

Projeto e Análise de Algoritmos

# Aula 15:

## Classes de Complexidade P, NP (GPV 8.1,8.2)

DECOM/UFOP

2012/2 – 5<sup>o</sup>. Período

Anderson Almeida Ferreira

Material elaborado por:

Andréa Iabrudi Tavares

BCC241/2012-2



# Objetivos

- Saber definir o que é um problema de busca e um problema de decisão
- Saber definir classes de complexidade P e NP
- Entender a pergunta  $P \neq NP$
- Bibliografia
  - GPV 8.1,8.2

# Melhor Rota: Meu primeiro emprego

- Você foi contratado por uma empresa de distribuição de produtos de limpeza. Você será responsável pelo sistema que gera as rotas para os 10 estabelecimentos atendidos.
- Seu chefe espera que você gere uma rota que passe uma única vez por cada um dos estabelecimentos, com a menor distância possível.

# Melhor Rota: Modelagem

- O modelo de seu problema é um grafo  $G=(V,E)$ :
  - cada cidade é um nó  $v$ ,  $n = |V|$
  - cada aresta  $e = (i,j)$  tem um peso  $w(e)$ , a distância entre os estabelecimentos.
- **O problema de busca é:**
  - Encontre um ciclo  $C$  em  $G$  que passe por cada nó exatamente uma vez e tenha tamanho mínimo.

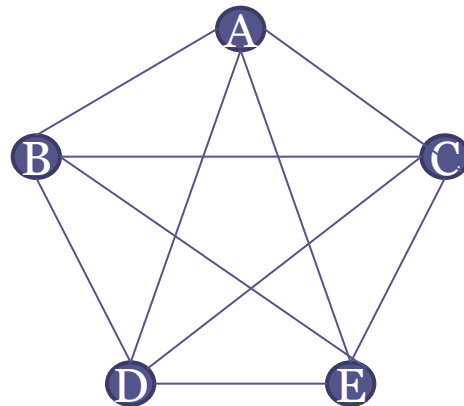
$$\min w(C) = \sum_{i=1}^{n-1} w(v_i, v_{i+1}) + w(v_n, v_1)$$

onde são válidos

$$C = (v_1, \dots, v_n) \mid \forall i, j \ i \neq j \Rightarrow v_i \neq v_j \ (v_n, v_1) \in E, \quad \forall 1 \leq i < n \ (v_i, v_{i+1}) \in E$$

# Exemplo com 5 estabelecimentos

	Aurora	Bacarão	Camarada	Dasdona	Enlevo
Aurora	0	5	2	1	3
Bacarão		0	2	3	4
Camarada			0	2	2
Dasdona				0	1
Enlevo					0



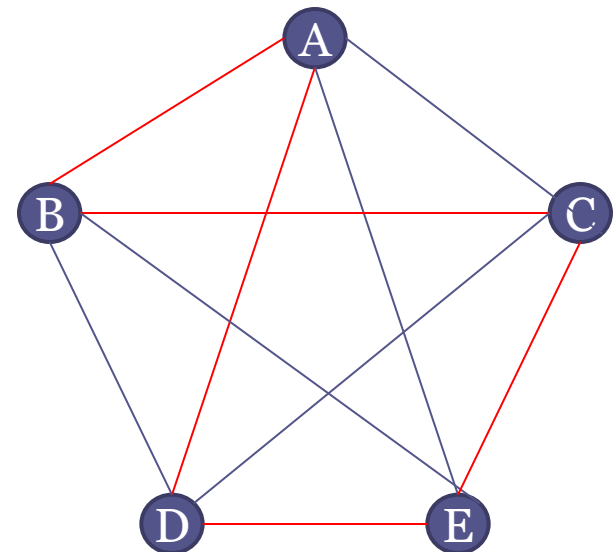
# Melhor Rota: Solução Gulosa

- Usando seus conhecimentos até o momento, você tenta uma abordagem gulosa para achar a solução para o problema.
- Como no algoritmo de Prim para menor árvore, seu algoritmo sempre escolhe um novo vértice para entrar no ciclo baseado no peso das arestas para o último vértice inserido.

# Exemplo com 5 estabelecimentos

	A	B	C	D	E
A	0	5	2	1	3
B	5	0	2	3	4
C	2	2	0	2	2
D	1	3	2	0	1
E	3	4	2	2	0

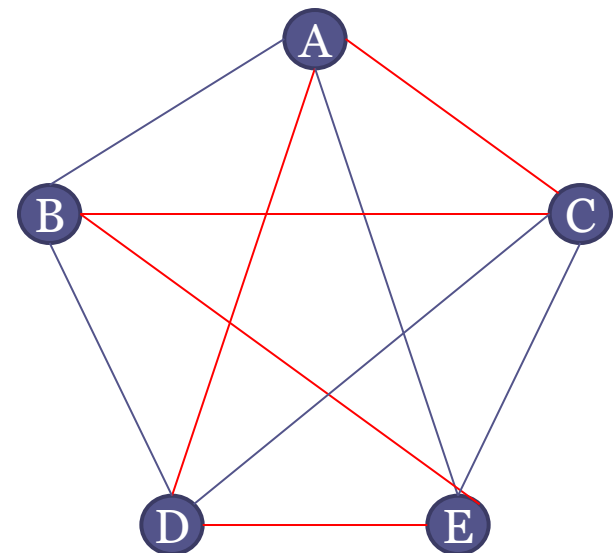
Tamanho = 11



# Infelizmente, há uma solução melhor

	A	B	C	D	E
A	0	5	2	1	3
B	5	0	2	3	4
C	2	2	0	2	2
D	1	3	2	0	1
E	3	4	2	2	0

Tamanho = 10





# Melhor Rota: Solução Ótima Exaustiva

- Após várias tentativas, você se entrega e decide inspecionar todos os ciclos possíveis.
- Como existem  $9!$  ciclos diferentes (todas as permutações de estabelecimentos), seu algoritmo é  $O(n!)$ . Mas, garantidamente, a menor rota será identificada.
- Você implementa seu algoritmo e encontra a rota num tempo bastante razoável. Seu chefe e você ficam felizes.

# Melhor Rota: A empresa se expande...

- **Boas notícias:** seu algoritmo gerou economia e, com os novos investimentos, o negócio está se expandindo e 20 novos estabelecimentos serão atendidos.
- **Péssimas notícias:** seu algoritmo, tão rápido para os 10 estabelecimentos, está executando há dois dias com o novo grafo de 30 estabelecimentos e até agora nada! Por quê?????

# Crescimento de funções de complexidade

Função de complexidade	Tamanho da Instância do Problema					
	10	20	30	40	50	60
$n$	0,00001 segundos	0,00002 segundos	0,00003 segundos	0,00004 segundos	0,00005 segundos	0,00006 segundos
$n^2$	0,0001 segundos	0,0004 segundos	0,0009 segundos	0,0016 segundos	0,0025 segundos	0,0036 segundos
$n^3$	0,001 segundos	0,008 segundos	0,027 segundos	0,064 segundos	0,125 segundos	0,216 segundos
$n^5$	0,1 segundos	3,2 segundos	24,3 segundos	1,7 minutos	5,2 minutos	13,0 minutos
$2^n$	0,001 segundos	1,0 segundos	17,9 segundos	12,7 dias	35,7 anos	366 séculos
$3^n$	0,059 segundos	58 minutos	6,5 anos	3855 séculos	$2 \times 10^8$ séculos	$1,3 \times 10^{13}$ séculos

# Será que a solução é comprar um computador melhor?

Maior instância que um computador resolve em 1 hora			
Função de complexidade	Computador Atual	Computador 100x mais rápido	Computador 1000x mais rápido
$n$	N	100 N	1000 N
$n^2$	M	10 M	31,6 M
$n^3$	Z	4,64 Z	10 Z
$n^5$	W	2,5 W	3,98 W
$2^n$	X	$X + 6,64$	$X + 9,97$
$3^n$	Y	$Y + 4,19$	$Y + 6,29$

# É possível resolver Melhor Rota?

- Você acha que não pode obter uma solução ótima em tempo razoável. Como você pode convencer seu chefe disso?



