Resumo – Aspectos Teóricos da Computação

Unidade I – Vídeo Aula I

➢ Linguagens e dispositivos reconhecedores

- Linguagens regulares: são geradas pelas gramáticas regulares e são **reconhecidas pelos autômatos finitos**.

- Linguagens livres de contexto: são geradas pelas gramáticas de mesmo nome e **reconhecidas pelos autômatos de pilha**.

- Linguagens recursivas e linguagens recursivamente enumeráveis: são reconhecidas pela máquina de **Turing**.

➢ Definição formal da máquina de Turing

Máquina de Turing: é um dispositivo computacional proposto por Alan Turing, um matemático, em 1936. É constituída basicamente por uma fita de entrada e uma máquina de controle finito.

Fita de entrada (ou fita de saída) é um dispositivo de armazenamento, onde são armazenadas a cadeia de entrada para ser processada pela máquina de Turing. Diferente do autômato de pilha e do infinito, a fita de entrada é ilimitada a direita, as células vazias são representadas em branco ou por símbolo de vazio. Pode ser lida ou escrita.

O cursor pode se movimentar tanto a esquerda quanto a direita (antes o cursor movimentava-se apenas a direita).

Definição formal da máquina de Turing:

M = (Q, A, “R”, g, q0, >, b, F)

Q é o conjunto finito não vazio de estados

A é o alfabeto de entrada, formado por um conjunto não vazio de símbolos.

“R” é o conjunto finito e não vazio de símbolos que podem ser lidos e/ou escritos na fita de trabalho.

Q0 é o estado inicial.

F é o conjunto de estados finais.

Unidade I – Vídeo Aula II

- MT reconhecedora da Linguagem L = {a^n b^n}

- A máquina de Turing é reconhecedora da linguagem livre de contexto de dublo balanceamento. Ocorre entre os símbolos A e B.

- Ela representa cinco estados: q0, q1, q2, q3 e qf.

Unidade I – Vídeo Aula III

- Método ou procedimento

- Um método, ou procedimento, M, que é empregado para se obter um resultado desejado é denominado “efetivo” ou mecânico e 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 auxilio de uma máquina, ou seja, empregando apenas “lápis e papel”.

// método ou procedimento não implica necessariamente que o problema deve ser resolvido por máquina, também pode ser resolvido por humanos.

- Algoritmo

“Informalmente, 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 ***sequencia de etapas computacionais*** *que transformam valores de entrada para valores de saída.”*

- Hipótese 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”.

- Estabelece-se, assim, a equivalência entre máquinas de Turing e algoritmos. Uma função de teoria dos números é computável por um algoritmo se, e somente se, for computável por Turing.

➢ Máquinas Equivalentes

- Múltiplas trilhas.

- Múltiplas fitas.

- Múltiplos cursores.

- Fita ilimitada em ambos os sentidos.

- Transições que deslocam o cursor um número variável de posições.

- Transições sem leitura ou gravação de símbolos.

Unidade I – Vídeo Aula IV

➢ Máquina de Turing não determinísticas

- A função de transição g remete o não determinismo inerente aos dispositivos aceitadores apresentados.

- Dada uma mesma combinação de estado corrente e de símbolo na fita de trabalho, é possível especificar múltiplas transições.

// Uma máquina de Turing comum (determinística) possui uma função de transição que, dado um estado e um símbolo na posição de execução da fita, especifica três coisas: um novo símbolo a ser escrito na posição de execução da fita, a direção para o qual a fita deve mover-se e um novo estado para o controle finito.

Uma máquina de Turing não determinística difere-se pois um estado e um símbolo de fita não mais definem estas três coisas de forma única – mais de uma ação pode ser aplicável dado um estado e um símbolo.

Uma máquina de Turing determinística possui um único caminho de computação, já a não determinística possui uma “árvore de computação”.

➢ Máquina de Turing Universal

- É uma máquina de Turing capaz de simular qualquer outra apropriadamente codificada.

// Uma máquina de Turing pode processar outra máquina de Turing.

- *"Processa programas que são codificações de máquinas de Turing".*

- **Simula a máquina descrita e produz como resultado o mesmo resultado que a máquina simulada produziria**.

- É universal, pois é capaz de executar qualquer algoritmo.

➢ Ordenação lexicográfica

É possível especificar uma máquina de Turing através de uma descrição passível de ser a entrada de outra máquina de Turing. A seguinte convenção pode ser adotada:

- Cada estado distinto da máquina de Turing é nomeado por uma cadeia constituída do símbolo q, que deve ser sucedido por uma cadeia de símbolos do alfabeto binário.

