Análise de Algoritmos
Problemas NP-Completos

Nelson Cruz Sampaio Neto
nelsonneto@ufpa.br

Universidade Federal do Pará
Instituto de Ciências Exatas e Naturais
Faculdade de Computação

12 de novembro de 2019

Introdução

- Já estudamos algoritmos de tempo polinomial sobre entradas de tamanho n com $O(n^k)$ para alguma constante k.
- Embora $\log(n)$ não seja um polinômio, seu valor é $O(n)$, logo é eficiente da mesma forma.
- Todos os problemas são resolvidos em tempo polinomial?
- Não. Existem os problemas que hoje podem ser resolvidos, mas não em tempo $O(n^k)$ para qualquer constante k.
- Outros nem podem ser resolvidos, não importa quanto tempo seja fornecido: Problema da parada (ou *halting problem*).

Introdução

- Em geral, imaginamos problemas que podem ser resolvidos em tempo polinomial como sendo **tratáveis**.
- E os problemas que exigem algoritmos de complexidade no tempo não-polinomial como **intratáveis**.
- A teoria de complexidade a ser apresentada não mostra como obter algoritmos polinomiais para problemas aparentemente intratáveis, nem afirma que não existem.
- Porém, é possível mostrar que esses problemas para os quais não há algoritmo polinomial conhecido estão relacionados.
- Eles formam a classe de problemas conhecida como NP.

Introdução

- A maioria dos estudiosos acredita que os *NP* são problemas intratáveis, dada a quantidade de problemas dessa natureza ainda sem solução polinomial.
- Contudo, até que se prove o contrário, não podemos eliminar a possibilidade de que os problemas *NP* possam ser resolvidos em tempo polinomial.
- Se puder estabelecer um problema como *NP*, você terá uma **boa evidência** de sua intratabilidade e poderá se dedicar ao desenvolvimento de um algoritmo aproximado ou a resolver um caso especial tratável.

Tipos de problemas

- Existem três tipos gerais de problemas. São os problemas de decisão, localização e otimização.

- Em um problema de decisão, o objetivo consiste em decidir se a resposta é SIM ou NÃO.

- Em um problema de localização, o objetivo é encontrar uma certa estrutura que satisfaça um conjunto de propriedades dadas.

- Se as propriedades envolvem encontrar a melhor estrutura possível, então o problema torna-se de otimização.

Tipos de problemas

- **Problema de Decisão:**
  Existe uma estrutura S que satisfaça uma propriedade P?
  Resposta: SIM ou NÃO.

- **Problema de Localização:**
  Encontrar uma estrutura S que satisfaça uma propriedade P.
  Resposta: Uma estrutura S.

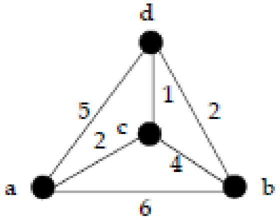
- **Problema de Otimização:**
  Encontrar uma estrutura S que satisfaça uma propriedade P e que seja a melhor, segundo algum critério α de medida.
  Resposta: Uma estrutura S tal que não haja outra melhor.

Exemplo: Problema do caixeiro viajante (PCV)

- **Problema clássico de otimização:** Dado um grafo, consiste na procura de um ciclo de Hamilton de custo mínimo.

- **Problema de localização relacionado:** Dado um grafo, ache um ciclo de Hamilton de custo total menor ou igual a k.

- **Problema de decisão relacionado:** Dado um grafo, existe um ciclo de Hamilton de custo total menor ou igual a k?



Tipos de problemas

- O problema de decisão é o mais simples, tendo dificuldade não maior do que os problemas de localização e otimização.
- Por exemplo, podemos resolver o PCV de decisão resolvendo o de otimização, e comparando o custo mínimo encontrado ao valor do parâmetro de decisão k.
- Alguma prova da possível intratabilidade de um problema de decisão pode ser estendida aos outros casos.
- Por esse motivo, os problemas em discussão daqui pra frente serão todos de decisão.

