério de eficiência: classes P, NP e co-NP

A classe NP (abreviação de *nondeterministic polynomial time*) é composta pelos problemas de decisão para os quais uma resposta afirmativa possui um certificado que pode ser verificado em tempo polinomial.

Ou seja, a classe NP é formada pelos problemas de decisão para os quais existe um problema Π' em P e uma função polinomial $p(n)$ tais que, para cada instância I do problema Π, existe um objeto C com $\langle C \rangle \leq p(\langle I \rangle)$ tal que a resposta a $\Pi(I)$ é SIM se e somente se a resposta a $\Pi(I, C)$ é SIM.

Critério de eficiência: classes P, NP e co-NP

O objeto C é dito um certificado polinomial (ou certificado curto) da resposta SIM a $\Pi(I)$.

Com um algoritmo polinomial para Π', um verificador incrédulo pode, após gastar uma quantidade de tempo polinomial, testar a validade de uma resposta SIM.

Critério de eficiência: classes P, NP e co-NP

Por exemplo, se a resposta a uma dada instância G do problema do circuito hamiltoniano é SIM então um circuito hamiltoniano em G é um objeto que certifica a resposta: dados um grafo G e um circuito H de G pode-se verificar em tempo $O(\langle G \rangle)$ se H é um circuito hamiltoniano.

Logo, o problema HCP pertence a NP.

Critério de eficiência: classes P, NP e $co\text{-}NP$

Claramente $P \subseteq NP$, já que se Π é um problema em P, então pode-se tomar a sequência de instruções realizadas por um algoritmo polinomial para resolver $\Pi(I)$ como certificado polinomial da resposta SIM a $\Pi(I)$.

Acredita-se que a classe NP é maior que a classe P, ainda que isso não tenha sido provado até agora.

Este é o intrigante problema matemático conhecido pelo rótulo "$P \neq NP$?".

Critério de eficiência: classes P, NP e $co\text{-}NP$

Um problema de decisão Π está em $co\text{-}NP$ se admite um certificado polinomial para a resposta NÃO.

É claro que $P \subseteq NP \cap co\text{-}NP$.

Redução

Uma redução de um problema Π a um problema Π' é um algoritmo eficiente σ, que transforma uma instância de Π em uma instância de Π', tal que

- se I é uma instância de Π que tem resposta SIM, então $\sigma(I)$ é uma instância de Π' que tem resposta SIM;
- se I é uma instância de Π que tem resposta NÃO, então $\sigma(I)$ é uma instância de Π' que tem resposta NÃO.

Redução

Usamos a notação $\Pi \leq_P \Pi'$ para denotar a existência de uma redução de Π a Π'.

Note que, se $\Pi \leq_P \Pi'$, a existência de um algoritmo polinomial para resolver o problema Π' implica na existência de um algoritmo polinomial para resolver Π.

Exemplo de redução

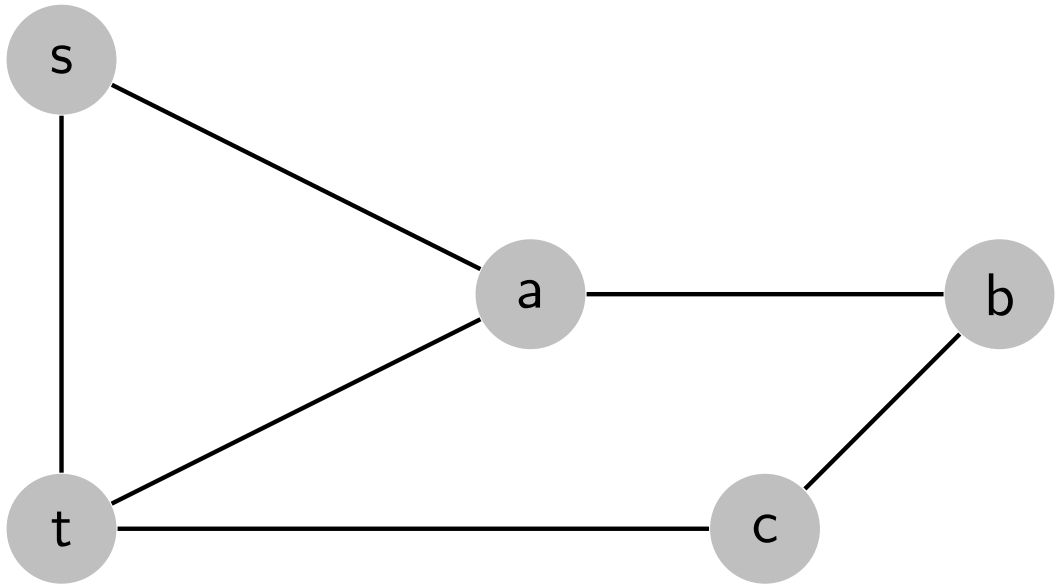
Considere o problema do caminho Hamiltoniano (s, t) (*Hamiltonian (s, t)-path problem*), denotado pela sigla HPP:

