- O que é o método da redutibilidade? Explique como é aplicado, dê um exemplo.
  R: A **redutibilidade** é um método de provar que certos problemas são computacionalmente **insolúveis** ou **indecidíveis**.
  A ideia é transformar um problema desconhecido (A) em outro problema conhecido (B). Se sabemos que B não pode ser resolvido por nenhum algoritmo, e conseguimos transformar A em B de maneira computável, então A também é insolúvel.

  Exemplo: Imagine que queremos viajar de Boston a Paris (problema A). Isso se reduz a comprar uma passagem aérea (problema B). Para comprar a passagem, precisamos de dinheiro (problema C). Para isso, precisamos de um emprego (problema D). Ou seja, se resolver o problema D (conseguir um emprego), conseguimos resolver todos os anteriores, por meio de **reduções sucessivas**.

- Qual a relação entre o método da redutibilidade e o problema de determinar se um problema é computacionalmente solúvel?
  R: A redutibilidade nos permite provar que um problema é indecidível.
  Se conseguimos reduzir um problema indecidível conhecido (como o problema da parada) a um novo problema, e essa redução for computável, então o novo problema também não pode ser decidido por uma máquina de Turing.
  Assim, usamos a redutibilidade como uma ferramenta para transferir a indecidibilidade de um problema para outro.

- O que é redutibilidade por mapeamento?
  R: É um tipo específico de redução, dizemos que um problema *A* é **redutível por mapeamento** a um problema *B* se existe uma **função computável *f*** tal que, para qualquer entrada *w*, tempos:

  $$w \in A \Leftrightarrow f(w) \in B$$

  Ou seja, transformamos instâncias do problema *A* em instâncias do problema *B*, e se conseguirmos resolver *B*, conseguimos resolver *A* também.

- Dê a descrição e a prova de 3 problemas computacionalmente insolúveis (indecidíveis)
  R:
  1 - Problema da parada:
  Dado uma máquina de Turing ***M*** e uma entrada *w*, determinar se ***M*** para ou entra em loop infinito ao processar *w*.
  HALT = { $\langle \boldsymbol{M}, w \rangle$ | ***M*** é uma MT que para com entrada *w* }

  Prova:
  Suponha que exista uma máquina ***H*** que resolve HALT. Podemos usá-la para construir outra máquina que leva a uma **contradição lógica**, o que prova que HALT é indecidível.

2 - Verificação de software:

Dado um programa ***P*** e uma especificação formal ***S***, determinar se ***P*** satisfaz ***S*** para toda entrada possível

Será que um algoritmo de ordenação **sempre** retorna uma lista ordenada para qualquer entrada válida? Esse tipo de verificação geral é indecidível.

Prova:
Reduz-se o problema da parada ao problema da verificação: se fosse possível verificar totalmente um programa, poderíamos verificar se ele para, o que contradiz a indecidibilidade do HALT.

3 - Problema da igualdade de linguagens:
$EQ\_TM = \{ \langle M_1, M_2 \rangle \mid M_1 \text{ e } M_2 \text{ são MTs e } L(M_1) = L(M_2) \}$

Esse problema pergunta se duas máquinas de Turing diferentes aceitam exatamente a mesma linguagem. Ou seja, dado dois autômatos, queremos saber se tudo o que uma aceita, a outra também aceita, e vice-versa.

Esse problema é indecidível porque, se fosse possível decidir EQ_TM, seria possível resolver problemas como o da parada, o que sabemos ser impossível. A prova normalmente envolve reduzir o problema da parada a esse problema, mostrando que se EQ_TM fosse decidível, o problema da parada também seria — o que é uma contradição. Portanto, EQ_TM é indecidível.