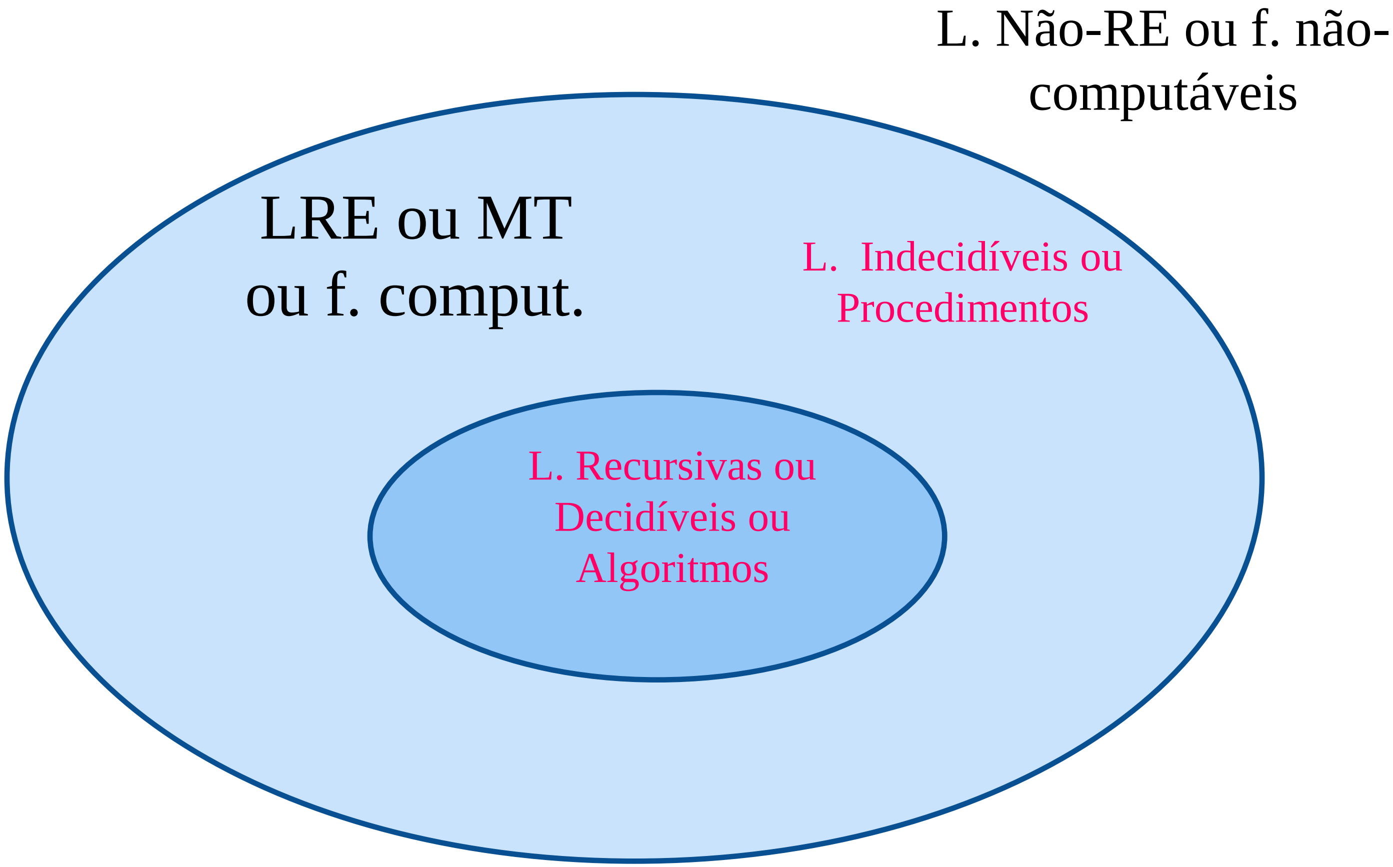
