UNIFAL-MG
Universidade Federal de Alfenas

Projeto e Análise de Algoritmos

NP Completude

Prof. Humberto Brandão
humberto@bcc.unifal-mg.edu.br

Universidade Federal de Alfenas
versão da aula: 0.4

Introdução

UNIFAL-MG
Universidade Federal de Alfenas

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

Introdução

UNIFAL-MG
Universidade Federal de Alfenas

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

- Problemas "fáceis":
  - resolvidos por algoritmos polinomiais;

Introdução

UNIFAL-MG
Universidade Federal de Alfenas

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

- Problemas **"fáceis"**:
  - resolvidos por **algoritmos polinomiais**;

- Problemas "difíceis":
  - no momento, conhecemos apenas algoritmos exponenciais para resolvê-los;

Introdução

UNIFAL-MG
Universidade Federal de Alfenas

- **Polinomial**:
  - função de complexidade é $O(p(n))$, onde $p(n)$ é um polinômio.

Introdução

UNIFAL-MG
Universidade Federal de Alfenas

- **Polinomial**:
  - função de complexidade é $O(p(n))$, onde $p(n)$ é um polinômio.
  - Por exemplo:
    - pesquisa binária $(O(\log n))$;
    - pesquisa seqüencial $(O(n))$;
    - ordenação por inserção $(O(n^2))$;
    - multiplicação de matrizes $(O(n^3))$;

Introdução

UNIFAL-MG
Universidade Federal de Alfenas

- **Exponencial**:
  - função de complexidade é $O(c^n)$, $c > 1$;

Introdução

UNIFAL-MG
Universidade Federal de Alfenas

- **Exponencial**:
  - função de complexidade é $O(c^n)$, $c > 1$;
  - Por exemplo:
    - **Problema do caixeiro viajante** (PCV);
    - Problema de localização de Facilidades;
    - Problema de Mochila;
    - Problema de Roteamento de Veículos.

Introdução

UNIFAL-MG
Universidade Federal de Alfenas

- **Exponencial**:
  - função de complexidade é $O(c^n)$, $c > 1$;
  - Por exemplo:
    - **Problema do caixeiro viajante** (PCV);
    - Problema de localização de Facilidades;
    - Problema de Mochila;
    - Problema de Roteamento de Veículos.
  - Mesmo problemas de pequeno e médio porte não podem ser resolvidos por algoritmos não-polinomiais.

Classe de Problemas: NP-Completo (Introdução)

UNIFAL-MG
Universidade Federal de Alfenas

- A teoria de complexidade a ser apresentada não mostra como obter algoritmos polinomiais para problemas que demandam algoritmos exponenciais, nem afirma que não existem;

Classe de Problemas: NP-Completo (Introdução)

UNIFAL-MG
Universidade Federal de Alfenas

- 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**.

Classe de Problemas: NP-Completo (Introdução)

UNIFAL-MG
Universidade Federal de Alfenas

- Estes problemas formam a classe conhecida como NP-Completo;

Classe de Problemas: NP-Completo (Introdução)

UNIFAL-MG
Universidade Federal de Alfenas

- Estes problemas formam a classe conhecida como NP-Completo;

- Propriedade:
  - *um problema da classe NP-Completo poderá ser resolvido em tempo polinomial se e somente se todos os outros problemas em NP também puderem;*

Classe de Problemas: NP-Completo (Introdução)

UNIFAL-MG
Universidade Federal de Alfenas

- Estes problemas formam a classe conhecida como NP-Completo;

- Propriedade:
  - *um problema da classe NP-Completo 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-Completo.

NP-Completo e os Problemas de Decisão

UNIFAL-MG
Universidade Federal de Alfenas

- Para o estudo teórico da complexidade de algoritmos considera-se problemas cujo resultado da computação seja "sim" ou "não";

NP-Completo e os Problemas de Decisão

UNIFAL-MG
Universidade Federal de Alfenas

- Para o estudo teórico da complexidade de algoritmos considera-se problemas cujo resultado da computação seja "sim" ou "não";

- Versão do Problema do Caixeiro Viajante (PCV) cujo resultado é do tipo "sim/não":
  - Dados: uma constante *k*, um conjunto de cidades $C = \{c_1, c_2, ..., c_n\}$ e uma distância $d(c_i, c_j)$ para cada par de cidades c_i, c_j pertencente a C.
  - Questão:
    - Existe um "roteiro" para todas as cidades em C cujo comprimento total seja menor ou igual a k?

NP-Completo e os Problemas de Decisão

UNIFAL-MG
Universidade Federal de Alfenas

- Característica fundamental da classe NP-Completo: problemas "sim/não" para os quais uma dada solução pode ser verificada facilmente.

