Grafos e Algoritmos Computacionais

NP-Completude

Prof. André Britto

Problemas NP-Completos

- Estudamos técnicas que resolvem algoritmos de forma "eficiente".
- Realidade: estas técnicas (nem outras) conseguem resolver alguns problemas "eficientemente".
- "**eficiente**" $\Rightarrow$ Complexidade do algoritmo é polinomial em relação ao tamanho da entrada.

Problemas NP-Completos

- Problema **tratável** $\Rightarrow$ Existe um algoritmo *A* polinomial dentre uma coleção *C* de algoritmos, que resolve um dado problema *P* $\Rightarrow$ tempo finito de solução de *P* por *A*.

- Problema **intratável** $\Rightarrow$ Não se conhece um algoritmo que resolve *P* em tempo Polinomial $\Rightarrow$ resolução de *P* por *A* pode durar séculos, mesmo se o tamanho da entrada for reduzido.

Problemas NP-Completos

- **tratável** $\Rightarrow$ exibição do algoritmo de complexidade polinomial.

- **intratável** $\Rightarrow$ prova que **todo possível** algoritmo que resolve *P* não possui complexidade polinomial.

Problemas de Decisão – Problema Algorítmico

- Caracteriza-se por:
  - Conjunto de dados (instância: objeto com dados específicos).
  - Objetivo do problema.

- Resolver o problema:
  - Desenvolver algoritmo $\Rightarrow$ solução.

Problemas de Decisão – Problema Algorítmico

- Ex.: Problema:
  - Instância : um par *(G,k)* específico.
  - Solução: um subgrafo completo de *G* com *k* ou mais vértices, se existir.

Classes Gerais de Problemas Algorítmicos

- Problemas de **Decisão** (sim / não)
- Problemas de **Localização** (Localizar estrutura *S*)
- Problemas de **Otimização** (Localizar estrutura *S* sobre critérios C)

otimização

Classes Gerais de Problemas Algorítmicos

- Podem ser associados:

  Ex.:

  Problema de Decisão

  existe estrutura *S* que satisfaça a propriedade *P?*

  Problemas de Localização

  Encontrar estrutura *S* que satisfaça propriedade *P*.

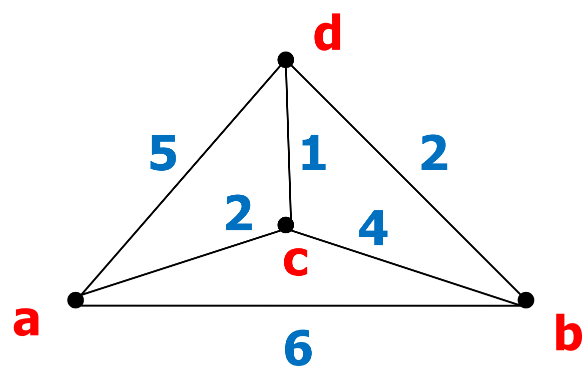
  Problema de Otimização

  Encontrar estrutura *S* que satisfaça *C* critérios de otimização

Classes Gerais de Problemas Algorítmicos

Ex.: Problema do Caixeiro Viajante

Percurso do Caixeiro Viajante ⇒ Ciclo Hamiltoniano de *G* cuja soma dos pesos das arestas seja mínimo.



um percurso : a b c d a ⇒ peso 16

um percurso ótimo : a b d c d ⇒ peso 11

Classes Gerais de Problemas Algorítmicos

1. Problema de Decisão

   Dados: um grafo G e um inteiro $k > 0$.

   Objetivo: Verificar se G possui um percurso de caixeiro viajante de peso $\leq k$.

2. Problemas de Localização

   Dados: um grafo G e um inteiro $k > 0$.

   Objetivo: Localizar em G um percurso de caixeiro viajante de peso $\leq k$.

3. Problema de Otimização

   Dados: um grafo G.

   Objetivo: Localizar em G um percurso de caixeiro viajante ótimo.

Classes Gerais de Problemas Algorítmicos

- Os problema estão relacionados:

Resolução 3 $\Rightarrow$ Resolução 2 $\Rightarrow$ Resolução 1

$\longleftarrow$

> dificuldade maior

Classes Gerais de Problemas Algorítmicos

- Por que estudar os de decisão ?
  - Mais simples que os outros dois.
  - Alguma prova de sua possível intratabilidade pode ser estendida facilmente aos demais problemas.

