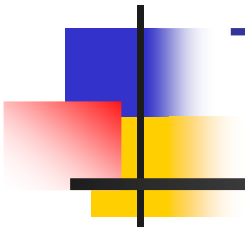
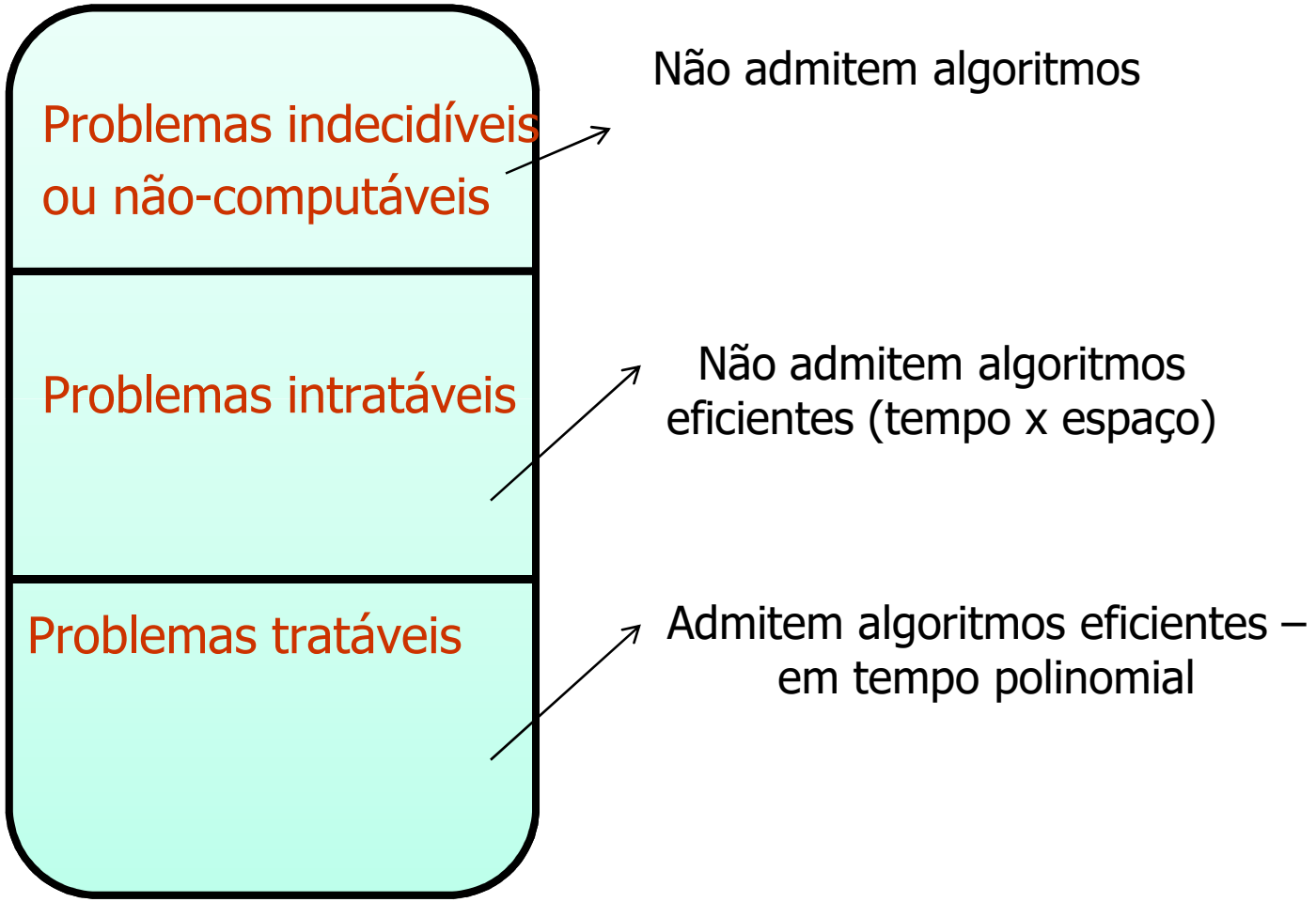


Teoria da Computação

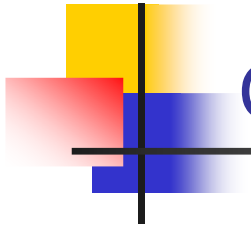
Complexidade computacional – classes de problemas

