



CEFET/RJ

Formalização



Eduardo Ogasawara

eduardo.ogasawara@cefet-rj.br

<https://eic.cefet-rj.br/~eogasawara>

Formalização matemática e computacional

- A formalização matemática traz solidez a conceitos abstratos e permite uma comunicação precisa e rigorosa
 - Ajuda a evitar ambiguidades e imprecisões
 - Facilita a análise de problemas e algoritmos
 - Permite validar resultados com maior confiança
 - Garante uma base sólida para aplicações computacionais
- Exemplo: formalização de um algoritmo simples de ordenação:
 - Descrição vaga: "O algoritmo percorre a lista e reorganiza os elementos até que fiquem ordenados."
 - Descrição formal: "o algoritmo recebe uma sequência $S = [s_1, s_2, \dots, s_n]$ e aplica comparações entre elementos s_i e s_{i+1} , trocando-os se $s_i > s_{i+1}$. Esse processo continua até que S esteja ordenada segundo uma relação \leq . A complexidade do algoritmo, no pior caso, é $O(n^2)$."
- Vantagens da formalização:
 - Define com precisão as operações realizadas
 - Permite a análise rigorosa da complexidade
 - Evita ambiguidades e facilita a compreensão

Definições, teoremas e provas

- Os conceitos fundamentais da matemática formal incluem:
- Definição: explica o significado matemático de um conceito
- Lema e proposição: uma proposição é uma afirmação verdadeira; um lema auxilia na demonstração de teoremas
- Teorema: uma proposição verdadeira de grande importância
- Corolário: um resultado que segue diretamente de um teorema
- Prova: demonstração rigorosa da veracidade de uma proposição
- Conjectura: afirmação que se acredita verdadeira, mas sem prova formal
- Axioma: suposição básica aceita sem necessidade de prova

[1] s.G. Krantz, 2017, A primer of mathematical writing. American mathematical soc.

[2] k. Houston, 2009, how to think like a mathematician: A companion to undergraduate mathematics. Cambridge university press.

Exemplo de teorema

- Teorema: se n é um número natural, então a soma dos primeiros n números naturais é dada por:
- $1 + 2 + \dots + n = \frac{n(n+1)}{2}$
- Prova por indução:
 - Base: para $n = 1$, temos $1 = \frac{1(1+1)}{2}$, que é verdadeiro
 - Hipótese de indução: suponha que a fórmula seja verdadeira para $n = k$.
 - Passo indutivo: para $n = k + 1$, somamos $k + 1$ dos dois lados, obtendo $\frac{k(k+1)}{2} + (k + 1) = \frac{(k+1)(k+2)}{2}$
 - Portanto, a fórmula vale para todo n natural
- Importância dos teoremas
 - Estabelecem verdades fundamentais na matemática e na computação
 - São essenciais para a análise de algoritmos recursivos e estruturas de dados

Leitura de teoremas

- Como interpretar corretamente um teorema?
 - Identifique hipóteses e conclusões
 - Analise se há restrições para sua aplicação
 - Compare com outros teoremas conhecidos
 - Verifique exemplos e contraexemplos
- Exemplo: Teorema do valor médio
 - Se uma função $f(x)$ é contínua e diferenciável, existe um ponto onde $f'(c)$ é igual à inclinação da reta secante
 - Se a função for constante? → A inclinação da reta secante será zero
 - Se a função não for contínua? → O teorema não se aplica
- A leitura crítica ajuda a determinar quando um teorema é válido e aplicável

[1] s.G. Krantz, 2017, *A primer of mathematical writing*. American mathematical soc.

[2] k. Houston, 2009, *how to think like a mathematician: A companion to undergraduate mathematics*. Cambridge university press.

Elementos essenciais na escrita de teoremas

- Boas práticas para escrever teoremas:
 - Seja claro e objetivo
 - Numere definições, lemas e teoremas para referência
 - Apresente hipóteses antes da conclusão
 - Estruture a prova de forma lógica
- Exemplo:
 - Ruim: "Seja S um conjunto e seja S ordenado."
(Ambíguo: dado que S é um conjunto, o que significa "ordenado"?)
 - Melhor: "Seja S um conjunto finito com uma relação de ordem \leq ."
(especificidade evita ambiguidades)

Provas matemáticas

- Uma prova garante que uma proposição é verdadeira
- Demonstração estruturada melhora a compreensão e evita ambiguidades
- Técnicas de provas: direta, por casos, por contradição, por indução
- Exemplo: Provar que $\sqrt{2}$ é irracional
 - Suponha, por contradição, que $\sqrt{2} = \frac{p}{q}$, com p, q inteiros primos entre si
 - Elevando ao quadrado: $2q^2 = p^2$
 - p^2 é par, logo p também é.
 - Se p é par, $p = 2k$, então $(2k)^2 = 4k^2$, o que implica que q também é par.
 - Contradição, pois p e q deveriam ser primos entre si


[1] s.G. Krantz, 2017, *A primer of mathematical writing*. American mathematical soc.

