Problemas NP-completo

Profa.: Loana T. Nogueira

November 30, 2011

Introdução

Considere o problema de avaliar satisfatoriamente a eficiência de algoritmos:

Até o momento, utilizamos a complexidade como medida de eficiência. Contudo...

- uma vez de posse dessa complexidade de que forma reconhecer se o algoritmo correspondente é ou não eficiente?

Até o momento, utilizamos a complexidade como medida de eficiência. Contudo...

- uma vez de posse dessa complexidade de que forma reconhecer se o algoritmo correspondente é ou não eficiente?
- Um algoritmo é eficiente precisamente quando a sua complexidade for um polinômio no tamanho de sua entrada.

Até o momento, utilizamos a complexidade como medida de eficiência. Contudo...

- uma vez de posse dessa complexidade de que forma reconhecer se o algoritmo correspondente é ou não eficiente?
- Um algoritmo é eficiente precisamente quando a sua complexidade for um polinômio no tamanho de sua entrada.
- Considere a coleção de todos os algoritmos que resolvem um certo problema P. O interessante é conhecer se nessa coleção existe algum que seja eficiente, isto é, de complexidade polinomial.

Até o momento, utilizamos a complexidade como medida de eficiência. Contudo...

- uma vez de posse dessa complexidade de que forma reconhecer se o algoritmo correspondente é ou não eficiente?
- Um algoritmo é eficiente precisamente quando a sua complexidade for um polinômio no tamanho de sua entrada.
- Considere a coleção de todos os algoritmos que resolvem um certo problema P. O interessante é conhecer se nessa coleção existe algum que seja eficiente, isto é, de complexidade polinomial.
- Se existir tal algoritmo, o problema P será denominado *tratável*, e *intratável* caso contrário.

Tratável X Intratável

- De acordo com a definição, um problema seria classificado como tratável, exibindo-se algum algoritmo de complexidade polinomial, que o resolvesse.

Tratável X Intratável

- De acordo com a definição, um problema seria classificado como tratável, exibindo-se algum algoritmo de complexidade polinomial, que o resolvesse.
- Para verificar que é intratável, há necessidade de provar que todo possível algoritmo que o resolva não possui complexidade polinomial.

Problemas Algorítmico

Um problema algorítmico pode ser caracterizado por um conjunto de todos os possíveis dados do problema, denominado conjunto de dados e por uma questão solicitada, denominada objetivo do problema.

- Em que consiste resolver o problema algorítmico?
- Consiste em desenvolver um algoritmo cuja entrada são dados específicos retirados desse conjunto (instância) e cuja saída, denominada solução, responda ao objetivo do problema.
- Exemplo: Elaborar um algoritmo para encontrar uma clique de tamanho $\geq k$ num grafo G dado.

DADOS: Um grafo G e um inteiro $k > 0$
OBJETIVO: Encontrar em G uma clique de tamanho $\geq k$, se existir.

Classificação dos Problemas Algorítmicos

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

Classificação dos Problemas Algorítmicos

- Problemas de Decisão
  - Dados: Um grafo G e um inteiro $k > 0$.
  - Questão: G possui clique de tamanho $\geq k$?
- Problemas de Localização
  - Dados: Um grafo G e um inteiro $k > 0$.
  - Questão: Encontrar em G uma clique de tamanho $\geq k$, caso exista.

Classificação dos Problemas Algorítmicos

- Problemas de Decisão
  - Dados: Um grafo G e um inteiro $k > 0$.
  - Questão: G possui clique de tamanho $\geq k$?
- Problemas de Localização
  - Dados: Um grafo G e um inteiro $k > 0$.
  - Questão: Encontrar em G uma clique de tamanho $\geq k$, caso exista.
- Problemas de Otimização
  - Dados: Um grafo G e um inteiro $k > 0$.
  - Questão: Encontrar em G a clique de tamanho máximo