Universo de problemas



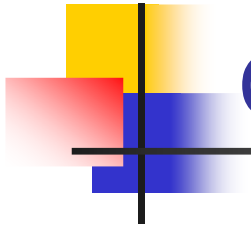
Classes de problemas tratáveis e intratáveis

- Se um algoritmo apresentar uma complexidade polinominal ele é dito tratável, caso contrário, é intratável
  - Um problema tratável pode ser solucionado por um computador em um tempo aceitável ➔ pode ser verificado a partir de um algoritmo de ordem polinomial
  - Algoritmos não polinomiais podem levar séculos, mesmo para entradas de tamanho reduzido.



Complexidade computacional

- A partir deste ponto, concentraremos a nossa atenção na questão de determinar se um **problema é tratável/intratável**.
  - Tratável: algoritmos com solução polinomial
  - Intratável: algoritmos com solução não-polinomial
- Precisaremos introduzir mais um pouco de teoria, primeiramente classificando os problemas conforme a resposta esperada. Os problemas podem ser classificados em:



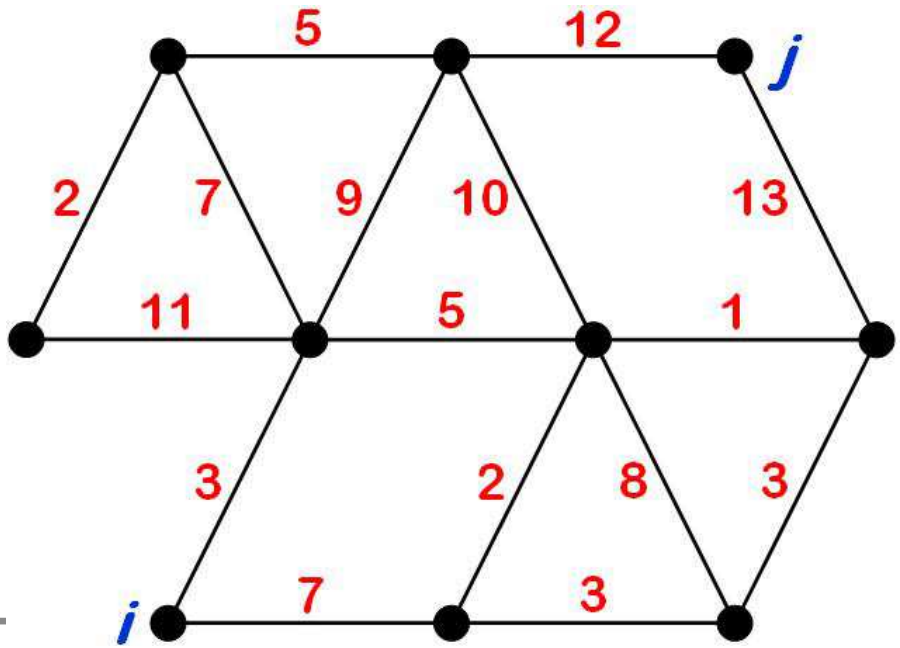
Classes de problemas algorítmicos

- Problemas de Decisão
  - consiste na verificação (decisão) da veracidade ou não de determinada questão para o problema (resposta SIM ou NÃO)
- Problemas de Localização
  - consiste na verificação da existência e identificação (localização) de uma solução para o problema
- Problemas de Otimização
  - consiste na verificação da existência e identificação da melhor (otimização) solução possível, dentre as soluções para o problema

Problemas de Decisão

- Objetivo: responder ***sim*** ou ***não*** à determinada indagação.