[2] k. Houston, 2009, *how to think like a mathematician: A companion to undergraduate mathematics*. Cambridge university press.

Estratégias para uma leitura eficiente de formalizações

- Método de leitura:
 - Leia com um propósito: Identifique se o foco é definição, teorema ou prova
 - Priorize definições e exemplos: Eles estabelecem a base do entendimento
 - Verifique a estrutura lógica: Veja se os argumentos fazem sentido
 - Teste conceitos com exemplos simples
- Exemplo prático:
 - Ao ler um teorema sobre complexidade de algoritmos, verifique:
 - Se as hipóteses são realistas
 - Se os exemplos testados validam a afirmação

Exemplo de leitura eficiente de formalizações

- Imagine que você está lendo um artigo sobre análise de algoritmos e encontra o seguinte teorema:
- Teorema:
 - "Para qualquer algoritmo de ordenação baseado em comparações, o número mínimo de comparações no pior caso é $\Omega(n \log n)$ "
-  Como abordar a leitura?
 - Entenda os termos: o que significa "baseado em comparações"? O que representa $\Omega(n \log n)$?
 - Identifique a estrutura: o teorema apresenta um limite inferior, não uma quantidade exata de comparações
 - Busque exemplos: verifique se o artigo traz um algoritmo de ordenação que se encaixa no resultado
 - Teste a aplicação: pegue um conjunto pequeno de elementos e veja quantas comparações são realmente necessárias

Escrita clara e bem estruturada de formalizações

- Boas Práticas:
 - Evite frases longas: Use notação matemática quando possível
 - Mantenha a estrutura lógica: Definições → Teoremas → Provas
 - Facilite a leitura: Numere resultados importantes
 - Destaque conceitos chave em negrito ou itálico
- Exemplo
 - Não-formalizado: “uma lista invertida para um determinado termo é uma sequência de pares, onde o primeiro elemento em cada par é o identificador do documento e o segundo é a frequência de um termo em um documento referente ao identificador correspondente”
 - Formalizado: “uma lista invertida para um termo t é uma sequência de pares na forma $\langle d, f \rangle$, onde d representa um documento e f é a frequência de t em d ”

Exemplo de escrita eficiente com uma boa formatação

- Numere teoremas, proposições e definições para facilitar referências
- Separe trechos importantes com espaçamento adequado
- Destaque palavras-chave para ajudar na organização mental do leitor
- Exemplo:
 - Ruim:
 - “A relação de equivalência pode ser definida formalmente considerando um conjunto de pares ordenados que satisfazem três propriedades fundamentais: reflexividade, simetria e transitividade.”
 - Melhor:
 - Definição: uma relação de equivalência em um conjunto X é uma relação R que satisfaz:
 - Reflexividade: $\forall x \in X, (x, x) \in R$
 - Simetria: $\forall x, y \in X, (x, y) \in R \Rightarrow (y, x) \in R$
 - Transitividade: $\forall x, y, z \in X, (x, y) \in R \text{ e } (y, z) \in r \Rightarrow (x, z) \in R$

Algoritmos e formalismo de algoritmos

- A contribuição principal de diversos artigos de computação é materializada por algoritmos
- Ao verificar um algoritmo, o leitor espera:
 - Os passos
 - Os dados de entrada, saída
 - As estruturas de dados usadas internamente
 - Propriedades de correção
 - Análise de complexidade
- Tipos de formalização
 - Descrição narrativa
 - Pseudocódigo
 - Prosecode

Pseudocódigo

- O pseudocódigo representa algoritmos sem depender da sintaxe de uma linguagem específica
- Deve ser claro, organizado e sem ambiguidades

The **WeightedEdit** function computes the edit distance between two strings, assigning a higher penalty for errors closer to the front.

Input: $S1, S2$: strings to be compared.
Output: weighted edit distance
Variables: $L1, L2$: string lengths
 $F[L1, L2]$: array of minimum distances
 W : current weighting
 M : maximum penalty
 C : current penalty

WeightedEdit($S1, S2$):

```
1.  $L1 = \text{len}(S1)$ 
2.  $L2 = \text{len}(S2)$ 
3.  $M = 2 \times (L1 + L2)$ 
4.  $F[0, 0] = 0$ 
5. for  $i$  from 1 to  $L1$ 
6.    $F[i, 0] = F[i - 1, 0] + M - i$ 
7. for  $j$  from 1 to  $L2$ 
8.    $F[0, j] = F[0, j - 1] + M - j$ 
9. for  $i$  from 1 to  $L1$ 
10.   $C = M - i$ 
11.  for  $j$  from 1 to  $L2$ 
12.     $C = C - 1$ 
13.     $F[i, j] = \min(F[i - 1, j] + C,$ 
                      $F[i, j - 1] + C,$ 
                      $F[i - 1, j - 1] + C \times \text{isdiff}(S1[i], S2[j]))$ 
14. WeightedEdit =  $F[L1, L2]$ 
```