- Cada símbolo da cadeia de entrada é nomeado por uma cadeia constituída do símbolo "a" e sucedido por uma cadeia de símbolos do alfabeto binário.

- Os estados e os símbolos da cadeia de entrada devem ser ordenados. Pode-se, por convenção, ordená-los em ordem lexicográfica crescente de tal forma que:

• O estado inicial é o primeiro e os estados de aceitação são os últimos

• Os símbolos especiais são os primeiros na seguinte ordem: branco, inicio de fita, movimento a esquerda e movimento a direita.

➢ Linguagem não recursivamente enumerável

- É possível codificar todas as máquinas de Turing como uma palavra sobre o alfabeto A, de tal forma que cada código represente uma única máquina de Turing.

- Decidibilidade

- É o estudo das propriedades exibidas pelas linguagens, com o objetivo de determinar a existência ou não de algum algoritmo capaz de aceitar ou rejeitar, em tempo e espaço finitos, uma cadeia qualquer apresentada para análise.

- Solucionável X não solucionável X parcialmente solucionável.

- Problema da parada

"Existe um algoritmo para decidir, dadas uma máquina de Turing T qualquer e uma cadeia a, se T começando com uma fita contendo a, vai parar?"

Teorema sobre o problema da parada: o problema da parada é insolúvel.

- Teorema: o problema de decidir se uma gramática livre de contexto arbitrária é ambígua é indecidível.

Unidade II – Vídeo Aula I

- Introdução

• Decidibilidade é o estudo das propriedades exibidas pelas linguagens com o objetivo de determinar a existência ou não de um algoritmo capaz de aceitar ou rejeitar, em tempo e espaço finitos, uma cadeia qualquer apresentada para análise (Diz se o algoritmo existe ou não).

• Não basta um problema ser decidível. Há que se considerar os **custos** dessa solução.

• **Custos** dizem respeito ao tempo total de execução e ao volume total de memória para se chegar à solução do problema.

- Complexidade Computacional

• Complexidade: "estudo das **propriedades** exibidas pelas linguagens com o objetivo de determinar o custo de seu processamento, em termos do tempo e espaço finitos"

• "A complexidade de tempo de uma computação mede a quantidade de trabalho gasto pela computação".

- Tempo de execução de um algoritmo depende do tamanho da **cadeia de entrada**.

➢ Estudo comparativo da grandeza de algumas funções

n	$\log_2(n)$	$n*\log_2(n)$	n^2	n^3	2^n
1	0	0	1	1	2
10	3.32	33	100	1000	1024
100	6.64	664	10^4	10^6	$1,268 \times 10^{30}$
1000	9.97	9970	10^6	10^9	$1,072 \times 10^{301}$

➢ Complexidade de tempo

Seja MT uma Máquina de Turing determinística.

• A complexidade de tempo de execução é uma função:

$$f: N \rightarrow N$$

Tal que f(n) é o número máximo de transições processadas por uma computação de MT quando iniciada uma cadeia de entrada de comprimento n, independentemente de MT aceitar ou não.

• Assume-se que a computação termina para toda a cadeia de entrada.

➢ A notação O – grande O

Sejam f:

$$g\ N \rightarrow R^+$$

// domínio do conjunto dos números naturais e contradomínio do conjunto dos números reais maiores ou iguais a zero.

Diz-se que f(n) = O(g(n)), se existirem números inteiros e positivos c e n° tais que:

$$\forall n, n \geq n_0, f(n) < c.\ g(n)$$

// qualquer que seja o valor n, com n maior ou igual a zero a função f(n) será sempre menor que c multiplicado por g(n).

// a função f(n) tem um crescimento limitado pela função g(n) multiplicada por uma constante c.

• A notação O é utilizada para dar um limite assintótico superior sobre uma função, dentro de um fator constante.

➢ Operações com a notação O

f(n)	= O(f(n))
c x f(n), c constante	= O(f(n))
O(f(n)) + O(f(n))	= O(f(n))
O(O(f(n))	= O(f(n))
O(f(n)) + O(g(n))	= O(max(f(n), g(n)))
O(f(n)) . O(g(n))	= O(f(n) . g(n))
f(n) . O(g(n))	= O(f(n) . g(n))

// Representa o processamento sequencial de dois trechos de algoritmos um com desempenho O(f(n)) e O(g(n)). O processamento dos dois trechos vai apresentar um desempenho O(max(f(n), g(n))).