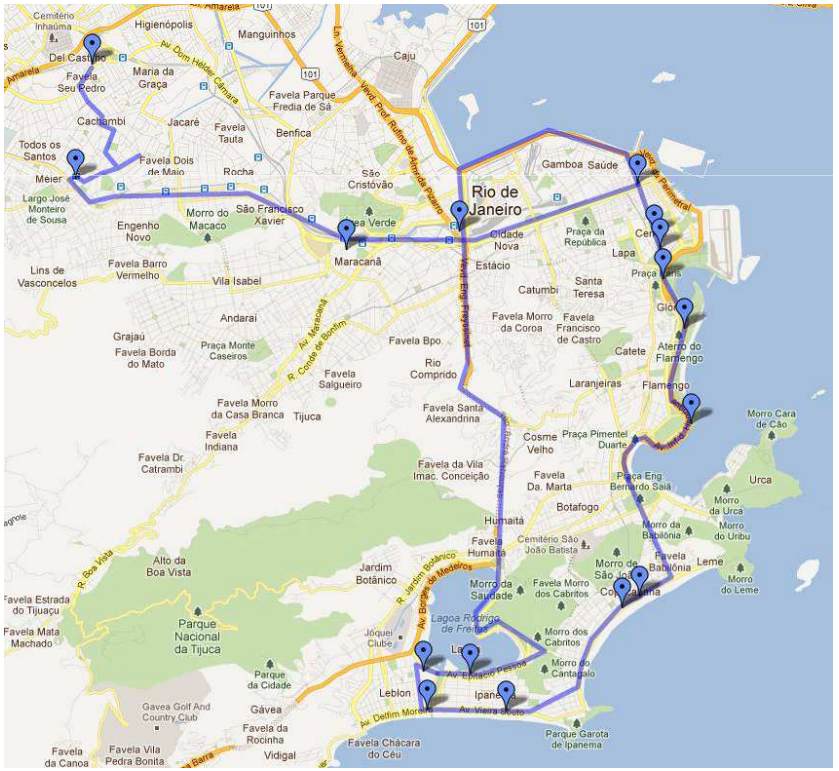
- Algoritmo de **Dijkstra**
  - Existe um caminho de **i** até **j** com peso <=***k***?



Problemas de Localização

- Objetivo: encontrar, caso exista, determinada estrutura satisfazendo requisitos especificados por uma questão.

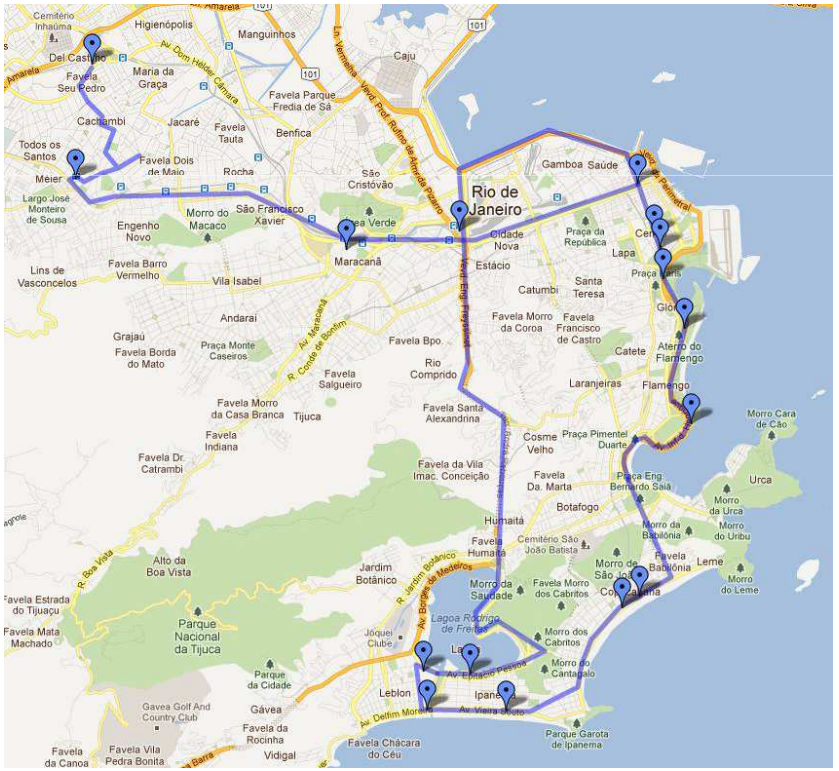
- Exemplo: Dado um conjunto P de pontos turísticos de uma determinada cidade, encontre um percurso turístico de um único dia que passa por todos os pontos em P com custo menor ou igual a k.
  - Resposta: Percurso turístico .



Problemas de Otimização

- Objetivo: obter determinada estrutura satisfazendo critério de otimização pré-definido

- Exemplo: Dado um conjunto P de pontos turísticos de uma determinada cidade, qual é o percurso que passa por todos os pontos em P na menor quantidade de tempo/custo possível.
  - Resposta: Percurso turístico (o mais rápido/melhor/menos custoso).



