ESTRUTURAS DE DADOS II

MSC. DANIELE CARVALHO OLIVEIRA

DOUTORANDA EM CIÊNCIA DA COMPUTAÇÃO - USP

MESTRE EM CIÊNCIA DA COMPUTAÇÃO – UFU

BACHAREL EM CIÊNCIA DA COMPUTAÇÃO - UFJF

2

PROBLEMAS NP-COMPLETOS

3 INTRODUÇÃO

- Problemas intratáveis ou difíceis são comuns na natureza e nas áreas do conhecimento.

- Problemas “fáceis”: resolvidos por algoritmos polinomiais.
- Problemas “difíceis”: somente possuem algoritmos exponenciais para resolvê-los.

4 PROBLEMAS DIFÍCEIS, PROBLEMAS FÁCEIS

- À direita temos problemas que são resolvidos eficientemente.
- À esquerda, temos problemas que escaparam da solução eficiente durante décadas ou séculos.

Problemas difíceis	Problemas fáceis
3SAT	2SAT, HORN SAT
CAIXEIRO VIAJANTE	ÁRVORE GERADORA MÍNIMA
LONGEST PATH	SHORTEST PATH
3D MATCHING	BIPARTITE MATCHING
MOCHILA	MOCHILA fracionária
INDEPENDENT SET	INDEPENDENT SET em árvores
ZERO-ONE EQUATIONS	ZOE fracionário
RUDRATA PATH	EULER PATH
MAXIMUM CUT	MINIMUM CUT

5 PROBLEMAS DIFÍCEIS, PROBLEMAS FÁCEIS

- Os problemas à direita são resolvidos por algoritmos especializados e diversos: programação dinâmica, fluxo de rede, busca em grafo, guloso.
  - Estes problemas são fáceis por várias razões diferentes.

- Em contraste, os problemas da esquerda são todos difíceis pela mesma e única razão:
  - No fundo, eles são todos o mesmo problema, apenas com diferentes disfarces!
  - Ou seja, todos são equivalentes: cada um pode ser reduzido a qualquer outro.

6 PROBLEMAS NP-COMPLETO

- A teoria de complexidade a ser apresentada não mostra como obter algoritmos polinomiais para problemas que demandam algoritmos exponenciais, nem afirma que não existem.
- É possível mostrar que os problemas para os quais não há algoritmo polinomial conhecido são computacionalmente relacionados.
  - Formam a classe conhecida como NP.

- Propriedade: um problema da classe NP poderá ser resolvido em tempo polinomial se e somente se todos os outros problemas em NP também puderem.
- Este fato é um indício forte de que dificilmente alguém será capaz de encontrar um algoritmo eficiente para um problema da classe NP.

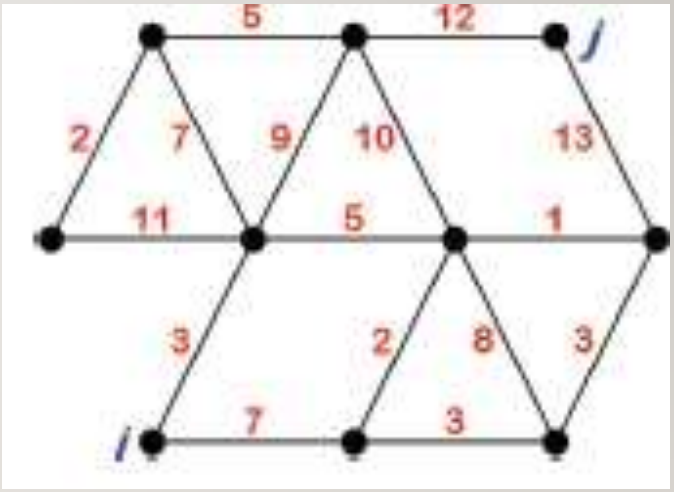
7 NP – PROBLEMAS DE BUSCA

- Um problema de busca é aquele onde uma solução proposta pode ser rapidamente verificada quanto à correção.
  - cujo resultado da computação seja "sim" ou "não".

- A classe NP é a classe de todos os problemas de busca
- Muitos problemas NP podem ser resolvidos em tempo polinomial
  - A solução pode ser muito difícil ou impossível de ser obtida, mas uma vez conhecida ela pode ser verificada em tempo polinomial.

8 EXEMPLOS: CAMINHO EM UM GRAFO

- Considere um grafo com peso, dois vértices i, j e um inteiro k > 0.