# Problemas de busca

- É uma relação  $R$  entre instâncias e soluções.

- **Verificação**

- Dadas uma instância  $x$  e uma solução  $y$  do problema, deve-se **verificar** se  $(x,y) \in R$ .

- Algoritmo  $C_R$  
$$C_R(x, y) = \begin{cases} \text{sim} & (x, y) \in R \\ \text{não} & (x, y) \notin R \end{cases}$$

- **Busca**

- Dada uma instância  $x$  do problema, deve-se **encontrar** uma solução  $y$  tal que  $(x,y) \in R$ .

- Algoritmo  $F_R$  
$$F_R(x) = \begin{cases} y & (x, y) \in R \\ \perp & \nexists y \mid (x, y) \in R \end{cases}$$

# Exemplos de Problemas de Busca

Satisfabilidade	Horn, 2-SAT
Caixeiro Viajante	Árvore Geradora Mínima
Maior caminho entre dois vértices	Menor caminho entre dois vértices
Casamento 3D	Casamento em grafo bipartido
Programação Linear Inteira	Programação Linear
Caminho de Rudrata (Hamiltoniano)	Caminho Euleriano

# Otimização x Decisão

- Problema de **otimização**

$$\min w(C) = \sum_{i=1}^{n-1} w(v_i, v_{i+1}) + w(v_n, v_1)$$

$$C = (v_1, \dots, v_n) \mid \forall i, j \ i \neq j \Rightarrow v_i \neq v_j \ (v_n, v_1) \in E, \quad \forall 1 \leq i < n \ (v_i, v_{i+1}) \in E$$

- Problema de **decisão** (sim/não)

$$\exists C \mid w(C) = \sum_{i=1}^{n-1} w(v_i, v_{i+1}) + w(v_n, v_1) \leq b?$$

$$C = (v_1, \dots, v_n) \mid \forall i, j \ i \neq j \Rightarrow v_i \neq v_j \ (v_n, v_1) \in E, \quad \forall 1 \leq i < n \ (v_i, v_{i+1}) \in E$$

- São equivalentes, então só problemas de decisão



# Classe NP

- Classe de problemas para os quais se conhece um algoritmo de tempo polinomial para  $C_R$ .
  - Por que classe?
    - Conjunto, nenhum específico
  - Por que algoritmo?
    - Equivalência polinomial dos modelos (tese de Church-Turing)
  - Por que tempo?
    - Análise assintótica
    - Pior caso
  - Por que polinomial?
    - Aceitável na prática
    - Fechada sob várias operações

# Definição de $C_R$ para ordenação

- Instância: um vetor  $A$  de  $n$  elementos inteiros
- Solução: uma permutação  $P$  dos índices (1 a  $n$ )
  - Representação através de um vetor  $P$  de  $n$  posições
- Algoritmo  $C_R$ 
  - Recebe  $A$  e  $P$  como parâmetros,
  - Verifica se é permutação (lembre-se do linear...)
  - Verifica se essa permutação resulta em uma sequência ordenada ( $A[P[i]] \leq A[P[i+1]]$ )

# Exemplos de Problemas de Busca

Satisfabilidade	Horn, 2-SAT
Caixeiro Viajante	Árvore Geradora Mínima
Maior caminho entre dois vértices	Menor caminho entre dois vértices
Casamento 3D	Casamento em grafo bipartido
Programação Linear Inteira	Programação Linear
Caminho de Rudrata (Hamiltoniano)	Caminho Euleriano

# Problema de Satisfabilidade

- $n$  variáveis lógicas (falso/verdadeiro)
- uma fórmula lógica com and, or e not
  - **Conjunção de disjunções**
- Existe uma atribuição para as variáveis que torna a fórmula verdadeira?

$$(x \vee y \vee z) (x \vee \bar{y}) (y \vee \bar{z}) (z \vee \bar{x}) (\bar{x} \vee \bar{y} \vee \bar{z})$$

# O problema de **decisão** da Melhor Rota pertence a NP?

- Existe um algoritmo polinomial que, dados o grafo, um limite e uma permutação dos nós de entrada, verifica se é uma solução?
- Quem não pertence a NP? Problemas de otimização!

# Classe P

- Classe de problemas para os quais se conhece um algoritmo de tempo polinomial para  $F_R$