Classes de problemas algorítmicos

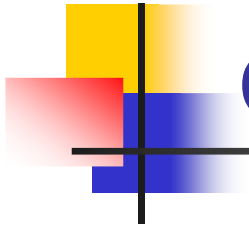
- Classes de problemas e grau de dificuldade

Problemas de decisão $\leq$ Problemas de localização $\leq$ Problemas de otimização

A obtenção de resposta para o *Problema de Decisão* em geral é mais fácil;

É fácil transformar um problema genérico em um *Problema de Decisão*;

Utiliza-se este problema para obter alguma indicação quanto à possível intratabilidade;



Classificação dos problemas

- A seguir, classificaremos os problemas segundo o tempo necessário para:
  1. encontrar uma solução
  2. verificar se uma resposta fornecida é realmente uma solução para o problema.

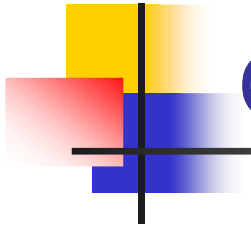
- As classes mais importantes nesses dois quesitos são denominadas **P** e **NP**

Classes de problemas P e NP

- **Classe P**

- Classe de problemas que compreende precisamente aqueles problemas que admitem **algoritmo polinomial**. (= classe de problemas que podem ser resolvidos por uma MT determinística em tempo polinomial ao tamanho da entrada)

  - Exemplos: algoritmos de ordenação, problema da Conectividade, o problema da equação do segundo grau (dados números inteiros a, b e c, encontrar um número inteiro x tal que $ax^2 + bx + c = 0$), entre outros.



Classes de problemas P e NP

- **Classe NP**

- A classe NP consiste nos problemas que são "verificáveis" em tempo polinomial.

  - É possível verificar, em tempo polinomial, se uma suposta solução de uma instância de X é de fato uma solução, ou seja, se existe um algoritmo que, ao receber uma instância I de X e uma suposta solução S de I, responde sim ou não conforme S seja ou não solução de I, e consome tempo limitado por um polinômio no tamanho de I para responder sim.

Classes de problemas P e NP

- **Classe NP**

- Para estes problemas são conhecidos algoritmos não-determinísticos polinomiais, ou seja, o algoritmo gera uma solução candidata ao problema e verifica sua viabilidade em tempo polinomial.

- Alguns exemplos:

  - problema do caminho hamiltoniano - é possível verificar em tempo polinomial se uma dada permutação dos vértices é um ciclo do grafo;
  - Problema da fatoração - é fácil verificar se um dado número natural p é divisor de n;
  - Problema do campo minado - é possível verificar em tempo polinomial se uma dada distribuição das minas é consistente com a configuração dada

Classes de problemas P e NP

- **Classe NP**

- Para que um problema p esteja na Classe NP é preciso que:

  - Seja possível verificar, em tempo polinomial, se uma candidata à solução de p seja de fato uma solução. Ou seja, a verificação/certificação/decisão da propriedade que faz dela uma solução para p tem que ser feita em tempo polinomial

    - resolver o problema de decisão subjacente já conhecido.

Classes de problemas P e NP

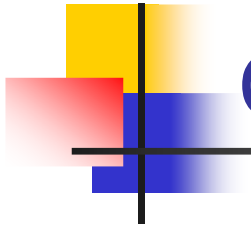
- **Classe NP – exemplos**
  - Caixeiro Viajante: Problema de decisão: Dada uma sequência de vértices do grafo, ela é um ciclo hamiltoniano?
  - Coloração de Grafos: Problema de decisão: dados G e um inteiro positivo k, existe uma coloração de G usando k cores?
- Se esses problemas de decisão forem polinomiais, então seus problemas originais, para os quais não se conhece solução polinomial, estão em NP.

Classes de problemas P e NP

- **Classe NP**
- Observa-se que:
  - Não se exige uma solução polinomial para os problemas de NP; somente que uma certificação possa ser verificada em tempo polinomial;
  - 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 em tempo polinomial também. Logo, P $\subseteq$ NP.
    - Para toda solução determinística pode ser construída uma não-determinística