- Notação

$\pi(D,Q)$ →

- π : problema de decisão.
- D : dados
- Q : questão (decisão)

$\pi(I) \rightarrow \pi(D,Q)$ aplicada à instância I.

A Classe *P*

- Algoritmos eficientes: $O(1)$, $O(n)$, $O(n^2 \log n)$, $O(n^{10})$,...

- Algoritmos ineficiente: $O(2^n)$, $O(n!)$, ...

- São mais comuns algoritmos de ordem de polinômios baixos: *0,1,2,3*.

A Classe *P*

- **Classe *P*** $\Rightarrow$ compreende problemas de decisão *P* que admitem algoritmo polinomial.

- Exemplo de um problema de classe *P* :
  - Determinar se um grafo *G* é ou não acíclico.

- Observação:
  - Se os algoritmos conhecidos para resolver um certo problema π forem todos exponenciais, **não** necessariamente $\pi \notin P$.

A Classe *P*

- Para afirmar $\pi \notin P$ precisamos de uma prova que todo possível algoritmo para resolver π é não polinomial.

- Por exemplo os algoritmos conhecidos até agora para resolver o problema do Caixeiro Viajante são todos exponenciais. Contudo, não é conhecida nenhuma prova de que seja impossível a formulação de algoritmo polinomial para o problema.

$\Downarrow$

Desconhece-se se Caixeiro Viajante pertence ou não a *P*.

Essa incerteza está em Grande parte dos problemas.

Alguns Problemas aparentemente difíceis

- Só se conhecem algoritmos exponenciais para resolvê-los.
- Desconhece-se uma prova que não existe algoritmos polinomiais que resolvam os problemas.

1. Problema da Satisfabilidade

Dados: Uma expressão Booleana *E* na forma **FNC**(Forma Normal Conjuntiva)

Decisão: *E* é satisfatível ?

Alguns Problemas aparentemente difíceis

1. Problema da Satisfabilidade

A seguinte fórmula é uma gramática formal para FNC:

1. <ou> $\rightarrow$ $\vee$
2. <e> $\rightarrow$ $\wedge$
3. <não/negação> $\rightarrow$ $\neg$
4. <conjunção> $\rightarrow$ <disjunção>
5. <conjunção> $\rightarrow$ <conjunção> <e> <disjunção>
6. <disjunção> $\rightarrow$ <literal>
7. <disjunção> $\rightarrow$ (<disjunção> <ou> <literal>)
8. <literal> $\rightarrow$ <termo>
9. <literal> $\rightarrow$ <não><termo>

Expressão booleana satisfatível: atribuição de V ou F às variáveis e resultado V

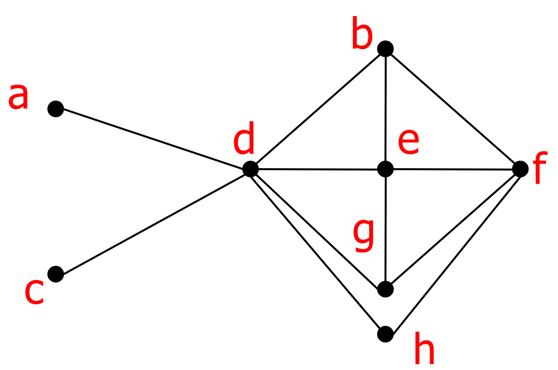
Alguns Problemas aparentemente difíceis

2. Problema do Conjunto Independente de Vértices

Dados: Um grafo *G* e um inteiro $k > 0$.

Decisão: *G* possui um conjunto independente de vértices de tamanho $\geq k$?

Dado um grafo *G*, um conjunto independente de vértices é um subconjunto $V' \leq VG$, tal que todo par de vértices de V' não é adjacente.



