Complexidade de Algoritmos

Profa. Mariana Kolberg

Material baseado nos slides do prof. Edson

Intratabilidade

- O limite superior de complexidade de um problema refere-se ao melhor algoritmo que o resolve.
- O limite inferior de um problema refere-se a um resultado teórico que determina qual é a melhor complexidade possível que um algoritmo pode alcançar.
- Quando a diferença entre esses limites desaparece, a complexidade mínima do problema é conhecida e o problema é chamado fechado.

Intratabilidade

- O problema de classificação por comparação é um problema com limite de complexidade determinada, nlogn.

- Não existe algoritmo de classificação por comparação com complexidade melhor que nlogn.

- Este é um exemplo de problema fechado

Intratabilidade

A **classe P** consiste nos problemas que podem ser **resolvidos** em tempo Polinomial (Problemas tratáveis)

A **classe NP** consiste nos problemas que podem ser **verificados** em tempo polinomial (Problemas Intratáveis).

> Dada uma entrada, é possível verificar se ela corresponde a uma solução do problema: o conjunto de vértices $<v_1, v_2, \ldots, v_n>$ corresponde a um ciclo hamiltoniano? Isto pode ser feito em tempo polinomial.

Sabemos que $P \subseteq NP$, mas não sabemos se P = NP ! (1 milhão de dolares!!!)

A **classe NP-completo** contém problemas NP que possuem a característica de que se um deles puder ser resolvido em tempo polinomial então todo problema NP terá uma solução em tempo polinomial.

Intratabilidade

Um aspecto interessante é que vários problemas NP-Completos são, a principio, semelhantes a problemas que têm algoritmos de tempo polinomial.

- Circuito Euleriano x Ciclo Hamiltoniano

- Satisfabilidade 2-CNF x Satisfabilidade 3-CNF
  - Forma normal k-conjuntiva, ou k-CNF : **And** de claúsulas **Or** de exatamente k variáveis

$$(\neg x \vee y) \wedge (\neg y \vee z) \wedge (x \vee \neg z) \wedge (z \vee y) \text{ - 2-CNF}$$

$$(\neg x \vee y \vee \neg z) \wedge (\neg y \vee x \vee \neg z) \wedge \ldots \wedge (x \vee z \vee y) \text{ - 3-CNF}$$

Intratabilidade

Como mostrar se um problema é NP-completo?

- Queremos mostrar que provavelmente não existe nenhum algoritmo eficiente para o problema.
- Idéia é: mapear o problema que se sabe ser NP-completo através de reduções para o problema que se quer provar a dificuldade
- Uso de técnicas:
  - Problemas de decisão
  - Reduções
  - Um primeiro problema NP-completo

Intratabilidade

Problemas de Decisão x Problemas de otimização

Problemas de otimização : dentre as soluções viáveis, qual é a melhor?

Problemas de decisão: existe uma solução para um dado problema?

É possível formular um problema de otimização como um problema de decisão impondo um limite sobre o valor a ser otimizado.

Isto nos beneficia quando queremos mostrar que um **problema de otimização é difícil**, pois o problema de decisão associado **é mais fácil ou não mais difícil que o de otimização** (limite inferior).

Intratabilidade

Problemas de Decisão x Problemas de otimização

Problemas de otimização : desejamos encontrar o caminho do vértice u a v que utiliza o menor número de arestas em um grafo (shortest path)

Problemas de decisão: dado um grafo G, vértices u e v, e um inteiro k, existe um caminho de u até v consistindo em no máximo k arestas (path)

Ex: podemos resolver path resolvendo o shortest path e depois comparando o número de arestas no caminho mais curto encontrado ao valor do parâmetro de problema de decisão k.

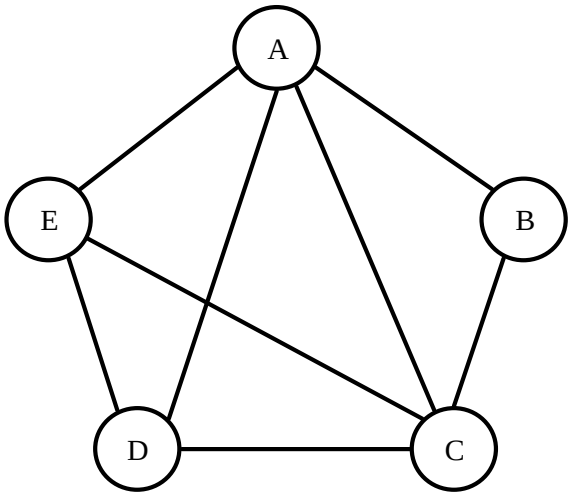
- Se o problema de otimização é fácil, seu problema de decisão relacionado também é facil.
- Se o problema de decisão é difícil, então o problema de otimização relacionado também é difícil.