Dado um grafo G e dois vértices s e t de G, existe um caminho de s a t, que passe exatamente por todos os vértices de G?

Vamos reduzir o problema $\mathrm{HPP}(G, s, t)$ ao problema $\mathrm{HCP}(G)$.

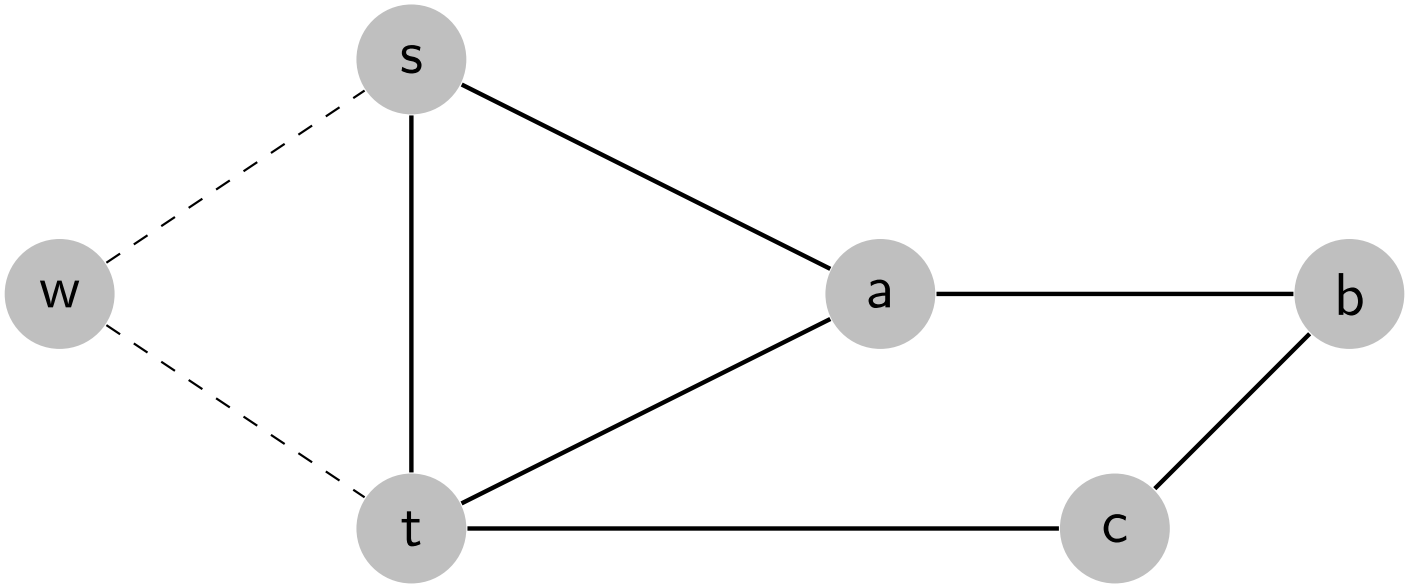
Exemplo de redução

Tome uma instância HPP, ou seja, um grafo $G = (V, E)$ e dois vértices s e t ao problema $\mathrm{HCP}(G)$.



Exemplo de redução

Construa o grafo $G' = (V', E')$ com os mesmos vértices de G, acrescidos de um vértice w, e as mesmas arestas de G, acrescidas das arestas $\{w, s\}$ e $\{s, t\}$. Ou seja, $V' = V \cup \{w\}$ e $E' = E \cup \{\{w, s\}, \{w, t\}\}$.