Problemas de Decisão, Localização e otimização

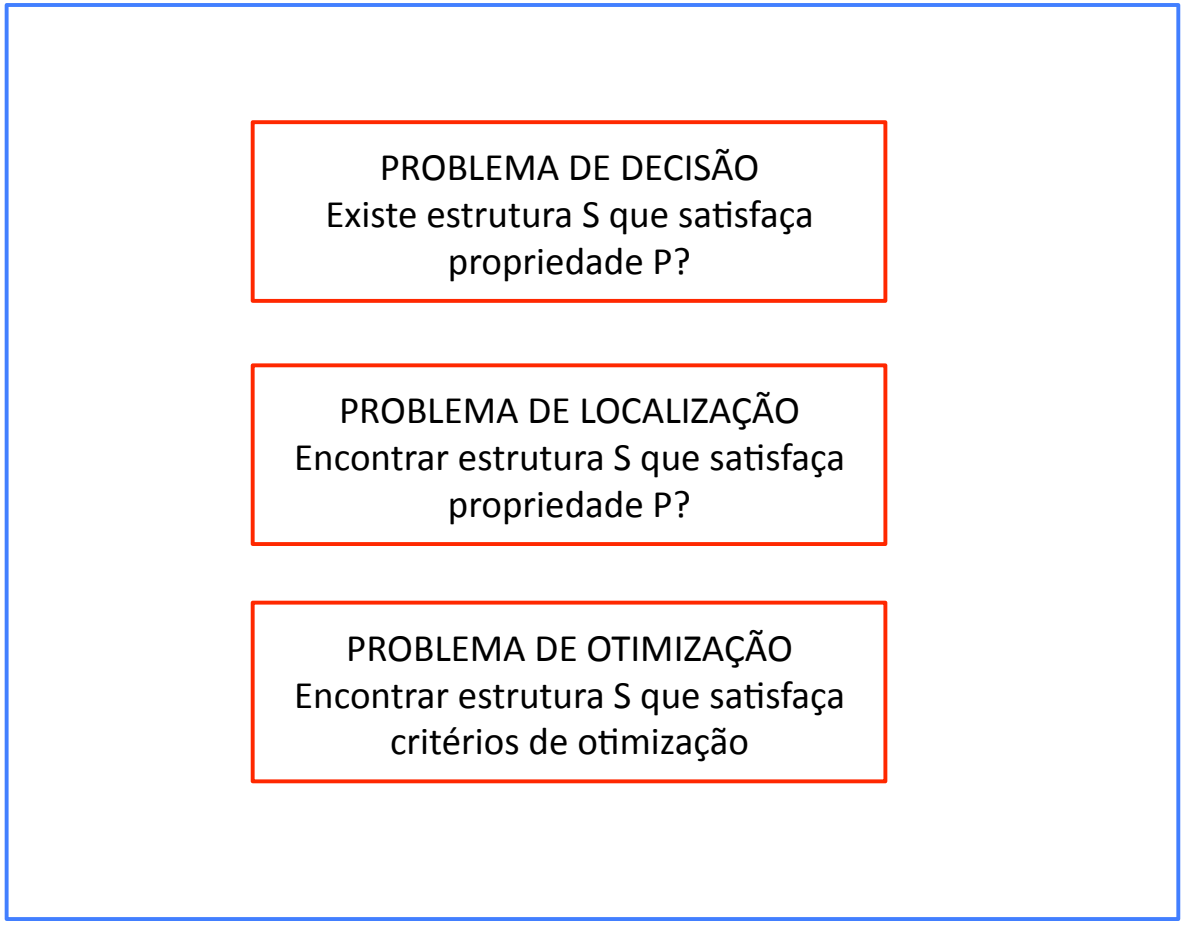


Figure: Problemas Algorítmicos

A Classe P

Um algoritmo eficiente é aquele cuja complexidade é uma função polinomial nos tamanhos dos dados de entrada.

Observe que para um problema de decisão, o tamanho da saída é constante, o que possibilita ignorá-lo.

Complexidades de algoritmos eficientes:

1. $O(1)$;
2. $O(n)$;
3. $O(n^2 logn)$;
4. $O(n^{10})$.

seriam todos classificiados como eficientes.

Por outro lado...

1. $O(2^n)$;
2. $O(n!)$;

corresponderiam a algoritmos não eficientes.

A Classe P

Definição:

A classe P de problemas de decisão é aquela que compreende precisamente aqueles que admitem algoritmo polinomial.

Exemplo:

O problema de buscar um elemento x numa lista não ordenada L com n elementos pertence à classe P. Por quê?

Observação:

Ainda que todos os algoritmos conhecidos para um certo problema π sejam todos exponenciais, não necessariamente $\pi \notin P$. Se de fato $\pi \notin P$ então deve existir alguma provade que todo possível algoritmo para resolver π não é polinomial.

Alguns problemas aparentemente difíceis

PROBLEMA 1: SATISFABILIDADE
DADOS: Uma expressão booleana E na forma FNC
DECISÃO: E é satisfatível?

Satisfabilidade

Considere um conjunto de variáveis booleanas: $x_1, x_2, x_3 \ldots$ e denote por $\bar{x}_1$, $\bar{x}_2$, $\bar{x}_3$,... seus complementos.

Isto é, cada variável x_i assume um valor *verdadeiro* (V) ou *falso* (F) e x_i é verdadeiro $\iff$ $\bar{x}_i$ é falso.