# Exemplos de Problemas de Busca em P

## C e F Polinomiais

Horn, 2-SAT

Árvore Geradora Mínima

Menor caminho entre dois vértices

Casamento em grafo bipartido

Programação Linear

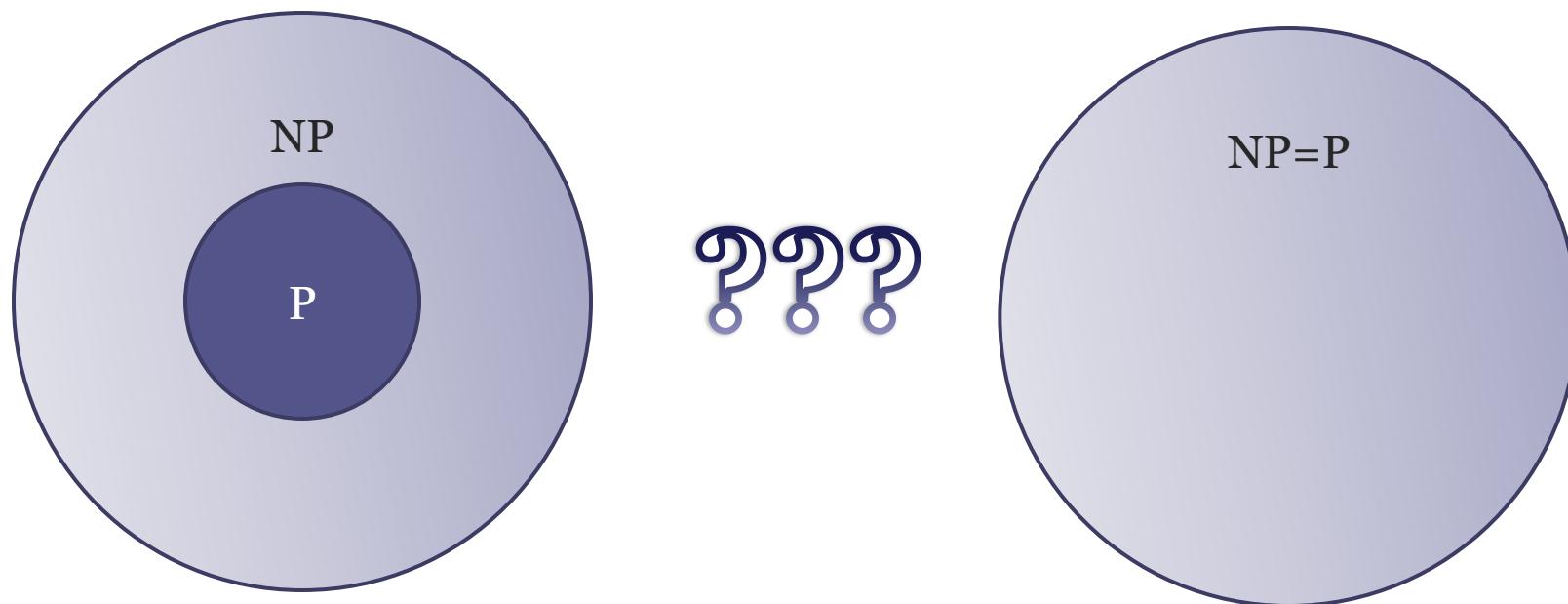
Caminho Euleriano

# Melhor rota pertence a P?

- Aparentemente não, pois você (que é quase um gênio) não conseguiu nem projetar um algoritmo nem reduzir seu problema a outro que você conhece que tenha solução polinomial...
- E daí???????



# Relação de P e NP



Achar a solução de qualquer problema  
é tão fácil quanto verificar se ela existe?

# Relação de P, NP e SAT

- Cook(1971) mostrou que o SAT é o problema mais difícil dentro dos NP, ou seja, todo problema em NP pode ser “transformado” no SAT.
- Então, se o SAT pode ser resolvido em tempo polinomial,  $P = NP$ ...
- Agora, vamos definir os problemas tão difíceis quanto o SAT, ou seja, transformar o SAT em outro problema.

# P = NP?

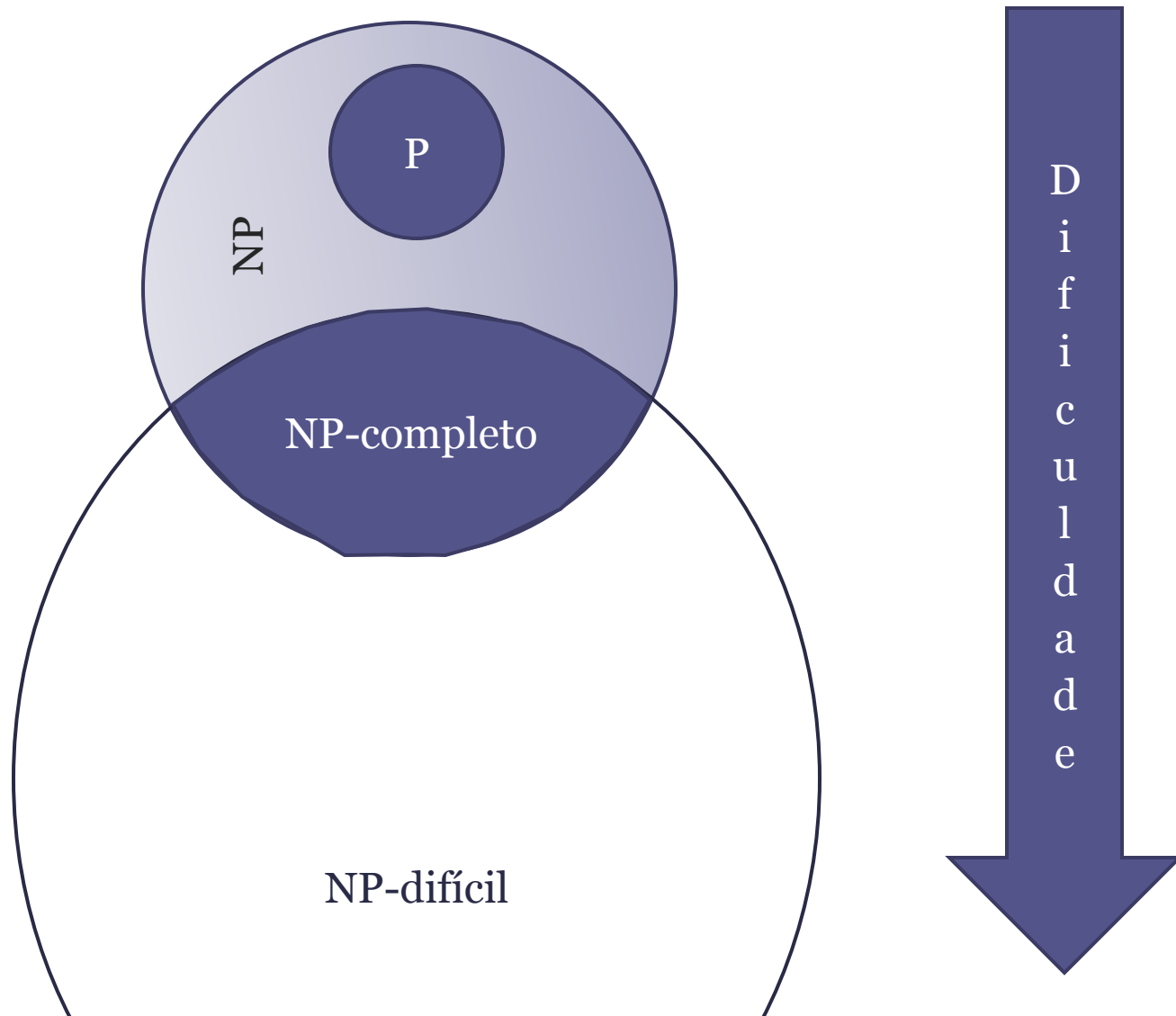
- Como formular esse problema?
  - Achar uma classe de problemas tais que todos os problemas de NP podem ser polinomialmente reduzidos a eles.

**NP-difícil: tão ou mais difíceis que qualquer um em NP**

- Achar uma classe de problemas **em NP** onde todos os problemas de NP podem ser polinomialmente reduzidos a eles.

**NP-completo: em NP, tão ou mais difíceis que qualquer outro em NP**

# P, NP, NP-difícil e NP-completo



# Problema de Satisfabilidade (SAT)

- $n$  variáveis lógicas (falso/verdadeiro)
- uma fórmula lógica com and, or e not
  - **Conjunção de disjunções**
- Existe uma atribuição para as variáveis que torna a fórmula verdadeira?

$$(x \vee y \vee z) (x \vee \bar{y}) (y \vee \bar{z}) (z \vee \bar{x}) (\bar{x} \vee \bar{y} \vee \bar{z})$$

# SAT como relação

- **Instância:** fórmula lógica como vetor de disjunções

$$(x \vee y \vee z) \quad (x \vee \bar{y}) \quad (y \vee \bar{z}) \quad (z \vee \bar{x}) \quad (\bar{x} \vee \bar{y} \vee \bar{z})$$

- **Solução potencial:** vetor atribuição de F/V para variáveis

$$(T, T, T)$$

- **Relação:** para cada fórmula, todas as atribuições que a tornam válida.