Intratabilidade

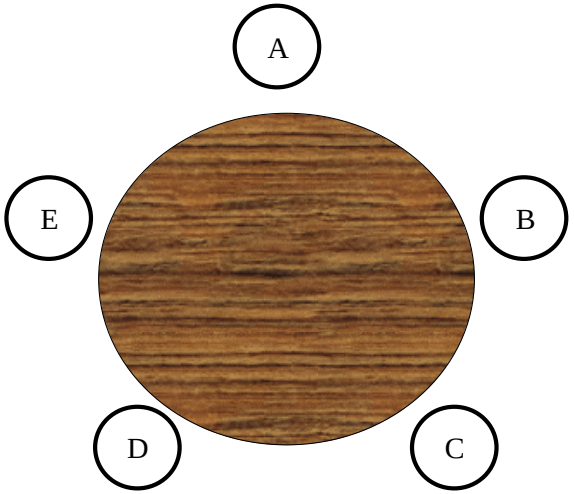
Reduções

Permite determinar afinidades entre problemas computacionais.

Visitar cada cidade em um conjunto de cidades apenas uma única vez



Organizar pessoas ao redor de uma mesa redonda de acordo com uma dada preferência



Intratabilidade

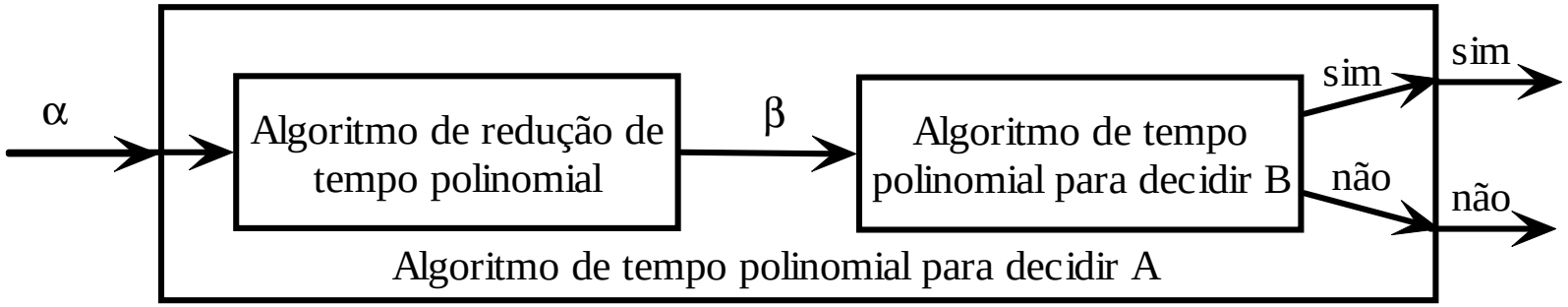
Reduções

Considere dois problemas A e B.

A redução consiste em um procedimento que transforma qualquer instância α (entrada) de **A** em alguma instância β de **B** de forma que:

- A transformação ocorre em **tempo polinomial**
- As respostas são as mesmas tanto para A quanto para B. Isto é, a resposta para α é sim sse a resposta para β é sim