{a, c, b, g,h}

É um conjunto independente de vértices de tamanho 5

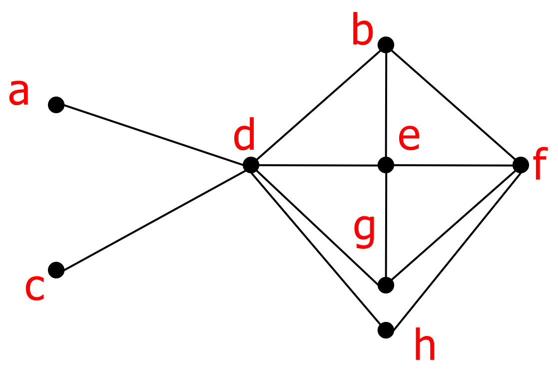
Alguns Problemas aparentemente difíceis

3. Problema da Clique

Dados: Um grafo *G* e um inteiro *k* > 0.

Decisão: *G* possui uma clique de tamanho ≥ *k?*

Exemplo:



{d, b, e}

clique de tamanho 3

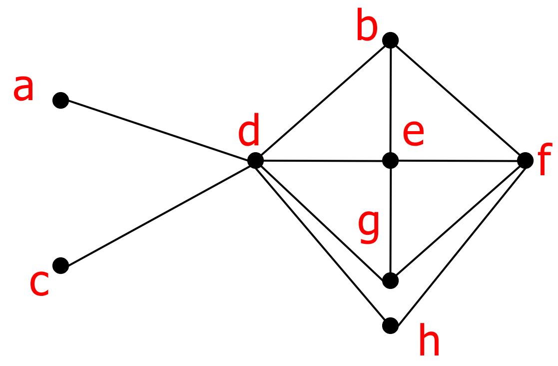
Alguns Problemas aparentemente difíceis

4. Problema da Cobertura de Vértices

Dados: Um grafo *G* e um inteiro $k > 0$.

Decisão: *G* possui uma cobertura de vértices de tamanho $\leq k$?

Para um grafo *G*, um subconjunto $V' \subseteq VG$ é chamado cobertura de vértices quando toda aresta de *G* possui (pelo menos) um de seus extremos em V'.



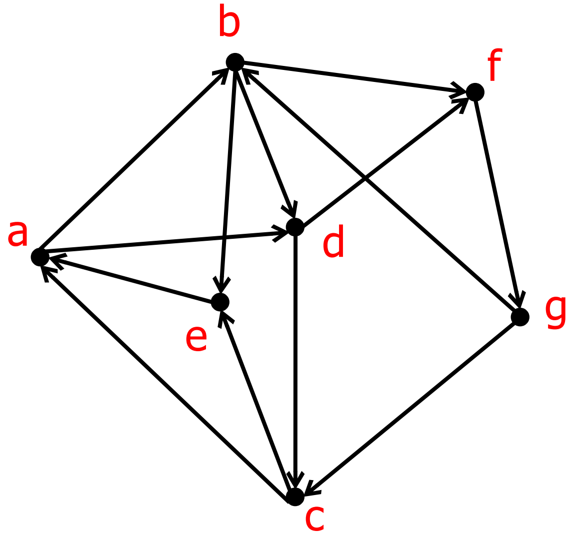
{d,f,e} é uma cobertura de vértices de tamanho 3

Alguns Problemas aparentemente difíceis

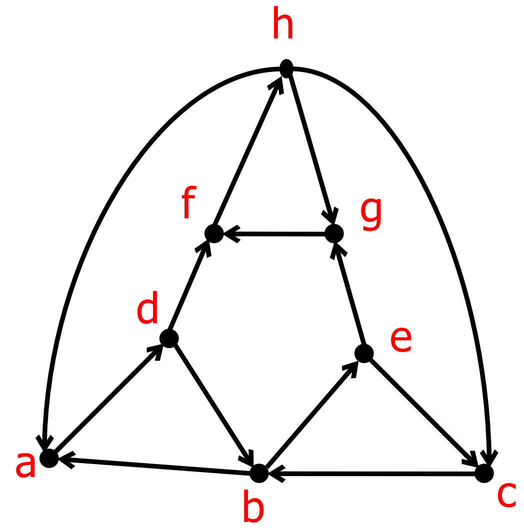
5. Problema do Ciclo Hamiltoniano Direcionado

Dados: Dígrafo *D*.

Decisão: *D* possui um ciclo hamiltoniano ?



