Computabilidade

Aula 5 – Classes de Complexidade

Prof. Rodrigo Mafort

Últimas aulas

- Revisão de Complexidade Computacional
- Revisão de Autômatos
- Máquinas de Turing
- Problemas Indecidíveis Computacionalmente (Problema da Parada)

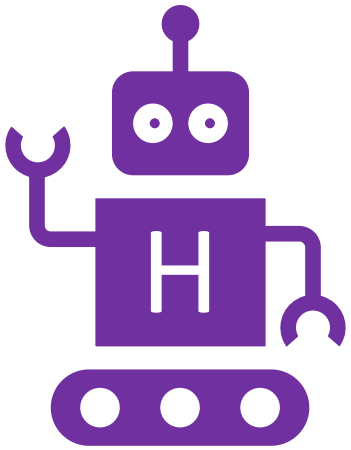
Máquina de Turing

- Composta por três partes:
  - Fita
  - Unidade de Controle
  - Programa

Na última aula

- Problemas Indecidíveis Computacionalmente
  - Problema de Hilbert (ou Décimo Problema de Hilbert)
    - Decidir se um determinado polinômio com mais de uma variável possui ou não raízes inteiras.
    - Como resolver: testar todos os inteiros possíveis em busca de uma atribuição cuja raiz seja 0.
    - Problema: São infinitas possibilidades
  - Problema da Parada:
    - Existe uma máquina capaz de verificar se uma outra máquina funciona (entra em aceitação ou rejeição) ou não (a máquina não para - entra em loop infinito)?
    - Provamos na última aula que essa máquina não pode existir.

Problema da Parada



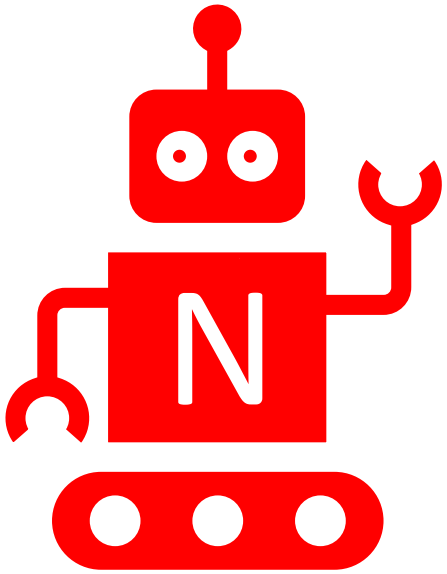
- Vamos criar agora uma máquina hipotética H

- H recebe:
  - Como uma máquina qualquer funciona (diagramas, programas, etc...)
  - Uma entrada qualquer para essa máquina

- Dadas as entradas, H responde:
  - A máquina vai funcionar corretamente (vai parar em um estado de aceitação ou rejeição)
  - Ou vai resultar em um erro (a máquina vai entrar em loop – não vai parar).

- Vamos supor que a máquina H funciona e retorna sempre a resposta correta: Dada a máquina e a entrada a máquina funciona (para) ou não (entra em loop). Isso significa que **H resolve o Problema da Parada**.

Problema da Parada



- Vamos definir agora uma máquina N

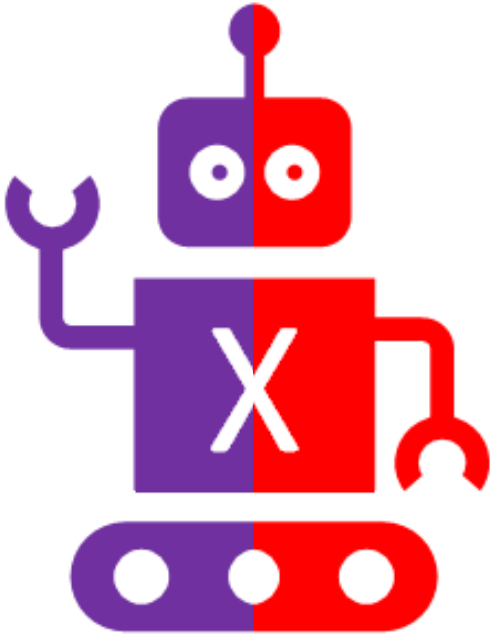
- Seu funcionamento é simples:
  - Se ela receber uma informação: Funciona (Para em aceitação ou rejeição)
  - Ela resultará em: Não Funciona (Entra em Loop – Não para)

  - Se ela receber uma informação: Não Funciona (Entra em Loop – Não ´para)
  - Ela resultará em: Funciona (Para em aceitação ou rejeição)

- Em resumo: ela retorna a negação da informação recebida (o contrário da informação: Sim vira Não e Não vira Sim).

Problema da Parada

- Vamos juntar a máquina H e a máquina N como uma única máquina: a Máquina X.

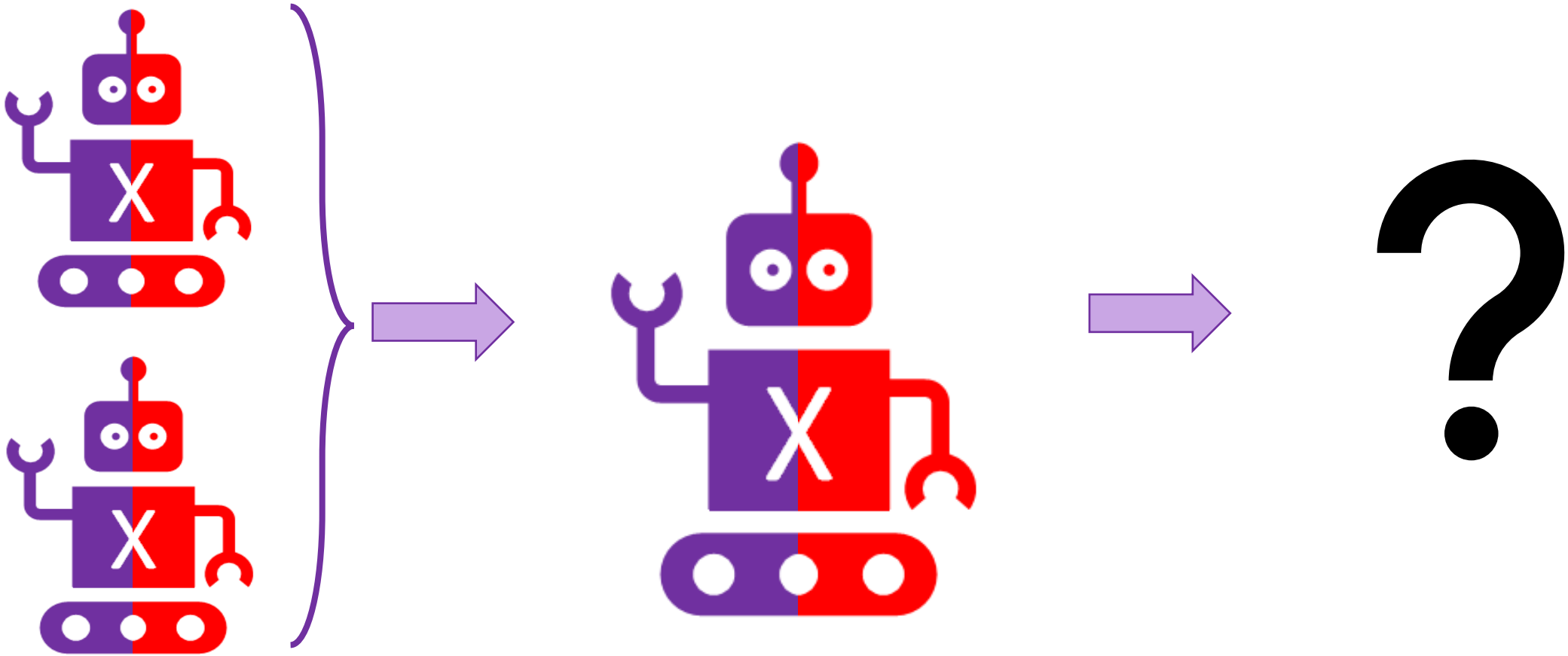


Problema da Parada

- O que a máquina X recebe?
  - O “esquema” da máquina – o programa (função de transição)
  - A entrada que desejamos avaliar
- A entrada é correspondente à máquina em questão:
  - Máquina de cálculos aritméticos → cálculos
  - Máquina de encaixe geométrico → formatos
- O que a máquina X retorna?
  - Se a máquina funciona (para)
  - Ou se a máquina não funciona (entra em loop – não para).

Problema da Parada

- Vamos considerar o seguinte caso:



Problema da Parada

- O que a máquina X recebe?
  - “Esquemas” de máquinas – os programas (função de transição)
  - As entradas que desejamos avaliar
- E nesse caso:
  - Esquema da máquina: o programa da própria máquina X
  - A entrada que desejamos avaliar: o programa da própria máquina X
- Em outras palavras:
  - Vamos usar uma máquina X para avaliar:
  - A máquina X funciona ou não quando recebe o esquema da máquina X?

Problema da Parada

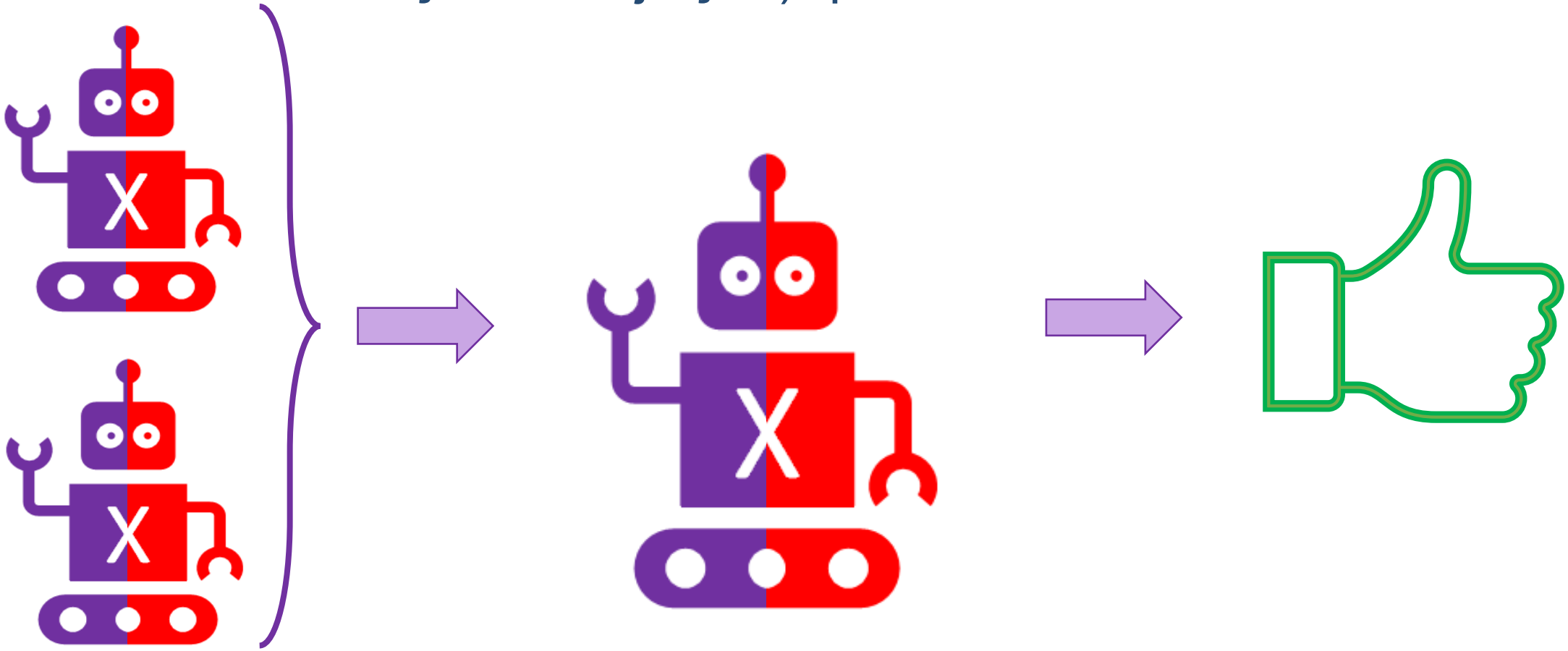
- O que esse caso diz?