Classes de problemas P e NP

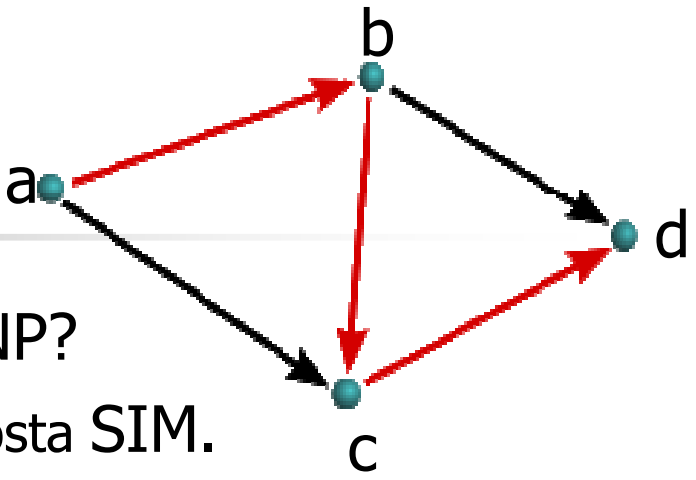
- **Classe NP**
- Observa-se que:
  - Assim, os problemas que estão em NP e não estão em P são aqueles cuja certificação é polinomial, mas para os quais não se conhece solução polinomial.
  - **NP NÃO significa "Não-Polinomial", significa:** Classe de problemas certificáveis por uma MT NÃO-Determinística, em tempo POLINOMIAL.



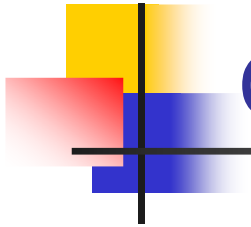
Classes de problemas NP

- Como mostrar que um determinado problema está em NP?
  - Elabore um algoritmo não-determinístico que execute em tempo polinomial para resolver o problema.
  - OU
  - Define-se uma justificativa J para a resposta SIM.
  - Elabora-se um algoritmo polinomial para reconhecer se J está correta. A entrada desse algoritmo consiste de J e da entrada.

Classes de problemas NP



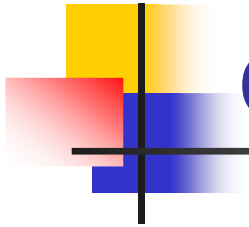
- Exemplo: ciclo Hamiltoniano está em NP?
  1. Define-se uma justificativa J para a resposta SIM.
     - R.: Uma sequência de vértices
  2. Para verificar a justificativa SIM
     - verificar se cada vértice aparece exatamente uma vez – contar quantos existem, O(n) e verificar se não há repetido O(n) – basta marcar enquanto verifica e observar se não marcou mais de uma vez.
     - verificar se existem arestas ligando os vértices consecutivos, O(n) – à medida que percorre a sequência observar o valor em uma matriz de adjacência
  - Como tudo pode ser feito em tempo polinomial, então está em NP



Classe de problemas P ou NP?

- Para demonstrar que um problema ***pertence à classe P*** basta mostrar um algoritmo polinomial que o resolva.

- Classe NP: devemos provar que não existe algoritmo (determinístico) polinomial para resolve-lo.



Classe NP-Completo

- A classe NP-completo tem a propriedade de que se um problema NP- completo puder ser resolvido em tempo polinomial todos os problemas em NP tem solução polinomial.

- Para definir formalmente a classe NP-completo **precisamos da noção de Redução Polinomial**.

Redução polinomial

Suposição 1

❖ Tem-se um problema de decisão A que se deseja resolver em tempo polinomial.

❖ Chama-se de instância a entrada para um determinado problema.

Suposição 2

❖ Existe um problema de decisão B, que já se sabe como resolver em tempo polinomial

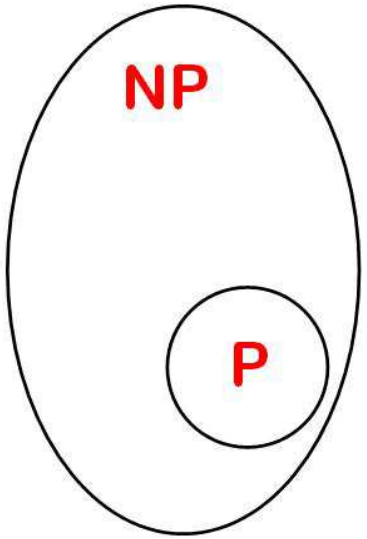
Suposição 3

❖ Tem-se um procedimento que transforma qualquer instância α de A em alguma instância β de B com as seguintes características:

(1) transformação demora tempo polinomial

(2) As respostas são as mesmas, isto é, a resposta para α é "sim" se e somente se a resposta para β também é sim.