{a, b, d, f, g, c, e, a}



Não tem

Alguns Problemas aparentemente difíceis

6. Problema do Ciclo Hamiltoniano não direcionado

Dados: grafo *G*.

Decisão: *G* possui um ciclo Hamiltoniano?

7. Problema de Coloração

Dados: Um grafo *G* e um inteiro $k > 0$.

Decisão: *G* possui uma coloração com um número $\leq k$ cores?

A Classe *NP*

Justificativa : Conjunto de argumentos que interpretados podem atestar a veracidade da resposta sim ou não dada ao problema.

Problema de Decisão π. Se π for solúvel através da aplicação de algum processo, então existe uma justificativa para a solução de π.

A Classe *NP*

- Exemplo:

1. Problema do Ciclo Hamiltoniano

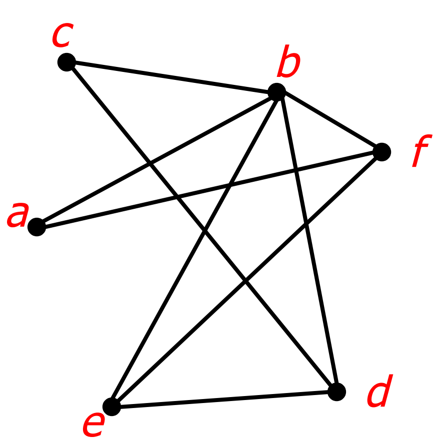
- justificativa do sim: exibição do ciclo *C* do grafo e reconhecimento que *C* é um ciclo Hamiltoniano.
- justificativa do não: listagem de todos os ciclos do grafo e reconhecimento de que nenhum deles é Hamiltoniano.

A Classe *NP*

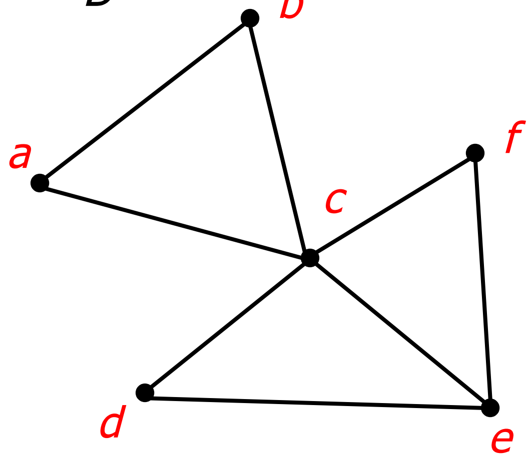
2. Problema do Ciclo Hamiltoniano

Ex.:

A



B



Justificativa SIM:

a, b, c, d, e, f, a

(passo da exibição)

Justificativa NÃO:

a, b, c, a

e, c, f, e

e, c, d, e

e, d, c, f, e

(passo da exibição)

A Classe *NP*

3. Problema da Clique

- justificativa SIM:

  exibição de uma clique P de tamanho $\geq k$.

  verificando para reconhecer

  (a) Se P é de fato uma clique.

  (b) $|P| \geq k$.

- justificativa NÃO:

  lista de todas as cliques do grafo.

  (a) Verificar se a lista é completa e de fato o tamanho de cada clique é $< k$.

A Classe *NP*

- Justificar respostas a problemas de decisão compreende duas fases distintas:
    - **Exibição**: consiste em exibir a justificativa.
    - **Reconhecimento**: consiste em verificar que a justificativa apresentada na fase de exibição é, de fato satisfatória.

A Classe NP

- Exemplo: Voltando ao problema do ciclo Hamiltoniano:

  - Justificativa SIM do grafo *A:*

    Exibição: sequência *C* de vértices *a*, *b*, *c*, *d*, *e*, *f*, *a*.

    Reconhecimento: Verificar:

    (i) *C* é ciclo.

    (ii) *C* contém cada vértice de *G* exatamente uma vez.

  - Observe que dado uma sequência de vértices, reconhecer se é um ciclo e se este ciclo é Hamiltoniano é possível ser realizado com um algoritmo polinomial.