# SAT está em NP

- **Algoritmo** verificador polinomial

# Cláusulas de Horn

Conjunção de cláusulas de disjunção com no máximo uma variável não-negada

- Subconjunto de problemas de satisfabilidade

$$(w \wedge y \wedge z) \Rightarrow x, \quad (x \wedge z) \Rightarrow w, \quad x \Rightarrow y, \quad \Rightarrow x, \quad (x \wedge y) \Rightarrow w, \quad (\bar{w} \vee \bar{x} \vee \bar{y}), \quad (\bar{z}).$$



# Cláusulas de Horn: relação e NP

- Igual a SAT, já que é subconjunto

# Cláusulas de Horn está em P

- Apresentar algoritmo resolvidor polinomial no tamanho da fórmula

```
set all variables to false
```

```
while there is an implication that is not satisfied:
```

```
    set the right-hand variable of the implication to true
```

```
if all pure negative clauses are satisfied: return the assignment
```

```
else: return ``formula is not satisfiable``
```

# SAT x Horn ou $P = NP$ ?

- Parecem iguais
- Ambos em NP, Horn também em P
- Até hoje, ninguém mostrou que SAT está em P

# O problema de Melhor Rota pertence a NP?

- Existe um algoritmo polinomial que, dados o grafo, um limite e uma permutação dos nós de entrada, verifica se é uma solução?

# Otimização x Decisão é P x NP?

- Problema de **otimização**

$$\min w(C) = \sum_{i=1}^{n-1} w(v_i, v_{i+1}) + w(v_n, v_1)$$

$$C = (v_1, \dots, v_n) \mid \forall i, j \ i \neq j \Rightarrow v_i \neq v_j \ (v_n, v_1) \in E, \quad \forall 1 \leq i < n \ (v_i, v_{i+1}) \in E$$

- Problema de **decisão** (sim/não)

$$\exists C \mid w(C) = \sum_{i=1}^{n-1} w(v_i, v_{i+1}) + w(v_n, v_1) \leq b?$$

$$C = (v_1, \dots, v_n) \mid \forall i, j \ i \neq j \Rightarrow v_i \neq v_j \ (v_n, v_1) \in E, \quad \forall 1 \leq i < n \ (v_i, v_{i+1}) \in E$$

# Decisão a partir de otimização

$$\min w(C) = \sum_{i=1}^{n-1} w(v_i, v_{i+1}) + w(v_n, v_1) \quad \exists C \mid w(C) = \sum_{i=1}^{n-1} w(v_i, v_{i+1}) + w(v_n, v_1) \leq b?$$

# Otimização a partir de decisão

$$\min w(C) = \sum_{i=1}^{n-1} w(v_i, v_{i+1}) + w(v_n, v_1) \quad \exists C \mid w(C) = \sum_{i=1}^{n-1} w(v_i, v_{i+1}) + w(v_n, v_1) \leq b?$$

# Otimização x Decisão

- Problema de **otimização (NP-difícil)**
  - Verificador é tão difícil quanto resolvedor
- Problema de **decisão (NP-completo)**
  - Verificador parece ser mais fácil do que resolvedor
  - Fácil mostrar que está em NP (verificador polinomial)
- São **polinomialmente** equivalentes, então **só** problemas de decisão



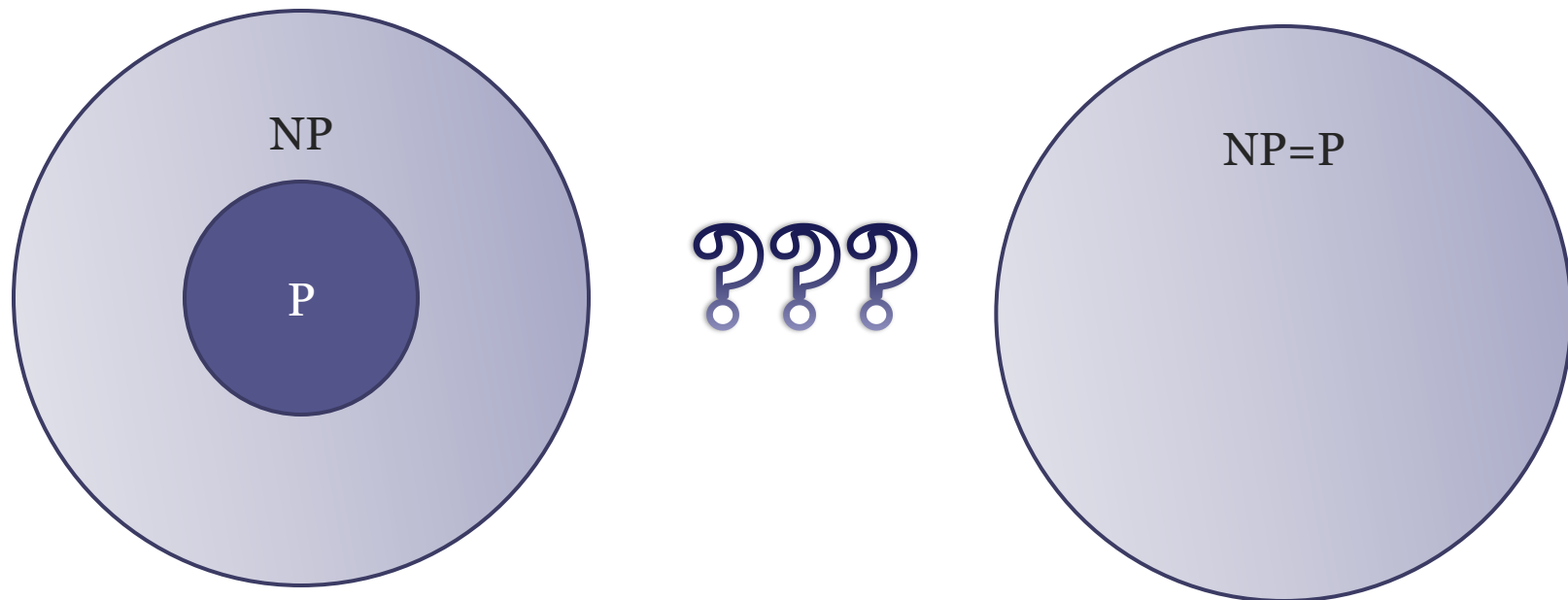
# O problema de **decisão** da Melhor Rota pertence a NP?

- Verificador polinomial com grafo, **limite** e permutação dos nós

# Melhor rota pertence a P?

- Aparentemente não, pois você (que é quase um gênio) não conseguiu nem projetar um algoritmo nem reduzir seu problema a outro que você conhece que tenha solução polinomial...
- E daí????????

# Relação de P e NP



Achar a solução de qualquer problema  
é tão fácil quanto verificar se ela existe?

# Relação de P, NP e SAT

- Cook(1971) mostrou que o SAT é o problema mais difícil dentro dos NP, ou seja, todo problema em NP pode ser “transformado” no SAT.
- Então, se o SAT pode ser resolvido em tempo polinomial,  $P = NP$ ...
- Agora, vamos definir os problemas tão difíceis quanto o SAT, ou seja, transformar o SAT em outro problema.

# P = NP?

- Como formular esse problema?
  - Achar uma classe de problemas tais que todos os problemas de NP podem ser polinomialmente reduzidos a eles.