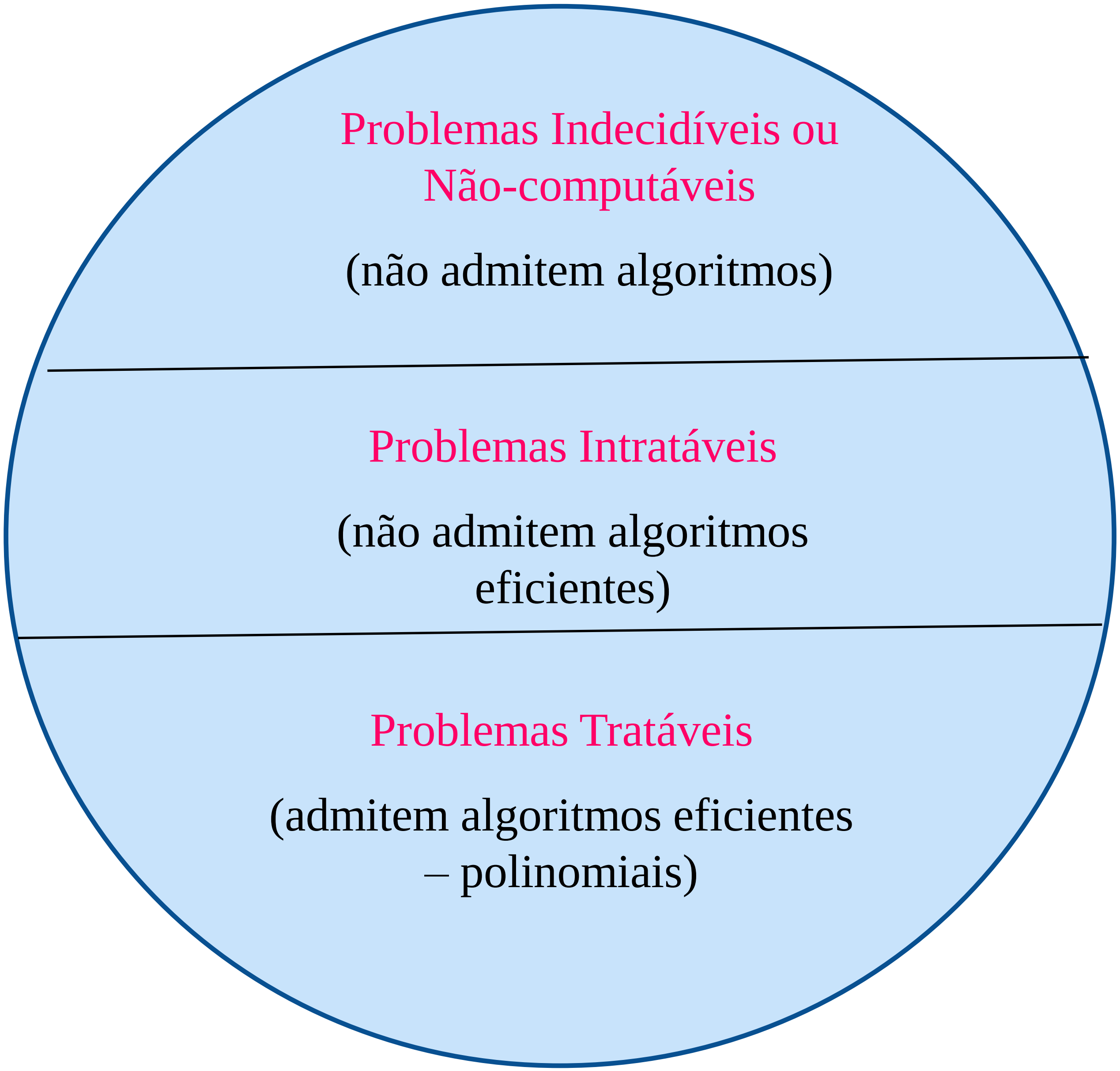
Uma forma de classificação



