UNIP
UNIVERSIDADE PAULISTA
Interativa

Aspectos Teóricos da Computação

APRESENTAÇÃO

Aspectos Teóricos da Computação

I – EMENTA

Máquinas de Turing e a tese de Turing-Church, Problemas Solucionáveis e Não Solucionáveis, Complexidade Computacional e Problemas NP-Completos. Problemas NP-Difíceis. Teorema da Incompletude de Gödel.

II – OBJETIVOS GERAIS

Permitir que os alunos travem contato com resultados teóricos da Ciência da Computação e avaliem adequadamente a importância dos mesmos.

III – OBJETIVOS ESPECÍFICOS

– Explicar a tese de Turing-Church e seu significado;

– Explicar como alguns problemas não apresentam solução algorítmica;

– Apresentar exemplos de problemas que não são computáveis;

– Definir as classes de problemas P e NP;

– Explicar o que são problemas NP-completos e NP-difíceis;

– Apresentar o Teorema da Incompletude de Gödel

IV – CONTEÚDO PROGRAMÁTICO

1. Introdução

1.1 Hierarquia de Chomsky

1.2 Máquina de Estados Finitos

1.3 Máquina de Turing: modelo que simula procedimentos computacionais mais gerais que a máquina de estados finitos;

2 Máquinas de Turing – Parte I

2.1 A definição formal da Máquina de Turing

2.2 A computação na Máquina de Turing: funções recursivas e linguagens recursivamente enumeráveis

3 Máquinas de Turing – Parte II

3.1 Extensões da Máquina de Turing

3.2 Máquinas de Turing com Acesso Aleatório

3.3 Máquinas de Turing Não Determinísticas

4. Máquinas de Turing como Calculadora de Funções Numéricas

5 Problemas Indecidíveis – Parte I

5.1 A Tese de Turing Church

5.2 Máquinas de Turing Universais

5.3 O Problema da Parada

6. Problemas Indecidíveis – Parte II.

6.1Problemas Não Solucionáveis sobre as Máquinas de Turing e sobre as Gramáticas.

6.2 Propriedades das Linguagens Recursivas

7. Tempo de Execução de um Programa

7.1Comportamento Assintóptico de Funções

7.2Classes de Comportamento Assintóptico: complexidade logarítmica, complexidade polinomial (complexidade linear, complexidade quadrática etc.); complexidade exponencial;

8. Complexidade Computacional – Parte I

8.1 A Classe P: definição;

8.2 Grafos Eulerianos e Hamiltonianos;

9. Complexidade Computacional – Parte II

9.1 Problema do Caixeiro Viajante;

9.2 Clique (Máximo e Mínimo);

9.3 Problema da Cobertura dos Nós;

9.4 Problema do Particionamento

10. Complexidade Computacional – Parte III

10.1 Satisfabilidade e Satisfabilidade Booleana

10.2 Problema da Mochila;

11. Completude NP e Problemas NP-Difíceis

11.1 Definição

11.2 Teorema de Cook

11.3 Problemas NP-Difíceis.

12. O Teorema de Gödel

BIBLIOGRAFIA BÁSICA

DIVERIO, T. A.; MENEZES, P. B. "Teoria da Computação". Porto Alegre: Bookman, 2008.

LEWIS, Harry R. & PAPADIMITRIOUS, Christos H. "Elementos de Teoria da Computação". 2. ed. Porto Alegre: Bookman, 2004.

SIPSER, Michael. "Introdução à Teoria da Computação". São Paulo: Thomson Pioneira, 2007.

Complementar BRAINERD, W. S.; LANDWEBER, L. H. "Theory of computation". New York: John Wiley & Sons, 1974.

DIVERIO, T. A.; MENEZES, P. B. "Teoria da Computação Máquinas Universais e Computabilidade". Série Livros Didáticos Número 5, Instituto de Informática, da UFRGS, Editora Sagra Luzzatto, 1. Ed. 1999.