A classe P de problemas

- A classe P contém os problemas cuja complexidade no pior caso é uma função polinomial no tamanho da entrada.
- Para provar que um problema é da classe P, basta mostrar um algoritmo eficiente que o resolva. Ex.: Ordenação.
- Para provar que um problema não é da classe P, é necessário provar que não existe algoritmo polinomial que o resolva.
- Por exemplo, todos os algoritmos conhecidos para resolver o PCV são não-polinomiais. Contudo, ainda não se provou que todo possível algoritmo que o resolva não é eficiente.
- Não se sabe, portanto, se o PCV pertence ou não à classe P.

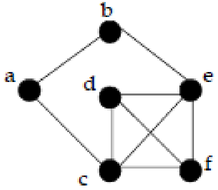
Problemas aparentemente intratáveis

- Essa incerteza sobre pertencer, ou não, à classe P também é vista em vários outros problemas.
- Esses problemas apresentam geralmente um grande número de alternativas que precisam ser verificadas para saber qual delas responde ao problema.
- Só que quando o número de alternativas é exponencial, fica difícil desenvolver um algoritmo eficiente.
- **Suspeita-se** que esses problemas sejam intratáveis.

- **Exemplo:** Problema do ciclo de Hamilton.

  Dados: Um grafo G.
  Decisão: G possui um ciclo de Hamilton?

- Por exemplo, considere o grafo G abaixo:

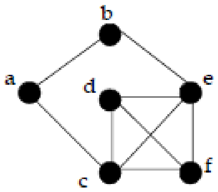


- O ciclo $\{a, b, e, f, d, c, a\}$ é obviamente hamiltoniano.

- **Exemplo:** Clique em grafo.

  Dados: Um grafo G e um inteiro $k > 0$.
  Decisão: G possui um clique de tamanho $\geq k$?

- Por exemplo, considere o grafo G abaixo:

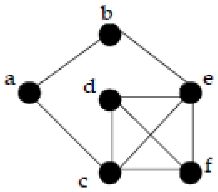


- Caso $k = 4$, a resposta seria SIM. O subgrafo completo de G $\{c, d, e, f\}$ é um clique de tamanho 4.

- **Exemplo:** Conjunto independente (c.i.) de vértices.

  Dados: Um grafo G e um inteiro $k > 0$.
  Decisão: G possui um c.i. de tamanho $\geq k$?

- Por exemplo, considere o grafo G abaixo:



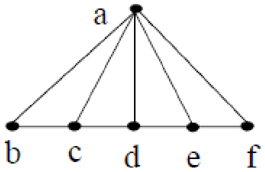
- Considerando $k = 2$, a resposta seria SIM. O conjunto de vértices $\{a, e\}$ é independente de tamanho 2.

- **Exemplo:** Coloração de vértices.

  Dados: Um grafo G e um inteiro $k > 0$.
  Decisão: G possui uma coloração $\leq k$ cores?

- Por exemplo, o grafo G abaixo admite uma coloração no mínimo com 3 cores.

a $\rightarrow$ vermelho
b $\rightarrow$ azul
c $\rightarrow$ amarelo
d $\rightarrow$ azul
e $\rightarrow$ amarelo
f $\rightarrow$ azul



- **Exemplo:** Satisfabilidade de fórmulas (SAT).

  Dados: Expressão booleana E na forma normal conjuntiva.
  Decisão: E é satisfeita, ou seja, é verdadeira?

- Por exemplo, considere a expressão booleana E_1 abaixo:

  $$E_1 = (x_1 \vee \overline{x_2}) \wedge (\overline{x_1} \vee \overline{x_2} \vee x_3) \wedge (\overline{x_1} \vee x_2 \vee \overline{x_3})$$

  Resposta: Sim. Com os valores $x_1 = V$, $x_2 = V$ e $x_3 = V$.

- Por exemplo, considere a expressão booleana E_2 abaixo:

  $$E_2 = (x_1 \vee x_1) \wedge (\overline{x_1} \vee \overline{x_1})$$

  Resposta: Não. Nenhum valor dado a x_1 torna E_2 verdadeira.

A classe NP de problemas

- A classe *NP* engloba **problemas de decisão** π, tal que existe uma justificativa à resposta SIM, cuja verificação é feita por um algoritmo polinomial.
- São os problemas ditos "verificáveis" em tempo polinomial.
- Note que não se exige uma solução polinomial para π, só que exista um algoritmo eficiente para **verificar** a resposta SIM.
- Também não é exigido nada em relação a resposta NÃO.
- De fato, há problemas da classe *NP* que admitem algoritmos polinomiais para suas justificativas NÃO. Como há também aqueles para os quais tais algoritmos não são conhecidos.