- Fácil: Existe um caminho de i até j com peso ≤ k?.
  - Há um algoritmo eficiente com complexidade de tempo O(E log V), sendo E o número de arestas e V o número de vértices (algoritmo de Dijkstra).
- Difícil: Existe um caminho de i até j com peso ≥ k?
  - Não se conhece algoritmo eficiente. É equivalente ao PCV em termos de complexidade.

9 COLORAÇÃO DE UM GRAFO

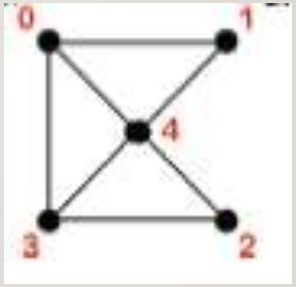
- Em um grafo G = (V,E), mapear C :V → S, sendo S um conjunto finito de cores tal que se (v,w) ∈ E então c(v) != c(w) (vértices adjacentes possuem cores distintas).
- O número cromático X(G) de G é o menor número de cores necessário para colorir G, isto é, o menor k para o qual existe uma coloração C para G e |C(V)| = k.
- O problema é produzir uma coloração ótima, que é a que usa apenas X(G) cores.
- Formulação do tipo "sim/não": dados G e um inteiro positivo k, existe uma coloração de G usando k cores?
  - Fácil: k = 2.
  - Difícil: k > 2.
- Aplicação: modelar problemas de agrupamento (clustering) e de horário (scheduling).

10 COLORAÇÃO DE UM GRAFO: PROBLEMA DO HORÁRIO

- Suponha que os exames finais de um curso tenham de ser realizados em uma única semana.
- Disciplinas com alunos de cursos diferentes devem ter seus exames marcados em horários diferentes.
- Dadas uma lista de todos os cursos e outra lista de todas as disciplinas cujos exames não podem ser marcados no mesmo horário, o problema em questão pode ser modelado como um problema de coloração de grafos.

11 CIRCUITO HAMILTONIANO

- Circuito Hamiltoniano: passa por todos os vértices uma única vez e volta ao vértice inicial (ciclo simples).
- Caminho Hamiltoniano: passa por todos os vértices uma única vez (ciclo simples).
- Exemplos:
  - Circuito Hamiltoniano: 0 1 4 2 3 0
  - Caminho Hamiltoniano: 0 1 4 2 3
- Existe um ciclo de Hamilton no grafo G?
  - Fácil: Grafos com grau máximo = 2.
  - Difícil : Grafos com grau > 2.
- É um caso especial do PCV.
  - Pares de vértices com uma aresta entre eles tem distância 1; e
  - Pares de vértices sem aresta entre eles têm distância infinita.



12 PROBLEMA DA SATISFABILIDADE

- Considere um conjunto de variáveis booleanas $x_1, x_2, \ldots, x_n$, que podem assumir valores lógicos verdadeiro ou falso.
- A negação de x_i é representada por $\neg x_i$.
- Expressão booleana: variáveis booleanas e operações ou ($\vee$) e e ($\wedge$) (também chamadas respectivamente de adição e multiplicação).
- Uma expressão booleana E contendo um produto de adições de variáveis booleanas é dita estar na forma normal conjuntiva.
- Dada E na forma normal conjuntiva, com variáveis x_i, $1 \leq i \leq n$, existe uma atribuição de valores verdadeiro ou falso às variáveis que torne E verdadeira ("satisfaça")?
- $E_1 = (x_1 \vee x_2) \wedge (x_1 \vee \neg x_3 \vee x_2) \wedge (x_3)$ é satisfatível ($x_1 = F, x_2 = V, x_3 = V$).
- A expressão $E_2 = x_1 \wedge \neg x_1$ não é satisfatível.

13 PROBLEMA DA SATISFABILIDADE

- O algoritmo AvalND(E,n) verifica se uma expressão E na forma normal conjuntiva, com variáveis x_i, $1 \leq i \leq n$, é satisfatível.

```
AVALND(E, n)
1  for i ← 1 to n
2  do x_i ← ESCOLHE(true, false)
3     if E(x_1, x_2, ..., x_n)
4        then sucesso
5        else insucesso
```

- O algoritmo obtém uma das 2^n atribuições possíveis de forma não-determinística em O(n).
- Melhor algoritmo determinístico: $O(2^n)$.
- Aplicação: definição de circuitos elétricos combinatórios que produzam valores lógicos como saída e sejam constituídos de portas lógicas e, ou e não.