Outra forma de classificação



Classes de Problemas P e NP

Eficiência de Algoritmos

- Um algoritmo é eficiente precisamente quando sua complexidade for baixa.
- Um algoritmo é eficiente precisamente quando a sua complexidade for um polinômio nos tamanhos da entrada e saída.
- Este critério não é absoluto, mas na maioria dos casos é realmente aceitável.

Classes de Problemas P e NP

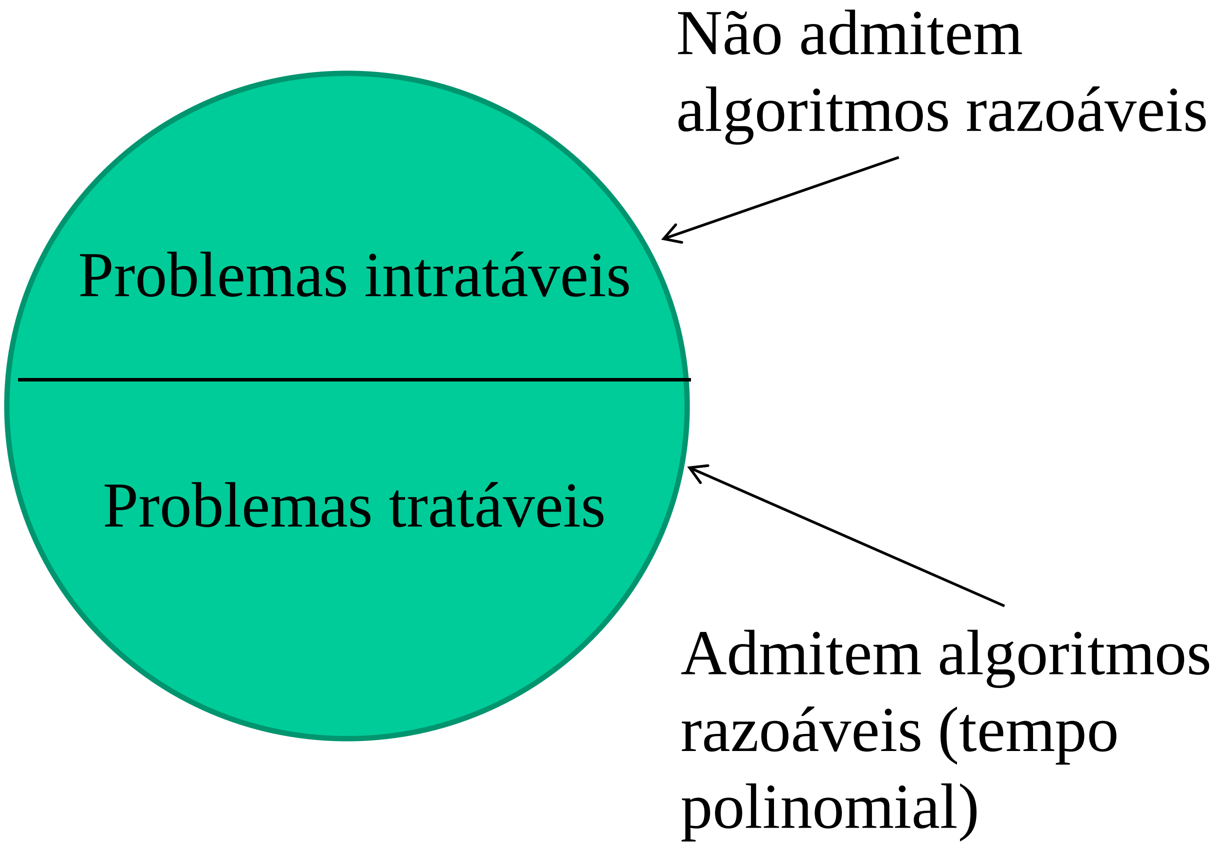
Problemas Tratáveis e Intratáveis

Considere a coleção de todos os algoritmos que resolvem um certo problema *p*.