A Classe NP

- Exemplo: Voltando ao problema do ciclo Hamiltoniano:

  - Justificativa NÃO do grafo *B:*

    Exibição: Conjunto de 4 sequências de vértices:

    {*a, b, c, a / e, c, f, e / e, c, d, e / e, d, c, f, e* }

    Reconhecimento: Comprovar que:

    (i) cada sequência de vértices é um ciclo não Hamiltoniano.

    (ii) todo ciclo do grafo está no conjunto.

  - Observe que o algoritmo não é mais tão simples. É de natureza experimental.
  - Não se conhece um algoritmo polinomial para se fazer o reconhecimento da justificativa não do problema

A Classe NP

- **Classe NP** : Compreende todos os problemas de decisão π, tais que existe uma justificativa à resposta SIM para π, cujo passo de reconhecimento pode ser realizado por um algoritmo polinomial do tamanho da entrada de π.

  (Isto não implica numa solução polinomial para o problema)

- Para existir um algoritmo de reconhecimento polinomial é necessário (mas não suficiente) que o tamanho da justificativa dada pelo passo de exibição seja polinomial no tamanho da entrada do problema.

A Classe NP

- Exemplo: Justificativa NÃO para o problema do ciclo Hamiltoniano.
    - Exibição da lista de todos os ciclos do grafo
      → Exponencial no tamanho do grafo
    - Qualquer algoritmo para checar uma entrada exponencial leva tempo exponencial de processamento (muito embora para cada item da entrada ele possa ser polinomial)

A Classe NP

- **Observação**: Nada se exige sobre a justificativa NÃO para enquadrar um problema na classe *NP*.

- Existem problemas *NP* que admitem algoritmos polinomiais para justificativa NÃO e outros que não se sabe se isso é possível.

- Exemplo: Ciclo Hamiltoniano é *NP*.

A Classe *NP*

- Para verificar se um problema π pertence ou não a *NP* procede-se da seguinte maneira:

  (i) Define-se uma justificativa J conveniente para a resposta SIM ao problema.

  (ii) Elabora-se um algoritmo para reconhecer se J está correta.

  Se algoritmo é polinomial $\pi \in NP$

A Classe *NP*

- Problemas anteriormente vistos são *NP* ?

1. Problema da Satisfabilidade

   Dados: Uma expressão Booleana *E* na forma **FNC**(Forma Normal Conjuntiva)

   Decisão: *E* é satisfatível ?

Justificativa SIM:

(i) Exibição : A expressão Booleana *E* e um atribuição para cada variável de *E*.

(ii) Reconhecimento:

Algoritmo : Substitui-se em *E* cada variável pelo seu valor atribuído (*V* ou *F*). É imediato concluir que a justificativa está correta se e só se cada cláusula de *E* possui pelo menos uma variável com atribuição *V*.

Conclusão $\in$ ***NP***

A Classe *NP*

2. Problema do Conjunto Independente de Vértices

Dados: Um grafo G e um inteiro $k > 0$.

Decisão: G possui um conjunto independente de vértices de tamanho $\geq k$?

Justificativa SIM:

(i) Exibição : O grafo G e um subconjunto de vértices $V' \subseteq VG$.

(ii) Reconhecimento:

Algoritmo : Examina-se cada lista de adjacências $Adj(v')$, $v' \in V'$, para verificar se todo $w \in Adj(v')$ é tal que $w \notin V'$. Seja agora $k' = |V'|$. A justificativa está correta se e só se essas verificações forem satisfeitas e além disso $k' \geq k$.

Conclusão: Algoritmo Polinomial $\Rightarrow$ problema $\in$ *NP*

A Classe *NP*

3. Problema da Cobertura de Vértices

   Dados: Um grafo G e um inteiro $k > 0$.

   Decisão: G possui uma cobertura de vértices de tamanho $\leq k$?

Justificativa SIM:

(i) Exibição : O grafo G e um subconjunto de vértices $V' \subseteq VG$.

(ii) Reconhecimento:

Algoritmo : Examina-se cada aresta $(v,w) \in EG$ com o intuito de verificar se v ou $w \in V'$. Seja agora $k' = |V'|$. A justificativa está correta se e só se as verificações forem todas satisfeitas e além disso $k' \leq k$?.