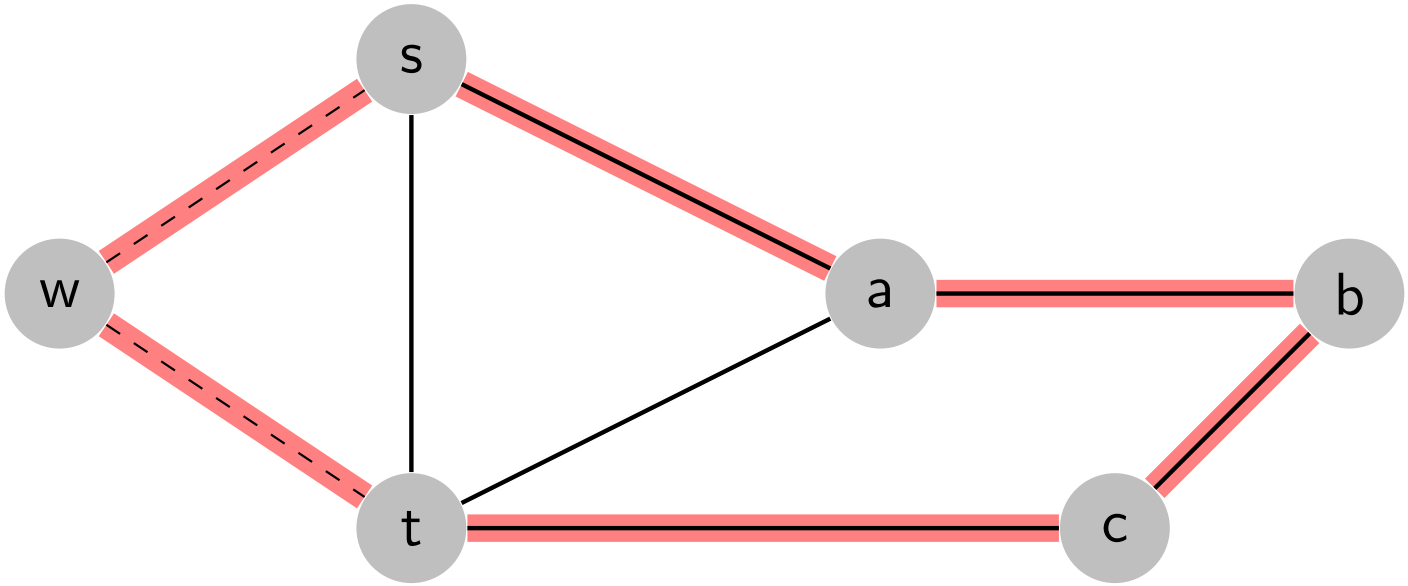
Esta transformação é polinomial no tamanho da instância de HHP.