HOPCROFT J.; ULLMAN, J.; MOTWANI, R. "Introdução à Teoria dos Autômatos, Linguagens e Computação". Rio de Janeiro: Campus, 2002.

LEISERSON, C. E. ; RIVEST, R. L. ; CORMEN, T. H.; STEIN, C. "Algoritmos – Teoria e Prática". Rio de Janeiro: Campus, 2. ed. 2002

Revisão da Hierarquia de Chomsky

INTRODUÇÃO

No primeiro módulo da presente disciplina, será apresentada uma revisão da Hierarquia de Chomsky, bem como da máquina de estados finitos, tópicos estudados em disciplina anterior. Em seguida, será apresentada a Máquina de Turing, modelo que simula procedimentos computacionais mais gerais que a máquina de estados finitos. Ao final, a definição formal da Máquina de Turing será aduzida.

Unidade I

HIERARQUIA DE CHOMSKY

Noam Chomsky classificou as linguagens em quatro tipos, a saber:

Linguagens regulares;

Linguagens livres de contexto;

Linguagens dependentes de contexto;

Linguagens irrestritas.

Em uma Linguagem de Programação (Java, C#, por exemplo) ocorrem três componentes da Hierarquia de Chomsky: a componente regular, a livre de contexto e a dependente de contexto. O compilador, que é o *software* que traduz a linguagem de programação de alto nível para a linguagem de máquina, emprega algoritmos advindos do estudo das Linguagens Regulares e Livres de Contexto.

A análise de cada palavra de um programa escrito em uma linguagem de programação qualquer, denominada análise léxica, usa os algoritmos obtidos do estudo das **linguagens regulares**. Esses algoritmos são a realização do modelo computacional denominado **máquina de estados finitos**.

A análise de cada comando (***if*, *while*, atribuição**) e demais estruturas sintáticas (**classes, declarações de variáveis** etc.) de um programa desenvolvido em uma linguagem de programação, a análise sintática, emprega algoritmos advindos de estudo das **linguagens livres de contexto**.

A componente **dependente de contexto** em uma linguagem de programação pode ser identificada na concordância entre a declaração de tipos das variáveis de uma variável e uso delas na concordância entre o número de parâmetros na declaração de um método de uma classe e o número de argumentos no uso do método de um objeto, em sobrecarga de métodos etc. Como será visto na disciplina **Compiladores**, tais problemas são resolvidos computacionalmente através da utilização de uma extensão das **Gramáticas Livres de Contexto**, denominada Gramática de Atributos. O estudo das **linguagens dependentes de contexto** ainda é objeto de pesquisa.

A linguagem natural (português, italiano, inglês etc) é um exemplo de **linguagem irrestrita.**

A título de revisão, considere-se o alfabeto A = {a, b, c}. A partir desse alfabeto, podem ser definidas diferentes linguagens.

Exemplos:

$L_R = \{w \mid w = a^n\ bb\ c^m, n > 0, m > 0\}$

Trata-se de uma linguagem regular, pois não existe um vínculo entre *o número* de ocorrência dos diferentes símbolos "a", "b" e "c".

$L_L = \{w \mid w = a^n\ bb\ c^n, n > 0\}$

Trata-se de uma linguagem livre de contexto, pois o número de ocorrências do símbolo "c" deve ser igual ao do símbolo "a". Ao se verificar se uma palavra pertence a L_L, um reconhecedor deveria, ao identificar cada símbolo "c", "lembrar-se", que anteriormente foi encontrado um símbolo "a" correspondente. Sabe-se que para isso uma memória organizada em pilha resolve conceitualmente o problema.

$L_D = \{w \mid w = a^n\ (bb)^n c^n, n > 0\}$

Trata-se de uma linguagem dependente de contexto. Ao se verificar se uma palavra pertence a L_D, um reconhecedor deveria memorizar não somente cada ocorrência do símbolo "a", como também cada ocorrência da cadeia "bb" para poder, finalmente, aceitar cada ocorrência do símbolo "c". Pode-se constatar que, conceitualmente, para se resolver este problema seriam necessárias duas pilhas.