➢ Máquina de Turing de k fitas

• Teorema: seja L uma linguagem aceita por uma máquina de Turing determinística de k fitas M com complexidade de tempo f(n). Então L é aceita por uma máquina de Turing padrão de uma fita com complexidade de tempo **O(f(n)²)**.

• Observe-se que a eliminação de fitas adicionais (de k fitas para 1 fita) transforma o tempo de execução para no máximo uma potência de 2.

// O processamento em máquinas de turing determinísticas é diferente do processamento em máquinas de turing não determinísticas.

• Seja uma máquina de turing não determinística deve-se considerar todas as computações possíveis para uma cadeia de entrada.

• Cálculos em MT em tempo exponencial são ineficientes.

• Problemas para os quais não existe um algoritmo em tempo polinomial são ditos intratáveis.

• A teoria da complexidade classifica os problemas decidíveis em **tratáveis** e **intratáveis**.

Unidade II – Vídeo Aula II

➢ A classe P

• Uma linguagem L é denominada polinomialmente decidível se existe uma MT determinística polinomial que a decide.

• Uma MT é denominada polinomial se a máquina sempre para qualquer que seja a entrada de comprimento n, após p(n) transições.

• P(n) é uma função polinomial do comprimento n da cadeia de entrada.

• P é a união de todos os conjuntos de linguagens **decidíveis** por uma Máquina de Turing em tempo limitado por um polinômio de grau d.

• Todo algoritmo prático/ eficiente pode ser reduzido a uma máquina de Turing limitada em tempo polinomial.

São exemplos:

- Caminho de Euler

- Problema da Satisfabilidade Booleana SAT 2.

- Caminho de Euler

- Dado um grafo G, existe algum caminho em G que use todas as arestas exatamente uma vez?

• Teorema: existe um caminho de Euler em um grafo conexo se, e somente se, **não existem nós impares (circuito)** ou **existem exatamente dois nós impares**. No caso em que não existem dois nós impares, o caminho pode iniciar em qualquer nó e terminar aí; no caso de dois nós impares, o caminho precisa começar em um deles e terminar no outro.

- **Solução Polinomial.**

- Satisfabilidade booleana

• Uma fórmula booleana é composta por variáveis e operações booleanas.

• Literal: variável booleana ou variável booleana negada.

• Cláusula: fórmula booleana composta por literais e a operação "**OU**".

• Fórmula normal conjuntiva: composta por cláusulas conectadas pelo operador "**E**".

• 2SAT: instancia do problema da Satisfabilidade booleana apresentam 2 ou menos literais em cada cláusula, na sua forma conjuntiva.

- **Desempenho Polinomial**.

- Alcançabilidade

• Linguagens codificam problemas.

• Um problema é um conjunto de entradas, tipicamente infinito e uma questão do tipo sim/não arguida para cada entrada (propriedade que a entrada pode ter ou não).

- **Solução com desempenho $O(n^3)$** (polinomial cúbico).

- Circuito hamiltoniano

- Dado um grafo G, existe um circuito 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 circuito hamiltoniano.

• **Trata-se, portanto, de um problema O(n!)** (polinomial fatorial)

// fatorial cresce mais rápido que a exponencial

- Problema do caixeiro viajante

- Um caixeiro viajante, partindo de sua cidade, deve visitar exatamente uma única vez cada cidade de uma dada lista de cidades e retornar para casa de modo que a distancia total percorrida seja menor possível. (problema de decisão)

• Este problema tem inúmeras aplicações práticas, como minimização de rotas de veículos, sequenciamento de atividades, entre outros.

• Há que se fazer uma pesquisa exaustiva sobre o espaço de busca.

• **Trata-se de um problema cuja solução é O(n!)**

- Problema de otimização

“São problemas cujas soluções possuem melhor ‘custo’ (determinado por uma função de custo) entre um conjunto de possíveis soluções.

- A classe NP

• NP é o conjunto de todas as linguagens que são decidíveis em tempo polinomial p(n) em MT não determinísticas.

• P é diferente de NP, por não ter nada que se prove o contrário até então.

• Apresentam um verificador de tempo polinomial. Algoritmo que não gera uma solução para um problema NP, porem verifica se uma determinada entrada coincide com a solução de um problema NP.

- Redução

• Consiste basicamente na construção de um algoritmo de mapeamento entre as linguagens que traduzem os problemas.

• Se a classe de uma dessas linguagens é conhecida, então pode-se estabelecer algumas conclusões sobre a linguagem que se deseja investigar.