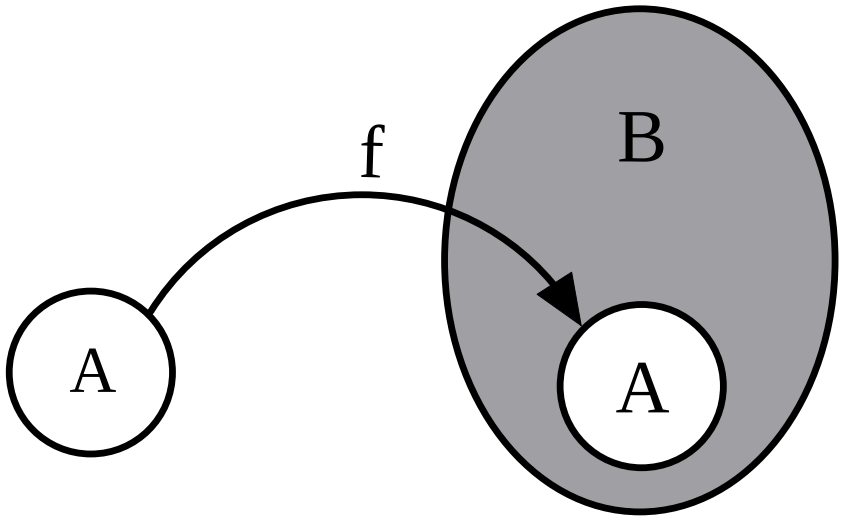
Intratabilidade



- A existência de uma redução polinomial de A para B indica que B é no mínimo tão dificil que A.
- Se B é eficientemente solúvel então A também será.
- Se A exige tempo exponencial então B também exige.

Intratabilidade

A redução é expressa por $A \leq_P B$
Observe que B pode ser mais dificil que A.



Este processo pode ser chamado de

- mapeamento em tempo polinomial
- redução Karp em tempo polinomial

f(A) pode ser a parte fácil de B.

Intratabilidade

- Reduções x problemas np-completos
  - Suponha:
    - Existe um problema de decisão A para o qual já sabemos que não existe algoritmo polinomial
    - Existe um algoritmo de redução de tempo polinomial transformando instancias de A em instancias de B
  - Logo, não é possivel existir algoritmo polinomial para resolver B

Intratabilidade

- Um primeiro problema NP-Completo?
  - Satisfabilidade (próxima aula)

Intratabilidade

Em termos de linguagem, dizemos que uma linguagem L é **C**-Completa, onde **C** é uma classe de linguagens, se

1. L $\in$ **C**
2. Para qualquer linguagem **A** $\in$ **C**, $A \leq_p L$

Intratabilidade

Problemas Abstratos

Relaciona um conjunto **I** de instâncias e um conjunto **S** de soluções.

Em um problema de decisão o conjunto de soluções é igual a {0 (não) ,1 (sim) }, e a instância é igual a entrada que será processada.

Quando as instâncias de um problema são codificadas em cadeias binárias para serem executadas em um computador, este problema passa de um **problema abstrato para um problema concreto**.

Intratabilidade

Um algoritmo **resolve** um problema concreto no tempo O(T(n)) se ele é capaz de produzir uma resposta em um tempo máximo O(T(n)), onde **n=| I |** (tamanho da entrada) e **I** corresponde a uma instância do problema.

Um problema pode ser resolvido em **tempo polinomial** se existe um algoritmo para resolvê-lo no tempo $O(n^k)$, $k \in$ **R.**

Intratabilidade

Linguagem Formal

O conjunto de instâncias para qualquer **problema de decisão Q** pode ser visto como uma linguagem L definida sob o alfabeto $\Sigma = \{0,1\}$

$$L = \{ x \in \Sigma^* \mid Q(x) = 1 \}$$

(x é o conjunto de todas cadeias binarias)

Um algoritmo A **aceita** uma cadeia $x \in \Sigma^*$ se a saída de A(x) é 1.

Um algoritmo A **rejeita** uma cadeia $x \in \Sigma^*$ se a saída de A(x) é 0.

Uma linguagem é **decidida** por um algoritmo A, se toda cadeia que pertença a L é aceita por A e toda cadeia que não pertença a L é rejeitada por A.

Intratabilidade

Linguagem Formal

Uma linguagem L é **decidida** em *tempo polinomial* por um algoritmo **A**

se existe uma constante **k** tal que , para qualquer cadeia $x \in \{0,1\}^*$

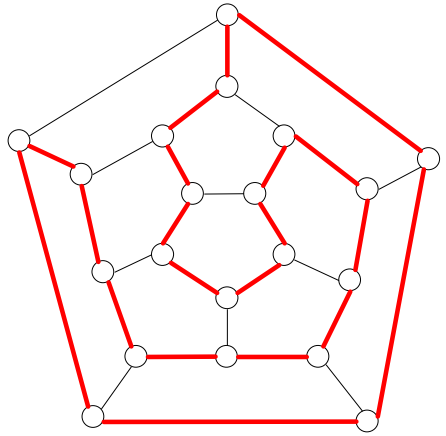
A decide corretamente se $x \in L$ em $O(n^k)$, onde n=| x |.

A **classe P** = { $L \subseteq \{0,1\}^*$ | existe um algoritmo A que decide L em tempo polinomial}.