REFERÊNCIAS BIBLIOGRÁFICAS

RAMOS, Marcus Vinicius Midena; NETO, João José. VEGA, Italo Santiago. "Linguagens Formais. Teoria, Modelagem e Implementação". Porto Alegre: Bookman, 2009.

MENEZES, Paulo Blauth. "Linguagens formais e autômatos". Porto Alegre: Bookman, 2008.

Definição formal de Máquinas de Turing, linguagens recursivas e recursivamente enumeráveis

DEFINIÇÃO FORMAL DA MÁQUINA DE TURING (LEWIS, PAPADIMITRIOUS)

Uma máquina de Turing é uma quíntupla (Q, A, q0, F, g), em que:

Q é o conjunto de estados;

A é o alfabeto, contendo o símbolo de espaço em branco β e o símbolo de início de fita ·. Não contém os símbolos $\leftarrow$ e $\rightarrow$ (indicadores de movimento do cursor para a esquerda e para a direita, respectivamente).

$q0 \in Q$ é o estado inicial;

$F \subseteq Q$ é o conjunto de estados de parada;

g é a função de transição, com:

$g: (Q\text{-}F) \times A \rightarrow (Q \times (A \cup \{\leftarrow, \rightarrow\})$, tal que:

a) para todo $q \in Q - F$, se $g(q,\bullet) = (p, b)$, então $b = \rightarrow$

b) para todo $q \in Q - F$, se $a \in A$, se $g(q, a) = (p, b)$ então $b \neq \bullet$

Linguagem Recursivamente Enumerável

(Lewis, Papadimitrious)

Seja uma máquina de Turing $M = (Q, A, , q0, F, g)$, seja $A_0 \subseteq A - \{\beta, \bullet\}$ um alfabeto e seja a linguagem $L \subseteq A_0^*$.

Observação: A_0^* significa o conjunto de todas as palavras que podem ser formuladas, a partir do alfabeto A, exceção feita aos símbolos β e $\bullet$.

Diz-se que M semidecide L, se para qualquer palavra $w \in A_0^*$: se e somente se M para quando a palavra w pertencer a L.

Uma linguagem L é recursivamente enumerável se e somente se existe uma Máquina de Turing que semidecide L.

Note-se que a definição acima deixa em aberto dois comportamentos possíveis para a máquina M ao se verificar se uma palavra não estiver em L, a saber:

- M pode parar em um estado que não seja de "parada" ou
- M pode não parar.

Linguagem Recursiva

Considere uma Máquina de Turing $M = (Q, A, g, q0, F)$, em que $F = \{sim, não\}$

Uma configuração de aceitação é aquela cujo estado final é sim.

Uma configuração de rejeição é aquela cujo estado final é não.

Seja $A_0 \subseteq A - \{\beta, \bullet\}$

Diz-se que uma Máquina de Turing M aceita uma palavra $w \in L \subseteq A_0^*$ se o seu processamento resulta em uma configuração de aceitação. Por outro lado, diz-se que M rejeita w se o processamento resulta em uma configuração de rejeição.

Diz-se que M decide uma Linguagem $L \subseteq A_0^*$, se M aceita a cadeia $w \in L$ e M rejeita toda cadeia $w \notin L$.

Uma linguagem é recursiva se existe uma Máquina de Turing que a decide.

Teorema: se uma linguagem é recursiva, então também é recursivamente enumerável.

Exercício resolvido 1: considere a Máquina de Turing M = (Q, A, g, q0, F), em que:

Q = {q0, q1, q2, q3}

A = {0, β, •, I, P}

F = {q3 }

Ainda:

g(q0, •) = (q0, →)

g(q0, 0) = (q1, →)

g(q1, 0) = (q2, →)

g(q1, β) = (q3, I)

g(q2, 0) = (q1, →)

g(q2, β) = (q3, P)

Considere a seguinte fita de entrada:

•	0	0	0	β	β	β	β	β	β	β	...
⇑											
q0											

Pede-se a configuração final da fita de entrada, após o reconhecimento da cadeia de entrada.