• Se existir algum algoritmo de complexidade polinomial, então *p* é dito *tratável*; caso contrário, é dito *intratável*.

• A ideia é que um problema tratável sempre pode ser resolvido por um processo automático (computador, p.ex.) em tempo factível.

• Algoritmos não polinomiais podem levar séculos, mesmo para entradas de tamanho reduzido.


Não admitem
algoritmos razoáveis
Problemas intratáveis
Problemas tratáveis
Admitem algoritmos
razoáveis (tempo
polinomial)

Classes de Problemas P e NP

Verificando a Tratabilidade

• Um problema é tratável se for possível exibir um algoritmo de complexidade polinomial que o resolva.

Verificando a Intratabilidade

•Por outro lado, para verificar se um problema é intratável, há a necessidade de se provar que todo possível algoritmo que o resolva possui complexidade maior que a polinomial.

• Há uma classe de problemas para os quais todos os algoritmos conhecidos são de complexidade exponencial (pior caso). Envolvem, em geral, tarefas de *scheduling* e *matching*. Em geral, atingem o pior caso para entradas razoavelmente pequenas.

• Por outro lado, não se conhecem provas, até o momento, de que seja impossível encontrar um algoritmo polinomial para esses problemas.

Classes de Problemas P e NP

Exemplos de problemas para os quais NÃO se conhece solução polinomial:

•<u>Problema do Caixeiro Viajante (*matching*) </u>: qual é o caminho (ciclo) simples mais barato/curto entre 2 cidades? (Grafo correspondente possui um ciclo hamiltoniano – contém todos os vértices do grafo, sem repetição - de peso mínimo?);

•<u>Problema da Coloração de Grafos (*scheduling*)</u>: qual é o número mínimo de cores necessárias para colorir (as arestas de) um grafo, sem que 2 vértices adjacentes tenham a mesma cor? (Encontrar o <u>número cromático</u> de um grafo)

A Classe P - Easy to find

Define-se a classe P de problemas como sendo aquela que compreende precisamente aqueles problemas que admitem algoritmo polinomial.

(= *classe de problemas que podem ser resolvidos por uma Máquina de Turing determinística em tempo polinomial ao tamanho da entrada – pior caso*)

Problemas para os quais não se conhecem algoritmos polinomiais estão fora de P; podem estar em NP.

A Classe NP - Easy to check

Dizemos que um problema *p* está em NP se:

1. Não se conhece nenhuma solução polinomial para *p*;

2. E se for possível decidir, em tempo polinomial, se uma candidata à solução de *p* é de fato solução; se tem a propriedade da solução.

(em outras palavras, se o problema de decisão subjacente é polinomial, i.e. se existe uma MT que decide - algoritmo, sempre termina – se uma cadeia pertence ou não à linguagem das soluções de *p*)

Ou seja, não se consegue achar a solução em tempo polinomial, mas se consegue verificar uma candidata à solução, em tempo polinomial.

A Classe NP - Easy to check

São polinomiais os problemas de decisão associados aos exemplos:

Caixeiro Viajante:

Problema de decisão: Dada uma sequência de vértices do grafo, ela é um ciclo hamiltoniano?

Coloração de Grafos:

Problema de decisão: dados G, e uma atribuição de cores para G, ela consiste no número cromático de G?

Mas, encontrar a tal sequência de vértices ou a atribuição de cores não é (ou tem sido) possível em tempo polinomial, apenas em tempo exponencial.
Dizemos então que os problemas acima estão em NP.

A Classe NP- Easy to check

Observa-se que:

1. Não se exige uma solução polinomial para os problemas de NP; somente que uma certificação possa ser verificada em tempo polinomial;
2. Todo problema que está em P também está em NP, pois, se é possível achar uma solução em tempo polinomial, então é possível verificá-la (por meio do algoritmo de geração) em tempo polinomial também. Logo, $P \subseteq NP$.
3. Assim, os problemas que estão em NP e não estão em P são aqueles cuja <u>certificação</u> é polinomial, mas para os quais não se conhece <u>solução</u> polinomial.