NP-Completo e os Problemas de Decisão

UNIFAL-MG
Universidade Federal de Alfenas

- **Característica** fundamental da classe **NP-Completo**: problemas "sim/não" para os quais uma **dada solução pode ser verificada facilmente**.

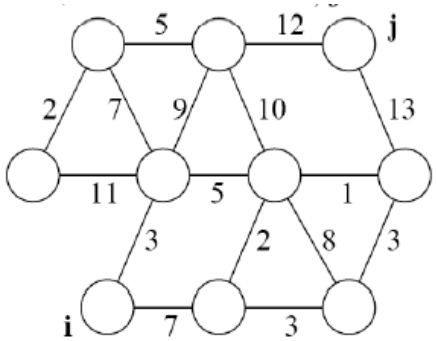
- *A solução pode ser muito difícil de ser obtida, mas uma vez conhecida ela pode ser verificada em tempo polinomial.*

UNIFAL-MG
Universidade Federal de Alfenas

Problemas
Fáceis vs. Difíceis

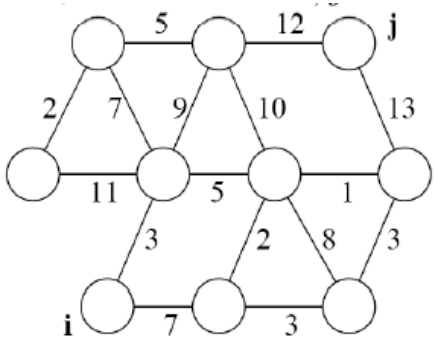
Fácil vs. Difícil

- Considere um grafo valorado, dois vértices *i, j* e um inteiro $k > 0$.



Fácil vs. Difícil

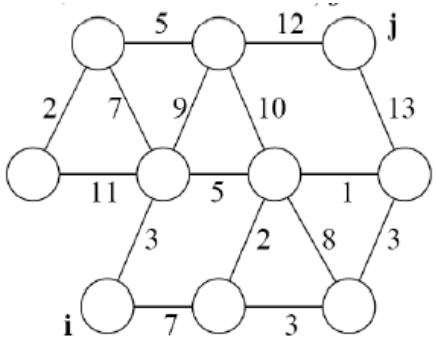
- Considere um grafo valorado, dois vértices *i*, *j* e um inteiro *k* > *0*.



- 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);

Fácil vs. Difícil

- Considere um grafo valorado, dois vértices *i*, *j* e um inteiro *k* > *0*.

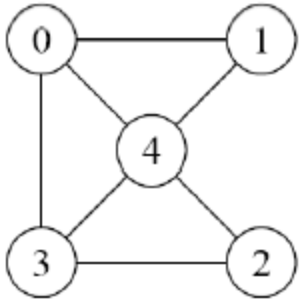


- Fácil:
  - Existe um caminho de *i* até *j* com peso <= *k*?
  - Há um algoritmo eficiente com complexidade de tempo O(E log V), sendo E o número de arestas e V o número de vértices (algoritmo de Dijkstra);

- Difícil:
  - Existe um caminho de i até j com peso >= k?
  - Não existe algoritmo eficiente. É equivalente ao PCV em termos de complexidade. É preciso enumerar todas as possibilidades.

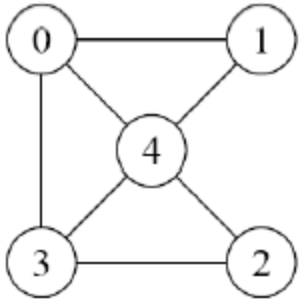
Fácil vs. Difícil

- Caminho que passa por todos os vértices uma única vez e retorna ao vértice inicial;
- Exemplo de circuito Hamiltoniano: 0 1 4 2 3 0.



Fácil vs. Difícil

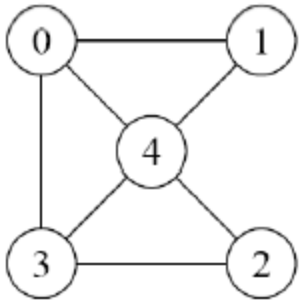
- Caminho que passa por todos os vértices uma única vez e retorna ao vértice inicial;
- Exemplo de circuito Hamiltoniano: 0 1 4 2 3 0.



- Existe um ciclo de Hamilton no grafo G?
  - Fácil: Grafo onde cada vértice tem grau máximo = 2 (vértices com no máximo duas arestas incidentes);

Fácil vs. Difícil

- Caminho que passa por todos os vértices uma única vez e retorna ao vértice inicial;
- Exemplo de circuito Hamiltoniano: 0 1 4 2 3 0.