Conclusão: Algoritmo Polinomial $\Rightarrow$ problema $\in$ *NP*

A Classe *NP*

- Problema para o qual se desconhece a pertinência ou não a *NP*.
- Problema da Clique Máxima

Dados: Um grafo G e um inteiro $k > 0$.

Decisão: A clique de tamanho máximo de G tem tamanho k?

A Classe *NP*

Problema da Clique Máxima

Justificativa SIM:

(i) Exibição : Apresentação de um conjunto S contendo todas as cliques maximais de G.

(ii) Reconhecimento:

Algoritmo : Comprova-se que S tem de fato todas as cliques maximais de G. Seja k' o tamanho da maior clique de S. A justificativa está correta se $k = k'$.

S pode ser exponencial $\Rightarrow$ algoritmo exponencial.

$\Rightarrow$ Nada se pode afirmar sobre a pertinência ou não do problema à classe *NP*.

A Questão $P = NP$

- Relação entre classes P e NP.

Lema

$$P \subseteq NP$$

A Questão $P = NP$

Prova

Seja $\pi \in P$ um problema de decisão. Então existe um algoritmo α que apresenta a solução de π, em tempo polinomial no tamanho de sua entrada. Em particular, α pode ser utilizado como algoritmo no reconhecimento para uma justificativa à resposta SIM de P. Logo $\pi \in NP$.

A Questão $P = NP$

- $P \neq NP$?

  $\Rightarrow$ Existe algum problema da classe *NP* que é intratável ?

  $\Rightarrow$ Todo problema de *NP* admite necessariamente algoritmo polinomial?

- Evidência $\rightarrow P \neq NP$

A Questão *P = NP*

- Outra questão:

  Admitindo-se que $P \neq NP$ seria possível ao menos resolver em tempo exponencial todo problema da classe *NP* ?

Lema 7.2 do Szwarcfiter $\Rightarrow$ prova SIM

com complexidade $O(\,|A|^k\, C_d)$,

onde k é o tamanho de π

Complemento de Problemas

- Classe *NP*
  - Exige que justificativa SIM seja reconhecida em tempo polinomial
  - Nada se exige sobre a justificativa NÃO

- Inversão de papéis $\Rightarrow$ nova classe de problemas

- **A classe *Co-NP*** $\Rightarrow$ Compreende todos os problemas de decisão π, tais que existe uma justificativa à resposta NÃO, cujo passo de reconhecimento corresponde a um algoritmo polinomial na entrada de π.

Complemento de Problemas

- Complemento $\bar{\pi}$ de um problema de decisão π.

- A resposta ao problema π é SIM se e somente se a resposta para $\bar{\pi}$ for NÃO.

- Classe *Co-NP* $\Rightarrow$ Compreende exatamente os complementos dos problemas da classe *NP*

  - $P \subseteq Co\text{-}NP$
  - Se $\pi \in P \Rightarrow \pi \in NP \cap Co\text{-}NP$

Complemento de Problemas

- Existem problemas $\pi \in NP$ para os quais não se sabe se $\bar{\pi} \in NP$. Analogamente para *Co-NP*.

Exemplo : $\overline{\text{O problema da Clique}}$ (Já vimos que Clique $\in NP$)

Dados: Um grafo G e um inteiro $k > 0$.

Decisão: G não possui uma clique de tamanho $\geq k$?

Algoritmo que reconhece em tempo polinomial a justificativa SIM de $\overline{\text{Clique}}$?

$\hookrightarrow$ Equivale a algoritmo polinomial para reconhecer a justificativa NÃO de clique $\rightarrow$ **Desconhecido**

Complemento de Problemas

- Conclusão: Não se sabe se $\overline{\text{Clique}}$ $\in$ *NP.*

- Clique Máxima: Desconhece-se se $\pi \in NP$ e também se $\overline{\pi} \in Co\text{-}NP$ e $\overline{\pi} \in NP$.

- Existem problemas tais que π e $\overline{\pi} \in$ *a NP.*

Complemento de Problemas

- Problema dos Numeros Compostos

  Dados: Um inteiro $k > 0$.

  Decisão: Existem inteiros $p,q > 1$ tais que $k = pq$?