A classe NP de problemas

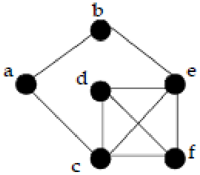
- O processo para justificar respostas a problemas de decisão é composto por duas fases distintas:

- **FASE 1:** Exibição.
  - Exibir uma justificativa conveniente.

- **FASE 2:** Reconhecimento ou verificação.
  - Verificar que a justificativa apresentada no processo de exibição é, de fato, satisfatória.

A classe NP de problemas

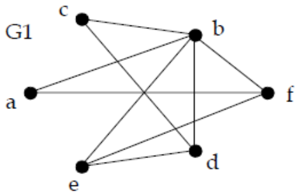
- **Exemplo:** O problema do clique em grafo é classe NP?



- Para mostrar que é NP, temos que exibir uma justificativa à resposta SIM e verificar em $O(n^k)$ que ela é satisfatória.
- **FASE 1:** A justificativa é um subconjunto V' de vértices.
- **FASE 2:** Para verificar a justificativa SIM basta:
  - Verificar o tamanho do subconjunto: $O(V')$
  - Verificar se o tamanho é $\geq k$: $O(1)$
  - Verificar se existe arestas entre todos os pares de V': $O(V'^2)$
- Como a verificação é feita em tempo polinomial: Clique $\in NP$.

A classe NP de problemas

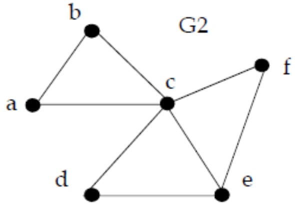
- **Exemplo:** O problema do ciclo de Hamilton é classe *NP*?



- Para mostrar que é *NP*, temos que exibir uma justificativa à resposta SIM e verificar em $O(n^k)$ que ela é satisfatória.
- **FASE 1:** Exibição
  - Como justificativa a sequência de vértices: $\{a, b, c, d, e, f, a\}$.
- **FASE 2:** Reconhecimento
  - Verificar se a sequência é um ciclo e visita exatamente uma vez cada vértice, o que é implementável em tempo polinomial.
- Logo, ciclo de Hamilton pertence à classe *NP*.

A classe NP de problemas

- E a resposta NÃO pode ser verificada em tempo polinomial?



- **FASE 1:** Exibição
  - Consiste da apresentação do conjunto de sequências: $\{(a, b, c, a), (e, c, d, e), (e, c, f, e), (e, d, c, f, e)\}$.
- **FASE 2:** Reconhecimento
  - Verificar se cada sequência não é um ciclo de Hamilton e que o conjunto engloba **todos** possíveis ciclos simples, o que pode ser exponencial com o tamanho o grafo.

A classe NP de problemas

- A justificativa SIM possui como passo de reconhecimento um algoritmo polinomial, logo, ciclo de Hamilton é *NP*.
- Até o momento, é desconhecido se existe outro processo para justificar a resposta NÃO, tal que o passo de reconhecimento corresponda a um algoritmo polinomial.
- Como sua inexistência também não foi provada, é impossível afirmar se o problema do ciclo de Hamilton (e muitos outros) é tratável ou intratável.
- O fato do problema pertencer à classe *NP* **não** garante sua tratabilidade ou intratabilidade.

Todo problema de decisão é NP?

- Por outro lado, existem muitos problemas que se desconhece sua pertinência, ou não, à classe NP. Por exemplo:

- **Exemplo:** Clique máximo.

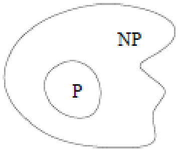
  Dados: Um grafo G e um inteiro $k > 0$.
  Decisão: O clique máximo de G possui k vértices?

- Para justificar a resposta SIM do problema seria preciso exibir um conjunto S com **todos** os cliques maximais de G, o que pode ser exponencial no tamanho de G.

- Uma justificativa polinomial ainda não foi encontrada, porém, também não foi provado que ela não possa existir. Logo, não é sabido se o problema é ou não NP.

A questão P $\subseteq$ NP?