Intratabilidade

Verificação de tempo polinomial

Considere o problema de encontrar um ciclo hamiltoniano em um grafo não orientado. Ele é definido através da seguinte linguagem



Ham-ciclo ={ <G> | G é um grafo hamiltoniano}

Um algoritmo para **decidir** a linguagem Ham-ciclo necessitaria listar todas as possíveis combinações de vértices de G, e depois verificar cada permutação para ver se ela corresponde a um caminho hamiltoniano.

Este processo demora O(n!).

Logo, este algoritmo **não é executado em tempo polinomial**.

Intratabilidade

Verificação de tempo polinomial

É possível verificar em tempo polinomial se uma dada configuração de vértices é alguma permutação dos vértices de G e se cada aresta consecutiva ao longo do ciclo existe realmente no grafo e forma um ciclo.

- Este algoritmo pode ser implementado para execução no tempo $O(n^2)$
- Essa possível solução é chamada de certificado
- Instancia é neste caso o grafo

A partir dai, definimos um algoritmo **A** de verificação, como sendo um algoritmo de dois argumentos : uma cadeia de entrada **x** (instancia) e uma cadeia binária **y** chamada **certificado**.

A linguagem verificada por A é

$$L = \{ x \in \{0,1\}^* \mid \text{existe } y \in \{0,1\}^* \text{ tal que } A(x,y)=1 \}$$

Verifica se um grafo X, com um certificado y (conjunto de vertices) é ciclo hamiltoniano

Intratabilidade

A classe de complexidade NP

A classe NP (*nondeterministic polynomial time*) é a classe de linguagens que podem ser **verificadas** por um algoritmo em tempo polinomial.

Uma linguagem pertence a NP sse existe um algoritmo **A** que verifica L em tempo polinomial, ou seja,

$$L = \{ x \in \{0,1\}^* \mid \text{existe } y \in \{0,1\}^* \text{ tal que } A(x,y)=1 \}$$

Exemplo : O problema Ham-ciclo $\in$ NP.

- Para provar que é NP:
  - Elaborar um algoritmo de verificação em tempo polinomial (certificado)

Intratabilidade

A classe de complexidade NP

Note que se $L \in P$ então $L \in NP$

existe um algoritmo de tempo polinomial para decidir **L**, o qual pode ser convertido em um algoritmo de verificação que ignora o **certificado** e aceita as cadeias fornecidas como entrada.

Logo $P \subseteq NP$, mas não sabemos se P = NP !

Intratabilidade

A classe de complexidade NP

Em 1971, Stephen Cook mostrou que todos os problemas da Classe NP podem ser reduzidos ao problema SAT.

Se SAT admitir um algoritmo polinomial, então qualquer problema em NP pode ser resolvido por este algoritmo. Assim, SAT é dito NP-Completo.

→primeiro problema a ser provado como NP-Completo!!

Em 1972, Richard Karp reduziu SAT a diversos outros problemas da Classe NP.

Ele foi o autor da pergunta P = NP?

Intratabilidade

Classe NP - Completo

Possui a propriedade de que se qualquer problema NP-completo puder ser resolvido em tempo polinomial então todo problema NP terá uma solução em tempo polinomial e P=NP.

Para provar que um problema P $\in$ NP é um problema NP-Completo, devemos o reformular em termos de um problema Q já conhecido como sendo NP-Completo.

Isto pode é feito através de redução.

Intratabilidade

Classe NP - Completo

A linguagem L_1 é redutivel em tempo polinomial a uma linguagem L_2 se existe uma função f : $\{0,1\}^* \rightarrow \{0,1\}^*$, computável em tempo polinomial, tal que para todo x $\in$ $\{0,1\}^*$,

$$x \in L_1 \text{ se e somente se } f(x) \in L_2$$

A função **f** é chamada função de redução, e o algoritmo que calcula f é chamado algoritmo de redução.

Isto é representado por

$$L_1 \leq_p L_2$$

Intratabilidade

Classe NP – Completo

Se $L_1 \leq_p L_2$, então L_1 não é mais difícil que L_2

Uma linguagem L é NP-completa se

1. $L \in NP$
2. $L' \leq_p L$ para $L' \in NP$ **-Completo**

- A linguagem pertence a classe NP
- Existe uma linguagem L´ da classe NP-completo que pode ser reduzida em tempo polinomial para L

Intratabilidade

Classe NP – Completo

Se $L_1 \leq_p L$ onde L_1 é NP-Completa então L é NP-Difícil .