Tem-se a seguinte sequência do reconhecimento:

•	0	0	0	β	β	β	β	β	β	β	...
⇑											
q0											

•	0	0	0	β	β	β	β	β	β	β	...
	⇑										
	q0										

•	0	0	0	β	β	β	β	β	β	β	...
		⇑									
		q1									

•	0	0	0	β	β	β	β	β	β	β	...
			⇑								
			q2								

•	0	0	0	β	β	β	β	β	β	β	...
				⇑							
				q1							

•	0	0	0	I	β	β	β	β	β	β	...
				⇑							
				q3							

Exercício resolvido 2: considere a Máquina de Turing M = (Q, A, g, q0, F), em que:

Q = {q0, q1, q2, q3, q4}

A = {0, 1, β, •, I, P}

F = {q4 }

Ainda:

g(q0, •) = (q0, →)

g(q0, 0) = (q1, →)

g(q1, 0) = (q2, →)

g(q1, β) = (q3, I)

g(q2, 0) = (q1, →)

g(q2, β) = (q3, P)

g(q3, P) = (q3, ←)

$g(q3, |) = (q3, \leftarrow)$

$g(q3, 0) = (q3, 1)$

$g(q3, 1) = (q3, \leftarrow)$

$g(q3, \bullet) = (q4, \rightarrow)$

Considere a seguinte fita de entrada:

•	0	0	0	β	β	β	β	β	β	β	...
⇑											
q0											

Pede-se a configuração final da fita de entrada, após o reconhecimento da cadeia de entrada.

Tem-se a seguinte sequência do reconhecimento:

•	0	0	0	β	β	β	β	β	β	β	...
⇑											
q0											

•	0	0	0	β	β	β	β	β	β	β	...
	⇑										
	q0										

•	0	0	0	β	β	β	β	β	β	β	...
		⇑									
		q1									

•	0	0	0	β	β	β	β	β	β	β	...
			⇑								
			q2								

•	0	0	0	β	β	β	β	β	β	β	...
				⇑							
				q1							

•	0	0	0	|	β	β	β	β	β	β	...
				⇑							
				q3							

•	0	0	0	I	β	β	β	β	β	β	...
			⇑								
			q3								

•	0	0	1	I	β	β	β	β	β	β	...
			⇑								
			q3								

•	0	0	1	I	β	β	β	β	β	β	...
		⇑									
		q3									
•	0	1	1	I	β	β	β	β	β	β	...
		⇑									
		q3									
•	0	1	1	I	β	β	β	β	β	β	...
	⇑										
	q3										

•	1	1	1	I	β	β	β	β	β	β	...
	⇑										
	q3										

•	1	1	1	I	β	β	β	β	β	β	...
⇑											
q3											

•	1	1	1	I	β	β	β	β	β	β	...
	⇑										
	q4										

REFERÊNCIAS

LEWIS, H. R.; PAPADIMITRIOU, C. H. "Elementos da Teoria da Computação".

Tese de Turing Church

TESE DE TURING – CHURCH

(Zalta, E. N.)

"[...] A tese de Turing-Chuch diz respeito à noção de um método mecânico e efetivo no contexto da Lógica e Matemática. Efetivo e seu sinônimo mecânico são termos dessas disciplinas. Não apresentam o significado do nosso dia a dia.

Um **método, ou procedimento**, M, que é empregado para se obter um resultado desejado é denominado 'efetivo' ou mecânico.

Assim, M:

Deve ser descrito em um número finito de instruções exatas (cada instrução sendo expressa por um número finito de símbolos);

Se M não apresentar erros, deve produzir o resultado desejado em um número finito de etapas;

M pode ser realizado por um ser humano sem o auxílio de uma máquina, ou seja, empregando apenas 'lápis e papel';

O método M não deve exigir nenhuma inferência e nem mesmo inventividade do ser humano que o executa."