- Existe um ciclo de Hamilton no grafo G?
  - **Fácil:** Grafo onde cada vértice tem grau máximo = 2 (vértices com no máximo duas arestas incidentes);
  - **Difícil:** Grafo onde os vértices têm grau > 2.
    - É um caso especial do PCV: Pares de vértices com uma aresta entre eles têm distância 1 e pares de vértices sem aresta entre eles têm distância infinita.

UNIFAL-MG
Universidade Federal de Alfenas

Algoritmos Não-Determinísticos

Algoritmos Não-Determinísticos

UNIFAL-MG
Universidade Federal de Alfenas

- Antes de falar de *não-determinismo,* vamos definir os Algoritmos determinísticos:

Algoritmos Não-Determinísticos

UNIFAL-MG
Universidade Federal de Alfenas

- Antes de falar de *não-determinismo*, vamos definir os Algoritmos determinísticos:
  - Algoritmos Determinísticos: o resultado de cada operação é definido de forma única;

Algoritmos Não-Determinísticos

UNIFAL-MG
Universidade Federal de Alfenas

- Os algoritmos podem conter operações cujo resultado não é definido de forma única;

Algoritmos Não-Determinísticos

UNIFAL-MG
Universidade Federal de Alfenas

- Os algoritmos podem conter operações cujo resultado não é definido de forma única;

- **Algoritmo não-determinístico**: capaz de *escolher* uma dentre as várias alternativas possíveis a cada passo;

Algoritmos Não-Determinísticos

UNIFAL-MG
Universidade Federal de Alfenas

- Os algoritmos podem conter operações cujo resultado não é definido de forma única;
- **Algoritmo não-determinístico**: capaz de *escolher* uma dentre as várias alternativas possíveis a cada passo;
- Algoritmos não-determinísticos contêm operações cujo resultado não é unicamente definido, ainda que limitado a um conjunto definido de possibilidades.

Algoritmos Não-Determinísticos

UNIFAL-MG
Universidade Federal de Alfenas

- Os algoritmos podem conter operações cujo resultado não é definido de forma única;
- **Algoritmo não-determinístico**: capaz de *escolher* uma dentre as várias alternativas possíveis a cada passo;
- Algoritmos não-determinísticos contêm operações cujo resultado não é unicamente definido, ainda que limitado a um conjunto definido de possibilidades.
- Vamos definir uma função irreal para os algoritmos não determinísticos...

Algoritmos Não-Determinísticos
Função *Escolhe*

UNIFAL-MG
Universidade Federal de Alfenas

- Algoritmos não-determinísticos utilizam uma função escolhe(C), que escolhe um dos elementos do conjunto C de forma arbitrária.

Algoritmos Não-Determinísticos
Função *Escolhe*

UNIFAL-MG
Universidade Federal de Alfenas

- Algoritmos não-determinísticos utilizam uma função escolhe(C), que escolhe um dos elementos do conjunto C de forma arbitrária.

- O comando de atribuição X = escolhe(1 : n) pode resultar na atribuição a X de qualquer dos inteiros no intervalo [1, n];

Algoritmos Não-Determinísticos
Função *Escolhe*

UNIFAL-MG
Universidade Federal de Alfenas

- **Algoritmos não-determinísticos utilizam uma função escolhe(C)**, que escolhe um dos elementos do conjunto C de forma arbitrária.

- O comando de atribuição X = escolhe(1 : n) pode resultar na atribuição a X de qualquer dos inteiros no intervalo [1, n];

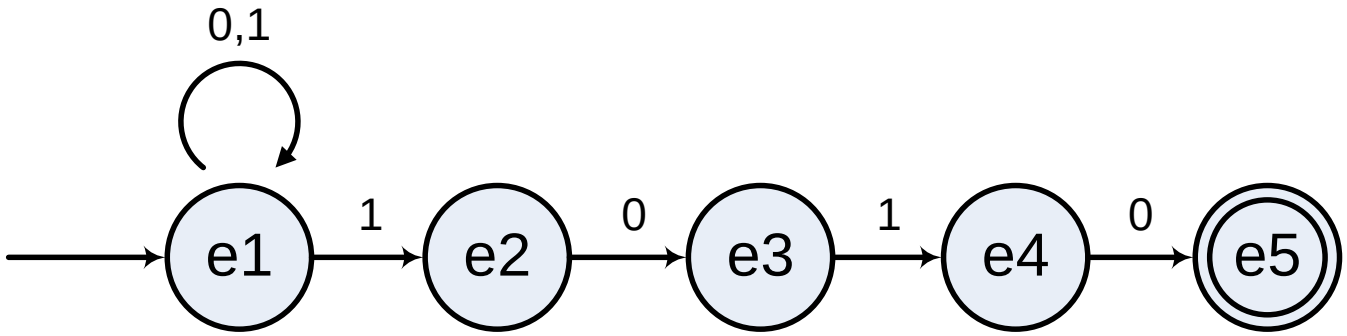
- A complexidade de tempo para cada chamada da função *escolhe* é *O(1)*.