14 CARACTERIZAÇÃO DAS CLASSES P E NP

- P: conjunto de todos os problemas que podem ser resolvidos por algoritmos determinísticos em tempo polinomial.
- NP: conjunto de todos os problemas que podem ser resolvidos por algoritmos não-determinísticos em tempo polinomial.
- Para mostrar que um determinado problema está em NP, basta apresentar um algoritmo não-determinístico que execute em tempo polinomial para resolver o problema.
- Outra maneira é encontrar um algoritmo determinístico polinomial para verificar que uma dada solução é válida.

15 EXISTE DIFERENÇA ENTRE P E NP?

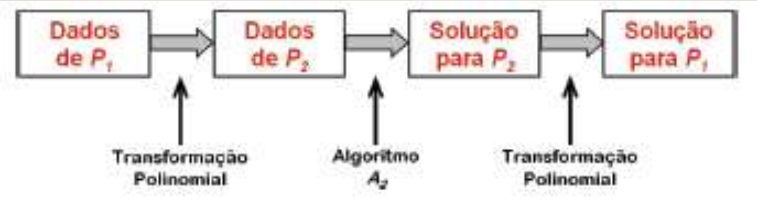
- P ⊆ NP, pois algoritmos determinísticos são um caso especial dos não-determinísticos.
- A questão é se P = NP ou P != NP.
- Esse é o problema não resolvido mais famoso que existe na área de ciência da computação.
- Se existem algoritmos polinomiais determinísticos para todos os problemas em NP, então P = NP.
- Por outro lado, a prova de que P != NP parece exigir técnicas ainda desconhecidas.

16 NP ⊃ P OU NP = P? CONSEQUÊNCIAS

- Muitos problemas práticos em NP podem ou não pertencer a P (não conhecemos nenhum algoritmo determinístico eficiente para eles).
- Se conseguirmos provar que um problema não pertence a P, então não precisamos procurar por uma solução eficiente para ele.
- Como não existe tal prova, sempre há esperança de que alguém descubra um algoritmo eficiente.
- Quase ninguém acredita que NP = P.
- Existe um esforço considerável para provar o contrário, mas a questão continua em aberto!

17 TRANSFORMAÇÃO POLINOMIAL

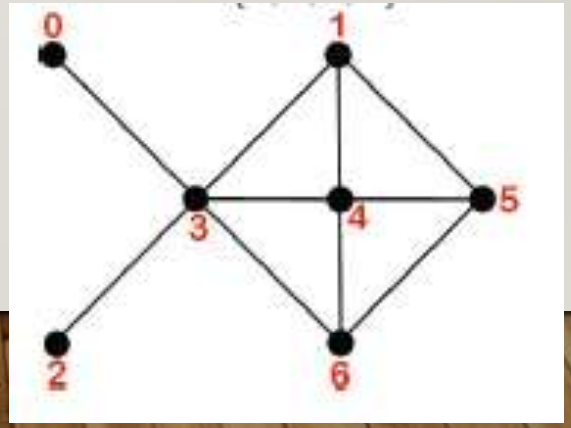
- Sejam P1 e P2 dois problemas "sim/não".
- Suponha que um algoritmo A2 resolva P2.
- Se for possível transformar P1 em P2 e a solução de P2 em solução de P1, então A2 pode ser utilizado para resolver P1.
- Se pudermos realizar as transformações nos dois sentidos em tempo polinomial, então P1 é polinomialmente transformável em P2.



- Esse conceito é importante para definir a classe NP-completo.
- Para mostrar um exemplo de transformação polinomial, definiremos clique de um grafo e conjunto independente de vértices de um grafo.

18 CONJUNTO INDEPENDENTE DE VÉRTICES DE UM GRAFO

- O conjunto independente de vértices de um grafo G = (V,E) é constituído do subconjunto $V' \subseteq V$, tal que $v, w \in V' \Rightarrow (v, w) \notin E$.
- Todo par de vértices de V' é não adjacente (V' é um subgrafo totalmente desconectado).
- Exemplo de cardinalidade 4: $V' = \{0, 2, 1, 6\}$.