- Vamos usar a própria máquina X para descobrir se a máquina X funciona ou não quando recebe como entrada o esquema da própria máquina X.

- Vamos considerar duas possibilidades:
  1. Assumir que a máquina X funciona (Para – Entra em aceitação ou rejeição)
  2. Assumir que a máquina X não funciona (Não Para – Entra em Loop)

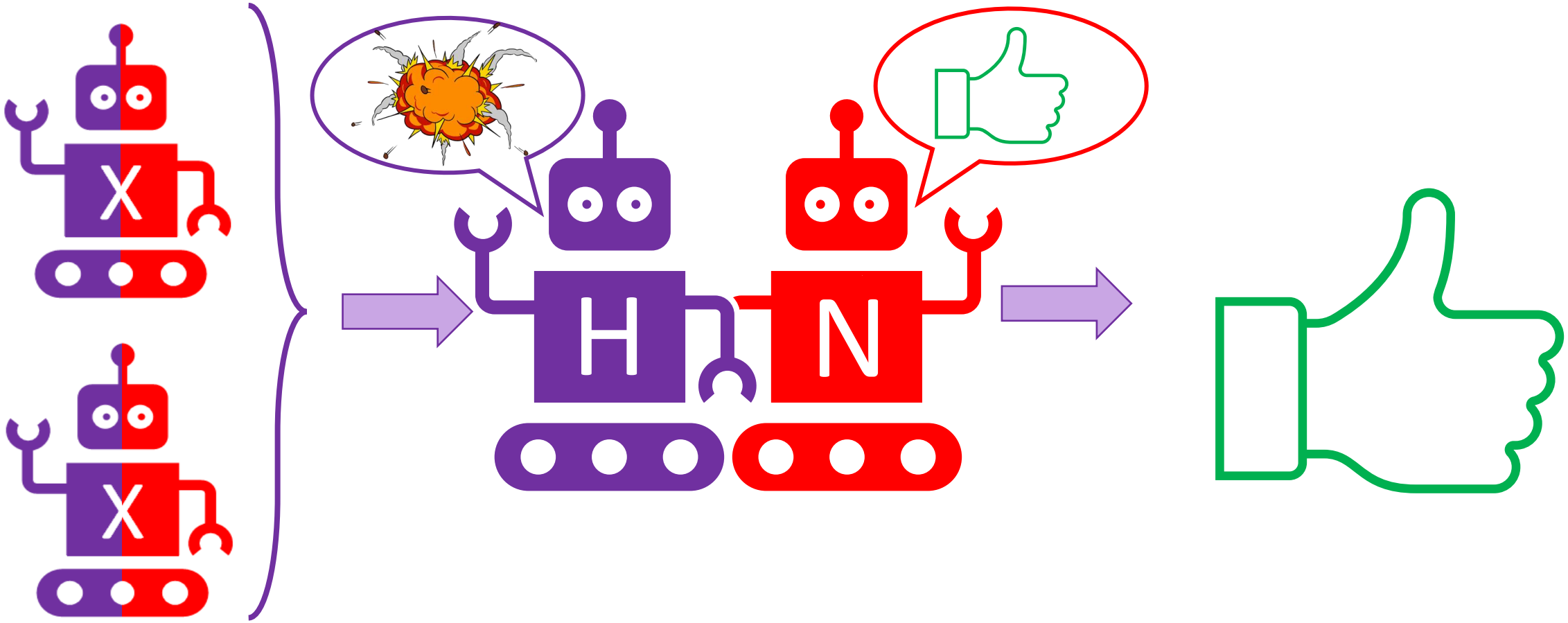
Problema da Parada

- Caso 1: Vamos assumir que a máquina X funciona (para em um estado de aceitação ou rejeição) quando avalia.



Problema da Parada

- Caso 1: Vamos assumir que a máquina X funciona (para) quando avalia.



Problema da Parada

- Temos um problema...

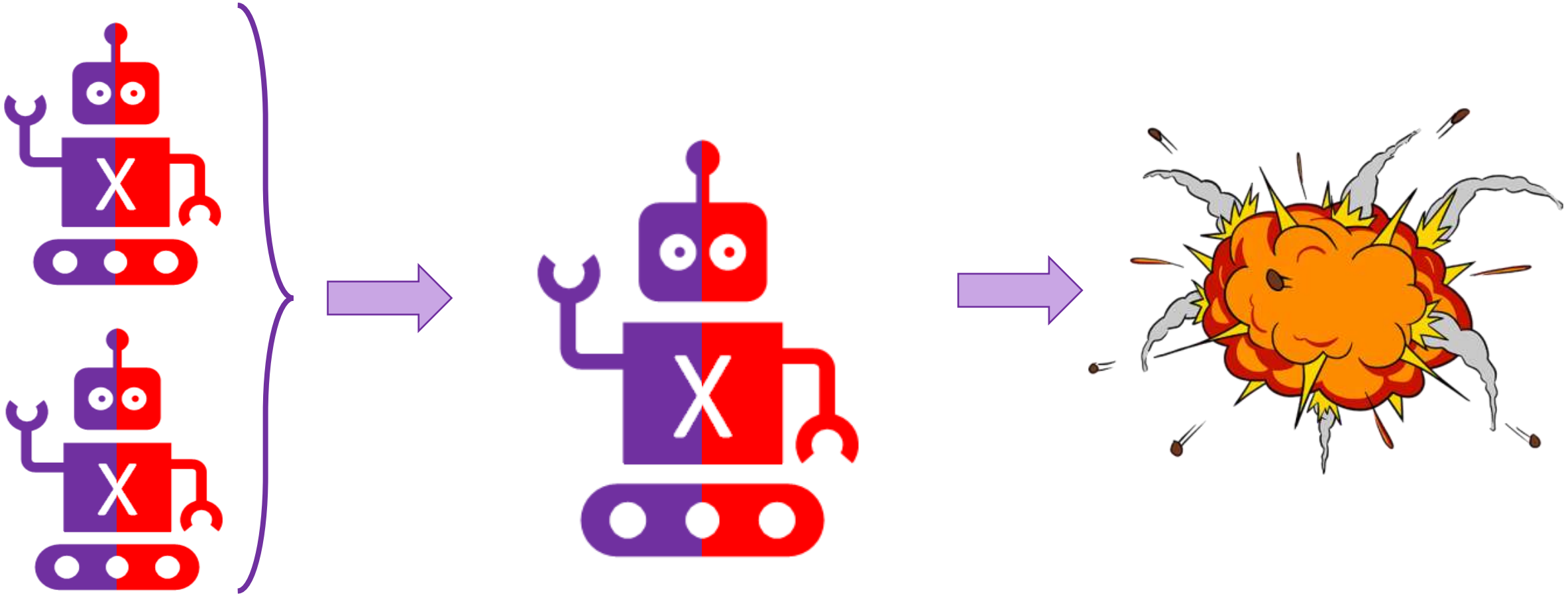
- A máquina H disse que a máquina X não ia parar (entrar em loop)
- Entretanto, ela entrou em um estado de aceitação ou rejeição.

- A máquina H disse que não ia parar (entrar em loop)
- Mas a máquina X parou (não entrou em loop)...

- Logo: A Máquina H estava errada nesse caso.

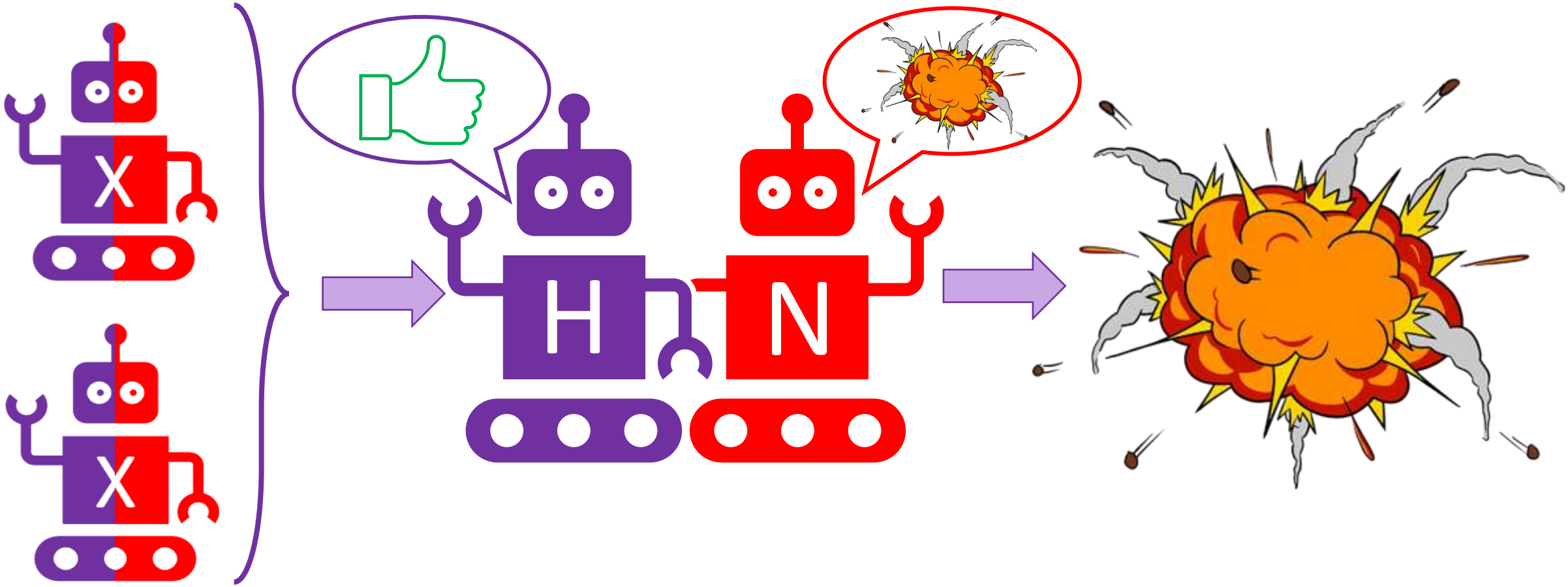
Problema da Parada

- Caso 2: Vamos assumir que a máquina X não para (entra em loop) quando avalia.



Problema da Parada

- Caso 2: Vamos assumir que a máquina X não para (entra em loop) quando avalia.



Problema da Parada

- Temos outro problema...

- A máquina H disse que a máquina X ia funcionar (entrar em estado de aceitação ou rejeição).
- Entretanto, ela entrou em um loop (não para).

- Logo: A Máquina H estava errada nesse caso também.