Algoritmos Não-Determinísticos
Função *Escolhe*

UNIFAL-MG
Universidade Federal de Alfenas

- Vamos fazer um paralelo com a Teoria da Computação:

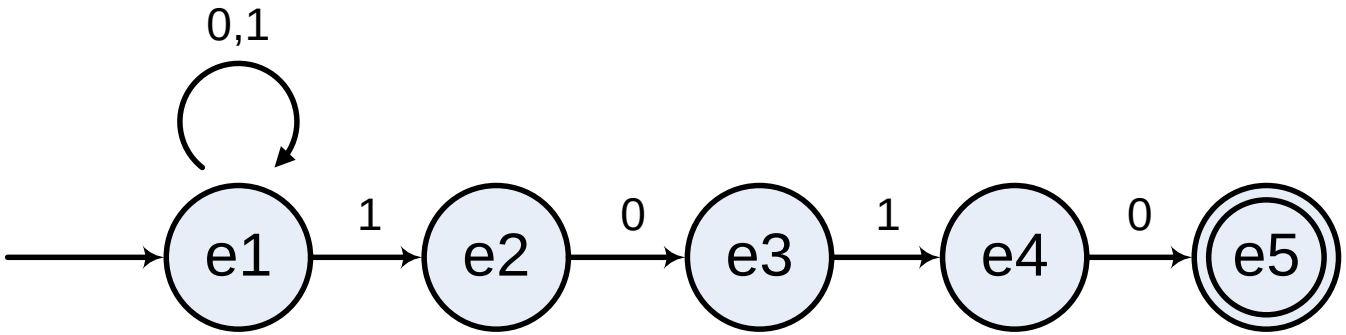
- A palavra $w = 1010$ é reconhecida pelo AFN definido a seguir?



Algoritmos Não-Determinísticos
Função *Escolhe*

UNIFAL-MG
Universidade Federal de Alfenas

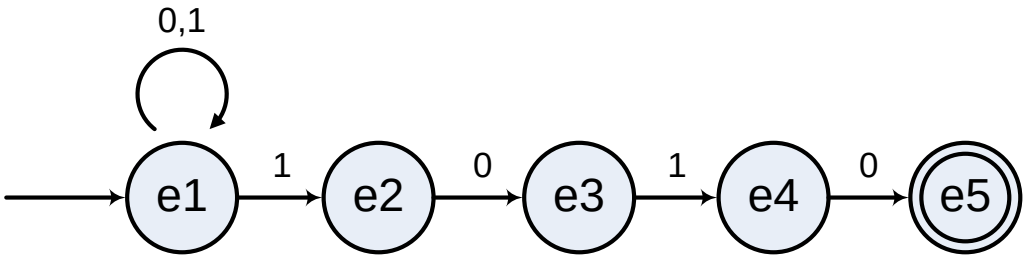
- *w = 1010*
- Como um estudante de C.C deve pensar: Devemos criar uma árvore de possibilidades;
  - Primeira tentativa: e1, e1, e2, e3 (estado não final). Não foi reconhecido...
    - O *backtracking* volta na árvore...
  - Segunda tentativa: e1, e2, e3, e4, e5 (estado final). Foi reconhecido.



Algoritmos Não-Determinísticos
Função *Escolhe*

UNIFAL-MG
Universidade Federal de Alfenas

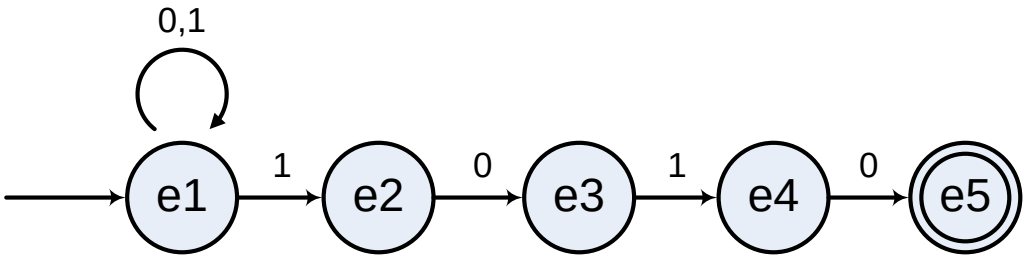
- Podemos visualizar a função *escolhe* de duas formas:



Algoritmos Não-Determinísticos
Função *Escolhe*

UNIFAL-MG
Universidade Federal de Alfenas

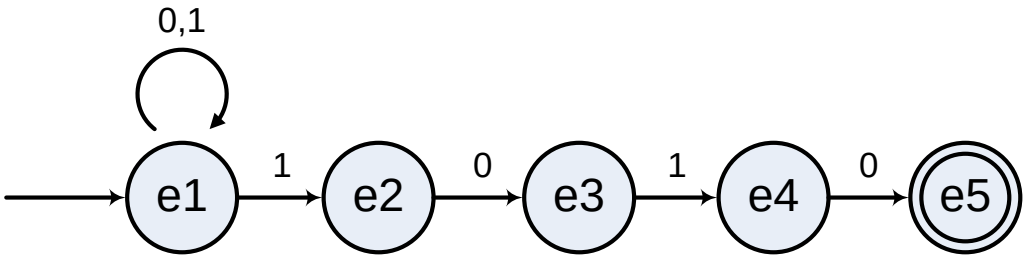
- Podemos visualizar a função *escolhe* de duas formas:
  - Ou ela faz a escolha certa a cada não determinismo;



Algoritmos Não-Determinísticos
Função *Escolhe*

UNIFAL-MG
Universidade Federal de Alfenas

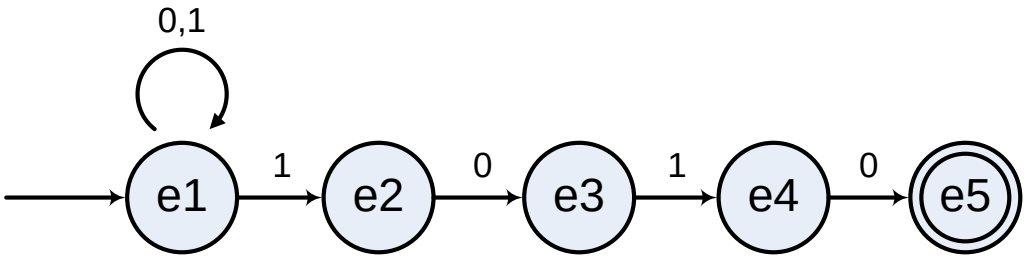
- Podemos visualizar a função *escolhe* de duas formas:
  - Ou ela faz a escolha certa a cada não determinismo;
  - Ou ela recebe um ou mais processadores para continuar a computação em paralelo;



Algoritmos Não-Determinísticos
Função *Escolhe*

UNIFAL-MG
Universidade Federal de Alfenas

- Podemos visualizar a função *escolhe* de duas formas:
  - Ou ela faz a escolha certa a cada não determinismo;
  - Ou ela recebe um ou mais processadores para continuar a computação em paralelo;



- Por que ambas as escolhas são irreais com os computadores existentes?
  - Mesmo com a área do processamento paralelo e distribuído bem desenvolvida na C.C...

Algoritmos Não-Determinísticos
Função *Escolhe*

UNIFAL-MG
Universidade Federal de Alfenas

- Uma máquina capaz de executar a função *escolhe* admite a capacidade de **computação não-determinística**.

Algoritmos Não-Determinísticos
Função *Escolhe*

UNIFAL-MG
Universidade Federal de Alfenas

- Uma máquina capaz de executar a função *escolhe* admite a capacidade de **computação não-determinística**.

- Uma máquina não-determinística é capaz de produzir cópias de si mesma quando diante de duas ou mais alternativas, e continuar a computação independentemente para cada alternativa.

Algoritmos Não-Determinísticos
Função *Escolhe*

UNIFAL-MG
Universidade Federal de Alfenas

- Uma máquina capaz de executar a função *escolhe* admite a capacidade de **computação não-determinística**.

- Uma máquina não-determinística é capaz de produzir cópias de si mesma quando diante de duas ou mais alternativas, e continuar a computação independentemente para cada alternativa.

- A máquina não-determinística que acabamos de definir não existe na prática, mas ainda assim fornece fortes evidências de que certos problemas não podem ser resolvidos por algoritmos determinísticos em tempo polinomial.