Portanto, a classe NP-completo é uma subclasse da **NP-difícil**

Prova: Como L_1 é NP-Completa então para todo $L_2 \in$ NP, $L_2 \leq_p L_1$

Por hipótese, $L_1 \leq_p L$ e por transitividade $L_2 \leq_p L$. Logo L é NP-Díficil.

- **NP- completo**
  - classe que contém os problemas mais difíceis
- **NP-difícil**
  - qualquer problema **pertencente ou não a classe NP**, que seja **redutível a um problema NP-completo**, terá a propriedade de não ser resolvido em tempo polinomial, a menos que P=NP.
  - Estes problemas pertencem a classe NP-difícil
  - Não há verificação em tempo polinomial (não conseguimos provar que a linguagem pertence a NP).

Intratabilidade

Classe NP - Completo

Se existe algum problema NP-Completo que pode ser decidido em tempo polinomial então para todo $A \in NP$, teremos $A \in P$.

Prova: Se $A \in NP$, então por definição de NP-completude, $A \leq_P B$.
Se $B \in P$, então $A \in P$.

Intratabilidade

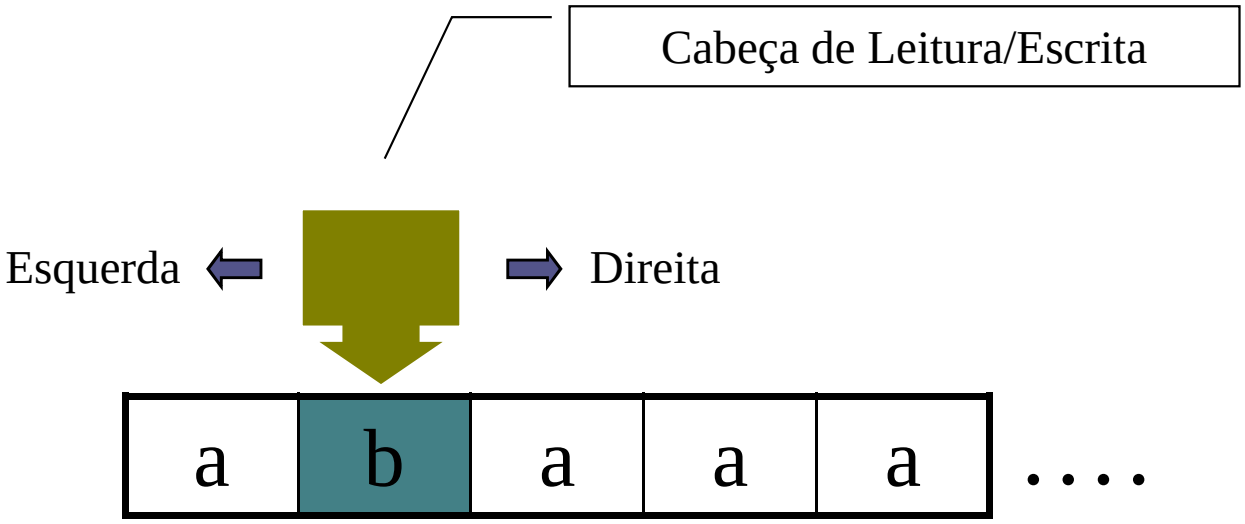
Máquina de Turing

Ela consiste basicamente de 3 partes:

- **Fita**: usada como dispositivo de entrada e saída
- **Unidade de controle:** reflete o estado corrente da máquina. É composta de uma unidade de leitura e gravação (cabeça da fita) que acessa **uma célula** da fita por vez e movimenta-se para esquerda ou direita
- **Função de transição**: comanda as leituras e gravações, o movimento da cabeça da fita e o estado da máquina