Problema da Parada

- Em ambos os casos observamos que a resposta de H está errada.
- Mas como? Assumimos que H existe e está sempre correta.

- Isso é uma contradição!
- Logo a assumpção inicial estava errada

- Essa máquina H não pode existir (assim como X).

- Isso que dizer: não existe uma máquina capaz de decidir se alguma outra máquina qualquer, quando iniciada a partir de uma entrada qualquer, eventualmente funciona (para ou entra em loop).

Problema da Parada

- Em ambos os casos observamos que a resposta de H está errada.
- Mas como? Assumimos que H existe e está sempre correta.

- Isso é uma contradição!
- Logo a assumpção inicial estava errada

- Essa máquina H não pode existir (assim como X).

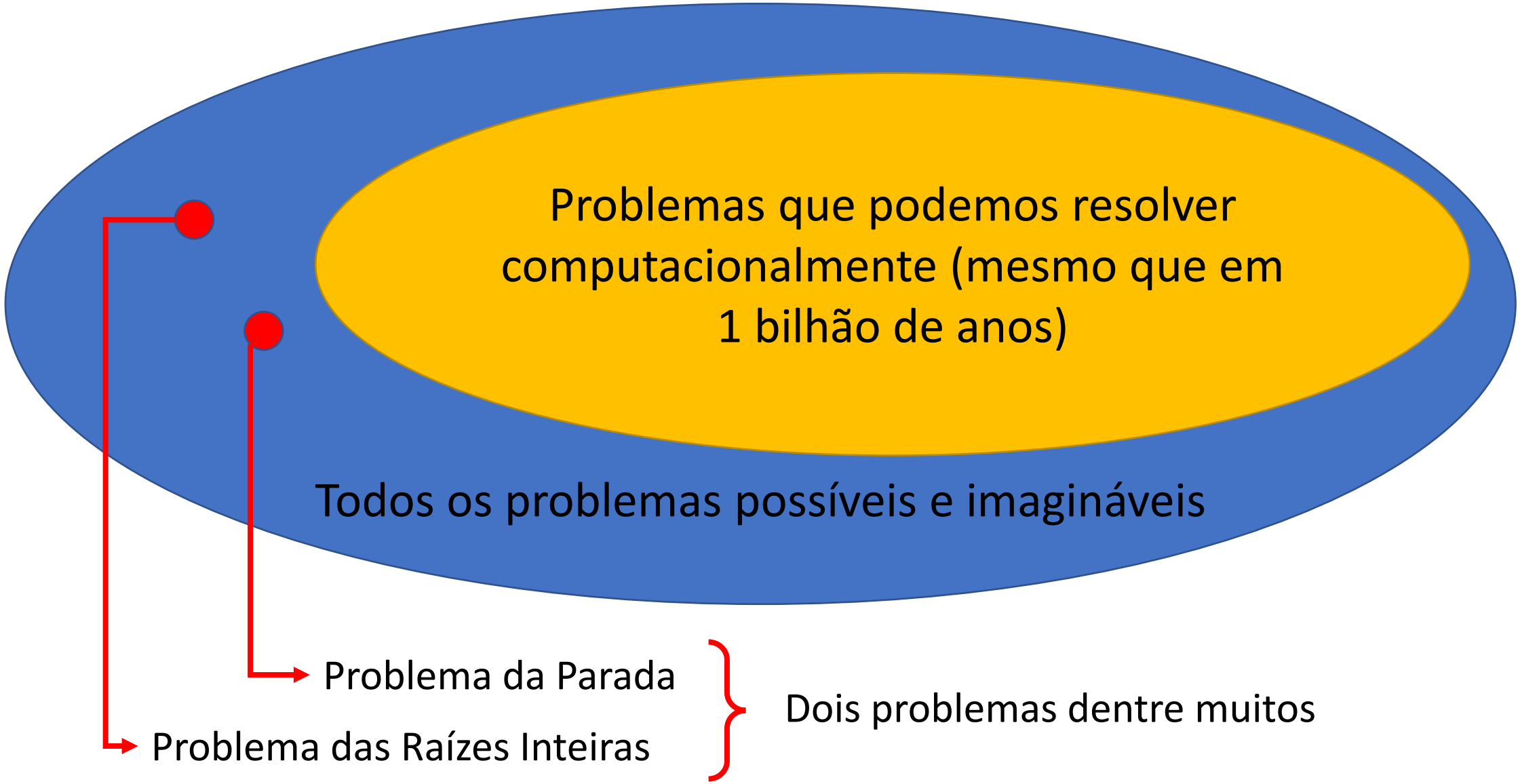
- Isso que dizer: não existe um **algoritmo** capaz de decidir se algum outro **algoritmo** qualquer, quando iniciado a partir de uma entrada qualquer, eventualmente funciona (para ou entra em loop infinito).

Problema da Parada

- Isso que dizer: não há algoritmo para decidir se alguma determinada máquina, quando iniciada a partir de uma entrada qualquer, eventualmente funciona (para ou entra em loop).

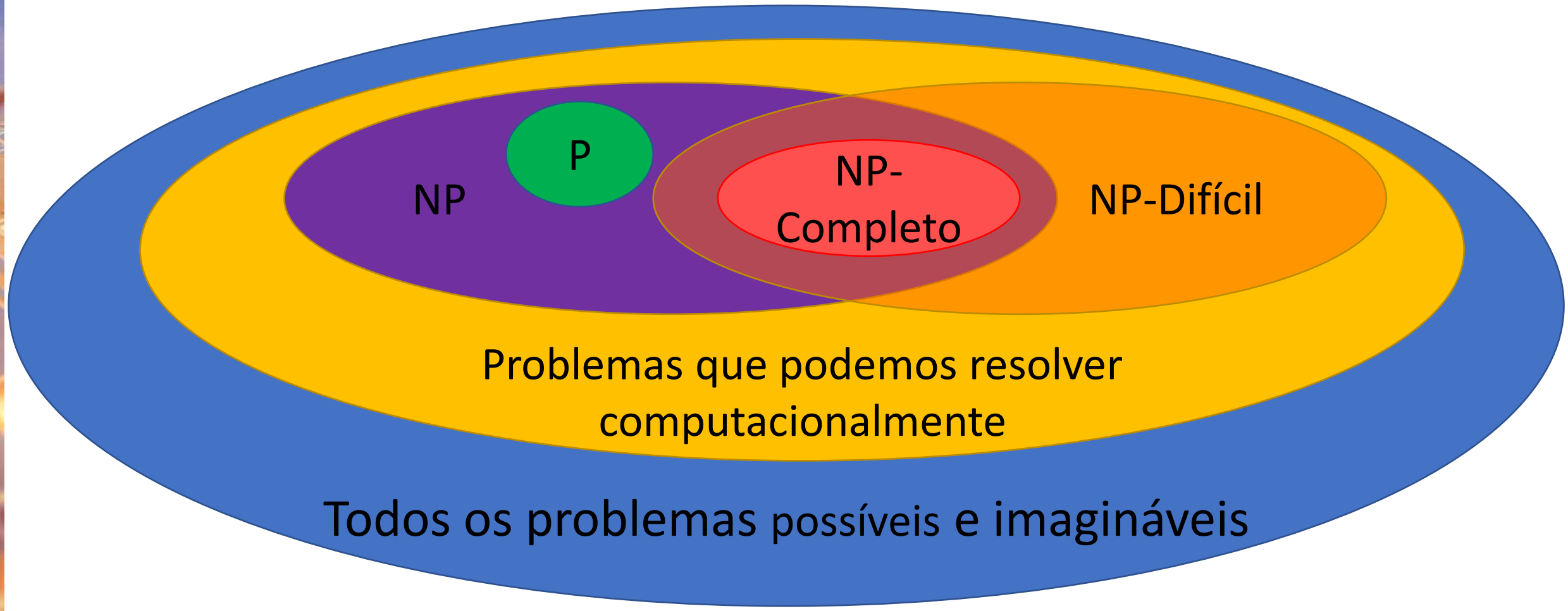
- **Não existe um algoritmo que determina se um outro algoritmo qualquer funciona ou não a partir de uma entrada qualquer.**

Universo de Problemas



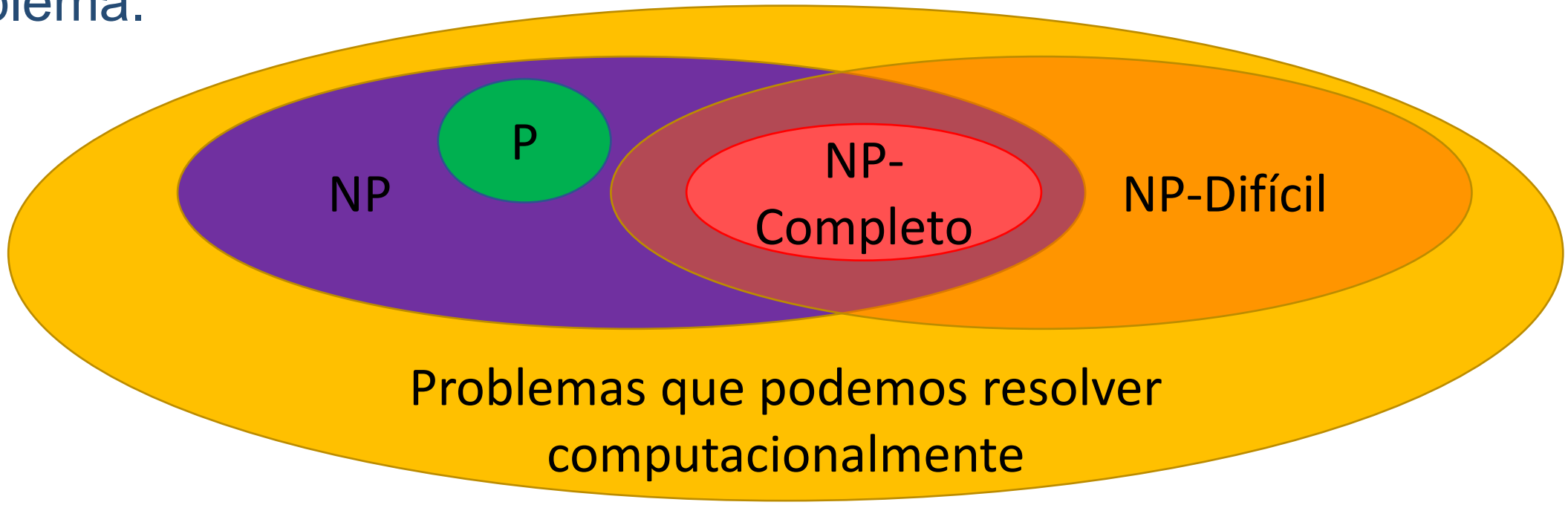
Universo de Problemas

- Podemos classificar os problemas contidos no conjunto de problemas que podemos resolver computacionalmente.