A Classe NP- Easy to check

4. Considere uma MT Não-determinística. Uma MTND funcionando em tempo polinomial tem a habilidade de pressupor um número exponencial de soluções possíveis para um problema e verificar cada uma, em tempo polinomial, "em paralelo". Essa MT pode, então, resolver os problemas de NP.
5. NP NÃO significa "Não-Polinomial"; significa: Classe de problemas que podem ser resolvidos por uma MT NÃO-Determinística, em tempo POLINOMIAL.

A Classe NP – Easy to check

Exemplo 1: Considere o problema do CAIXEIRO VIAJANTE (CV): a pergunta "um grafo G possui um CICLO HAMILTONIANO?", necessária para resolver o problema, já torna o CV um problema de NP.

Problema de Decisão subjacente: dad uma sequência de vértices de G, ela possui a propriedade de ser um ciclo hamiltoniano? (ou seja, ser um ciclo simples, contendo todos os vértices de G)

Para encontrar (*find*) um ciclo hamiltoniano no grafo *G* (n vértices):

seria necessário exibir todos os ciclos de *G* e verificar se há um hamiltoniano entre eles. Para tanto, é preciso, para cada sequência possível de vértices:

(i) Verificar se é um ciclo;

(ii) Verificar se cada um desses ciclos é simples (sem vértices repetidos);

(iii) Verificar se todo vértice de *G* está presente na sequência exibida.

As operações (i), (ii) e (iii) são $O(n)$ e devem ser realizadas para cada ciclo exibido;

No entanto, teríamos que enumerar todas as sequências de vértices do grafo, ou seja, *O(n!)*, portanto, um algoritmo de natureza exponencial (até o momento, não se conhece outro processo que corresponda a um algoritmo polinomial).

$n! > 2^{cn}$ para qualquer constante c

$n! > n^k$

$n^n > n!$

•**Algoritmo de Certificação (*check*):** exibe-se uma sequência de vértices *C* de *G* e atesta-se que ela é hamiltoniana - o que consiste em verificar:

(i) se *C* é um ciclo, isto é, se vértices consecutivos na sequência são adjacentes no grafo e, além disso, o primeiro e o último coincidem; *O(n)*

(ii) se *C* é um ciclo simples, ou seja, se não há vértice repetido em *C*, a menos do primeiro e último; *O(n)*

(iii) se todo vértice de *G* participa de *C*. *O(n)*

O algoritmo correspondente a esse processo é simples e pode ser implementado em tempo linear *O(n)*.

Logo, Caixeiro Viajante está em NP.

(
No entanto, se, ao invés de um ciclo Hamiltoniano, quiséssemos um ciclo Euleriano – passar por cada aresta do grafo uma única vez – o algoritmo seria polinomial (na verdade, linear), já que bastaria verificar as seguintes condições para concluir que um grafo possui um ciclo Euleriano:

- O grafo é conexo (todo vértice é alcançável a partir de qualquer outro): há um algoritmo O(m);
- Todo vértice tem grau par (número par de arestas incidentes ao vértice): algoritmo trivial O(n).

)

A Classe NP – Easy to check

Exemplo 2: Considere o problema da Coloração de um Grafo: qual é o número cromático de G?

Problema de Decisão subjacente: dados G e um inteiro positivo k, existe uma coloração de G usando k<n cores?

- Essa verificação é polinomial (confira!).

Porém, encontrar o número cromático:

Requer verificar, para cada atribuição possível de k cores a n vértices – k^n – se ela obedece o requisito. Logo, exponencial. Para grafos particulares – perfeitos, bipartidos, etc. – há algoritmos polinomiais.

Logo, esse problema está em NP.