(ZALTA, E. N. Stanford Encyclopedia of Philisophy disponível em http://plato.stanford.edu/entries/church-turing/. Última data de acesso: 19 de Julho de 2012)

O que é um algoritmo? (Cormen, Leiserson)

βnformalmente, um algoritmo é qualquer procedimento computacional que recebe como entrada um ou mais valores e produz como saída um ou mais valores. Um algoritmo é, portanto, uma sequência de etapas computacionais que transformam valores de entrada para valores de saídaβ.

Tese de Turing-Church: "As máquinas de Turing são versões formais de algoritmos e nenhum procedimento computacional é considerado um algoritmo a não ser que possa ser apresentado na forma de uma máquina de Turing" (José Neto, J.)

Exercício Resolvido 1: A Máquina de Turing permite a computação de números naturais. Seja I um símbolo fixo não branco. Um número natural **n** pode ser representado em notação unária, pela cadeia de símbolos I, de comprimento n+1. Considerando essa definição, selecione a representação unária para os números 0, 1 e 2, respectivamente, com I = 0

Tem-se:

$(0)_{10} = (0)_1$

$(1)_{10} = (00)_1$

$(2)_{10} = (000)_1$

Exercício Resolvido 2: Relativamente à Hipótese de Church, justifique:

a) Quais as implicações da Hipótese de Church?

Pela Hipótese de Church, as Máquinas de Turing são versões formais dos algoritmos. Adotar tal modelo de algoritmo implica na possibilidade de formalmente se provar que certos problemas não podem ser resolvidos por nenhum algoritmo.

b) Por que ela é chamada de Hipótese de Church ao invés de Teorema de Church?

A Hipótese de Church não é um resultado matemático e, portanto, não pode ser provado.

O Problema da Parada – problemas não solucionáveis sobre as Máquinas de Turing e sobre as gramáticas

MÁQUINAS DE TURING UNIVERSAIS - O PROBLEMA DA PARADA - PROBLEMAS NÃO SOLUCIONÁVEIS SOBRE AS MÁQUINAS DE TURING E SOBRE AS GRAMÁTICAS

Problema solucionável

Um problema é dito solucionável ou totalmente solucionável se existe um algoritmo que solucione o problema, sempre para qualquer entrada, com uma resposta afirmativa (aceita) ou negativa (rejeita).

Problema não solucionável

Um problema é dito não solucionável se não existe um algoritmo que solucione o problema, tal que sempre para, qualquer que seja a entrada.

Problema parcialmente solucionável ou computável

Um problema é dito parcialmente solucionável ou computável se existe um algoritmo que solucione o problema, tal que pare quando a resposta é afirmativa (aceita). Entretanto, quando a resposta esperada for negativa, o algoritmo pode parar (rejeita) ou permanecer processando indefinidamente.

Problema completamente insolúvel

Um problema é dito completamente insolúvel ou não computável se não existe um algoritmo que solucione o problema, tal que pare quando a resposta é afirmativa.

É importante observar que alguns problemas não solucionáveis são parcialmente solucionáveis. Ainda, existem problemas não solucionáveis que possuem solução parcial.

A classe dos problemas solucionáveis é equivalente à classe das linguagens recursivas.

A classe dos problemas parcialmente solucionáveis é equivalente à classe das linguagens recursivamente enumeráveis.

Problemas não solucionáveis sobre Máquinas de Turing

Provam-se insolúveis os seguintes problemas sobre gramáticas:

a) Dada uma máquina de Turing M e uma palavra w, ambas arbitrária, M para com uma entrada w?

b) Dada uma máquina de Turing M, M para com a fita de entrada vazia?

c) Dada uma máquina de Turing M arbitrária, existe uma palavra que a faça parar?

d) Dada uma máquina de Turing M arbitrária, M para com qualquer entrada possível?

e) Dadas duas máquinas de Turing M1 e M2 arbitrárias, elas param com as mesmas entrada".

f) Dada M arbitrária, a linguagem por ela aceita é regular? É livre de contexto? É recursiva?