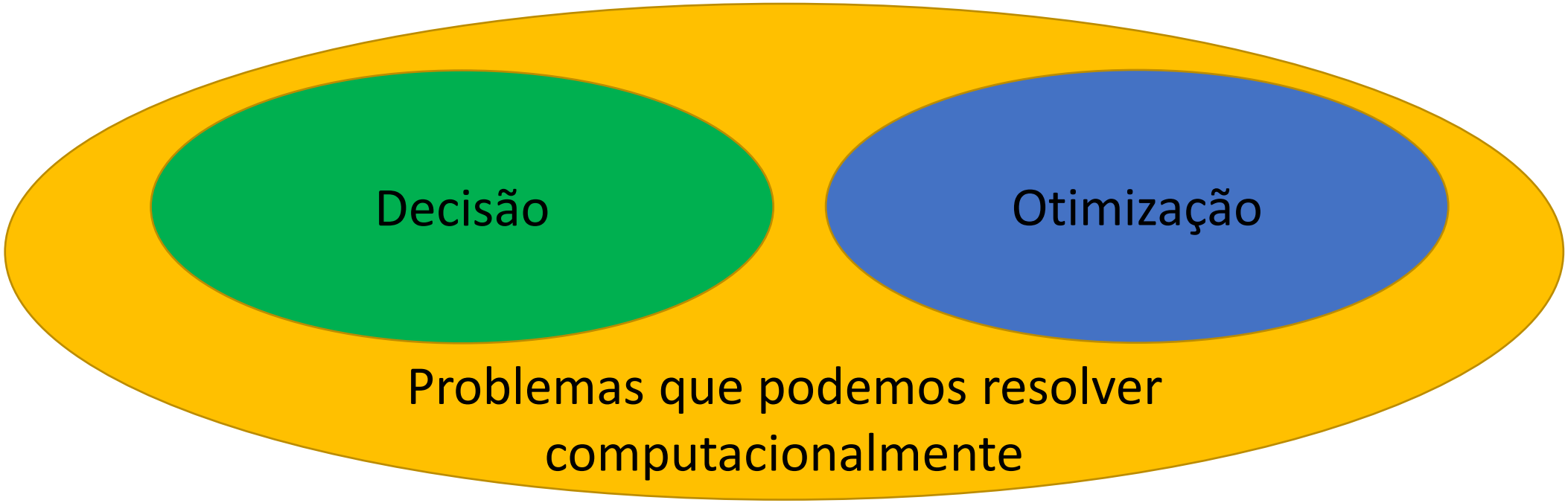
Classes de Problemas

- Essa classificação pode ser realizada em função da complexidade de tempo ou da complexidade de espaço.
- Em geral, os problemas são classificados em função da complexidade de tempo do melhor algoritmo já encontrado para esse problema.



Problemas de Decisão e Otimização

- Dentre o universo de problemas que podemos resolver computacionalmente, temos uma divisão importante:
  - Problemas de Decisão
  - Problemas de Otimização



Problemas de Decisão

- São problemas cuja resposta é SIM ou NÃO.

- Por exemplo:
  - É possível colorir o mapa ao lado usando apenas $k = 5$ cores de forma que dois países vizinhos não tenham cores iguais?

Problemas de Decisão

- São problemas cuja resposta é SIM ou NÃO.

- Por exemplo:
  - É possível colorir o mapa ao lado usando apenas $k = 5$ cores de forma que dois países vizinhos não tenham cores iguais?

  - Resposta: SIM

Problemas de Decisão

- São problemas cuja resposta é SIM ou NÃO.

- Por exemplo:
  - É possível colorir o mapa ao lado usando apenas $k = 4$ cores de forma que dois países vizinhos não tenham cores iguais?

Problemas de Decisão

- São problemas cuja resposta é SIM ou NÃO.

- Por exemplo:
  - É possível colorir o mapa ao lado usando apenas $k = 4$ cores de forma que dois países vizinhos não tenham cores iguais?

  - Resposta: SIM

Problemas de Decisão

- São problemas cuja resposta é SIM ou NÃO.

- Por exemplo:
  - É possível colorir o mapa ao lado usando apenas $k = 3$ cores de forma que dois países vizinhos não tenham cores iguais?

  - Resposta: ???

Problemas de Decisão

- São problemas cuja resposta é SIM ou NÃO.

- Por exemplo:
  - É possível colorir o GRAFO ao lado usando apenas $k = 3$ cores de forma que dois países vizinhos não tenham cores iguais?

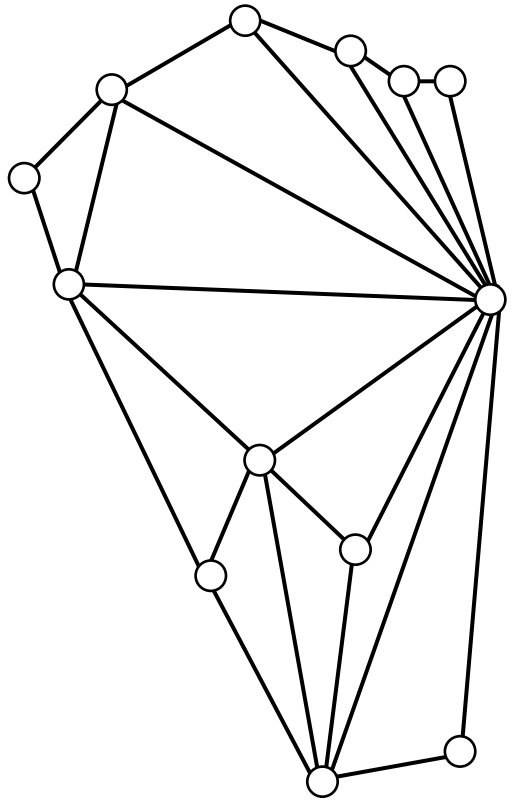
  - Resposta: ???

Problemas de Decisão

- São problemas cuja resposta é SIM ou NÃO.

- Por exemplo:
  - É possível colorir o GRAFO ao lado usando apenas $k = 3$ cores de forma que dois países vizinhos não tenham cores iguais?

  - Resposta: ???

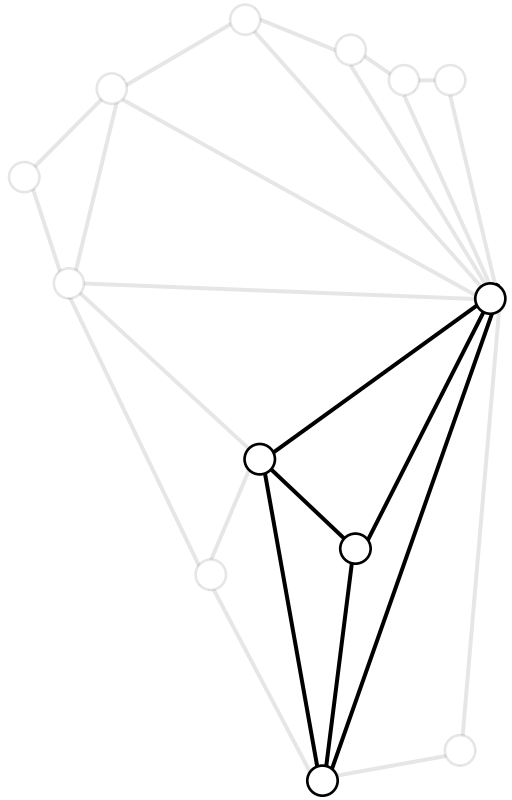


Problemas de Decisão

- São problemas cuja resposta é SIM ou NÃO.

- Por exemplo:
  - É possível colorir o GRAFO ao lado usando apenas $k = 3$ cores de forma que dois países vizinhos não tenham cores iguais?

  - Resposta: ???

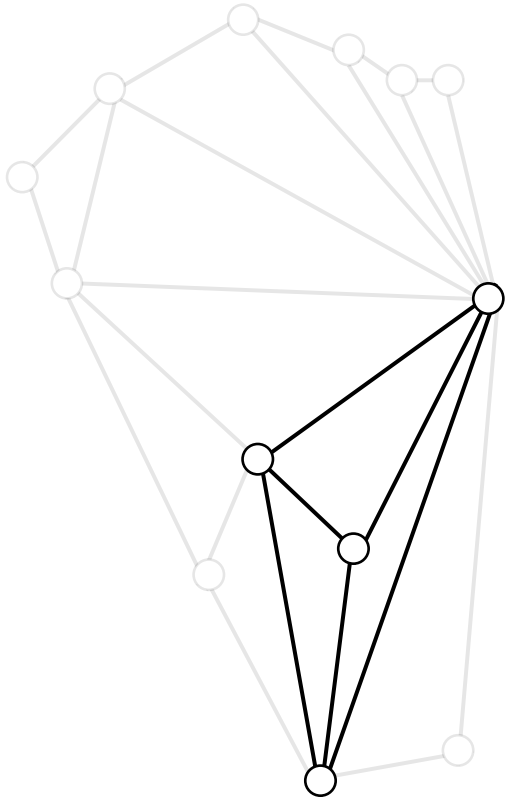


Problemas de Decisão

- São problemas cuja resposta é SIM ou NÃO.

- Por exemplo:
  - É possível colorir o GRAFO ao lado usando apenas $k = 3$ cores de forma que dois países vizinhos não tenham cores iguais?

  - Resposta: Não! Essa parte do grafo requer no mínimo 4 cores, pois cada vértice é vizinho dos outros três.

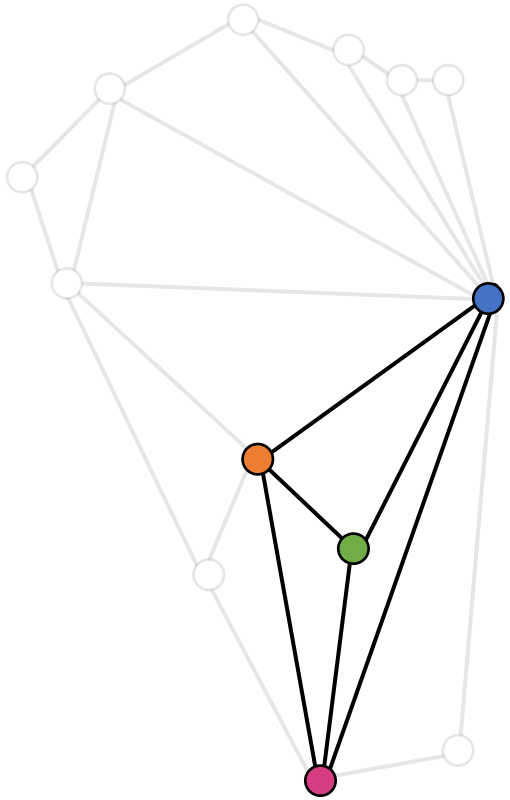


Problemas de Decisão

- São problemas cuja resposta é SIM ou NÃO.

- Por exemplo:
  - É possível colorir o GRAFO ao lado usando apenas $k = 3$ cores de forma que dois países vizinhos não tenham cores iguais?

  - Resposta: Não! Essa parte do grafo requer no mínimo 4 cores, pois cada vértice é vizinho dos outros três.

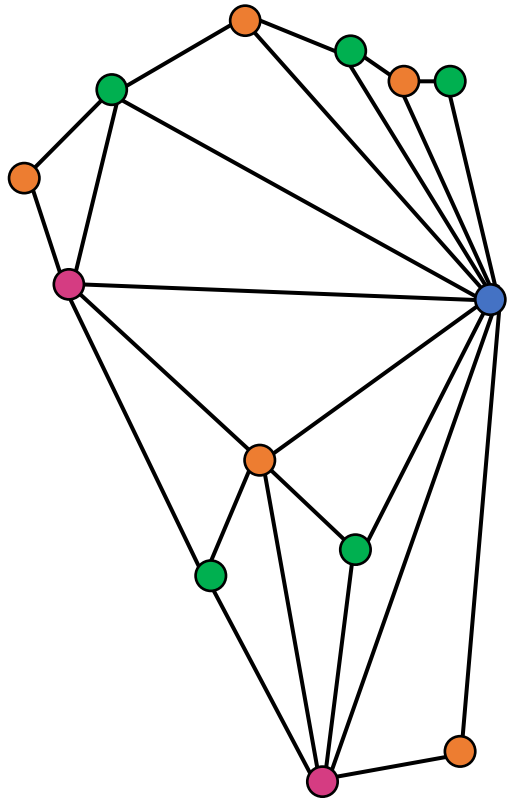


Problemas de Decisão

- São problemas cuja resposta é SIM ou NÃO.