A Classe NP: Definição

Define-se a Classe NP como sendo aquela que compreende todos os problemas para os quais existe um algoritmo de certificação que é polinomial em relação ao tamanho da entrada.

Repare que:

- Considerando que toda MT determinística é uma MTND com no máximo uma opção de movimento, então $P \subseteq NP$.
- Mas, $P \neq NP$?
- Em <u>caso afirmativo</u>, então existem problemas na classe NP para os quais nunca se achará solução polinomial.
- Em <u>caso negativo</u>, então todo problema de NP admite necessariamente uma solução polinomial, mas está em NP pois ainda não se conhece tal solução. E, nesse caso, a exigência de uma certificação polinomial seria condição suficiente para garantir a existência de um algoritmo polinomial para solucioná-lo.

- Até o momento, não se conhece a resposta a essa pergunta. Todas as evidências apontam na direção $P \neq NP$. O principal argumento para essa conjectura é que a classe NP incorpora um conjunto enorme de problemas para os quais inúmeros pesquisadores têm procurado algoritmos polinomiais, porém sem sucesso. Esses problemas pertenceriam, então, a NP - P, o que conduz à conjectura $P \neq NP$;

- Se não são polinomiais, então quão complexo pode ser um problema da classe NP? Ou seja, admitindo-se que $P \neq NP$, seria possível ao menos resolver em tempo exponencial todo problema da classe NP?

Resposta (pode ser provada): Sim. Se $p \in$ NP, então uma resposta pode ser obtida em tempo exponencial.

Não necessariamente toda entrada de $p \in NP$ requer tempo exponencial.

Se $p \in NP$, então $p = O(2^n)$, ou seja, seu pior caso é exponencial. No entanto, pode acontecer de $p = \Omega(n)$, ou seja, seu melhor caso ser linear ou polinomial de outra ordem.

A questão sobre esses problemas, não é tanto o quanto demora chegar a uma solução, mas sim, SE podemos ou não resolvê-los mesmo para entradas pequenas, ainda que em computadores poderosíssimos. Simplesmente não sabemos se são tratáveis ou intratáveis, já que seu limite inferior está na classe dos tratáveis, e o superior, na categoria dos intratáveis.

Há problemas que não estão em P e nem em NP?

Relembrando:

- Problemas de P correspondem a MTD polinomiais (fáceis de verificar e fáceis de achar);
- Problemas de NP correspondem a MTND polinomiais (fáceis de verificar).