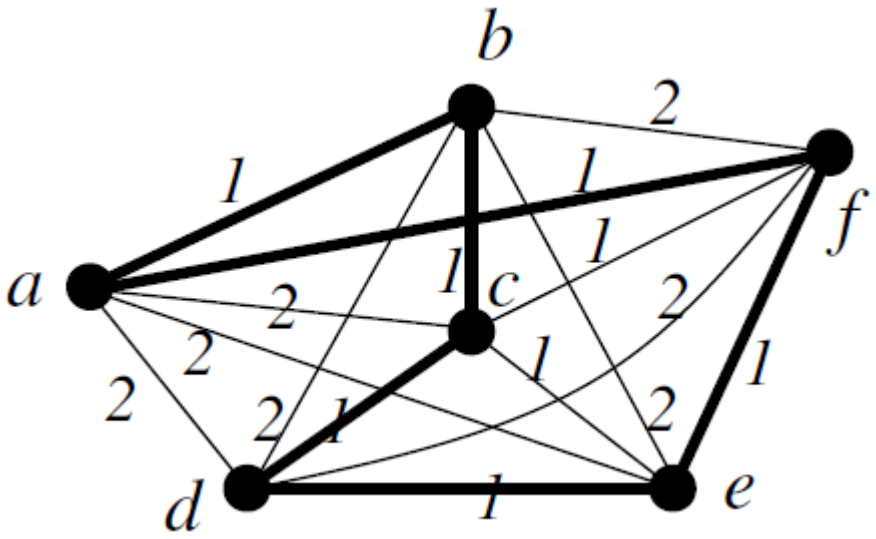
Classe NP-Completo



- NP-completo é um subconjunto de NP
  - Conjunto de todos os problemas de decisão os quais suas soluções podem ser verificadas em tempo polinomial;
  - Classe NP pode ser equivalentemente definida como o conjunto de problemas de decisão que podem ser solucionados em tempo polinomial em uma Máquina de Turing não determinística.

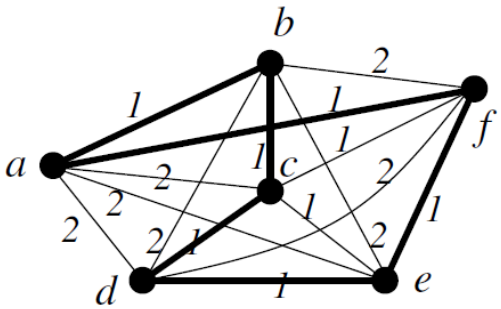
Exemplo: Problema do caixeiro viajante

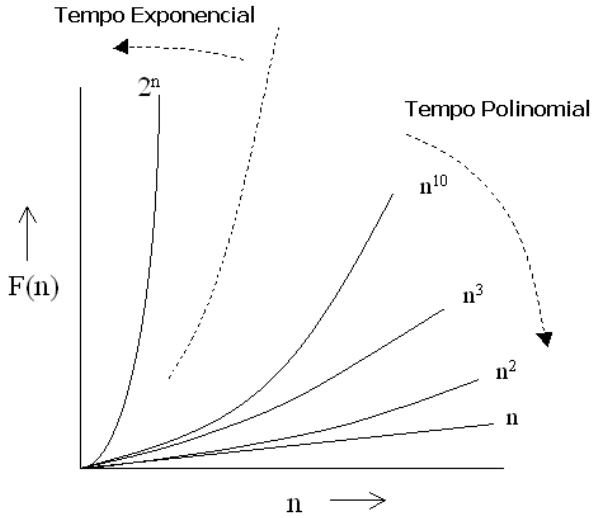
- Este problema de decisão consiste em se determinar se há um ciclo que passa por todos os vértices apenas uma vez com **peso total no máximo um valor k**.
  - É um caso especial do ciclo hamiltoniano



Exemplo: Problema do caixeiro viajante

- A resposta pode ser dada por uma sequência de vértices:
  - (1) verificar que cada vértice aparece na sequência apenas uma vez, O(n) – percorrer a lista marcando os vértices, observando se já não foram marcados, verificar se contém todos os vértices;
  - (2) O(1) – durante a marcação contar os vértices, e verificar se o peso total não ultrapassa o valor k;
  - (3) O(n) – percorrer a lista somando-se os pesos das arestas, descobrir cada peso é O(1) se os dados estão em uma matriz de adjacência.
  - Como o algoritmo é polinomial, então o **problema** é da **classe NP-Completo**.