Algoritmos Não-Determinísticos
Função *Escolhe*

UNIFAL-MG
Universidade Federal de Alfenas

- Exemplo do poder computacional da máquina não-determinística:

- Seja o seguinte algoritmo não-determinístico para pesquisar o elemento x em um conjunto de elementos $A[1 : n]$, $n >= 1$;

```
PESQUISAND(A, 1, n)
1  j ← ESCOLHE(A, 1, n)
2  if A[j] = x
3     then sucesso
4     else insucesso
```

Algoritmos Não-Determinísticos
Função *Escolhe*

UNIFAL-MG
Universidade Federal de Alfenas

- Exemplo do poder computacional da máquina não-determinística:

- Seja o seguinte algoritmo não-determinístico para **pesquisar o elemento x em um conjunto de elementos** $A[1 : n]$, $n >= 1$;

```
PESQUISAND(A, 1, n)
1  j ← ESCOLHE(A, 1, n)
2  if A[j] = x
3     then sucesso
4     else insucesso
```

- Determina um índice j tal que A[j] = x para um término com sucesso ou então insucesso quando x não está presente em A.

Algoritmos Não-Determinísticos
Função *Escolhe*

UNIFAL-MG
Universidade Federal de Alfenas

- Exemplo do poder computacional da máquina não-determinística:

- Seja o seguinte algoritmo não-determinístico para **pesquisar o elemento x em um conjunto de elementos** $A[1 : n]$, $n >= 1$;

```
PESQUISAND(A, 1, n)
1  j ← ESCOLHE(A, 1, n)
2  if A[j] = x
3     then sucesso
4     else insucesso
```

- Determina um índice j tal que A[j] = x para um término com sucesso ou então insucesso quando x não está presente em A.

- Qual é a complexidade para a máquina determinística?
- E para a não determinística?

Algoritmos Não-Determinísticos
Função *Escolhe*

UNIFAL-MG
Universidade Federal de Alfenas

```
PESQUISAND(A, 1, n)
1  j ← ESCOLHE(A, 1, n)
2  if A[j] = x
3     then sucesso
4     else insucesso
```

- Qual é a complexidade para a máquina determinística?
- E para a não determinística?
  - O algoritmo tem complexidade não-determinística *O(1)*.
  - Para um algoritmo determinístico a complexidade é *O(n)*.

UNIFAL-MG
Universidade Federal de Alfenas

A Classe NP

Classe NP

UNIFAL-MG
Universidade Federal de Alfenas

- Classe P:
  - conjunto de todos os problemas que podem ser resolvidos por algoritmos determinísticos em tempo polinomial;

Classe NP

UNIFAL-MG
Universidade Federal de Alfenas

- Classe P:
  - conjunto de todos os problemas que podem ser resolvidos por algoritmos determinísticos em tempo polinomial;

- Classe NP:
  - conjunto de todos os problemas que podem ser resolvidos por algoritmos não-determinísticos em tempo polinomial;

Classe NP

- **Classe P:**
  - conjunto de todos os problemas que podem ser resolvidos por algoritmos determinísticos em tempo polinomial;

- **Classe 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 (ou determinístico) que execute em tempo polinomial para resolver o problema;

UNIFAL-MG
Universidade Federal de Alfenas

A Classe NP-Completo

O Célebre
Problema da Satisfabilidade Booleana (SAT)

SAT
O tamanho do espaço de buscas pode ser grande demais

UNIFAL-MG
Universidade Federal de Alfenas

- SAT: Encontrar um conjunto de valores para variáveis booleanas de forma a avaliar uma expressão booleana qualquer como VERDADEIRA.

SAT
O tamanho do espaço de buscas pode ser grande demais

UNIFAL-MG
Universidade Federal de Alfenas

- SAT: Encontrar um conjunto de valores para variáveis booleanas de forma a avaliar uma expressão booleana qualquer como VERDADEIRA.

- Suponha a expressão:

$$F(x) = \overline{(x_{89} \vee x_{78} \wedge x_{01})} \wedge (\overline{x_{10}} \vee x_{99}) \vee \ldots \wedge (x_{56} \wedge x_{22})$$

  - Defina valores para cada componente do vetor *x*, para que a função *F(x)* seja avaliada como VERDADEIRA.

SAT
O tamanho do espaço de buscas pode ser grande demais

UNIFAL-MG
Universidade Federal de Alfenas

- Quantas diferentes atribuições existem para as componentes do vetor x?