- Todo problema de decisão pertencente à classe P também pertence à classe NP. Ou seja, P $\subseteq$ NP.
- Seja $\pi \in P$ um problema de decisão. Então, é certo que existe um algoritmo polinomial A que resolve π.
- Em particular, o algoritmo A pode ser utilizado para encontrar uma justificativa à resposta SIM de π.
- Logo, $\pi \in NP$.



Acredita-se que $NP \gg P$

A questão P = NP ou P $\neq$ NP?

- Existe algum problema na classe NP que seja intratável?
- Todo e qualquer problema pertencente à classe NP admite necessariamente algoritmo polinomial?
- Até então, **não** se conhece a resposta para essas perguntas. Todas as evidências apontam para $P \neq NP$.
- O fato é que nossa compreensão da relação entre P e NP é terrivelmente incompleta.
- Um prêmio de um milhão de dólares para quem a respondê-la está aberto: www.claymath.org/millennium/.

Exercícios

1. Usando o problema do clique em grafos, apresente a diferença entre problemas de decisão, localização e otimização.
2. Mostre que os problemas da satisfabilidade (de fórmulas) e do conjunto independente de vértices pertencem à classe NP.
3. Se fosse provado que um problema da classe NP é intratável, poderíamos afirmar que $P \neq NP$?
4. Se existissem algoritmos polinomiais para todos os problemas em NP, seria possível afirmar que $P = NP$?
5. Se um problema de decisão $\pi \in NP$, então $\pi \notin P$?

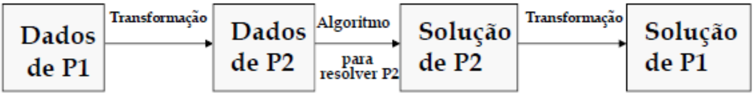
Transformação polinomial

- Suponha que queiramos resolver um problema π_1 e já temos um algoritmo A_2 para resolver um outro problema π_2.

- Para resolver π_1 usando A_2, é preciso transformar π_1 em π_2 da seguinte forma:

  (i) transformar os dados de entrada do problema π_1 em dados de entrada para o problema π_2.

  (ii) transformar a solução de π_2 numa solução de π_1.



Transformação polinomial

- π_1 é dito **polinomialmente transformável** em π_2, denotado por $\pi_1 \propto \pi_2$, quando os passos (i) e (ii) são polinomiais.

- Assim, caso o algoritmo A_2 seja polinomial e $\pi_1 \propto \pi_2$, então π_1 também pode ser resolvido em tempo polinomial.

- Note que a relação é **transitiva**:

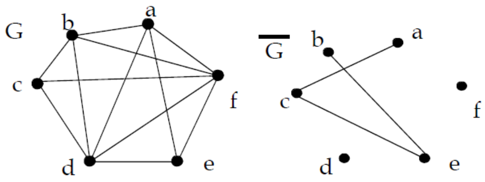
$$\pi_1 \propto \pi_2 \text{ e } \pi_2 \propto \pi_3 \Rightarrow \pi_1 \propto \pi_3$$

- Dois problemas são **polinomialmente equivalentes** quando:

$$\pi_1 \propto \pi_2 \text{ e } \pi_2 \propto \pi_1$$

Transformação polinomial

- Os problemas do clique e conjunto independente de vértices são ditos polinomialmente equivalentes.
- Encontrar um clique no grafo G equivale a obter um c.i. de vértices no seu complementar $\overline{G}$, e vice-versa.
  - **Passo (i):** A construção de $\overline{G}$ a partir de G é polinomial.
  - **Passo (ii):** G possui um clique de tamanho $\geq k$ se e somente se $\overline{G}$ possui um c.i. de vértices de tamanho $\geq k$, e vice-versa. Transformar a solução é, portanto, polinomial, aliás é direta.



A classe NP-Completo

- Os problemas da classe P são conhecidos como os de "menor complexidade" em NP.

- Em contrapartida, os problemas da classe NP-Completo são os de "maior dificuldade" dentre todos em NP.

- Um problema de decisão π é dito NP-Completo quando:

  - **Condição 1:** $\pi \in NP$.

  - **Condição 2:** Todo problema $\beta \in NP$ satisfaz $\beta \propto \pi$.

- A Condição 2 mostra que se **um** problema NP-Completo for resolvido em tempo polinomial, **todo** problema NP também admite solução polinomial, ou seja, teríamos $P = NP$.