**NP-difícil: tão ou mais difíceis que qualquer um em NP**

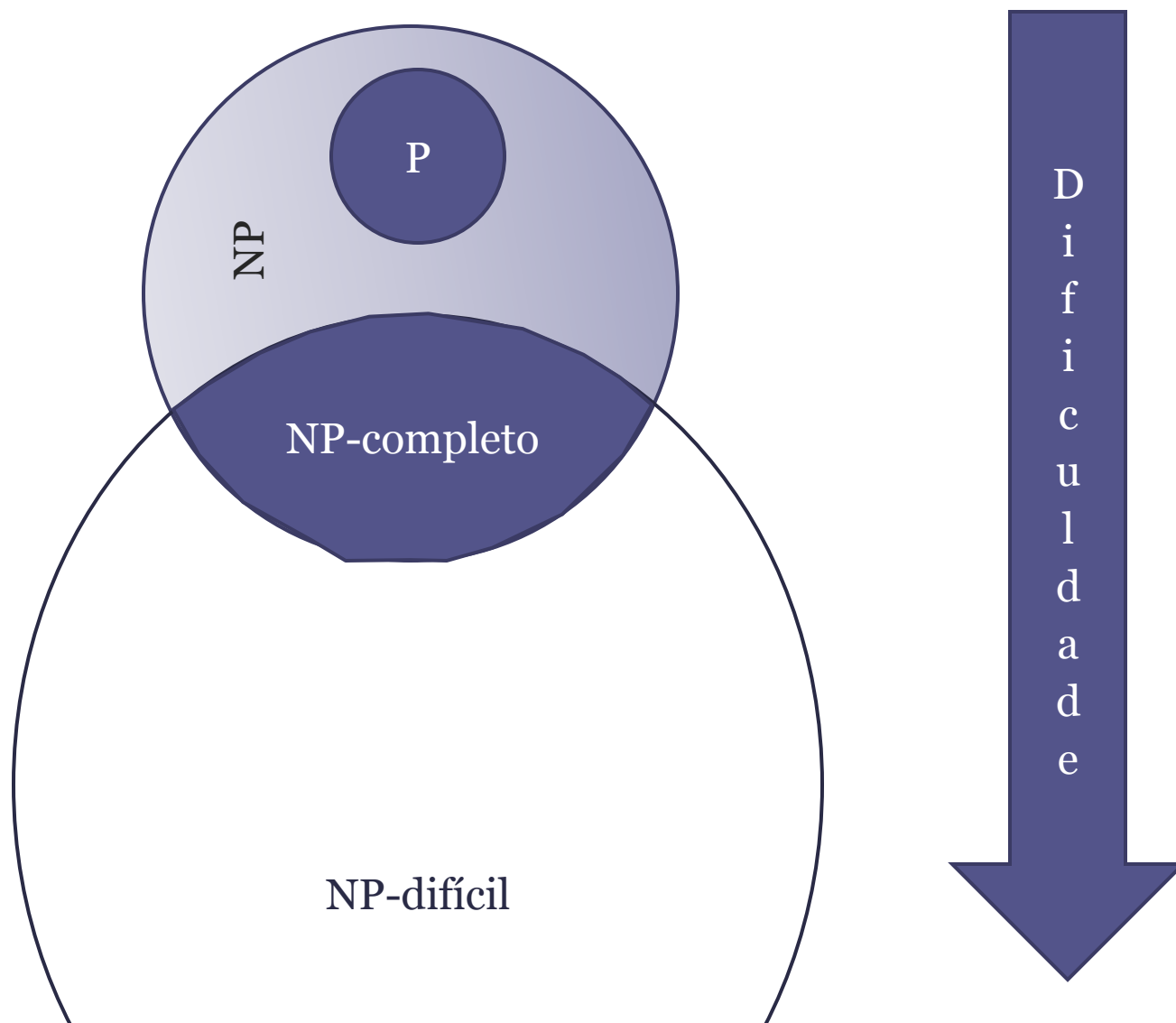
- Achar uma classe de problemas **em NP** onde todos os problemas de NP podem ser polinomialmente reduzidos a eles.

**NP-completo: em NP, tão ou mais difíceis que qualquer outro em NP**

# Classe NP-completo

- Os problemas **de busca na versão decisão MAIS difíceis**: todos se reduzem a eles
  - todos são o mesmo problema disfarçado...
- Subconjunto de NP e **nenhum NPC** possui solução **polinomial conhecida**
  - parece **intratável**.
- Seu problema está NPC, então
  - ou você não conseguirá um algoritmo polinomial
  - ou você ganhará o Prêmio Turing!

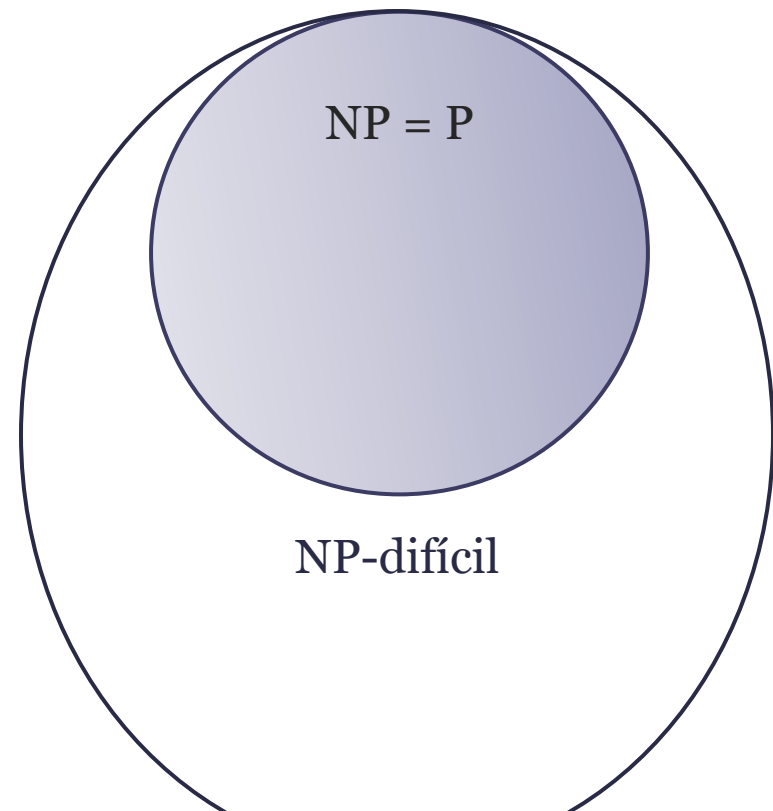
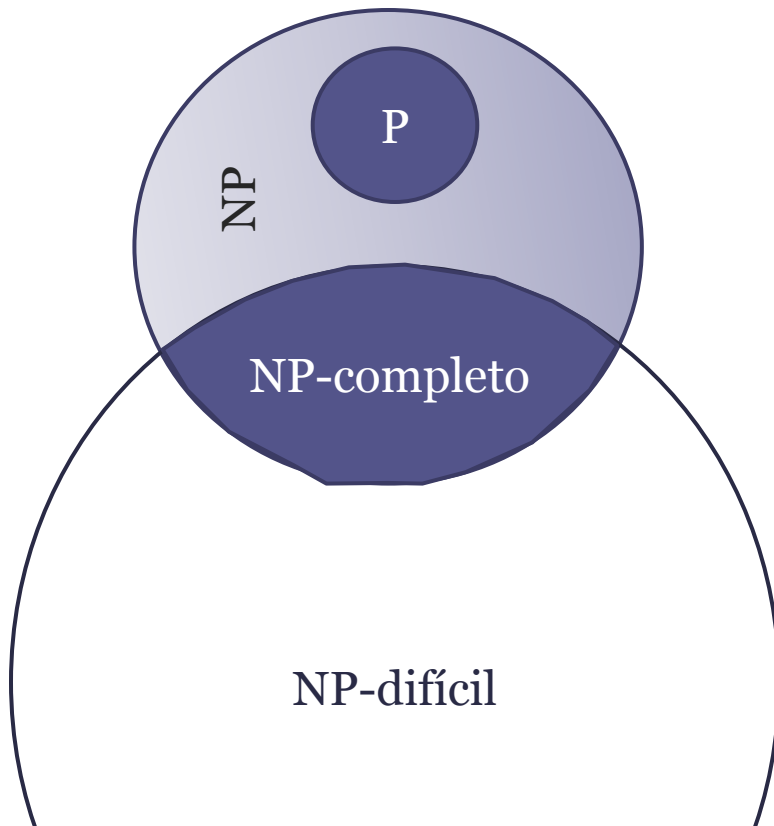
# P, NP, NP-difícil e NP-completo