- Em outras palavras, qual é o tamanho do espaço de busca do problema SAT?

$$F(x) = \overline{(x_{89} \vee x_{78} \wedge x_{01})} \wedge (\overline{x_{10}} \vee x_{99}) \vee \ldots \wedge (x_{56} \wedge x_{22})$$

SAT
O tamanho do espaço de buscas pode ser grande demais

UNIFAL-MG
Universidade Federal de Alfenas

$$F(x) = \overline{(x_{89} \vee x_{78} \wedge x_{01})} \wedge (\overline{x_{10}} \vee x_{99}) \vee \ldots \wedge (x_{56} \wedge x_{22})$$

- Qual é o tamanho do espaço de busca ?

$$|S| = 2^{|x|}$$

$$|S| = 2^{100} \approx 10^{30}$$

10.000.000.000.000.000.000.000.000.000

atribuições diferentes

SAT
O tamanho do espaço de buscas pode ser grande demais

UNIFAL-MG
Universidade Federal de Alfenas

- Qual é o tamanho do espaço de busca ?

$$|S| = 2^{100} \approx 10^{30}$$

10.000.000.000.000.000.000.000.000.000.000

atribuições diferentes

- Suponha que temos um computador que consegue avaliar 1000 atribuições diferentes por segundo.

SAT
O tamanho do espaço de buscas pode ser grande demais

UNIFAL-MG
Universidade Federal de Alfenas

- Qual é o tamanho do espaço de busca ?

$$|S| = 2^{100} \approx 10^{30}$$

10.000.000.000.000.000.000.000.000.000.000

atribuições diferentes

- Suponha que temos um computador que consegue avaliar 1000 atribuições diferentes por segundo.

- Se este computador estivesse funcionando desde o BIG BANG (a 15 bilhões de anos), ele não teria analisado nem 1% de todas as possibilidades.

SAT

UNIFAL-MG
Universidade Federal de Alfenas

- O SAT foi o primeiro problema a ser classificado como NP-completo por Cook no ano de 1971;

SAT

UNIFAL-MG
Universidade Federal de Alfenas

- O SAT foi o primeiro problema a ser classificado como NP-completo por Cook no ano de 1971;

- A SAT é um problema especial, pois todos os problemas que possuem algoritmos (polinomiais ou não) podem ser transformados no problema da SAT.

SAT

UNIFAL-MG
Universidade Federal de Alfenas

- O SAT foi o primeiro problema a ser classificado como NP-completo por Cook no ano de 1971;
- A SAT é um problema especial, pois todos os problemas que possuem algoritmos (polinomiais ou não) podem ser transformados no problema da SAT.
  - Prova usa definição matemática da **Máquina de Turing não-determinística** (MTND), capaz de resolver qualquer problema em NP. Incluindo uma descrição da máquina e de como instruções são executadas em termos de fórmulas booleanas.

    Estabelece uma correspondência entre todo problema em NP (expresso por um programa na MTND) *e alguma instância de SAT*.

SAT

UNIFAL-MG
Universidade Federal de Alfenas

- O **SAT foi o primeiro problema a ser classificado como NP-completo por Cook no ano de 1971**;
- A SAT é um problema especial, pois **todos os problemas que possuem algoritmos** (polinomiais ou não) **podem ser transformados no problema da SAT**.
  - Prova usa definição matemática da **Máquina de Turing não-determinística** (MTND), capaz de resolver qualquer problema em NP. Incluindo uma descrição da máquina e de como instruções são executadas em termos de fórmulas booleanas.

    Estabelece uma correspondência entre todo problema em NP (expresso por um programa na MTND) *e alguma instância de SAT.*
- Para completar o raciocínio, veremos a redução de problemas.

UNIFAL-MG
Universidade Federal de Alfenas

Redução de Problemas

Introdução

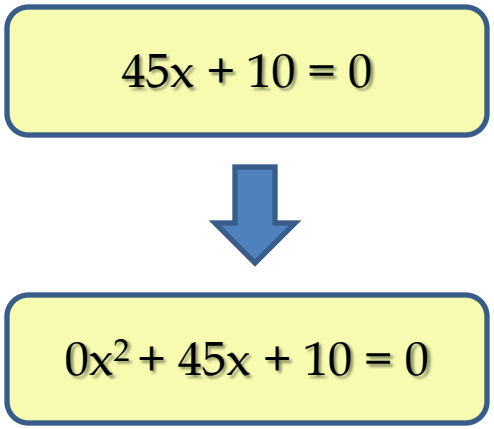
Redução de Problemas

UNIFAL-MG
Universidade Federal de Alfenas

- Ex.:
  - *Podemos resolver uma equação de primeiro grau através de um algoritmo que resolve equações de segundo grau...*

Redução de Problemas

- Ex.:
  - *Podemos resolver uma equação de primeiro grau através de um algoritmo que resolve equações de segundo grau...*



- Ou seja, as equações de primeiro grau são um caso particular das equações de segundo grau.