- Por exemplo:
  - É possível colorir o GRAFO ao lado usando apenas $k = 3$ cores de forma que dois países vizinhos não tenham cores iguais?

  - Resposta: Não!



Problemas de Decisão

- São problemas cuja resposta é SIM ou NÃO.

- Por exemplo:
  - É possível colorir o GRAFO ao lado usando apenas $k = 3$ cores de forma que dois países vizinhos não tenham cores iguais?

  - Resposta: Não!

Problemas de Decisão

- São problemas cuja resposta é SIM ou NÃO.

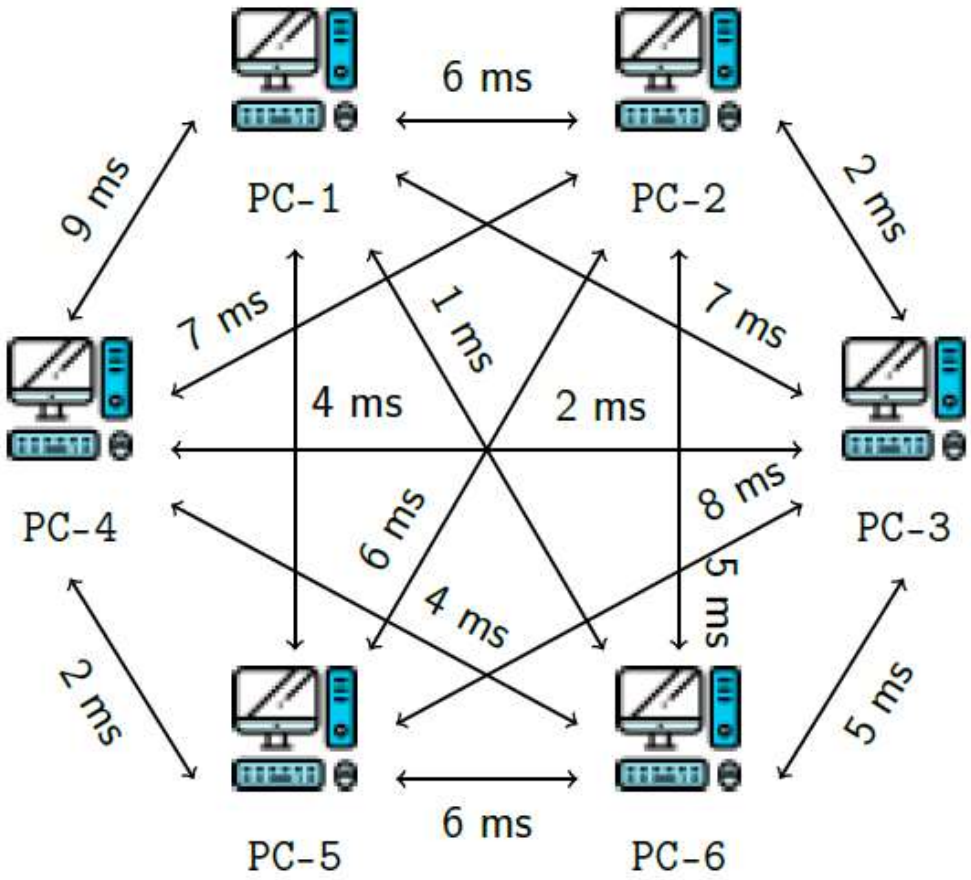
- Por exemplo:
  - É possível colorir o mapa ao lado usando apenas $k = 3$ cores de forma que dois países vizinhos não tenham cores iguais?

  - Resposta: Não!

Problemas de Decisão

- Existe alguma forma de enviar uma informação na rede abaixo, começando no computador $ini = 1$, passando por todos os demais computadores e retornando ao ponto de partida (ini) em menos de $t = 20ms$?

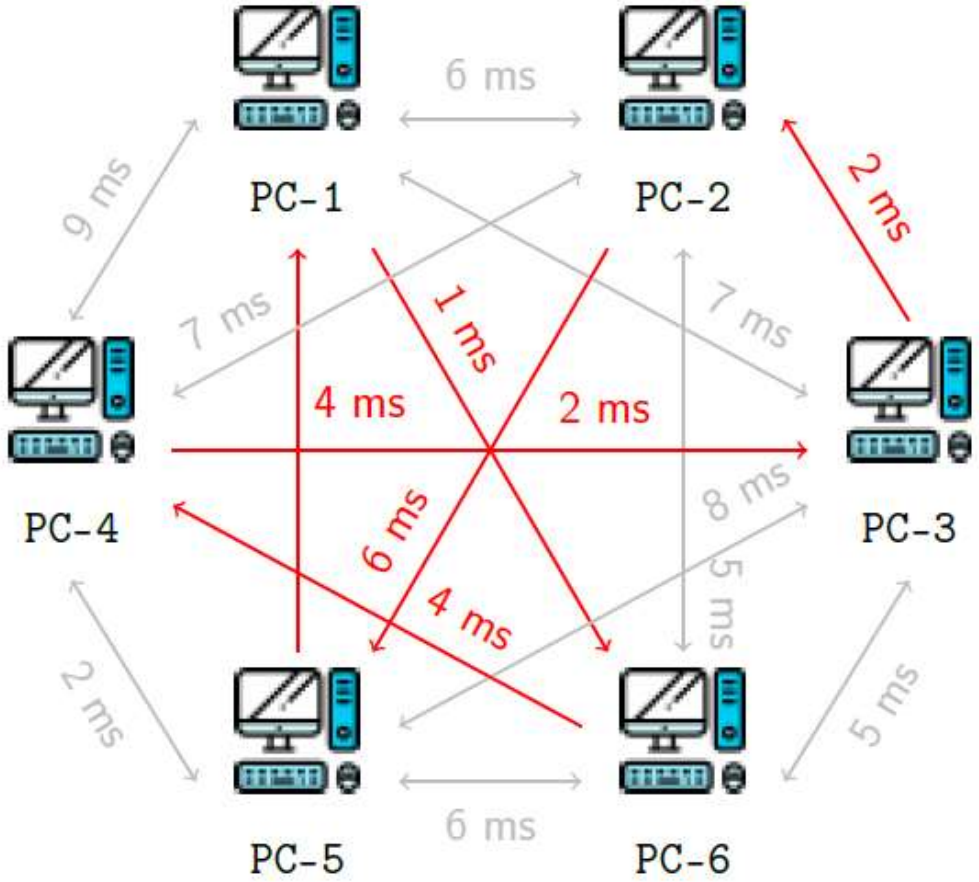
- Resposta:
  - Sim?
  - Não?



Problemas de Decisão

- Existe alguma forma de enviar uma informação na rede abaixo, começando no computador $ini = 1$, passando por todos os demais computadores e retornando ao ponto de partida (ini) em menos de $t = 20ms$?

- Resposta:
  - **Sim**



Problemas de Decisão

- Problema da Mochila 0-1:
  - O Problema da Mochila 0-1 considera que existem n itens, com pesos $p_1, p_2, \cdots, p_n$ e valores associados $v_1, v_2, \cdots, v_n$, e uma mochila de capacidade P
  - Pergunta: Existe alguma combinação dos itens $x_1, x_2, \cdots, x_n$, onde para cada item i: $x_i = \begin{cases} 1, & \text{se o item i for escolhido} \\ 0, & \text{em caso contrário} \end{cases}$ que satisfaça:
  - $\sum_{i=1}^{n} x_i * p_i \leq P$ (não exceder o peso máximo P
  - $\sum_{i=1}^{n} x_i * v_i \geq V$ (a mochila deve contar no mínimo o valor V)

Problemas de Decisão

- Objetivo: Verificar se existe uma solução que atenda as condições previamente determinadas.

- Problema Coloração: No máximo k cores
- Problema da Comunicação: No máximo t milissegundos
- Problema da Mochila 0-1: Levar no mínimo o valor V

- A resposta é SIM (existe essa solução) ou NÃO (não existe uma solução que atenda a essas exigências).

Problemas de Otimização

- Diferente dos problemas de decisão, os problemas de otimização busca a melhor solução possível.

- Problema da Coloração: Qual é o menor número possível de cores que permite pintar o mapa sem que dois países vizinhos tenham cores iguais?

- Problema da Comunicação: Qual é o menor tempo possível para que uma mensagem partindo do computador ini passe por todos os computadores uma única vez e retorne ao ponto de partida?

- Problema da Mochila: Qual é o maior valor que podemos carregar na mochila, considerando o limite de peso e os itens (valores e pesos unitários)?

Problemas de Decisão vs Otimização

- Vale observar que todo problema de otimização possui uma versão de decisão equivalente:
- Por exemplo: Problema da Coloração - Otimização: Qual é o menor número de cores necessárias para pintar o mapa:

- Problema da Coloração - Otimização:
  1. É possível pintar o mapa usando $k = 1$ cores?
  2. É possível pintar o mapa usando $k = 2$ cores?
  3. É possível pintar o mapa usando $k = 3$ cores?

     ⋮
  n. É possível pintar o mapa usando $k = n$ cores, onde n equivale ao número de países/regiões no mapa?

  O valor k do primeiro caso que retornar SIM é a resposta da versão de otimização.

Problemas de Decisão vs Otimização

- Vale observar que todo problema de decisão possui uma versão de otimização equivalente:
- Por exemplo: Problema da Mochila 0-1 - Decisão: Existe alguma combinação de itens que possibilite valor V?

- Problema da Mochila 0-1 - Otimização:
  - Qual é o maior valor X que podemos carregar na mochila, considerando o limite de peso e os itens (valores e pesos unitários)?

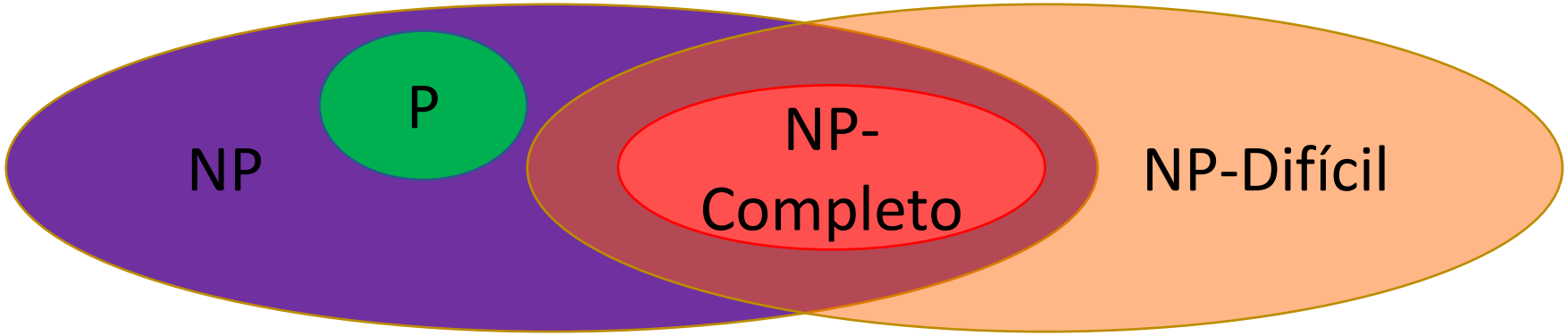
  - Se $X \geq V$, então a resposta da versão de decisão é SIM
  - Se $X < V$, então a resposta da versão de decisão é NÃO

Problemas de Decisão vs Otimização

- Essa relação entre Problemas de Decisão e Otimização implica que saber resolver um implica que sabemos resolver o outro (com algum trabalho extra).
- Nosso estudo das Classes de Problema vai focar nos Problemas de Decisão.
- Entretanto é importante lembrar que o nível de dificuldade de um é equivale ao outro.