Os símbolos x_1, $\bar{x}_1$, $x_2, \bar{x}_2$, $x_3, \bar{x}_3$,... são denominados *literais*.

Denote por $\wedge$: Conjunção e $\vee$: Disjunção

Uma *cláusula* é uma disjunção de literais.

Exemplo: $\bar{x}_2 \vee x_3 \vee \bar{x}_4 \vee x_1$ é uma cláusula.

Expressão Booleana

Uma *expressão booleana* é uma expressão cujos operandos são literais e cujos operadores são conjunções ou disjunções.

Uma expressão booleana é dita na *forma normal conjuntiva* (FNC) quando for uma conjunção de cláusulas.

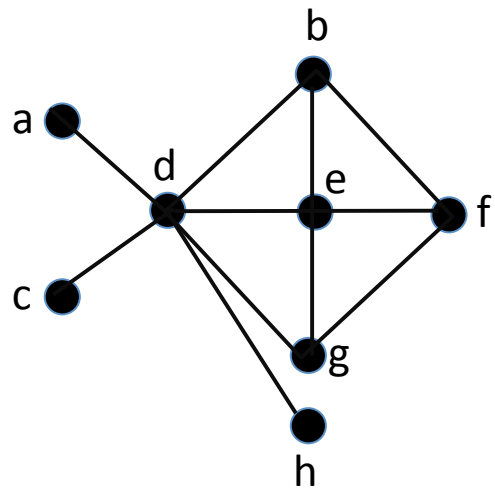
Exemplo: $(x_2 \vee \bar{x_1}) \wedge (\bar{x_1} \vee \bar{x_3} \vee x_2) \wedge (x_1 \vee \bar{x_3} \vee x_2)$

Uma expressão boolena é dita *satisfatível* se existe uma atribuição de valores, verdadeiro ou falso, às variáveis de tal modo que o valor da expressão seja verdadeira.

O **Problema de satisfabilidade** consiste em, da uma expressão booleana na FNC, verificar se ela é satisfatível.

Alguns problemas aparentemente difíceis

PROBLEMA 2: CONJUNTO INDEPENDENTE DE VÉRTICE
DADOS: Um grafo G e um inteiro $k > 0$
DECISÃO: G possui um conjunto indepenente de vértcies de tamanho $\geq k$?



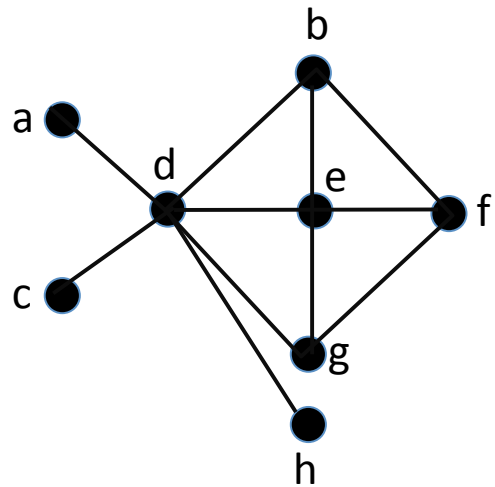
Conjunto independente

Dado um grafo $G(V, E)$, um *conjunto independente de vértices* é um subconjunto $V' \subseteq V$ tal que todo par de vértices de V' não é adjacente. Isto é, se $v, w \in V'$ então $(v, w) \notin E$.

Exemplo: $\{a, b, c, g, h\}$ é um conjunto independente de cardinalidade 5.

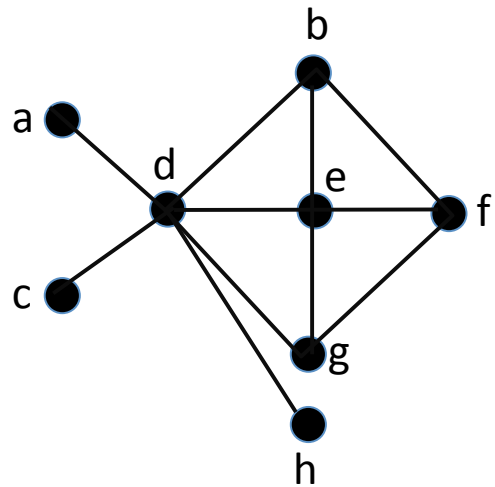
Alguns problemas aparentemente difíceis

PROBLEMA 3: CLIQUE
DADOS: Um grafo G e um inteiro $k > 0$
DECISÃO: G possui uma clique de tamanho $\geq k$?