- Problema dos $\overline{\text{Numeros Compostos}}$ (Números primos)

  Dados: Um inteiro $k > 0$.

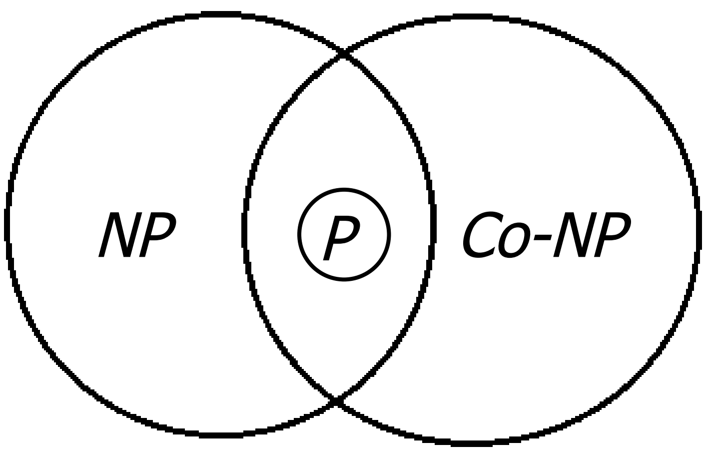
  Decisão: k é primo ?

  Algoritmo polinomial para reconhecer o SIM existe mas não é trivial.

  Conclusão: Números Compostos e Números primos pertecem ambos a *NP*. Mas não se sabe se $\in$ ou não a *P*.

Questões ainda não resolvidas

- $NP = Co\text{-}NP$?
- $P = NP \cap Co\text{-}NP$?
- Conjectura-se:



$P \neq NP$
$NP \neq Co\text{-}NP$
$P \neq NP \cap Co\text{-}NP$

Transformações Polinomiais

- $\pi_1(D_1,Q_1)$ e $\pi_2(D_2,Q_2)$ problemas de decisão

$\Uparrow$ resolve

Algoritmo A_2

Se for possível transformar o problema π_1 em π_2 e sendo conhecido um processo de transformar a solução de π_2 numa solução de π_1 , então o algoritmo A_2 pode ser usado para resolver o problema π_1.

Transformações Polinomiais

- Instância $I_1 \in D_1 \xrightarrow{T'}$ Instância $I_2 \in D_2 \xrightarrow{A_2}$
  Solução de $\pi_2 \xrightarrow{T''}$ Solução de π_1

Se ambos T' e T'' forem polinomiais, então diz-se que existe uma transformação polinomial de π_1 *em* π_2 , e que π_1 é polinomialmente transformável em π_2.

Transformações Polinomiais

- Formalmente, uma transformação polinomial de um problema de decisão $\pi_1(D_1,Q_1)$ no problema de decisão $\pi_2(D_2,Q_2)$ é uma função f: $\Delta_1 \rightarrow \Delta_2$ tal que valham:

  (i) *T* pode ser computada em tempo polinomial

  (ii) para toda instância $I \in D_1$ do problema π_1 tem-se: π_1 (I) possui resposta SIM se e somente se $\pi_2(f(I))$ também possuir.

Transformações Polinomiais

- Transformações polinomiais são operações importantes. Por que ?
  - Preservam a natureza (polinomial ou não) do algoritmo A_2 para π_2, quando utilizado para resolver π_1.

    ⇓

    A_2 for polinomial transformação polinomial de π_1 em π_2 > π_1 pode ser resolvido em tempo polinomial

  - Notação : $\pi_1 \;\alpha\; \pi_2$ (π_1 pode ser transformado polinomialmente em π_2)

Transformações Polinomiais

- Observação: α é transitiva, ou seja:

$$\boxed{\pi_1 \; \alpha \; \pi_2 \quad e \quad \pi_2 \; \alpha \; \pi_3 \quad \Rightarrow \quad \pi_1 \; \alpha \; \pi_3}$$

Exemplo: $\pi_1 \rightarrow$ Clique $\quad \pi_2 \rightarrow$ Conj. Indepedente de Vértices

$I_1 \rightarrow$ grafo G $\quad I_2\ f(I_1) \rightarrow$ Complemento $\bar{G}$ de G

inteiro $k > 0$ $\quad$ mesmo inteiro k

f é polinomial porque:

(i) $\bar{G}$ pode ser obtido a partir de G em tempo polinomial.