Classes de Problemas

- Os Problemas de Decisão podem ser classificados em algumas categorias.
- Dentre outras, as principais são:
  - P
  - NP
  - NP-Difícil
  - NP-Completo



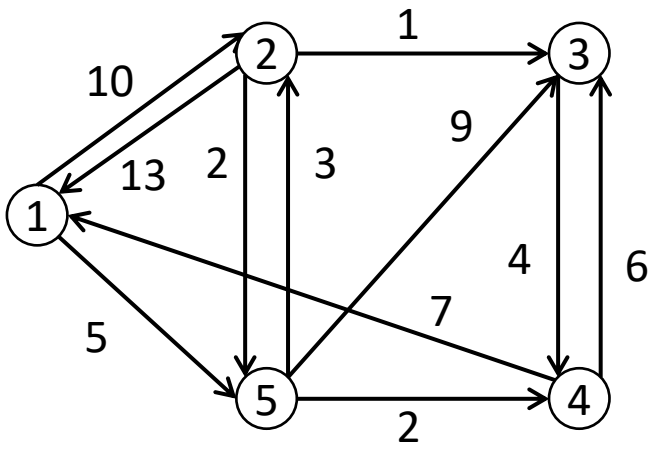
A Classe P

- A Classe P contém os problemas de decisão para os quais se conhece um algoritmo de tempo $O(n^k)$, onde k é uma constante (não cresce junto do n), capaz de resolver o problema.

- Entretanto, como vimos nos slides anteriores:
  - Se sabemos resolver a versão de decisão, sabemos resolver a versão de otimização.
  - Logo: A Classe P contém todos os problemas que podemos resolver computacionalmente em tempo $O(n^k)$

Exemplo – Classe P

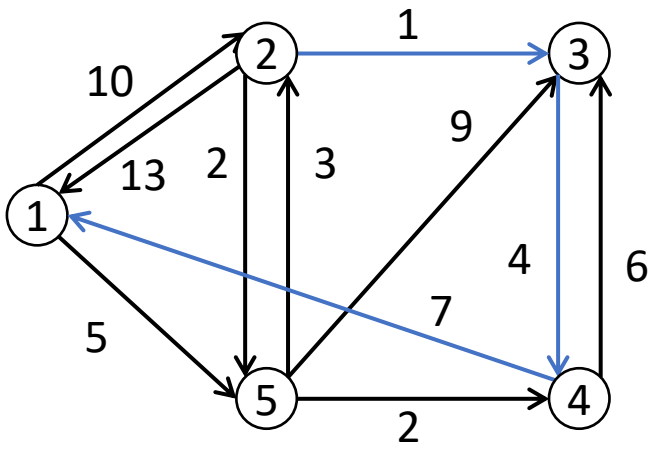
- Entradas: Um grafo com custos associados as arestas e dois vértices v e u
- Questão: Existe um caminho que parte de v e chega até u com custo menor ou igual à k?
- Resposta: Sim ou Não



Ponto de Partida: 2
Ponto de Chegada: 1
Custo Máximo: $k \leq 12$
Resposta: ???

Exemplo – Classe P

- Entradas: Um grafo com custos associados as arestas e dois vértices v e u
- Questão: Existe um caminho que parte de v e chega até u com custo menor ou igual à k?
- Resposta: Sim ou Não



Ponto de Partida: 2
Ponto de Chegada: 1
Custo Máximo: $k \leq 12$
Resposta: SIM

Existe um algoritmo que identifica o menor caminho e seu custo?

Sim. Algoritmo de Dijkstra.
Esse algoritmo é capaz de encontrar o menor caminho entre o vértice inicial e todos os demais vértices do grafo.

Complexidade do Algoritmo: $O(n^3)$

A Classe P

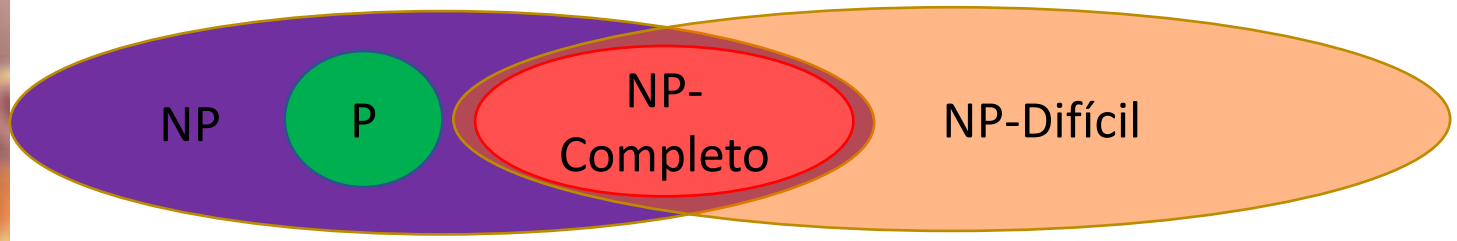
- Dado um problema qualquer.
- Existe um algoritmo de tempo polinomial que resolve esse problema?
  - Sim: O problema está na Classe P
  - Não: Precisamos avaliar mais para classificar esse problema

- Importante: Resolver um problema implica em ser capaz de encontrar a solução ótima para todas as entradas possíveis desse problema.
- Algoritmos aproximativos tem tempo polinomial, mas não encontram a solução ótima. As soluções nesses casos são aproximadas.

A Classe NP

- Se P contém todos os problemas de decisão para os quais se conhece um algoritmo de tempo polinomial.

- Então NP deve conter os problemas de decisão para os quais NÃO se conhece um algoritmo de tempo polinomial.
  - Não!
  - Se fosse assim $P \neq NP$.
  - Mas esse é um problema em aberto!
  - Hoje sabemos apenas que $P \subseteq NP$.



A Classe NP

- O nome NP vem de tempo polinomial não determinístico (*nondeterministic polynomial time*).

- Vale relembrar o conceito de não determinístico:
  - Máquina de Turing Determinística: Todas as possíveis transições devem ser apresentadas (equivale ao conceito de um algoritmo).
  - Máquina de Turing Não Determinística: Explora todos as possíveis transições. A máquina entra em estado de aceitação se alguma combinação de transições levar ao estado de aceitação.

- Como existe uma associação entre Máquinas de Turing e algoritmos: Existe um algoritmo hipotético que é capaz de resolver o problema em tempo polinomial.

A Classe NP

- Como os computadores atuais são determinísticos, eles (ainda?) não conseguem resolver esses problemas em tempo polinomial.

- Mas como verificamos se um problema está em NP se não temos como simular máquinas/algoritmos não determinísticos?

- Vamos considerar que existe uma máquina mágica e hipotética: um oráculo. Esse oráculo é supostamente capaz de resolver o problema e entregar uma resposta.

- Observe que esse oráculo pode ser uma máquina/algoritmo não determinístico. Basta que ele entregue uma resposta para o problema.

A Classe NP

- Sabendo a resposta do oráculo, podemos verificar se ela é uma resposta válida?

- Se for possível CERTIFICAR a resposta SIM em tempo polinomial, o problema está em NP.

- Certificar: Verificar se a resposta é válida para as entradas ou não.

A Classe NP

- Exemplo: Problema do Caixeiro Viajante

- Entradas: Um grafo completo (existe um caminho entre cada par de vértices) com pesos associados as arestas, vértice inicial i, parâmetro k.
- Pergunta: Existe um caminho que comece no vértice i, passe por todos os vértices do grafo e retorne ao ponto de partida com custo total menor ou igual à k?
- Resposta: SIM ou NÃO

A Classe NP

- Suponha que o oráculo respondeu que existe um caminho que atenda essas condições.

- Pergunta: O que você precisa para CERTIFICAR a resposta SIM do oráculo (verificar se ela é válida ou não)?
  - O caminho.

- Como você poderia CERTIFICAR esse caminho?
  - Percorrer o caminho e somar o custo de cada aresta percorrida.

- Qual é a complexidade desse algoritmo certificador?
  - $O(n)$ – o tempo necessário para percorrer o caminho

A Classe NP

- Observe que:
  - Existe um oráculo (máquina de Turing Não Determinística) que pode responder qualquer entrada do problema em tempo polinomial.
  - Podemos CERTIFICAR a resposta SIM em tempo polinomial.

- Logo:
  - Problema do Caixeiro Viajante está em NP.

- Por que esse problema não está em P?
  - Simples: Não sabemos até hoje um algoritmo (determinístico) de tempo polinomial que possa resolver esse problema
  - As soluções conhecidas até hoje são de tempo $O(n!)$ ou aproximativas.

A Classe NP

- Por que CERTIFICAR a resposta SIM e não TODAS as respostas?

- Simples:
  - Resposta SIM: Basta verificar se a resposta satisfaz as condições do problema e da entrada.
  - Resposta NÃO: Como verificar se a resposta NÃO é válida?
    - Seria necessário avaliar todas as possibilidades de respostas para verificar se alguma atende aos critérios (do problema e da entrada).
    - E isso NÃO pode ser feito em tempo polinomial.

A Classe NP

- Por que CERTIFICAR a resposta SIM e não TODAS as respostas?

- SIM: Basta uma resposta para demonstrar que existe uma resposta.
  - Basta um exemplo para provar que é possível

- NÃO: É necessário exaurir todas as respostas possíveis para provar que ela não existe. E isso é muito mais difícil...

A Classe NP

- Exemplo: Problema da Cobertura de Conjuntos

- Considere um conjunto $U = \{1, 2, \cdots, n\}$ e um conjunto S, formados por subconjuntos de U.
- É possível cobrir todos os elementos de U usando no máximo k elementos do conjunto S?