Problemas não solucionáveis sobre gramáticas

Provam-se insolúveis os seguintes problemas sobre gramáticas:

a) Para uma dada gramática geral G e uma palavra w, determinar se a palavra w βL(G) (L (G) significa: linguagem gerada pela gramática G).

b) Para uma dada gramática G, determinar se a palavra vazia e pertence à linguagem gerada por G, ou seja, L(G).

c) Dadas duas gramáticas gerais arbitrárias G1 e G2, determinar se as linguagens geradas pro G1 e G2 são iguais.

d) Para uma gramática geral arbitrária G determinar se a linguagem gerada é um conjunto vazio (ou seja, não existem palavras).

Propriedades das linguagens recursivas

Diz-se que uma Máquina de Turing M enumera a Linguagem L, se e somente se, para um determinado estado q de M, ao analisar uma fita de entrada com apenas um símbolo branco, passa periodicamente por um determinado estado q (q não pode ser estado de parada), insere na fita de entrada a palavra w que pertence à linguagem L.

Se as palavras forem enumeradas lexicograficamente por M, diz-se que a linguagem L é recursiva.

Teorema: uma linguagem é recursiva se e somente se for Turing-enumerável lexicograficamente.

Exercício resolvido 1: apresente argumentações para mostrar que o Teorema da Parada é insolúvel.

Considere uma Máquina de Turing X capaz de analisar qualquer Máquina de Turing T. As duas únicas possibilidades de X parar são descritas a seguir:

I – A Máquina de Turing X deve parar com a fita contendo apenas um algarismo 1, se e somente se, T aceitar uma cadeia α.

II – A Máquina de Turing X deve parar com a fita contendo apenas um algarismo 0, se e somente se T, nunca parar ao processar a cadeia α.

Resposta: Considere-se por absurdo que a Máquina de Turing X possa ser projetada. Uma vez que seja possível o projeto de X, é possível modificar X e obter uma Máquina de Turing Y com as seguintes características:

I – A Máquina de Turing Y não para se e somente se T parar ao processar a cadeia α. (basta acrescentar a X um loop infinito).

II – A Máquina de Turing Y deve parar com a fita contendo apenas um algarismo 0, se e somente se T nunca parar ao processar a cadeia α.

É, portanto, possível criar uma máquina de Turing Z que copie a cadeia de entrada α e em seguida submeta à máquina de Turing Y, a cadeia $\alpha\alpha$.

Sabe-se que é possível obter-se uma cadeia "M" que represente a máquina de Turing M, qualquer que seja.

Assim sendo, considere-se a situação em que Z processa sua própria representação "Z".

O desdobramento imediato é o seguinte paradoxo:

Z para ao processar "Z" porque Y (que faz parte de Z) não para ao processar ("Z", "Z").

Ou ainda, Z não para ao processar "Z", porque Y para ao processar ("Z", "Z).

Unidade II

COMPORTAMENTO ASSINTÓPTICO DE FUNÇÕES

Tempo de Execução de um Programa: Comportamento Assintóptico de Funções e Classes de Comportamento Assintóptico

Seja n um parâmetro que caracteriza o tamanho da entrada do algoritmo. Por exemplo, ordenar n números ou multiplicar duas matrizes quadradas n × n (cada uma com n^2 elementos).

Costuma-se medir um algoritmo em termos de tempo de execução ou o espaço (ou memória) usado. Para o tempo, podemos considerar o tempo absoluto (em minutos, segundos etc.). Medir o tempo absoluto não é interessante por depender da máquina.

Em Análise de Algoritmos conta-se o número de operações consideradas relevantes realizadas pelo algoritmo e expressa esse número como uma função de n. Essas operações podem ser comparações, operações aritméticas, movimento de dados etc.

Um caso de particular interesse é quando n tem valor muito grande ($n \to \infty$), denominado comportamento assintótico.