Alguns problemas aparentemente difíceis

PROBLEMA 3: COBERTURA DE VÉRTICES
DADOS: Um grafo G e um inteiro $k > 0$
DECISÃO: G possui uma cobertura de vértices de tamanho $\leq k$?



Cobertura de Vértices

Dado um grafo $G(V, E)$, uma *cobertura de vértices* é um subconjunto $V' \subseteq V$ tal que toda toda aresta de G possui pelo menos um extremo em V'.

Exemplo: $\{d, f, e\}$ é uma cobertura de vértices de tamanho 3.

A Classe NP

Considere um problema de decisão π. Se π for solúvel através de algum processo, então necessariamente existe uma *justificativa* para a solução de π. Essa justificativa pode ser representada por um determinado conjunto de argumentos, os quais, quando interpretados convenientemente, podem atestar a veracidade da resposta SIM ou NÃO dada ao problema.

Exemplo: A solução do problema de verificar se um dado grafo tem ou não uma árvore geradora de tamanho $\leq p$ pode ser obtida através do algoritmo de árvore geradora mínima. A justificativa é a própria corretudo do algoritmo.

A Classe NP

Considere agora o problema CLIQUE, onde o objetivo é decidir se um grafo G possui ou não clique de tamanho $\geq k$.

Uma justificativa para a resposta SIM pode ser obtida exibindo-se um conjunto de vértices de cardinalidade k e verificando que este é de fato uma clique. Uma justificativa para a resposta NÃO pode ser representada, por exemplo, listando todos os conjuntos de cardinalidade k e verificando que nenhum deles é uma clique.

Processo

Observe que o processo descrito anteriormente para justificar resposta a problemas de decisão se compõe de duas fases distintas:

FASE 1: EXIBIÇÃO

COnsiste em exibir a justificativa

FASE 2: RECONHECIMENTO

Consiste em verificar que a justificativa apresentada é, de tafo, satisfatória.

A classe *NP*

Define-se então a classe *NP* como sendo aquela que 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 plinomial no tamanho da entrada de π.

Obs.: Não se exige uma solução polinomial para π.

Reconhecendo um problema *NP*

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

- Define-se uma justificativa J conveniente para a resposta SIM ao problema;
- Elabora-se um algoritmo para reconhecer se J está correta. A entrada desse algoritmo consite de J e da entrada de π.

Se o algoritmo resultante do passo 2 for polinomial no tamanho da entrada de π, então π pertence a *NP*. O caso contrário, naturalmente, não implica necessariamente em não pertinência a *NP*.

Exemplos de Problemas *NP*: Satisfabilidade, Clique, Conjunto Independente, Cobertura de vértices.

A Questão $P = NP$

O seguinte lema é imediato:

LEMA: $P \subseteq NP$.

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

...Pergunta natural:

$$P = NP?$$

Transformações polinomiais

Sejam $\pi_1(D_1, Q_1)$ e $\pi_2(D_2, Q_2)$ problemas de decisão. Suponha que seja conhecido um algoritmo A_2 que resolva π_2. Se for possível trasformar 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 algoritmo A_2 pode ser utilizado para resolver o problema π_1.

Se a transformação de π_1 em π_2, bem como a da solução de π_2 para a de π_1 puder ser realizada em tempo polonimial, então diz-se que existe uma *transformação polinomial* de π_1 em π_2, e que π_1 é *polinomialmente transformável(reduzido)* em π_2.