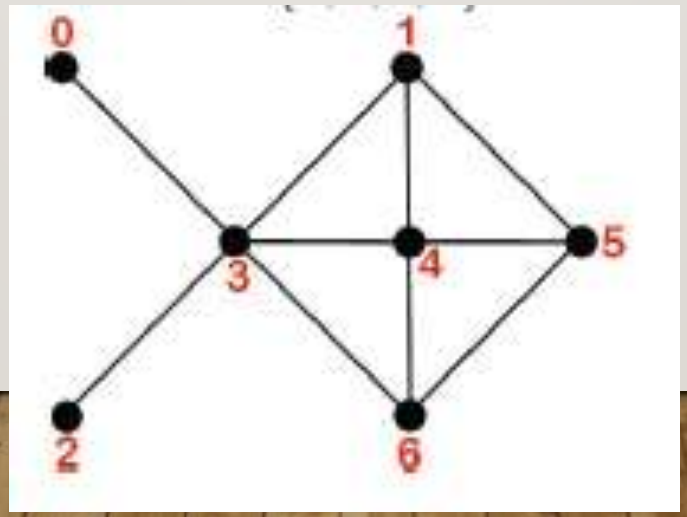


19 CONJUNTO INDEPENDENTE DE VÉRTICES: APLICAÇÃO

- Em problemas de dispersão é necessário encontrar grandes conjuntos independentes de vértices. Procura-se um conjunto de pontos mutuamente separados.
- Exemplo: identificar localizações para instalação de franquias.
- Duas localizações não podem estar perto o suficiente para competirem entre si.
- Solução: construir um grafo em que possíveis localizações são representadas por vértices, e arestas são criadas entre duas localizações que estão próximas o suficiente para interferir.
- O maior conjunto independente fornece o maior número de franquias que podem ser concedidas sem prejudicar as vendas.
- Em geral, conjuntos independentes evitam conflitos entre elementos.

20 CLIQUE DE UM GRAFO

- Clique de um grafo G = (V,E) é constituído do subconjunto $V' \subseteq V$, tal que $v,w \in V' \Rightarrow (v,w) \in E$.
- Todo par de vértices de V' é adjacente (V' é um subgrafo completo).
- Exemplo de cardinalidade 3: $V' = \{3, 1, 4\}$.



CLIQUE DE UM GRAFO: APLICAÇÃO

- O problema de identificar agrupamentos de objetos relacionados frequentemente se reduz a encontrar grandes cliques em grafos.
- Exemplo: empresa de fabricação de peças por meio de injeção plástica que fornece para diversas outras empresas montadoras.
- Para reduzir o custo relativo ao tempo de preparação das máquinas injetoras, pode-se aumentar o tamanho dos lotes produzidos para cada peça encomendada.
- É preciso identificar os clientes que adquirem os mesmos produtos, para negociar prazos de entrega comuns e assim aumentar o tamanho dos lotes produzidos.
- Solução: construir um grafo com cada vértice representando um cliente e ligar com uma aresta os que adquirem os mesmos produtos.
- Um clique no grafo representa o conjunto de clientes que adquirem os mesmos produtos.

22 TRANSFORMAÇÃO POLINOMIAL

- Considere P1 o problema clique e P2 o problema conjunto independente de vértices.
- A instância I de clique consiste de um grafo $\overline{G} = (V,A)$ e um inteiro $k > 0$.
- A instância f(I) de conjunto independente pode ser obtida considerando-se o grafo complementar $\overline{G}$ de G e o mesmo inteiro k.
- f(I) é uma transformação polinomial:
  - $\overline{G}$ pode ser obtido a partir de G em tempo polinomial.
  - G possui clique de tamanho ≥ k se e somente se $\overline{G}$ possui conjunto independente de vértices de tamanho ≥ k.

23 TRANSFORMAÇÃO POLINOMIAL

- Se existe um algoritmo que resolve o conjunto independente em tempo polinomial, ele pode ser utilizado para resolver clique também em tempo polinomial.
- Diz-se que clique $\propto$ conjunto independente.
- Denota-se $P_1 \propto P_2$ para indicar que P_1 é polinomialmente transformável em P_2.
- A relação $\propto$ é transitiva:
- $((P_1 \propto P_2) \wedge (P_2 \propto P_3)) \rightarrow P_1 \propto P_3$.

24 PROBLEMAS NP-COMPLETO E NP-DIFÍCIL

- Dois problemas P_1 e P_2 são polinomialmente equivalentes se e somente se $P_1 \propto P_2$ e $P_2 \propto P_1$.
  - Exemplo: problema da satisfabilidade (SAT). Se SAT $\propto P_1$ e $P_1 \propto P_2$, então SAT $\propto P_2$.
  - Um problema P é NP-difícil se e somente se SAT $\propto$ P (satisfabilidade é redutível a P).

- Um problema de decisão P é denominado NP-completo quando:
  - P $\in$ NP.
  - Todo problema de decisão P′ $\in$ NP-completo satisfaz P′ $\propto$ P.