Notação O

Para se estudar o comportamento assintótico – para n grande – emprega-se o termo de ordem superior.

A notação O pode ser formalizada como se segue:

Diz-se que T(n) = O(f(n)) se existirem inteiro m e constante c, tais que:

$T(n) \leq cf(n)$ para $n > m$

Exercício Resolvido 1:

Apresente um estudo comparativo da grandeza das funções:

F(n) = 1;

$F(n) = \log_2(n)$;

$F(n) = n$;

$F(n) = n \log_2(n)$;

$F(n) = n^2$;

$F(n) = n^3$

$F(n) = 2^n$

	F(n)						
n	1	$\log_2(n)$	n	$n \log_2(n)$	n^2	n^3	2^n
1	1	0	1	0	1	1	2
10	1	3.32	10	33	100	1000	1024
100	1	6.64	100	664	10000	1000000	$1,268 \times 10^{30}$
1000	1	9.97	1000	9970	1000000	10^9	$1,072 \times 10^{301}$

Exercício Resolvido 2:

Colocar em ordem crescente as seguintes complexidades:

$n\log_2(n)$, 3^{10}; n^3; $n^{1/3}$

A ordem crescente é:

3^{10}; $n^{1/3}$; $n\log_2(n)$; n^3

Problemas P e NP

Complexidade Computacional – A Classe P e a Classe NP – Grafos Eulerianos e Hamiltonianos

Ciclo de Euler:

Dado um grafo G, existe um caminho fechado em G, tal que esse caminho use uma aresta exatamente uma vez.

Teorema sobre Caminhos de Euler (para um grafo não orientado)

Existe um caminho de Euler em um grafo conexo se e somente se não existam nós ímpares ou existem exatamente dois nós ímpares. No caso em que não existam dois nós ímpares, o caminho pode terminar em qualquer nó e terminar aí; no caso de dois nós ímpares, o caminho precisa começar em um deles e terminar no outro.

Lewis e Papadimitrious apresentam a definição de um Grafo Euleriano quando o grafo é orientado:

Um grafo G é Euleriano se e somente se apresenta as seguintes duas propriedades:

a) Para quaisquer pares de nós u e v, nenhum dos quais isolados, existe um caminho de u para v;

b) Todos os nós apresentam número de arestas que nele incidem iguais ao número de arestas não incidentes.

Ciclo Hamiltoniano

Dado um grafo G, existe um ciclo que passe por todos os nós de G exatamente uma vez.

O único algoritmo conhecido para este problema consiste em examinar todas as permutações possíveis dos nós e para cada permutação, verificar se trata de um ciclo Hamiltoniano. Trata-se, portanto, de um problema O(n!).

Definição da classe P

P é a coleção de todos os conjuntos reconhecíveis por Máquinas de Turing em tempo polinomial.

Definição da classe NP

NP é a coleção de todos os conjuntos reconhecíveis por máquinas de Turing não determinísticas em tempo polinomial.

Uma máquina de Turing M = (Q, A, g, q0, H) é denominada polinomial se a máquina sempre para, qualquer que seja a entrada x de comprimento n após p(n) etapas em que p(n) é uma função polinomial do comprimento n da cadeia de entrada.

Exercício Resolvido 1:

Considere o grafo G = (V, A, g), em que:

V = {1, 2, 3, 4, 5} são os vértices

A = {a, b, c, d, e}

g(a) = 1-2

g(b) = 2 – 5

g(c) = 1 – 3

g(d) = 5- 5

g(e) = 3-4

Pede-se: verificar se existe um caminho de Euler

Resposta:

A matriz de adjacência de G é dada por:

$$M = \begin{matrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{matrix}$$

Assim identifica-se que grau(1) = grau(2) = grau(3) =grau(4) = grau(5) = 2

Portanto, o número de nós ímpares é 0. Assim sendo pelo Teorema de Euler, existe um caminho que passa por todas as arestas. Como o número de nós ímpares é 0, o caminho pode se iniciar em qualquer nó.

