Sistemas Lineares

Fabricio Murai

Aula passada

- Introdução: que tipos de problemas vamos estudar?
  - Correção no slide sobre Método de Newton

$$x_{n+1} = x_n - f(x_n)/f'(x_n)$$

- Conceitos fundamentais:
  - etapas no desenvolvimento de métodos numéricos
  - pseudo-código
- Revisão de conceitos em Álgebra Linear

Aula de hoje

- Revisar complexidade de algoritmos
    - objetivo: determinar a complexidade de um algoritmo simples usando a notação O
- Continuar revisão de Álgebra Linear
    - operações com transposta e inversa
    - autovalores e autovetores
    - normas vetoriais e matriciais

Anúncios

- Não haverá aula 16/03 quinta em função das Atividades Acadêmicas Complementares

- Quizz será realizado ao final de cada aula
  1o. quizz no dia 21/03 (terça)

- Vamos utilizar o app Socrative para "Questões do tipo clicker"
  Room: ANCN

Complexidade: notação O

- O(1), tempo constante
  Algoritmo que tem sempre mesmo tempo de execução, independente do tamanho da entrada

```
bool IsFirstElementNull(IList<string> elements)
{
    return elements[0] == null;
}
```

- O(n), linear
  Algoritmo cujo tempo de execução cresce linearmente e em proporção direta ao tamanho da entrada

```
bool ContainsValue(IList<string> elements, string value)
{
    foreach (var element in elements)
    {
        if (element == value) return true;
    }

    return false;
}
```

Complexidade: notação O

- $O(n^2)$, quadrático
  … tempo cresce proporcional ao quadrado da entrada

```
bool ContainsDuplicates(IList<string> elements)
{
    for (var outer = 0; outer < elements.Count; outer++)
    {
        for (var inner = 0; inner < elements.Count; inner++)
        {
            // Don't compare with self
            if (outer == inner) continue;

            if (elements[outer] == elements[inner]) return true;
        }
    }

    return false;
}
```

- $O(2^n)$, exponencial
  … tempo de execução cresce proporcional a 2 elevado ao tamanho da entrada

```
int Fibonacci(int number)
{
    if (number <= 1) return number;

    return Fibonacci(number - 2) + Fibonacci(number - 1);
}
```

Complexidade: notação O

$y = 3x$ $y = 6x-2$ $y = 15x + 44$

$y = x^2$ $y = x^2-6x+9$ $y = 3x^2+4x$

- Primeira família é O(n), segunda família é $O(n^2)$
- **Intuição**: Família de curvas, mesma forma
- Mais formalmente, O(f(n)) é um limite superior; para n grande o suficiente, cf(n) é maior.
- **Intuição**:
  função linear: dobra a entrada, dobra o tempo;
  função quadrática: dobra a entrada, quadriplica o tempo;

Fonte: https://rob-bell.net/2009/06/a-beginners-guide-to-big-o-notation/

Complexidade: notação O

- Determinante pela expansão de Laplace: quantas operações?

$$\det(A) = a_{11}\det(M_{11}) - a_{12}\det(M_{12}) + \cdots + (-1)^{n+1}a_{1n}\det(M_{1n}).$$

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \longrightarrow \det(A) = a_{11}a_{22} - a_{21}a_{12}$$

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \longrightarrow$$

$$\det(A) = a_{11}(a_{22}a_{33} - a_{32}a_{23}) - a_{12}(a_{21}a_{33} - a_{31}a_{23}) + a_{13}(a_{21}a_{32} - a_{31}a_{22}).$$

Clicker question: Complexidade da Expansão de Laplace

Sistemas Lineares (cont'd)

- Slides do Prof. Frederico Ferreira Campos Filho

Recapitulando

- Revisamos complexidade de algoritmos
  - objetivo: determinar a complexidade de um algoritmo simples usando a notação O
- Terminamos revisão de Álgebra Linear
  - operações com transposta e inversa
  - autovalores e autovetores
  - normas vetoriais e matriciais
- **Próxima aula**: 21 de março (haverá quizz!)