Redução de Problemas

UNIFAL-MG
Universidade Federal de Alfenas

- *Se qualquer problema NP-Completo for reduzido em tempo polinomial a um problema qualquer P, então, P é NP-Completo*

Redução de Problemas

UNIFAL-MG
Universidade Federal de Alfenas

- *Se qualquer problema NP-Completo for reduzido em tempo polinomial a um problema qualquer P, então, P é NP-Completo*

- Vamos juntar os fatos:

  - Todos os problemas que conhecemos algoritmos podem ser transformados na SAT;

Redução de Problemas

UNIFAL-MG
Universidade Federal de Alfenas

- *Se qualquer problema NP-Completo for reduzido em tempo polinomial a um problema qualquer P, então, P é NP-Completo*

- Vamos juntar os fatos:
  - Todos os problemas que conhecemos algoritmos podem ser transformados na SAT;
  - Se transformamos, por exemplo, a SAT em um problema P, então:
    - P é pelo menos tão difícil quanto a SAT;

Redução de Problemas

UNIFAL-MG
Universidade Federal de Alfenas

- *Se qualquer problema NP-Completo for reduzido em tempo polinomial a um problema qualquer P, então, P é NP-Completo*

- Vamos juntar os fatos:
  - Todos os problemas que conhecemos algoritmos podem ser transformados na SAT;
  - Se transformamos, por exemplo, a SAT em um problema P, então:
    - P é pelo menos tão difícil quanto a SAT;
    - E a SAT é pelo menos tão difícil quanto P;

Redução de Problemas

UNIFAL-MG
Universidade Federal de Alfenas

- *Se qualquer problema NP-Completo for reduzido em tempo polinomial a um problema qualquer P, então, P é NP-Completo*

- Vamos juntar os fatos:
  - Todos os problemas que conhecemos algoritmos podem ser transformados na SAT;
  - Se transformamos, por exemplo, a SAT em um problema P, então:
    - P é pelo menos tão difícil quanto a SAT;
    - E a SAT é pelo menos tão difícil quanto P;
    - Então, ambos possuem a mesma complexidade computacional;

Redução de Problemas

UNIFAL-MG
Universidade Federal de Alfenas

- O PCV já foi mostrado ser NP-Completo.

Redução de Problemas

UNIFAL-MG
Universidade Federal de Alfenas

- O PCV já foi mostrado ser NP-Completo.

- Se temos o Problema P1, e reduzimos o PCV a P1, isso quer dizer que:

Redução de Problemas

UNIFAL-MG
Universidade Federal de Alfenas

- O PCV já foi mostrado ser NP-Completo.

- Se temos o Problema P1, e reduzimos o PCV a P1, isso quer dizer que:

  - P1 é NP-Completo;

Redução de Problemas

UNIFAL-MG
Universidade Federal de Alfenas

- O PCV já foi mostrado ser NP-Completo.

- Se temos o Problema P1, e reduzimos o PCV a P1, isso quer dizer que:

  - P1 é NP-Completo;

  - E mais importante do que isso:
    - *Se algum dia alguém encontrar algum algoritmo polinomial para qualquer problema NPC, todos os problemas da classe NPC são também resolvidos em tempo polinomial;*

Redução de Problemas

UNIFAL-MG
Universidade Federal de Alfenas

- O PCV já foi mostrado ser NP-Completo.

- Se temos o Problema P1, e reduzimos o PCV a P1, isso quer dizer que:
  - P1 é NP-Completo;
  - E mais importante do que isso:
    - *Se algum dia alguém encontrar algum algoritmo polinomial para qualquer problema NPC, todos os problemas da classe NPC são também resolvidos em tempo polinomial;*
  - É a famosa tentativa de mostrar que P=NP;

Conclusões

- Quase ninguém acredita que P=NP;
- Atualmente, são conhecidos mais de 3.000 problemas NP-Completo, e para nenhum deles, foi encontrado algoritmo polinomial;
  - Este é um forte indício de que as classes são realmente distintas;
- Acredita-se também que NPC seja muito maior do que P.

Próxima aula

UNIFAL-MG
Universidade Federal de Alfenas

- Classe NP-Difícil (NP-Hard);
- Relacionamento entre as classes básicas de complexidade;
- Transformação polinomial de alguns problemas.

Bibliografia

- CORMEN, T. H.; LEISERSON, C. E.; RIVEST, R. L.; (2002). Algoritmos – Teoria e Prática. Tradução da 2ª edição americana. Rio de Janeiro. Editora Campus.

- TAMASSIA, ROBERTO; GOODRICH, MICHAEL T. (2004). Projeto de Algoritmos - Fundamentos, Análise e Exemplos da Internet.

- ZIVIANI, N. (2007). Projeto e Algoritmos com implementações em Java e C++. São Paulo. Editora Thomson;

- Material de aulas do Professor Loureiro (DCC-UFMG)
  - http://www.dcc.ufmg.br/~loureiro/paa/