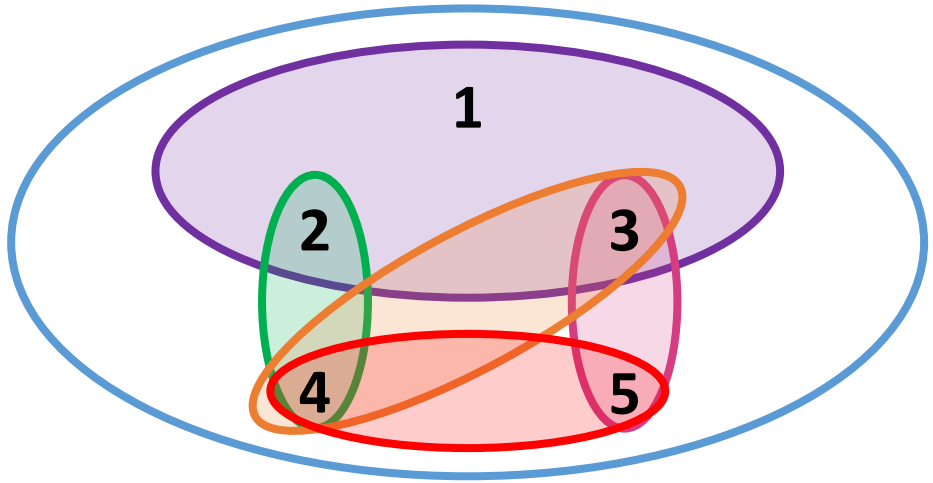
- Resposta: SIM ou NÃO

A Classe NP

- Exemplo: Problema da Cobertura de Conjuntos

- Considere um conjunto $U = \{1, 2, 3, 4, 5\}$ e um conjunto $S = \{\{1, 2, 3\}, \{2, 4\}, \{3, 4\}, \{4,5\}, \{3, 5\}\}$.
- É possível cobrir todos os elementos de U usando no máximo 3 elementos do conjunto S?

- Resposta: SIM ou NÃO

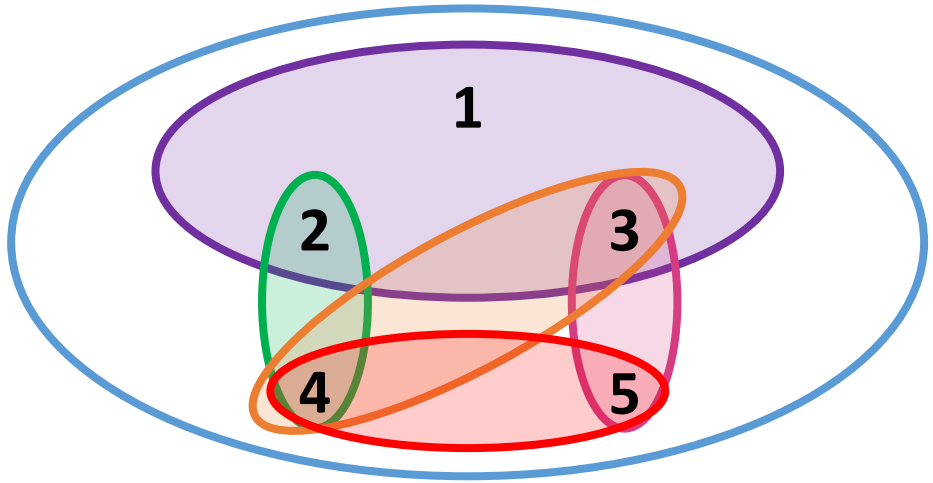


A Classe NP

- Exemplo: Problema da Cobertura de Conjuntos

- Considere um conjunto $U = \{1, 2, 3, 4, 5\}$ e um conjunto $S = \{\{1, 2, 3\}, \{2, 4\}, \{3, 4\}, \{3, 5\}\}$.
- É possível cobrir todos os elementos de U usando no máximo 3 elementos do conjunto S?

- Resposta: **SIM** ou NÃO
- Certificado: $\{1, 2, 3\}, \{2, 4\}, \{3, 5\}$

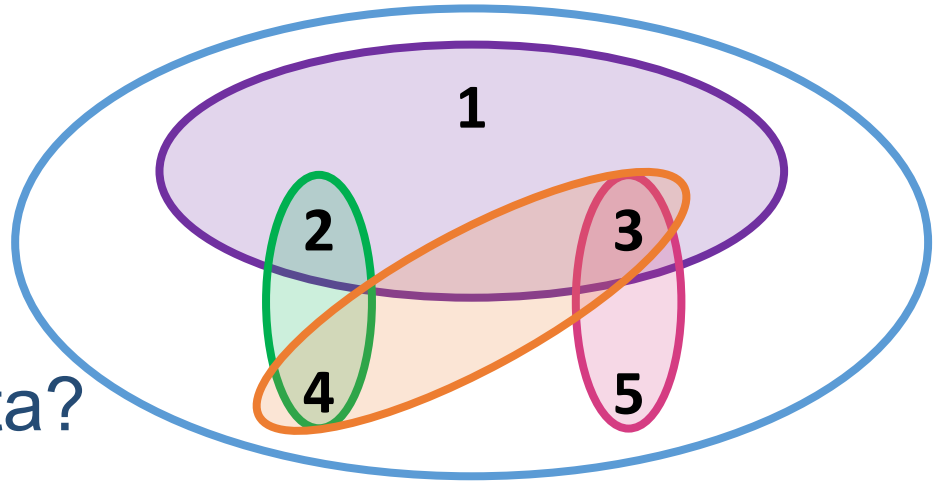


A Classe NP

- Exemplo: Problema da Cobertura de Conjuntos

- Considere um conjunto $U = \{1, 2, 3, 4, 5\}$ e um conjunto $S = \{\{1, 2, 3\}, \{2, 4\}, \{3, 4\}, \{3, 5\}\}$.
- É possível cobrir todos os elementos de U usando no máximo **2** elementos do conjunto S?

- Resposta: SIM ou NÃO
- NÃO (aparentemente)
- Mas como certificar essa resposta?



A Classe NP

- Como certificar a resposta SIM:
  - Basta percorrer cada conjunto da resposta, marcando os vértices.
  - Se, ao final, todos os vértices foram marcados e a resposta contém no máximo k conjuntos então a solução é válida.
  - Complexidade: $O(n)$.

A Classe NP

- Como certificar a reposta NÃO:
  - Necessário testar todas as combinações possíveis de conjuntos com no máximo k elementos.
  - Para cada combinação, marcar todos os vértices. Se todos forem marcados, a resposta NÃO está incorreta.
  - Se nenhuma combinação marcar todos os vértices, o NÃO foi certificado.
  - Complexidade:
    - Considerando que existem no máximo $\sum_{x=1}^{n} \frac{n!}{x!(n-x)!} = 2^n - 1$ subconjuntos.
    - Como no pior caso precisamos testar cada um deles
    - A complexidade dessa certificação é $O(2^n)$

Relação entre P e NP

- P: Sabemos resolver em tempo polinomial.
- NP: Dada uma solução SIM, podemos certificá-la em tempo polinomial.

- Sabendo disso: Qual é a relação entre P e NP?
  - $P \subseteq NP$ ($P = NP$ ou $P \subset NP$) . Por que?

- Se sabemos resolver, sabemos certificar (o SIM e o NÃO).
- Basta resolver e verificar se as respostas são equivalentes.

- Logo: Todo problema em P está também em NP.

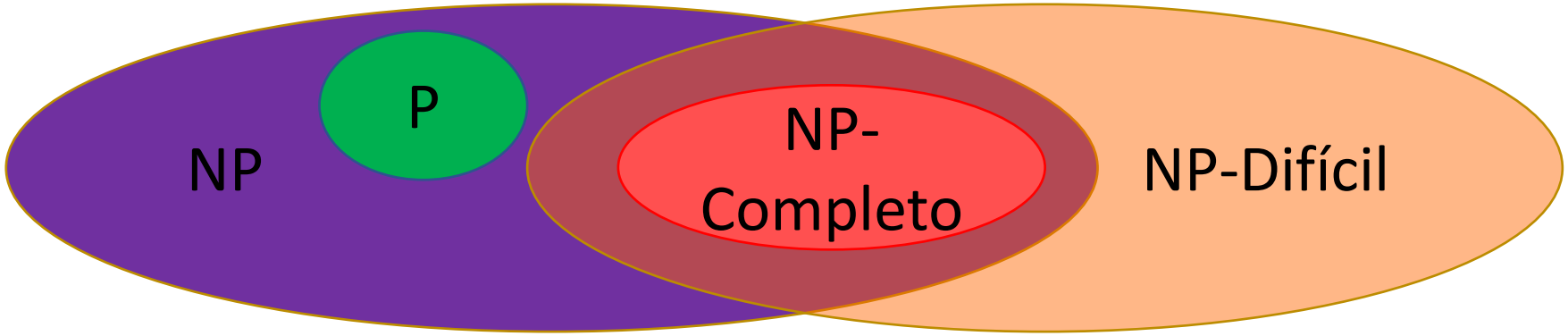
A Classe NP

- A Classe NP é restrita aos problemas de decisão.

- Por que?
  - Como certificar uma resposta para um problema de otimização?
  - Problema de Otimização: Melhor solução possível
  - Dada a solução, como podemos CERTIFICAR que ela é a melhor possível?
  - Seria necessário testar todas as outras soluções em busca de uma ainda melhor.
  - Como vimos, isso é muito custoso computacionalmente

Classes de Problemas

- Os Problemas de Decisão podem ser classificados em algumas categorias.
- Dentre outras, as principais são:
  - **P**
  - **NP**
  - NP-Difícil
  - NP-Completo



NP-Difícil

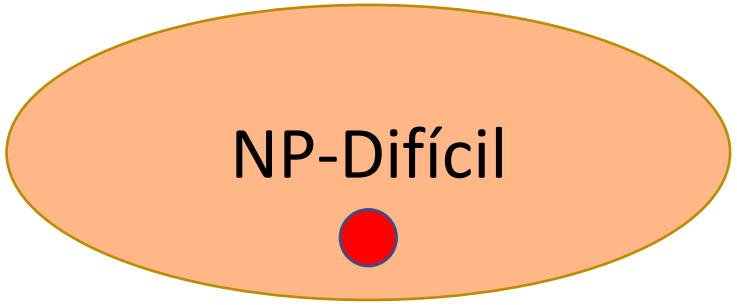
- A classe NP-Difícil contém os problemas mais difíceis de se resolver
- São problemas tão difíceis que TODO problema em NP pode ser transformado em tempo polinomial em um problema NP-Difícil.
- Isso implica que:
  - Se **UM** problema NP-Difícil for resolvido em tempo polinomial então **TODOS** os problemas em NP também podem ser resolvidos em tempo polinomial.
- Vale observar que os problemas de otimização podem estar contidos na classe NP-Difícil.

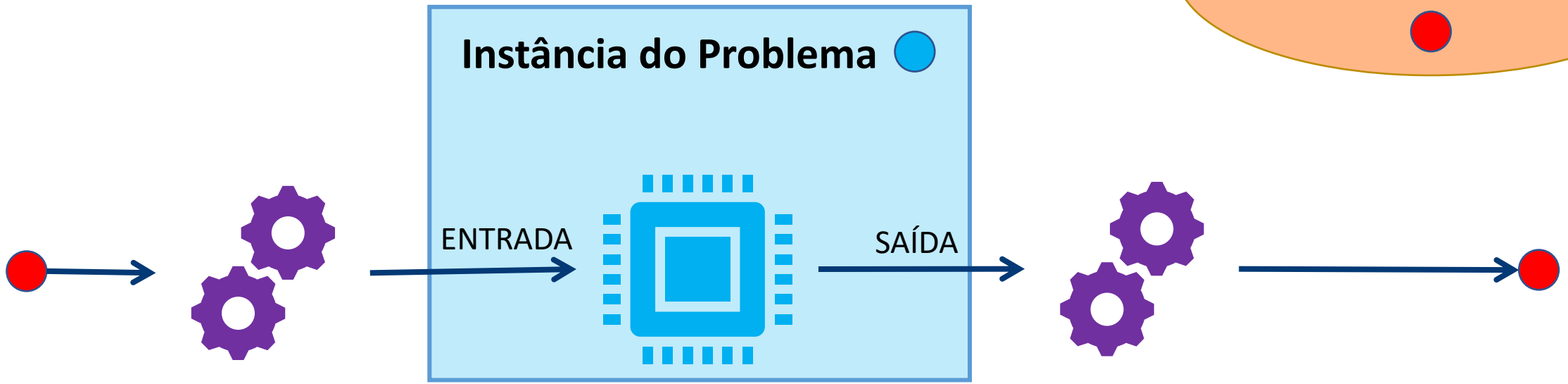
NP-Difícil

- Suponha um problema da classe NP-Difícil
- Suponha um outro problema qualquer
- Se:
  - For possível transformar a entrada do problema NP-Difícil em uma entrada do segundo problema **E**
  - For possível usar o segundo problema para resolver o primeiro **E**
  - Uma resposta SIM para o segundo problema significar uma resposta SIM para o segundo problema
- Então:
  - Podemos resolver o problema difícil usando o segundo problema
  - Isso prova que o segundo problema é pelo menos tão difícil quanto o primeiro.
  - Além disso, se eu conseguir resolver o segundo problema em tempo polinomial, eu posso resolver TODOS os problemas de NP em tempo polinomial.