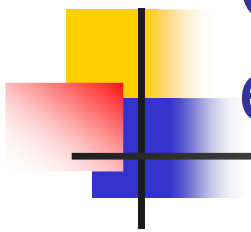




Problemas exponenciais

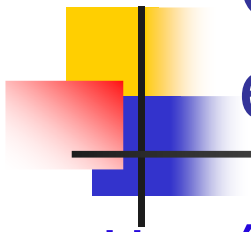
- É desejável resolver instâncias grandes de problemas de otimização em tempo razoável.
- Os melhores algoritmos para problemas NP-completo têm comportamento de pior caso exponencial no tamanho da entrada.
- Para um algoritmo que execute em tempo proporcional a 2^N, não é garantido obter resposta para todos os problemas de tamanho N >=100.
- Independente da velocidade do computador, ninguém poderia esperar por um algoritmo que leva 2^{100} passos para terminar sua tarefa.
- Um supercomputador poderia resolver um problema de tamanho N=50 em 1 hora, ou N=51 em 2 horas, ou N=59 em um ano.

O que fazer para resolver problemas exponenciais?



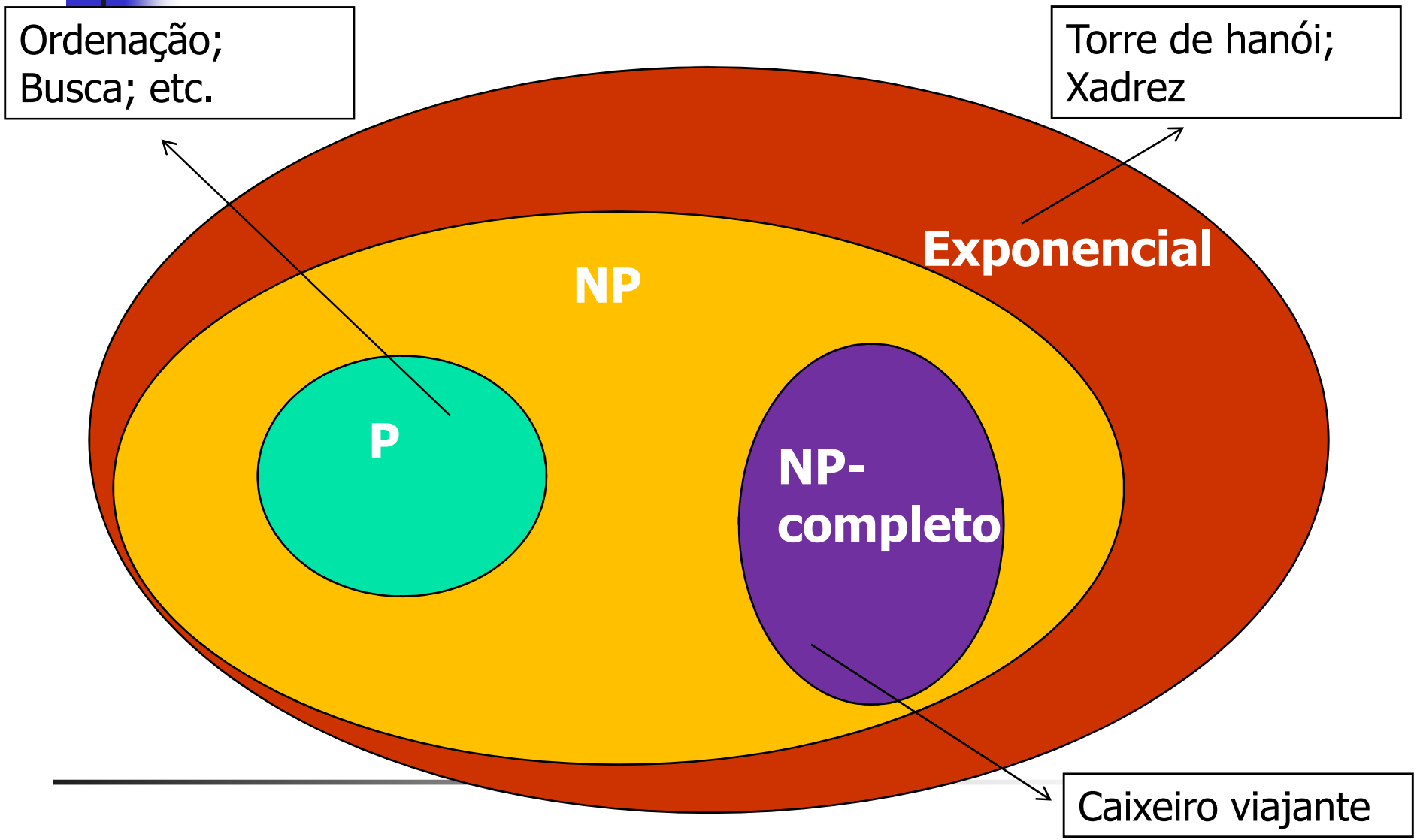
- Usar algoritmos exponenciais "eficientes" aplicando técnicas de tentativa e erro.
- Usar algoritmos aproximados. Acham uma resposta que pode não ser a solução ótima, mas é garantido ser próxima dela.
- Exemplo: backtracking (tentativa e erro), programação dinâmica, branch-and-bound (ramificação e poda), etc.