Intratabilidade

Máquina de Turing

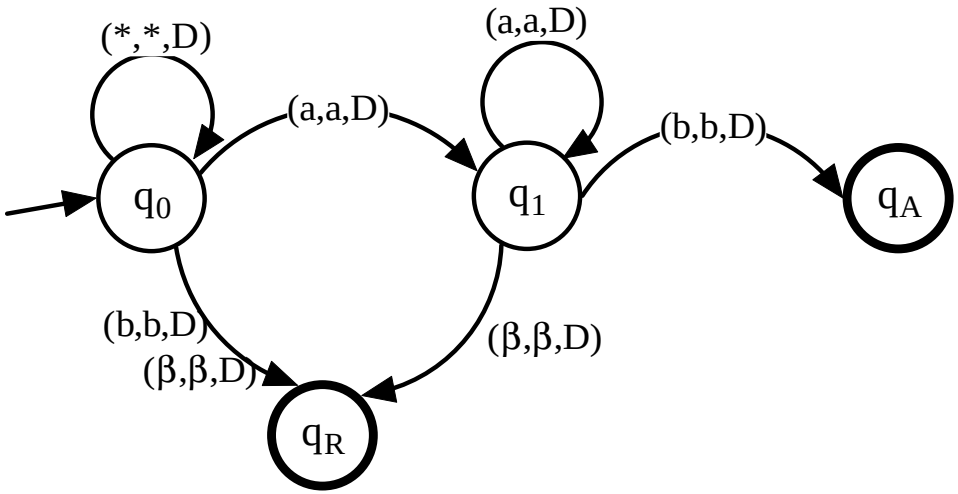


Intratabilidade

Máquina de Turing

É composta de diversos elementos

Q- conjunto de estados.
Σ - alfabeto de entrada.
Γ - alfabeto da fita
$\delta : Q \times \Sigma \rightarrow Q \times \Gamma \times \{E,D\}$ -
função de transição.
$\mathbf{q_0}$ - estado inicial.
$\mathbf{q_A} \in Q$ - estado de aceitação.
$\mathbf{q_R} \in Q$ - estado de rejeição.



Intratabilidade

Máquina de Turing Determinística (MTD)

- Uma máquina de Turing aceita uma sentença, se ela alcança uma configuração de aceitação;

- Uma MTD (ou um programa) M aceita $x \in \Sigma^*$ sse M pára no estado q_A após o processar x.

- A linguagem reconhecida é $L_M = \{ x \in \Sigma^* / M \text{ aceita } x\}$.

Intratabilidade

Máquina de Turing Não-Determinística (MTND)

É uma generalização da máquina determinística;

Um programa para uma MTND é definido exatamente como um programa para uma MTD, diferindo somente na execução.

Após a leitura de um símbolo é possível ir para mais que uma configuração da máquina.