# P = NP?

$D \in \text{NPC}$  tem solução polinomial  $\Leftrightarrow P = \text{NP}$

- Se  $P \neq \text{NP}$ , então todo  $D$  em NPC é intratável.
- Se  $D$  em NPC tem solução polinomial, então  $P = \text{NP}$ .





# Existe algum problema em NP-completo (intratável)?

- Cook mostrou que SAT é NP-completo, ou seja, que todos os problemas em NP podem ser reduzidos polinomialmente ao SAT. Logo, há fortes evidências de que SAT é intratável...

Isso foi feito utilizando a máquina de Turing não-determinística, então não vamos ver a prova...

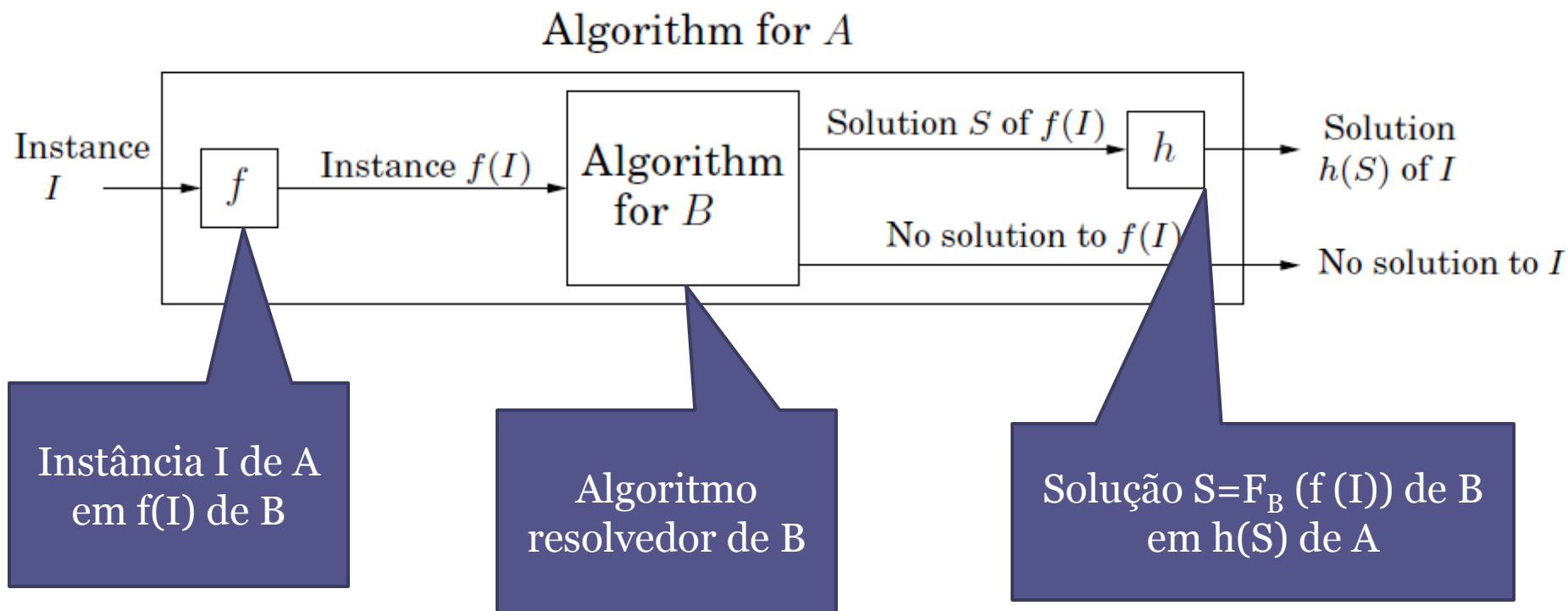
No livro-texto há uma outra versão, transformando todos NP para Circuito-SAT. Também não veremos.

# Redução polinomial de problemas

- $f$  e  $h$  são polinomiais
- Se  $A$  pode ser reduzido polinomialmente a  $B$ ,  $B$  é tão (ou mais) difícil quanto  $A$ .  $A \leq_p B$   
O que não pode acontecer é  $A$  ser exponencial e  $B$  polinomial, por exemplo... (Por quê?)

# Redução de Problemas

- Transforma um problema A em outro B



$$T(A) = T(f) + T(B) + T(h)$$

# Como mostrar problema A é P por redução?

1. Um algoritmo resolvidor polinomial para B
2. Uma redução polinomial (f e h) de A para B

Se  $B \in P$  e  $A \leq_p B$  então  $A \in P$

$$\Theta(A) = \Theta(B) + \Theta(f) + \Theta(g)$$

$$\Theta(A) = \max(\Theta(B) + \Theta(f) + \Theta(g))$$

# É possível resolver Melhor Rota?

- Não sabemos, mas se mostrarmos que ele pertence à classe dos problemas NP-completos há forte evidência de que não existe um algoritmo polinomial para resolvê-lo...
- Então, como mostrar que é NP-completo?

# Como mostrar problema é NP-Completo?

- O problema B é NPC e quero mostrar que A também é NPC.

Se  $B \in \text{NPC}$ ,  $A \in \text{NP}$  e  $B \leq_p A$  então  $A \in \text{NPC}$

- Para um problema de busca versão decisão:
  1. Um algoritmo polinomial para verificação
  2. Uma redução polinomial (f e h) de um problema NPC a ele

# Melhor Rota: Provando que é NPC

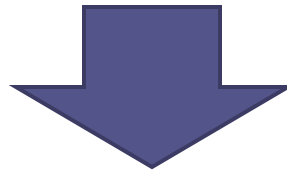
- Vamos provar que Melhor Rota é NPC:
  - Mostrando que ele é NP.
  - Reduzindo polinomialmente o problema do ciclo hamiltoniano, que é NPC, a ele.
- Melhor Rota é uma instância do conhecido problema do Caixeiro Viajante (*Travelling Salesman Problem – TSP*), que tem inspirado o desenvolvimento de inúmeras abordagens exatas e aproximadas.