25 PROBLEMAS NP-COMPLETO E NP-DIFÍCIL

- Um problema de decisão P que seja NP-difícil pode ser mostrado ser NP-completo exibindo um algoritmo não-determinístico polinomial para P.
  - Problemas de Decisão, Problemas de Pesquisa e Problemas de Otimização
- Apenas problemas de decisão (“sim/não”) podem ser NP-completo.
- Problemas de otimização podem ser NP-difícil, mas geralmente, se P_1 é um problema de decisão e P_2 um problema de otimização, é bem possível que $P_1 \propto P_2$.
- A dificuldade de um problema NP-difícil não é menor do que a dificuldade de um problema NP-completo.

26 EXEMPLO: PROBLEMA DA PARADA

- É um exemplo de problema NP-difícil que não é NP-completo.
- Consiste em determinar, para um algoritmo determinístico qualquer A com entrada de dados E, se o algoritmo A termina (ou entra em um loop infinito).
- É um problema indecidível. Não há algoritmo de qualquer complexidade para resolvê-lo.

- Mostrando que SAT $\propto$ problema da parada:
  - Considere o algoritmo A cuja entrada é uma expressão booleana na forma normal conjuntiva com n variáveis.
  - Basta tentar 2n possibilidades e verificar se E é satisfatível.
  - Se for, A pára; senão, entra em loop.
  - Logo, o problema da parada é NP-difícil, mas não é NP-completo.

PROVA DE QUE UM PROBLEMA É NP-COMPLETO

- São necessários os seguintes passos:
  - Mostre que o problema está em NP.
  - Mostre que um problema NP-completo conhecido pode ser polinomialmente transformado para ele.
- É possível porque existe uma prova direta de que SAT é NP-completo, além do fato de a redução polinomial ser transitiva
- $((SAT \propto P_1) \wedge (P_1 \propto P_2)) \rightarrow SAT \propto P_2$.

- Para ilustrar como um problema P pode ser provado ser NP-Completo, basta considerar um problema já provado ser NP-Completo e apresentar uma redução polinomial desse problema para P.

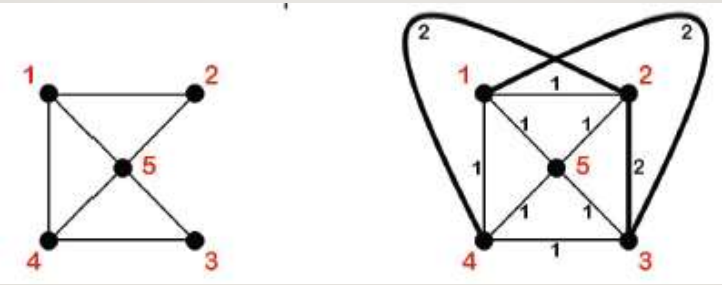
28 PCV É NP-COMPLETO: PARTE 1 DA PROVA

- Mostrar que o Problema do Caixeiro Viajante (PCV) está em NP.
- Prova: a partir do problema Circuito Hamiltoniano, um dos primeiros que se provou ser NP-completo.
- Isso pode ser feito:
  - apresentando (como abaixo) um algoritmo não-determinístico polinomial para o PCV ou
  - mostrando que, a partir de uma dada solução para o PCV, esta pode ser verificada em tempo polinomial.

```
procedure PCVND;
begin
  i := 1;
  for t := 1 to v do
    begin
      j := escolhe(i, lista-adj(i));
      antecessor[j] := i;
      i := j;
    end;
end;
```

29 PCV É NP-COMPLETO - PARTE 2 DA PROVA

- Apresentar uma redução polinomial do ciclo de Hamilton para o PCV. Pode ser feita conforme o exemplo abaixo.



- Dado um grafo representando uma instância do ciclo de Hamilton, construa uma instância do PCV como se segue:
  - Para cidades use os vértices.
  - Para distâncias use 1 se existir um arco no grafo e 2 se não existir.
- A seguir, use o PCV para achar um roteiro menor ou igual a V .
- O roteiro é o ciclo de Hamilton.

CLASSE NP-COMPLETO: RESUMO

- Problemas que pertencem a NP, mas que podem ou não pertencer a P.
- Propriedade: se qualquer problema NP-completo puder ser resolvido em tempo polinomial por uma máquina determinística, então todos os problemas da classe podem, isto é, P = NP.
- A falha coletiva de todos os pesquisadores para encontrar algoritmos eficientes para estes problemas pode ser vista como uma dificuldade para provar que P = NP.
- Contribuição prática da teoria: fornece um mecanismo que permite descobrir se um novo problema é "fácil" ou "difícil".
- Se encontrarmos um algoritmo eficiente para o problema, então não há dificuldade. Senão, uma prova de que o problema é NP-completo nos diz que o problema é tão "difícil" quanto todos os outros problemas "difíceis" da classe NP-completo.

FIM DA AULA 22

Fim da Disciplina

\o/