Exemplo de redução

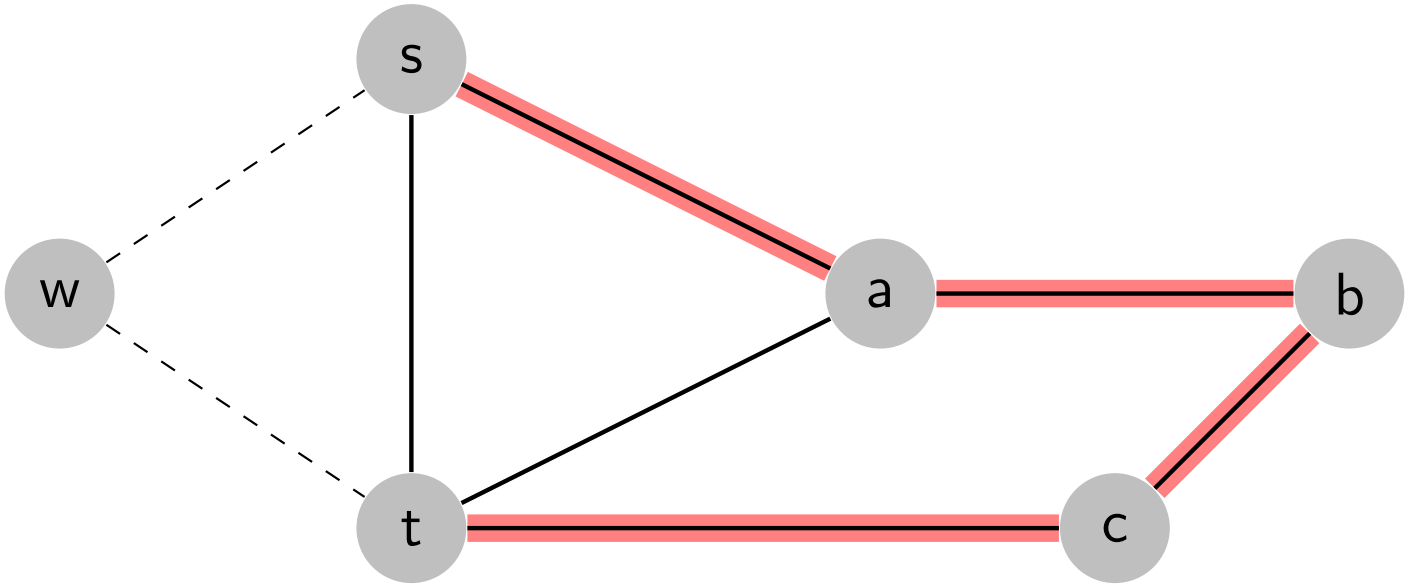
Note que, se é possível encontrar um circuito Hamiltoniano em G', é possível encontrar um caminho Hamiltoniano de s a t em G. Para isto, basta eliminar o trecho $\{s, w, t\}$ do circuito obtido.

Mais ainda, se é possível encontrar um caminho Hamiltoniano de s a t em G, basta unir este caminho ao trecho $\{s, w, t\}$ para obter um circuito Hamiltoniano em G',

Exemplo de redução



Exemplo de redução



Exemplo de redução

Portanto, $\mathrm{HPP} \leq_P \mathrm{HCP}$.

Ou seja, se existir um algoritmo polinomial para resolver o problema HCP, também existe um algoritmo polinomial para resolver HPP.

Isso dá uma ideia de que o problema HCP é ao menos tão difícil quanto o problema HPP.

NP-completude

Um problema Π em *NP* é *NP-completo* se cada problema em *NP* pode ser reduzido a Π.

É evidente que se Π' pertence a P e Π pode ser reduzido a Π', então Π pertence a P.

Isto implica que existe um algoritmo polinomial para um problema *NP-completo* se e somente se $P = NP$.

NP-completude

O primeiro problema que se demonstrou (por Cook e Levin) ser *NP-completo* é o problema da satisfatibilidade (*satisfiability problem*), denotado por $\textsc{Sat}$.

Problema $\textsc{Sat}(C)$: Dada uma coleção C de cláusulas booleanas, existe uma valoração que satisfaz todas as cláusulas em C?

NP-completude

Um problema Π, não necessariamente em *NP*, é *NP-difícil* se a existência de um algoritmo polinomial para Π implica em $P = NP$.

Por exemplo, considere o problema MaxSat.

Problema MaxSat(V, C): Dada uma coleção C de cláusulas booleanas sobre um conjunto V de variáveis, encontrar uma valoração x de V que satisfaça o maior número de cláusulas de C?

Note que, se temos um algoritmo polinomial A para resolver MaxSat, podemos definir um algoritmo A' da seguinte forma: execute A; verifique se todas as de cláusulas de C foram satisfeitas; em caso positivo, devolva SIM; em caso negativo, devolva NÃO.

Claramente, A' é um algoritmo polinomial que resolve Sat em tempo polinomial.

Como Sat é *NP-completo*, a existência de A (e, consequentemente, de A') implica em $P = NP$. Portanto, MaxSat é *NP-difícil*,

NP-completude

Alguns exemplos de problemas *NP-difíceis* são:

- Escalonamento;
- MinCC - Cobertura mínima por conjuntos;
- Mochila;
- TSPM - Caixeiro viajante métrico;
- TSP - Caixeiro viajante;
- MinMCut - Multicorte mínimo;
- MinCV - Cobertura mínima por vértices;
- Empacotamento.