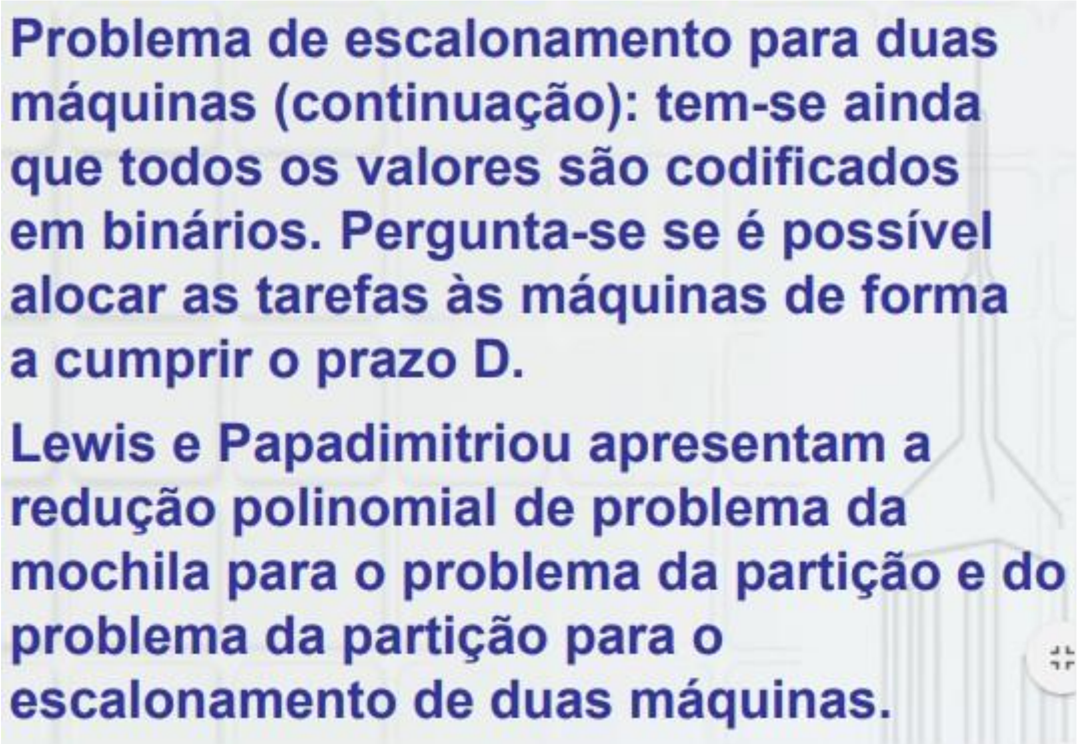
• Os problemas apresentados anteriormente são **NP – completos**, pois têm a seguinte propriedade de completude: todos os problemas em NP podem ser reduzidos aqueles via reduções polinomiais.

• Lewis e Papadimitriou apresentam o seguinte exemplo de redução: ciclo hamiltoniano para Satisfabilidade booleana.

➢ Redução polinomial

Problema da mochila: dado um conjunto $S = \{a_1, a_2, ..., a_n\}$ de números inteiros não negativos, representados em binário e um inteiro *K*, existe um subconjunto $P \subseteq S$ tal que a soma de todos os elementos de P resulte igual a K?

Problema de escalonamento para duas máquinas: sejam duas máquinas que têm a mesma velocidade, cada tarefa pode ser executada em qualquer máquina e não há restrições em ordenar as tarefas a serem executadas. Dados os tempos de execução $a_1, a_2, ..., a_n$ e um prazo D. (continua)



- Problemas **NP – difíceis** e **NP – completos**

• Um problema B é **NP – difícil** se a seguinte condição for verificada:

- Qualquer linguagem L pertencente a NP apresenta uma **redução** de tempo polinomial de L para B.

• A classe de problemas **NP – completos** é um subconjunto de problemas NP relacionados com a complexidade de todos os problemas NP.

- Problemas NP – completos

• **Teorema de Cook**: o problema da satisfabilidade é NP – completo

• **Teorema: MAX SAT** é NP – completo

• São também problemas NP – completos: **ciclo hamiltoniano, problema do caixeiro viajante**.

• Problema da mochila.

• Problema de escalonamento em duas máquinas.

P(n)	P(n^3)	P(n!)
Caminho de Euler	Alcançabilidade	Circuito Hamiltoniano
Satisfabilidade booleana		Problema do Caixeiro Viajante
		Permutação Simples