Transformações polinomiais

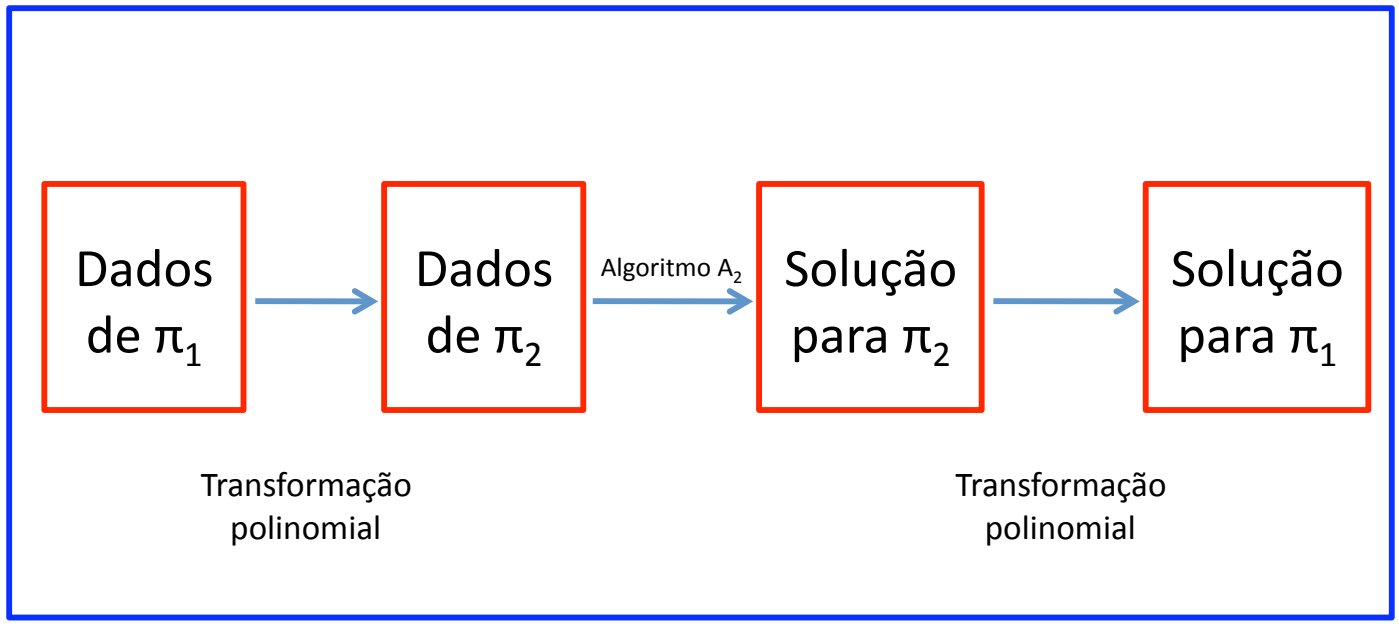


Figure: Transformação polinomial de π_1 para π_2

Exemplo de Transformação Polinomial

Considere como π_1 e π_2, respectivamente, os problemas CLIQUE e CONJUNTO INDEPENDENTE. A instância I de CLIQUE consiste de um grafo G e um inteiro $k \geq 0$. Para obter a instância $f(I)$ de CONJUNTO INDEPENDENTE, considera-se o grafo complementar $\overline{G}$ of G e o mesmo inteiro k. É então evidente que f é uma transformação plinomial porque:

- $\overline{G}$ pode ser obtido a partir de G, em tempo polinomial, e
- G possui uma clique de tamanho $\geq k$ se e somente se $\overline{G}$ possui um conjunto independente de vértices de tamanho $\geq k$.

Se existir um algoritmo A_2 que resolva CI em tempo polinomial, este algoritmo pode ser usado para resolver clique também tem tempo polinomial.

Problemas Equivalentes

Dados dois problemas π_1 e π_2. Se $\pi_1 \alpha \pi_2$ e $\pi_2 \alpha \pi_1$ então π_1 e π_2 são *equivalentes*.

NP-completo

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

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

Observe que (*ii*) implica que todo problema da classe *NP* pode ser transformado polinomialmente no problema π *NP*-completo. Isto é, se um problema *NP*-completo pode ser resolvido em tempo polinomial entaão todo problema de *NP* admite também algoritmo polinomial (e consequentemente, $N = NP$).

Caso somente a condição (*ii*) da definição de *NP*-completo seja considerada, o problema π é denominado *NP*-difícil.

Alguns problemas *NP*-Completo

- Como identificar problemas que pertençam a esta classe?
- DIFICULDADE?
- Condição (ii) - POR QUÊ?

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 também NP-completo.

Este lema é muito poderoso. Para provar que um certo problema π é NP-completo basta demonstrar que um problema NP-completo é polinomialmente transformável em π. Ou seja, é suficiente provar que:

- $\pi \in NP$
- um problema NP-completo π' é tal que $\pi' \alpha \pi$.

PROBLEMA?

Teorema: SATISFABILIDADE é *NP*-completo

Lema: O problema CLIQUE é *NP*-completo.

PROVA:
Dados de clique: Grafo G e inteiro positivo k. Seja C uma lcique do grafo . Pode-se reconhecer se C é uma clique e computar seu tamanho em tempo polinomial no tamanho da entrada de CLIQUE. Logo, CLIQUE $\in NP$.
Vamos transformar SATISFABILIDADE em CLIQUE:

A questão E é satisfatível será transformada no problema de verificar se um grafo G tem uma clique de tamanho $\geq p$.

Cria-se um vértice novo em G para cada ocorrência de literal em E. Existe uma aresta diferente (v_i, v_j) em G, para cada par de literais x_i, x_j de E tais que $x_i \neq \overline{x_j}$ e x_i, x_j ocorrem em cláusulas diferentes de E. Como decorrência...