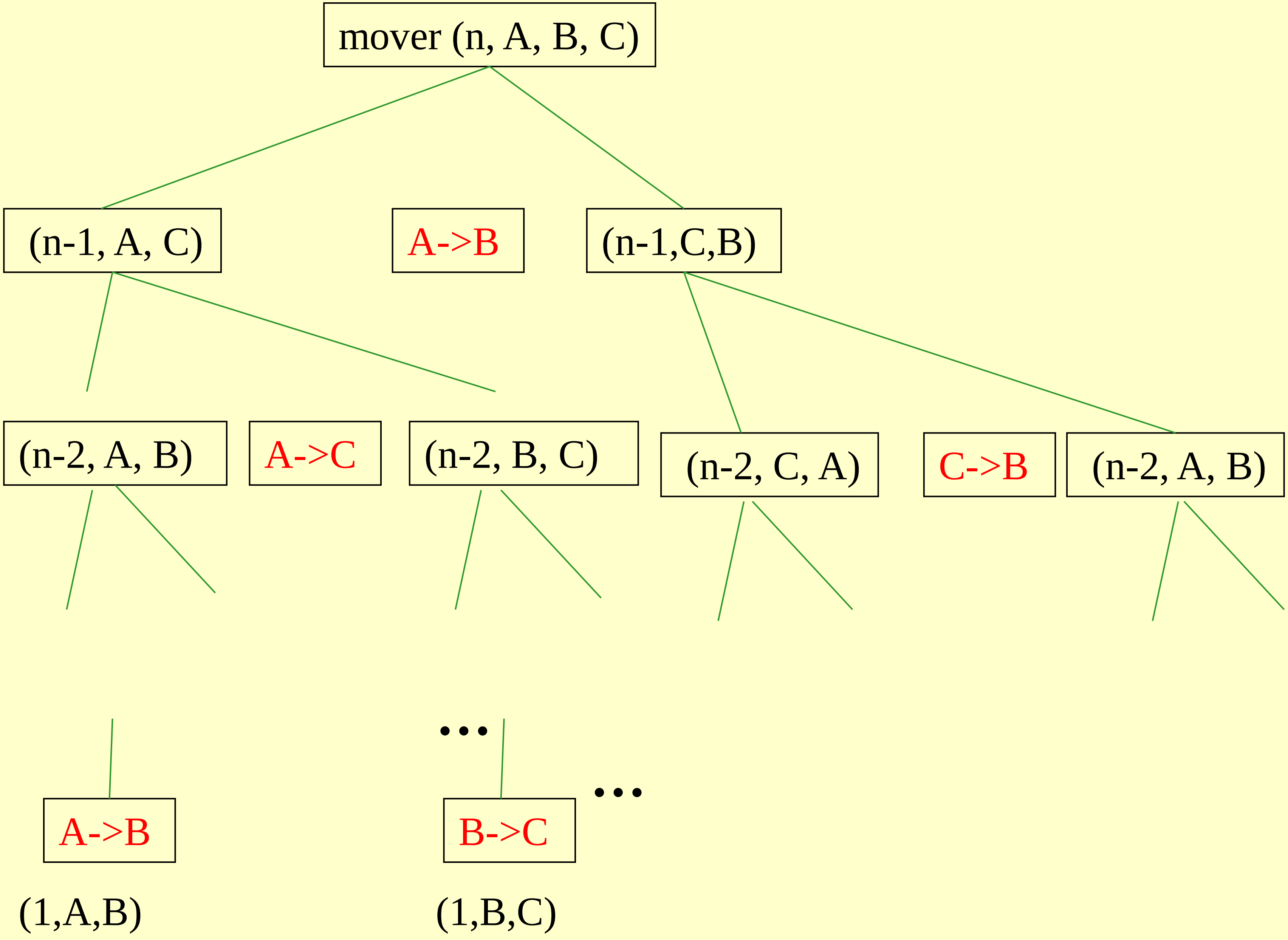
Há problemas que **comprovadamente** não são fáceis nem verificar, nem achar? Ou seja, correspondentes a MTND exponenciais?

- Sim, são os problemas exponenciais para os quais se comprovou que tanto achar quanto verificar só ocorrem em tempo exponencial.