# Melhor Rota - Otimização para Decisão

$$\min w(C) = \sum_{i=1}^{n-1} w(v_i, v_{i+1}) + w(v_n, v_1)$$

onde são válidos

$$C = (v_1, \dots, v_n) \mid \forall i, j \ i \neq j \Rightarrow v_i \neq v_j \ (v_n, v_1) \in E, \quad \forall 1 \leq i < n \ (v_i, v_{i+1}) \in E$$



$$\exists C \mid w(C) \leq b$$

onde são válidos

$$C = (v_1, \dots, v_n) \mid \forall i, j \ i \neq j \Rightarrow v_i \neq v_j \ (v_n, v_1) \in E, \quad \forall 1 \leq i < n \ (v_i, v_{i+1}) \in E$$



# Melhor Rota Decisão é NP

- Algoritmo checa se uma seqüência de vértices é solução para problema de decisão com limite  $b$ .
- O algoritmo é simples. Dada a seqüência de nós da rota proposta :  $s = (v_1, \dots, v_n)$

1. É uma permutação .  $v_i \neq v_j, \forall i \neq j$

2. Existe aresta entre nós consecutivos.

$$\forall i < n, e(v_i, v_{i+1}) \in E, e(v_n, v_1) \in E$$

3. A soma dos pesos das arestas tem que ser menor ou igual ao limite.

$$b \geq \sum_{i=1}^{n-1} w(e(v_i, v_{i+1})) + w(e(v_n, v_1))$$

# Melhor Rota Decisão é NP

# Encontrando um problema NPC para ser reduzido a Melhor Rota...

**Existem vários problemas NP-completos...**

[http://en.wikipedia.org/wiki/List\\_of\\_NP-complete\\_problems](http://en.wikipedia.org/wiki/List_of_NP-complete_problems)

## 1 Graph theory

- 1.1 Covering and partitioning
- 1.2 Subgraphs and supergraphs
- 1.3 Vertex ordering
- 1.4 Iso- and other morphisms
- 1.5 Miscellaneous

## 2 Network design

- 2.1 Spanning trees
- 2.2 Cuts and connectivity
- 2.3 Routing problems
- 2.4 Flow problems
- 2.5 Miscellaneous
- 2.6 Graph Drawing

## 3 Sets and partitions

## 4 Storage and retrieval

## 5 Sequencing and scheduling

## 6 Mathematical programming

## 7 Algebra and number theory

## 8 Games and puzzles

## 9 Logic

## 10 Automata and language theory

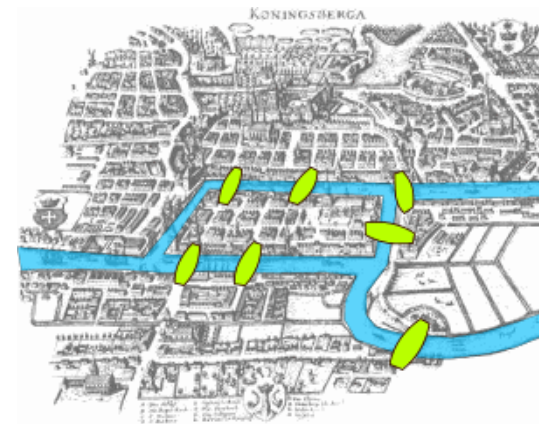
## 11 Computational geometry

## 12 Program optimization

# Caminho euleriano é P

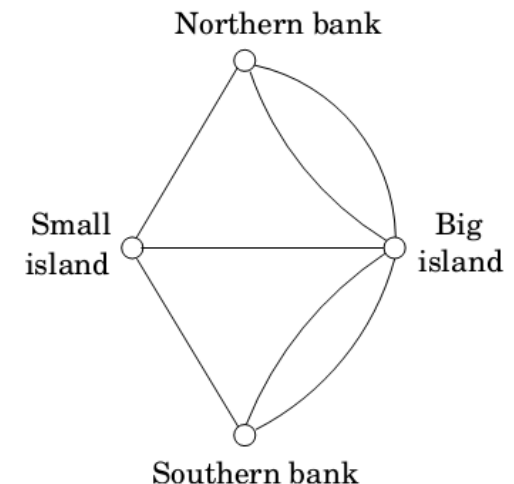
- Leonhard Euler (1735) – nascimento de teoria dos grafos

Possível percorrer todas as pontes sem repetir?



Existe caminho que passe por cada **aresta** exatamente uma vez?

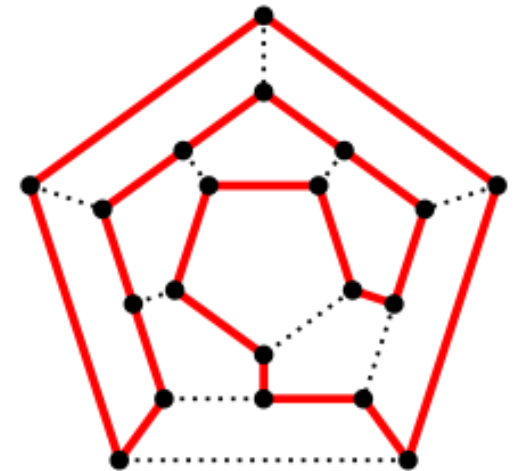
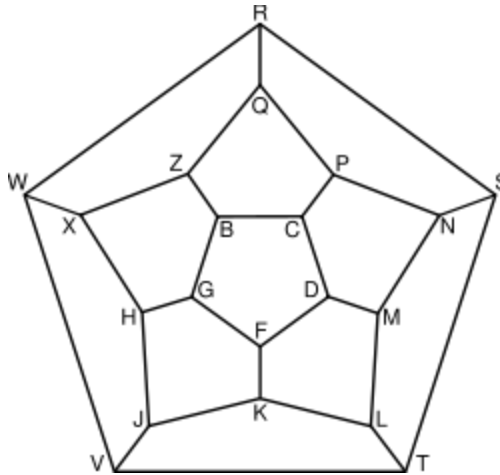
Basta testar cardinalidade dos vértices!



# Ciclo Hamiltoniano ou de Rudrata é NP-Completo

Redescoberto por William Hamilton (1857), físico, astrônomo e matemático.

Dado um grafo  $G=(V,E)$ , existe um ciclo que passa por cada **vértice** exatamente uma vez? (CH)



# Ciclo Hamiltoniano -> Melhor Rota

- A partir do grafo  $G=(V,E)$  do CH, montamos o grafo da Melhor Rota da seguinte forma:

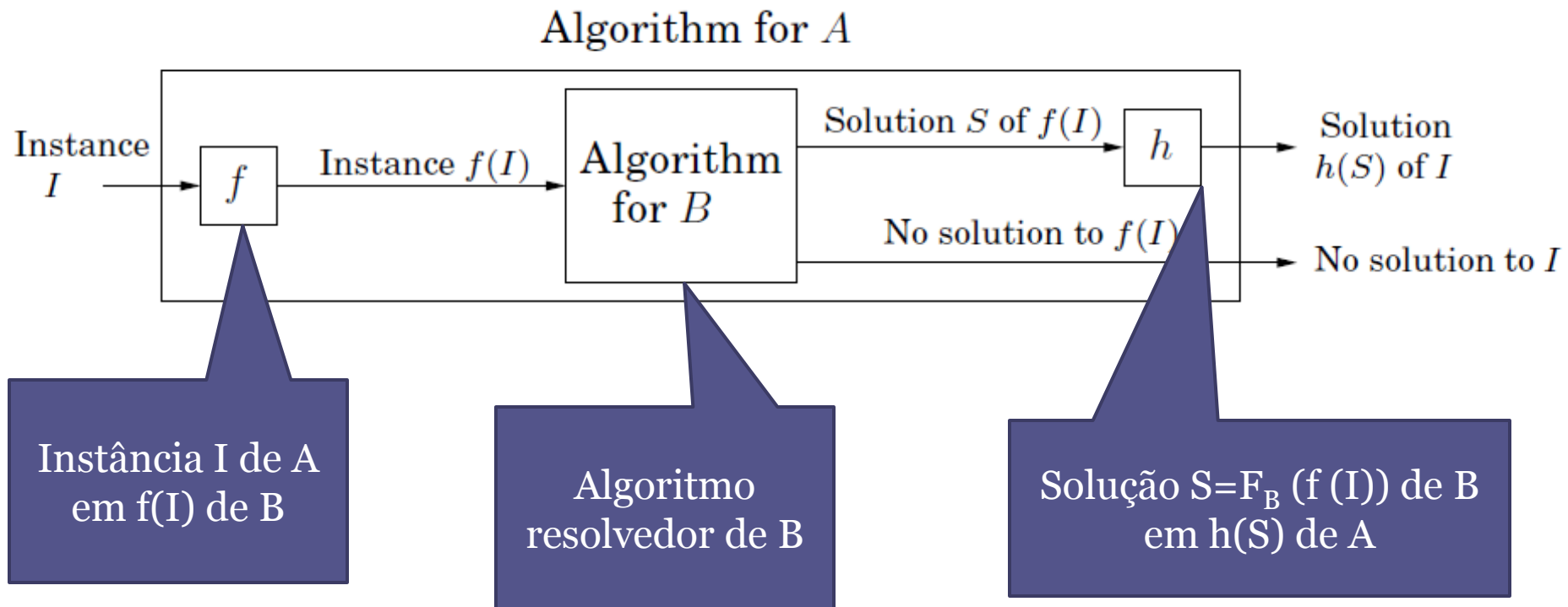
$G^{MR} = (V, E^{MR})$ , grafo completo com

$$w(v_i, v_j) = \begin{cases} 1, & (v_i, v_j) \in E \\ 2, & (v_i, v_j) \notin E \end{cases}$$

- Existe um CH em  $G$  se e somente se a melhor rota em  $G^{MR}$  tem tamanho  $n$ , pois nesse caso todas as arestas estavam originalmente em  $G$ .

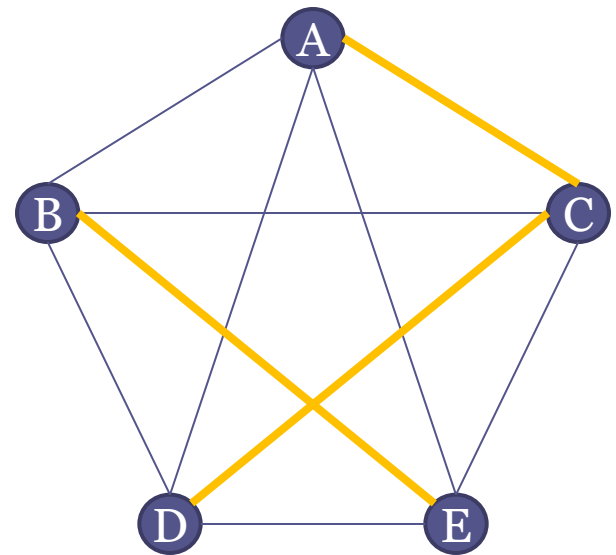
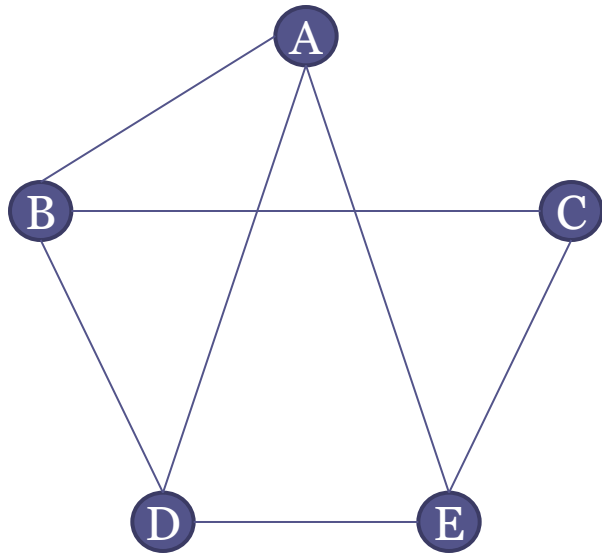
# CH -> Melhor Rota

- CH (A) se reduz polinomialmente a MR (B)



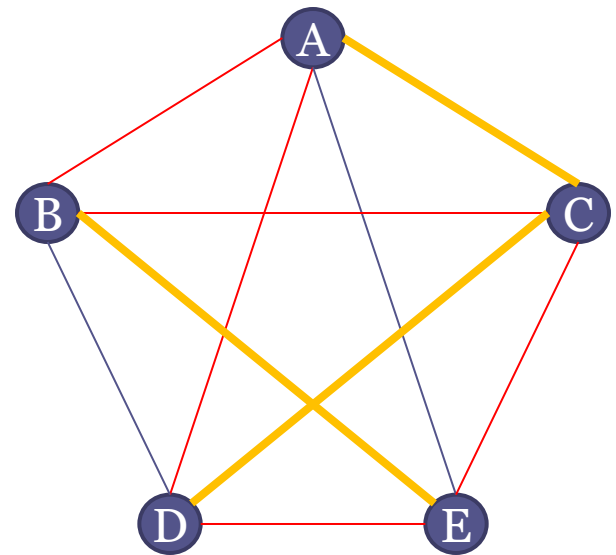
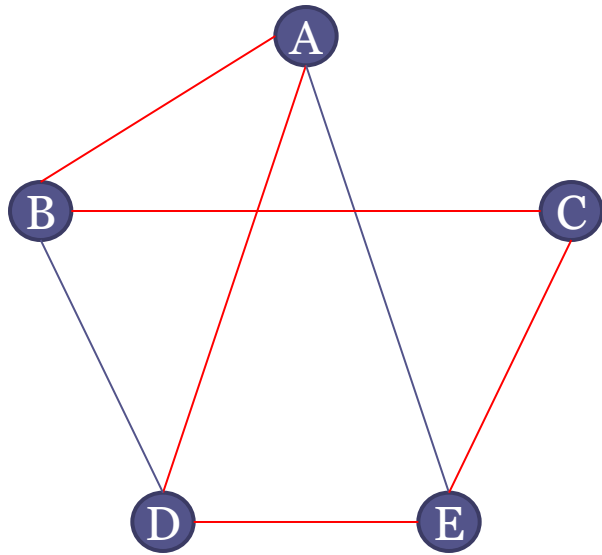
$$T(A) = T(f) + T(B) + T(h)$$

# Transformando CH em MR





# Solucionando MR e CH



# Melhor rota é intratável!

- Muito bem, você pode mostrar para seu chefe que é bastante improvável que algum algoritmo ofereça a solução ótima de roteamento.
- Contudo, como você agora sabe que esse é um conhecido problema TSP, é hora de procurar boas estratégias de exploração do espaço de solução ou soluções aproximadas...

# Resumo

- Problemas que pertencem a P têm solução polinomial.
- Problemas que pertencem a NP têm verificação polinomial.
- Problemas NP-Completos são os mais difíceis em NP.
- Há fortes evidências de que um NPC é intratável.
- Para provar que um problema é intratável:
  - Algoritmo verificador polinomial do problema na versão decisão.
  - Redução polinomial de um NPC conhecido ao novo problema.