(ii) G possui uma clique de tamanho ≥ k se e somente se $\bar{G}$ possui um conjunto independente de vértices de tamanho ≥ k.

Transformações Polinomiais

- Conclusão:

Se existir A2 que resolva o problema do conjunto independente de vértices em tempo polinomial, este algoritmo pode ser utilizado para resolver também o problema da clique em tempo polinomial.

Clique α Conjunto Independete de Vértices

Transformações Polinomiais

Se $\pi 1 \ \alpha \ \pi 2$ *e* $\pi 2 \ \alpha \ \pi 1 \Rightarrow \pi 1$ e $\pi 2$ são equivalentes $\Rightarrow$
idêntica dificuldade.

- Podemos utilizar a relação α para dividir *NP* em classes de problemas equivalentes entre si.
- Problemas pertencentes a *P* foram uma dessas classes: a classe de mínima dificuldade.

Transformações Polinomiais

- Temos outra classe de problemas equivalentes entre si na classe *NP* que são os de "maior dificuldade" entre todos em *NP.* São denominados ***NP-completos***.

- Um problema de decisão π é denominado ***NP-completo*** quando as seguintes condições forem satisfeitas:

  (i) $\pi \in NP$.

  (ii) todo problema de decisão $\pi' \in NP$ satisfaz $\pi' \,\alpha\, \pi$.

  => todo problema da classe *NP* pode ser transformável polinomialmente no π' *NP-completo*.

  → Se um problem π , *NP-completo,* puder ser resolvido em tempo polinomial **TODO** problema de *NP* admite algoritmos polinomial $\Rightarrow$ ***P = NP***

Transformações Polinomiais

- Caso somente a condição (ii) de NP-Completo seja satisfeita, não importante a satisfação da condição (i), o problema é denomidado NP-Difícil
- Problemas tão difíceis quantos os problemas mais difíceis da classe NP.

Alguns problemas *NP-completo*

- Passo de classificação de problema *NP*.
- Definição de *NP-completo* (disponível).
- Identificação $\Rightarrow$ aplicação da definição

⇩

  - Problemas: Todo problema *NP* precisaria ser transformado polinomialmente a π (problema de decisão a ser classificado)

Alguns problemas *NP-completo*

Lema

Sejam π_1 *e* π_2 problemas de decisão $\in$ *NP*, se π_1 é *NP-completo* e $\pi_1 \, \alpha \, \pi_2$ então π_2 é *NP-completo.*

Prova

Como $\pi_2 \in NP$, para mostrar que π_2 é *NP-completo* basta provar que (ii) vale, ou seja, que $\pi' \, \alpha \, \pi_2 \; \forall \pi' \in NP$. Como π_1 é *NP-completo* , então necessariamente $\pi' \, \alpha \, \pi_1 \; \forall \pi' \in NP$. Como $\pi_1 \, \alpha \, \pi_2$, por transitividade temos que $\pi' \, \alpha \, \pi_2 \; \forall \pi' \in NP$ $\square$

Alguns problemas *NP-completo*

- Lema simples mas **poderoso**.

- Agora para provar que um problema π é *NP-completo* basta provar que:

  (i) $\pi \in NP$ e

  (ii) um problema π', *NP-completo,* é tal $\pi' \,\alpha\, \pi$.

- Para aplicar o lema acima é preciso conhecer um problema π' *NP-completo*. Mas e para o primeiro ?
  - Neste caso aplica-se a definição.

Alguns problemas *NP-completo*

- Em 1971, Cook provou o problema da satisfabilidade a definição.

Teorema de Cook

O problema da satisfabilidade é *NP-completo*

- A partir dai vários problemas foram provados.
- Karp em 1972 efetivou *24* poblemas na classe *NP-completo.*
- Hoje temos centenas deles.
- Clique, conjunto independente de vértices

Referências

- Capítulo 7 do Szwarcfiter, J. L., *Grafos e Algoritmos Computacionais,* Ed. Campus, 1983.

- Capítulo 34 do Cormen, *Introduction to Algorithms,* MIT Press, 2001.