Assim, o caminho pode ser: 1a2b5d4e3c1.

Exercício Resolvido 2: 6 – Considere o grafo G = (V, A, g), em que:

V = {1, 2, 3, 4, 5, 6, 7, 8} são os vértices

A = {a, b, c, d, e}

g(a) = 2-6

g(b) = 4-3

g(c) = 2 – 3

g(d) = 1-4

g(e) = 1-2

g(f) = 5-6

g(g) = 5- 8

g(h)=8-7

g(i)= 6-7

g(j) = 7-3

g(k) = 8-4

Pede-se: verificar se existe um ciclo Hamiltoniano.

Resposta: O algoritmo para este problema consiste em identificar as 8! permutações dos nós e verificar se uma ou mais constituem um ciclo hamiltoniano. Por outro lado, por tentativa e erro é possível se constatar que este grafo apresenta o seguinte ciclo:

6i7j3c2e1d4k8g5f6

COMPLEXIDADE COMPUTACIONAL – PARTE II

Exercício Resolvido 1: O problema do carregamento de paletes (um estrado de madeira, metal ou plástico que é utilizado para movimentação de cargas) é um problema clássico e consiste em arranjar o máximo possível de itens (caixas) iguais sobre um palete (**Lorena, L. A. N.; Ribeiro, G. M. "Método de geração de colunas para o problema do carregamento de paletes do produtor").** Dado que se pode montar um grafo de conflitos para este problema, sendo os vértices as possíveis posições das caixas sobre o palete e as arestas os possíveis conflitos entre essas posições, este problema pode ser visto como um problema **de máximo conjunto independente de vértices**.

a) Qual é o algoritmo trivial conhecido para resolver este problema?

O problema do carregamento de paletes, pode ser visto como um problema de máximo conjunto independente de vértices. De fato, o problema de otimização dos conjuntos independentes pode ser assim enunciado: "Dado um grafo não orientado G e um inteiro $K \geq 2$, um subconjunto C do conjunto V dos vértices do grafo com $|C| \geq K$, tal que para todo par de vértices $v_i, v_j \in C$, não exista nenhuma aresta entre v_i e v_j.". O que se deseja, portanto, no problema de carregamento de paletes é obter o maior número de conjunto C de vértices, representando as caixas, de forma que quaisquer que seja $v_i, v_j \in C$, não exista aresta entre eles, representando que não há nenhum conflito em armazená-los na mesma palete.

Para este problema, o único algoritmo conhecido é enumerar todas os subconjuntos C de vértices não adjacentes. Trata-se de um algoritmo de natureza combinatória.

Exercício Resolvido 2: Sabe-se que o problema da Satisfabilidade Booleana é um problema NP. Por outro lado, o problema da Satisfabilidade – 2 está em P.

Considere as seguintes fórmulas booleanas em sua forma conjuntiva normal. Verifique se as mesmas são passíveis de satisfabilidade. Em caso afirmativo, apresente uma solução.

a) $F'=\{(x_1 \vee x_2 \vee x_3), (x'_1 \vee x_2), (x'_2 \vee x_3), (x_2 \vee x'_3), (x'_1 \vee x'_2 \vee x'_3)\}$.

Resp.: O algoritmo para se resolver este problema implica em aplicar as 2^3 possibilidades de combinações dos valores 1 e 0 nas 5 cláusulas da fórmula. Pode-se verificar que esta fórmula não é passível de satisfabilidade.

b) $F = \{(x_3 \vee x_4), (x'_3 \vee x_5), (x'_5 \vee x'_4), (x'_3 \vee x_4)\}$

Resp.: Esta fórmula é uma instância do problema da satisfabilidade- 2 (cada cláusula apresenta apenas duas variáveis). Os seguintes valores $x_3 = 1$, $x_5 = 1$, $x_4 = 0$ tornam a fórmula satisfatível.

Completude NP – Problemas NP Difíceis

Completude NP – Problemas NP-Difíceis – O Teorema de Gödel