NP-Difícil

- Suponha um problema 🔴 da classe NP-Difícil
- Suponha um outro problema qualquer 🔵





Transformar a entrada do problema 🔴 em uma entrada para o problema 🔵 em tempo polinomial

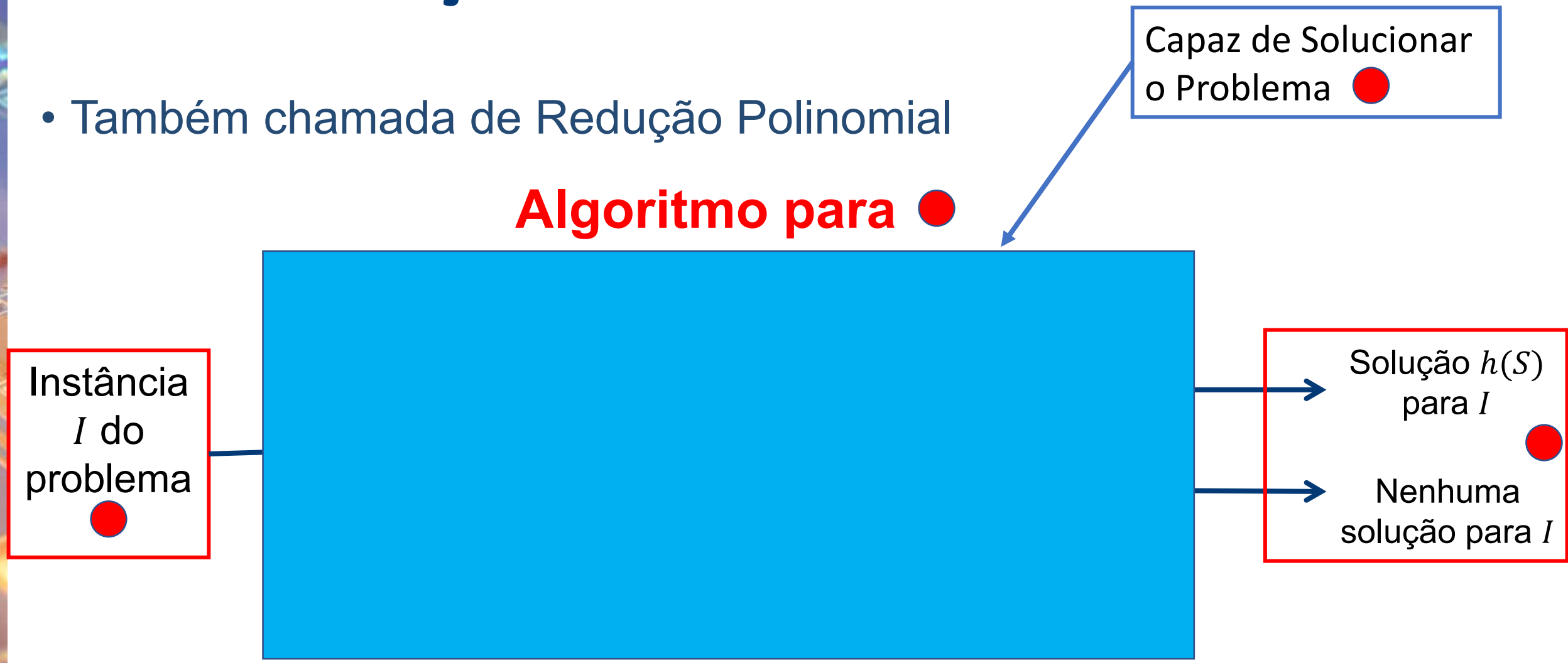
Resolver o problema 🔵

Transformar a saída do problema 🔵 em uma saída para o problema 🔴 em tempo polinomial

Saída para o problema 🔴

Transformação Polinomial

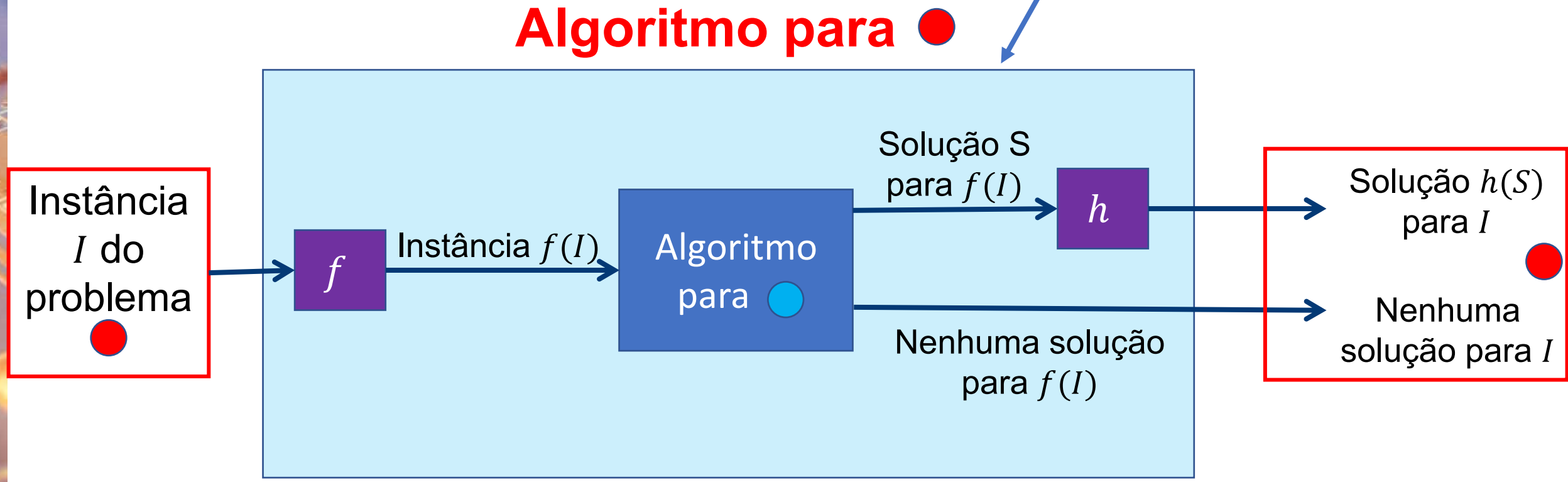
- Também chamada de Redução Polinomial



Transformação Polinomial

- Também chamada de Redução Polinomial

Capaz de Solucionar o Problema 🔴 **USANDO UM ALGORITMO PARA O PROBLEMA** 🔵

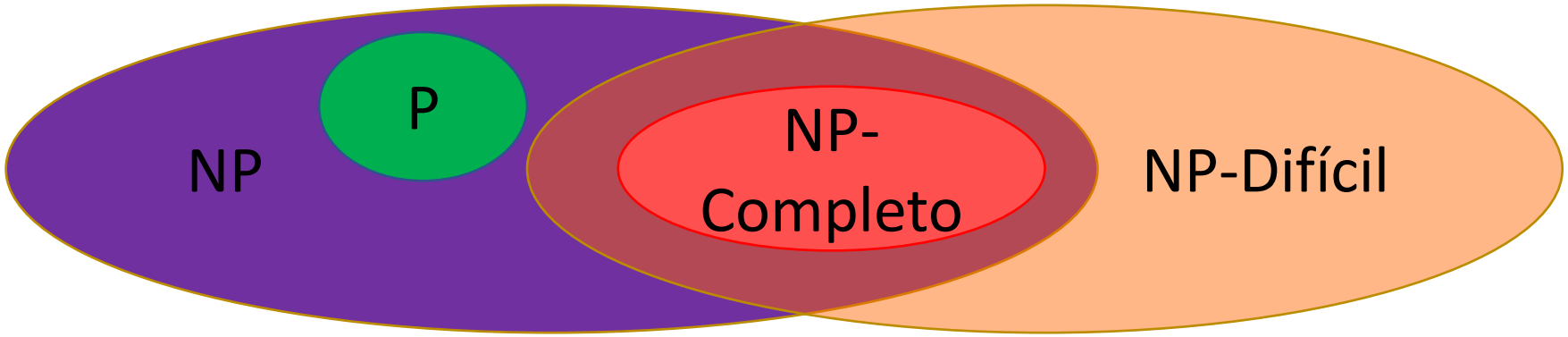


Onde f e h são transformações que requerem tempo polinomial

NP-Completo

- Um problema é classificado como NP-Completo se:
- Pertencer a classe NP
- Pertencer também a classe NP-Difícil

- Logo: São os problemas de decisão mais difíceis (está em NP-Difícil), para os quais podemos certificar uma resposta SIM em tempo polinomial (está em NP).



Por que classificar problemas?

- Seu chefe pediu que você resolvesse um problema.
- Por exemplo: Encontrar a melhor rota para fazer as entregas.

Por que classificar problemas?

- Após tentar bastante, você volta e diz:
  - Eu não consegui encontrar um algoritmo eficiente para resolver esse problema...

- Seu chefe não vai ficar muito feliz...
  - Acho que ele não é bom o bastante...

Por que classificar problemas?

- Após tentar bastante, você volta e diz:
  - Eu não consegui encontrar um algoritmo eficiente para resolver esse problema PORQUE ESSE ALGORITMO NÃO EXISTE

- Acho que seu chefe ainda não vai acreditar...
  - Como ele pode afirmar que o algoritmo não existe?

Por que classificar problemas?

- Após tentar bastante, você volta e diz:
  - Eu não consegui encontrar um algoritmo eficiente para resolver esse problema
  - Mas esse monte de gente famosa também não conseguiu

- Agora ele deve acreditar...

- Do livro do Garey e Johnson

Está faltando alguma coisa?

- Se para indicar que um problema B é NP-Completo precisamos apresentar uma transformação polinomial onde:
- Um problema NP-Completo A é resolvido usando um algoritmo para o problema B.
- Para demonstrar que um problema é NP-Completo precisamos de um outro problema NP-Completo.
- Sendo assim, como o primeiro problema NP-Completo foi categorizado???

Na Próxima Aula

- Formalização do conceito de Transformação Polinomial, com exemplos
- Teorema de Cook-Levin:
  - Satisfatibilidade booleana é NP-Completo
- Definição dos Temas dos Trabalhos sobre NP-Completude