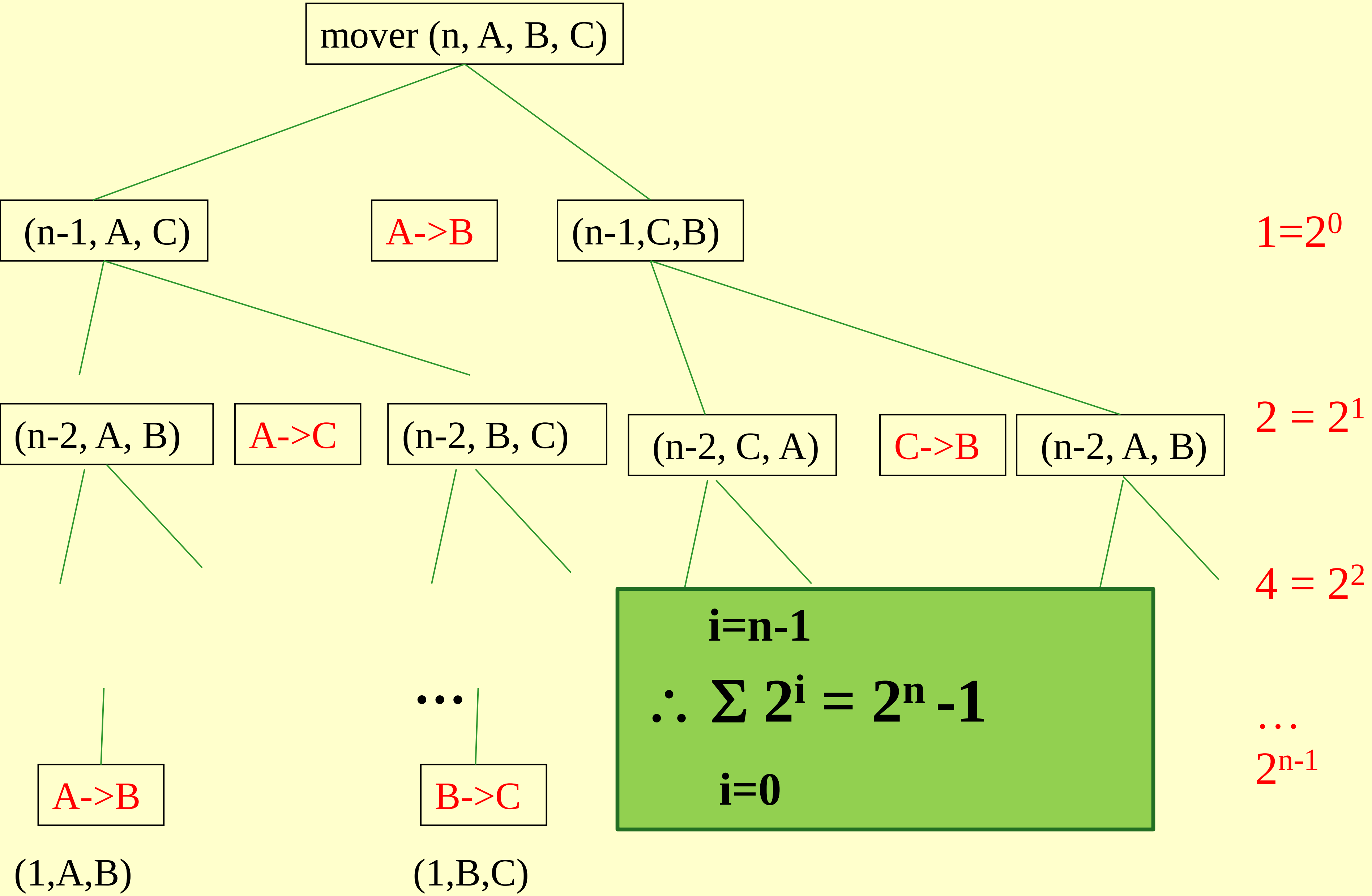
Torres de Hanoi

- **Objetivo**: transferir n discos, de diferentes tamanhos, um por vez, de uma torre A a outra torre B, sem que um disco maior fique em cima de um menor. Usa-se uma torre auxiliar C.
- Solução Recursiva:

```
mover (n, A, B, C) :-
    se n=1 transfere (A,B)
    senão {mover (n-1, A, C, B);
           transfere(A, B);
           mover (n-1, C, B, A)}
```


mover (n, A, B, C)
(n-1, A, C)
A->B
(n-1,C,B)
(n-2, A, B)
A->C
(n-2, B, C)
(n-2, C, A)
C->B
(n-2, A, B)
…
…
A->B
B->C
(1,A,B)
(1,B,C)

Quantos movimentos são necessários?



$$\therefore \sum_{i=0}^{n-1} 2^i = 2^n - 1$$

Problemas comprovadamente exponenciais

- São problemas cujas MT têm custo exponencial para verificar candidatos à solução. Podem ser MTD ou MTND. O custo de cada caminho (# transições do estado inicial ao final) é exponencial em relação ao tamanho da entrada.
  - Exs. Torres de Hanoi, Xadrez.

Complexidade de Algoritmos