A classe NP-Completo

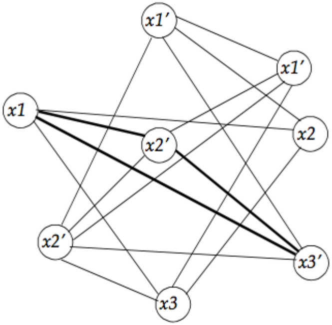
- Provar a Condição 2 é uma tarefa bastante árdua!
- Contudo, o lema abaixo vem contornar essa questão:

  "Sejam π_1 e π_2 problemas em *NP*. Se π_1 for *NP*-Completo e $\pi_1 \propto \pi_2$, então π_2 também é *NP*-Completo."
- Assim, bastaria encontrar o primeiro problema *NP*-Completo.
- **Teorema de Cook[1]:** Satisfabilidade (SAT) é *NP*-Completo.
- Por exemplo, para provar que o problema do clique em grafo é *NP*-Completo, mostre que clique $\in$ *NP* e SAT $\propto$ clique.

[1] https://dl.acm.org/citation.cfm?doid=800157.805047

- Considere a seguinte expressão booleana E e o grafo G construído a partir de E:

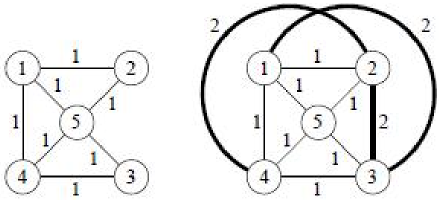
$$E = (x_1 \vee \overline{x_2}) \wedge (\overline{x_1} \vee \overline{x_2} \vee x_3) \wedge (\overline{x_1} \vee x_2 \vee \overline{x_3})$$



- Cada clique de tamanho 3 corresponde a uma atribuição de "Verdadeiro" às variáveis correspondentes em E de tal forma que E seja satisfeita. Logo, SAT $\propto$ clique.

Demonstração: Decisão do PCV é NP-Completo

- A prova é feita a partir do problema ciclo de Hamilton, um dos primeiros que se provou ser *NP*-completo.
- **Condição 1:** PCV $\in NP$. A verificação de uma justificativa à resposta SIM pode ser feita em tempo polinomial.
- **Condição 2:** Dado $G = (V, E)$ representando uma instância do ciclo de Hamilton, gere uma instância do PCV:



- Em seguida, use o PCV para achar um ciclo de custo menor ou igual a V. O resultado é um ciclo de Hamilton.

Restrições de problemas

- Diz-se que π' é uma restrição de π quando os dados de π' são iguais ou mais restritos que os dados de π.

- Por exemplo, o problema 3-SAT é uma restrição (ou seja, um caso particular) de SAT.

  Problema: 3-satisfabilidade de fórmulas (3-SAT).

  Dados: Expressão booleana E na forma normal conjuntiva, tal que cada cláusula de E possui exatamente 3 variáveis.

  Decisão: E é satisfeita, isto é, verdadeira?

- Claramente, tem-se que $\pi' \propto \pi$. Logo, se π' é NP-Completo e $\pi \in NP$, então π é NP-Completo.

Restrições de problemas

- Método alternativo para provar que π é NP-Completo:
  - **Condição 1:** $\pi \in NP$.
  - **Condição 2:** Encontrar uma restrição π' de π, que seja NP-Completo.
- **Problema:** Isomorfismo de subgrafos.

  Dados: Grafos G_1 e G_2.
  Decisão: G_1 contém um subgrafo isomorfo a G_2?
- O problema do clique pode ser visto como uma restrição do isomorfismo de subgrafos: basta fixar G_2 como um clique.
- Isomorfismo de subgrafos $\in NP$ e clique $\in NP$-Completo, logo isomorfismo de subgrafos é NP-Completo.

Todo NP é NP-Completo?

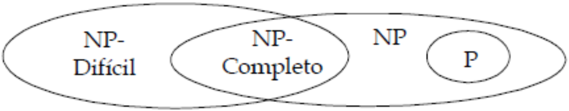
- A resposta para essa pergunta é **não**, e um exemplo que ainda persiste é o problema de isomorfismo de grafos.
- Dois grafos são isomorfos se pode-se transformar um no outro simplesmente renomeando os vértices.