O que fazer para resolver problemas exponenciais?



- Heurísticas: são modificações no tratamento do problema, de forma a produzir algoritmos polinomiais para problemas exponenciais a custo de otimalidade.
  - Um algoritmo heurístico pode não produzir uma solução ótima, na verdade, uma solução heurística pode nem mesmo produzir uma solução. O que se faz é assumir certos riscos e fazer a computação ignorando-se o espaço de soluções que consideramos dispensável.
    - Exemplo: algoritmos gulosos

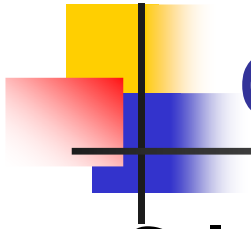
Complexidade de algoritmos



Complexidade computacional

- **Exemplo:**
- 400 estudantes universitários pleiteiam acomodação no alojamento, que acomoda apenas 100. Para complicar, o reitor forneceu uma lista com pares de estudantes que não podem figurar na lista dos escolhidos. Deseja-se uma lista-solução com 100 estudantes que podem ocupar o alojamento.

  - Este é um problema NP, uma vez que encontrar uma solução a partir da lista dos 400 candidatos e da lista de pares incompatíveis do reitor consiste num algoritmo exponencial. No entanto, o problema de decisão correspondente (dada uma lista de 100 estudantes, certificar se ela é uma solução possível) tem certificação polinomial.

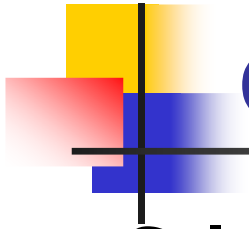


Complexidade computacional

- **Solução**

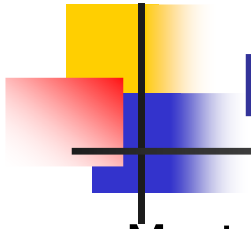
- Para certificar, basta verificar que nenhum par de estudantes da lista do reitor ocorre na lista candidata.

- Já para construir uma lista-solução, seria necessário certificar cada uma das diferentes combinações de 100 estudantes, a partir dos 400 estudantes. Nenhum supercomputador do futuro será capaz de solucionar esse problema por força bruta.



Complexidade computacional

- **Solução**
  - Este problema é modelado como um grafo (cada vértice corresponde a um estudante; cada aresta liga dois estudantes que não podem ficar juntos no alojamento), e sua solução corresponde em verificar, para 100^{400} possibilidades, se é possível "colorir" o grafo com 100 cores.



Exercícios

1. Mostre que o problema da coloração de vértices é NP.
2. Dado um grafo G e um inteiro k, mostre que o problema de se colorir os vértices do grafo de forma tal que nenhum vértice adjacente tenha a mesma cor, com um número de cores >= k é um problema que pertence à classe NP.
3. Um funcionário precisa visitar n cidades, sem repeti-las, em qualquer ordem. Como as passagens aéreas são fornecidas pela empresa, ele deseja maximizar o programa de milhagens, ou seja, escolher um caminho tal que a distância percorrida seja a máxima possível. Sabendo-se que o Problema do Caixeiro Viajante é NP-completo, mostre que o problema do funcionário também é NP-completo.
4. Mostre qual é a classe de complexidade do Problema da Mochila.