Intratabilidade

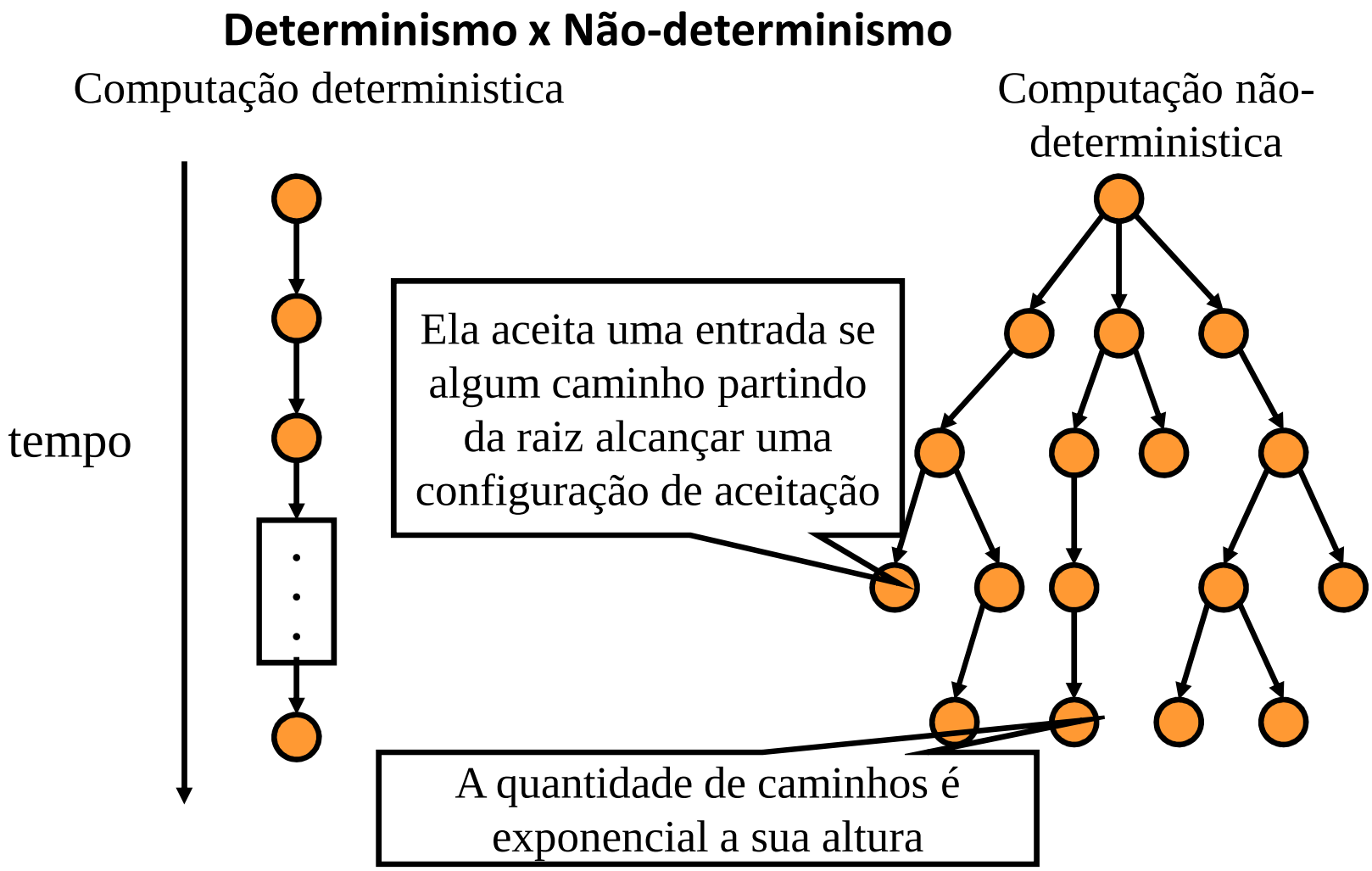
Máquina de Turing Não-Determinística

- Um MTND M aceita uma entrada x se pelo menos uma, dentre todas as possíveis computações de M, pára alcançando um estado de aceitação.

- A linguagem reconhecida é $L_M = \{ x \in \Sigma^* \ / \ M \text{ aceita } x \}$.

- A MTND tem a mesma expressividade da MTD.

Intratabilidade



Intratabilidade

Classe de Complexidade

Classe P

P = {L / existe uma MTD M de tempo polinomial t.q. L_M = L}

Classe NP

NP = { L / existe uma MTND M de tempo polinomial t.q. L_M= L }.

Se existe um algoritmo determinístico de tempo polinomial que resolve um problema de decisão, então este problema é também resolvivel por um algoritmo não determinístico, logo,

$$P \subseteq NP$$

Trabalho

- Assunto:
  mostrar a prova de que um problema da sua escolha pertence a classe NP-completo
- Entrega de artigo
- Apresentação de aprox. 20min
- Trabalho será realizado em duplas

- Datas das apresentações
  - 04/06, 06/06, 11/06, 13/06
- Entrega do artigo:
  - No dia da aula ANTERIOR a apresentação (limite: até a meia noite)

- Escolha do problema:
  - Enviar por e-mail (mariana.kolberg@inf.ufrgs.br) a partir de 18/05 às 8:00 até 22/11: o problema escolhido e o nome dos integrantes do grupo
    - Ordem de apresentação inversa
    - Escolha do problema

Avaliação

- Apresentação (4 ptos)
  - Qualidade dos slides
  - Segurança/dominio do conteudo
  - Erros primários
- Artigo (6 ptos)
  - Caracterização do problema
  - Prova que pertence a NP
    - Algoritmo de verificação
    - Análise da complexidade
  - Prova que pertence a NP-completo
    - Caracterização do problema np-completo a ser usado
    - Redução inst a→inst b
    - Algoritmo de redução
    - Análise da complexidade
  - Escrita

Trabalho

Uma linguagem L é NP-completa se

1. $L \in NP$
2. $L' \leq_p L$ para $L' \in NP$

- Mostrar que pertence a classe NP
  - Elaborar um algoritmo de verificação em tempo polinomial (certificado)
- Mostrar que é possivel reduzir um problema NP-completo ao problema alvo
  - Mostrar algoritmo polinomial de redução

Intratabilidade

Problemas NP - Completo

- Conjunto independente máximo
- Circuito Hamiltoniano
- Caixeiro Viajante
- Clique
- Coloração de grafos
- 3 SAT
- Escalonamento de processos,
- ….