  Problema: Isomorfismo de grafos.

  Dados: Grafos G_1 e G_2.
  Decisão: G_1 é isomorfo a G_2?
- Suspeita-se que o problema de isomorfismo de grafos não está em P e nem em NP-completo, ainda que esteja em NP.
- Trata-se de um problema aparentemente intratável, mas não tanto para estar em NP-completo.

A classe NP-Difícil

- Um problema é *NP*-Difícil se a Condição 2 for satisfeita.
- Ou seja, um problema π_2 é *NP*-Difícil se $\pi_1 \propto \pi_2$, onde π_1 é um problema *NP*-Completo.
- Não importa se π_2 é ou não *NP*.
- Podemos dizer que a classe *NP*-Completo é a interseção das classes *NP*-Difícil e *NP*.



Demonstração

- Sabendo que o problema de decisão do PCV é *NP*-Completo, podemos afirmar que a sua otimização é *NP*-Difícil?
- Ao resolvermos a otimização do PCV para um grafo G, o peso total p do ciclo encontrado é o menor valor possível. Se $p \leq k$, então o problema de decisão tem resposta SIM, senão NÃO.
- Assim, é possível transformar polinomialmente o problema de decisão em otimização.
- Então, como a decisão é *NP*-Completo, podemos afirmar que a otimização do PCV é *NP*-Difícil.
- Lembrando que, somente problemas de decisão podem ser *NP* e, consequentemente, *NP*-Completo.

Demonstração

- **Problema da parada:** Consiste em se decidir para qualquer algoritmo e qualquer entrada se o algoritmo vai terminar ou entrar em *loop* infinito.
- Esse problema é **indecidível**, ou seja, não existe algoritmo de qualquer complexidade que o resolva.
- O problema da parada não pertence a classe P, obviamente, e nem a classe NP. Contudo, ele é *NP*-Difícil.
- Para mostrar que SAT $\propto$ problema da parada, consideramos o algoritmo A cuja entrada é a expressão E com n variáveis:
$\Rightarrow$ Basta tentar 2^n possibilidades e verificar se a expressão é satisfeita. Se for, A para; senão, entra em *loop* infinito.
- A dificuldade de um problema *NP*-Difícil não é menor do que a dificuldade de um *NP*-Completo.

Considerações

- O caráter *NP*-Completo consiste em mostrar o quanto um problema é "difícil", e não o quanto ele é "fácil".
- Muitos são os problemas aparentemente intratáveis presentes em diversas áreas de estudo.
- Há dezenas de anos busca-se desenvolver algoritmos eficientes para resolver tais problemas, mas ainda ninguém teve sucesso. Por isso, **acredita-se** fortemente que $P \neq NP$.
- Para se tornar um bom projetista de algoritmos, você precisa entender os rudimentos da teoria *NP*-Completo.
- Na prática, a contribuição desse conhecimento é possibilitar descobrir se um novo problema é "fácil" ou "difícil".

6. Prove que o problema do c.i. de vértices é *NP*-Completo.
7. Se existisse um algoritmo que resolvesse o problema do clique em tempo polinomial, poderíamos afirmar que esse algoritmo resolve o problema do c.i. em tempo polinomial? Por quê?
8. A dificuldade para resolver em tempo polinomial um problema *NP*-Difícil é menor que a de um *NP*-Completo? Por quê?
9. Os problemas da classe *NP*-Completo são polinomialmente equivalentes entre si?
10. Se $\pi_1 \propto \pi_2$ e $\pi_1 \in P$, então $\pi_2 \in P$? Por quê?
11. Se $\pi_1 \propto \pi_2$ e $\pi_2 \in P$, então $\pi_1 \in P$? Por quê?

Bibliografia

- Algoritmos : teoria e prática / Thomas H. Cormen [et al.] ; tradução da segunda edição [americana] Vandenberg D. de Souza. - Rio de Janeiro : Elsevier. 2002 - 6a Reimpressão.

- Grafos e algoritmos computacionais / Jayme Luiz Szwarcfiter. - Rio de Janeiro : Campus, 1984.

- Projeto de algoritmos : com implementação em Java e C++ / Nivio Ziviani ; consultoria em Java e C++ de Fabiano Cupertino Botelho. - São Paulo : Thomson Learning, 2007.