Pseudocódigo - encapsulamento em funções

- Dividir o algoritmo em funções reutilizáveis melhora clareza e modularidade

```
SCHEDULER(Workflow W, ExecutionProcessesSet execProc)
1.  FragmentSet fragSet = optimizeWorkflow(W);
2.  while (hasElements(fragSet))
3.      frag = getReadyFragment(fragSet);
4.      FAISet faiSet = generateActivations(frag);
5.      DispatchStrategy dispStrat = getDispatchStrategy(frag);
6.      for each FAI f in faiSet
7.          dispatch(f, dispStrat, execProc);
8.      fragSet = removeCompleted(fragSet, frag);
```

Figure 8. Scheduler Algorithm.

Prosecode

- O Prosecode mistura linguagem natural com estrutura algorítmica
- Foca na explicação intuitiva do algoritmo

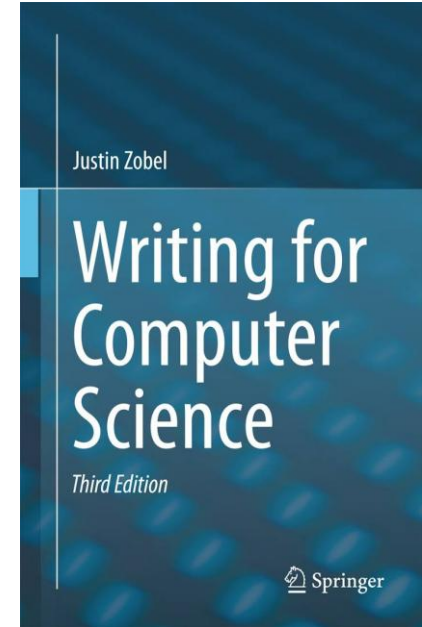
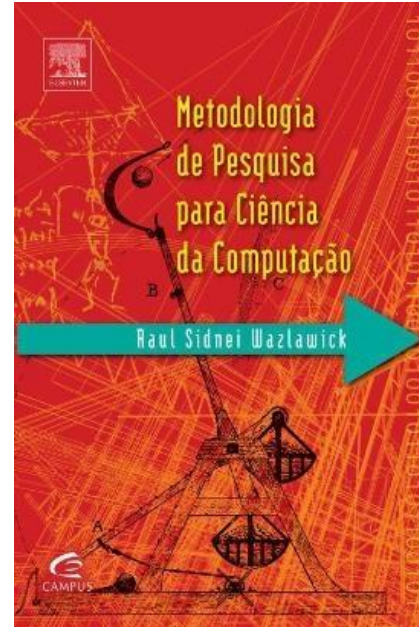
WeightedEdit(s, t) compares two strings s and t , of lengths k_s and k_t respectively, to determine their edit distance—the minimum cost in insertions, deletions, and replacements required to convert one into the other. These costs are weighted so that errors near the start of the strings attract a higher penalty than errors near the end.

We denote the i th character of string s by s_i . The principal internal data structure is a 2-dimensional array F in which the dimensions have ranges 0 to k_s and 0 to k_t , respectively. When the array is filled, $F_{i,j}$ is the minimum edit distance between the strings $s_1 \cdots s_i$ and $t_1 \cdots t_j$; and F_{k_s, k_t} is the minimum edit distance between s and t .

The value p is the maximum penalty, and the penalty for a discrepancy between positions i and j of s and t , respectively, is $p - i - j$, so that the minimum penalty is $p - k_s - k_t = p/2$ and the next-smallest penalty is $p/2 + 1$. Two errors, wherever they occur, will outweigh one.

1. (Set penalty.) Set $p \leftarrow 2 \times (k_s + k_t)$.
2. (Initialize data structure.) The boundaries of array F are initialized with the penalty for deletions at start of string; for example, $F_{i,0}$ is the penalty for deleting i characters from the start of s .
 - (a) Set $F_{0,0} \leftarrow 0$.
 - (b) For each position i in s , set $F_{i,0} \leftarrow F_{i-1,0} + p - i$.
 - (c) For each position j in t , set $F_{0,j} \leftarrow F_{0,j-1} + p - j$.
3. (Compute edit distance.) For each position i in s and position j in t :
 - (a) The penalty is $C = p - i - j$.
 - (b) The cost of inserting a character into t (equivalently, deleting from s) is $I = F_{i-1,j} + C$.
 - (c) The cost of deleting a character from t is $D = F_{i,j-1} + C$.
 - (d) If s_i is identical to t_j , the replacement cost is $R = F_{i-1,j-1}$. Otherwise, the replacement cost is $R = F_{i-1,j-1} + C$.
 - (e) Set $F_{i,j} \leftarrow \min(I, D, R)$.
4. (Return.) Return F_{k_s, k_t} .

Referências



- [1] D. G. Perovano, Manual de metodologia da pesquisa científica. Editora Intersaberes, 2016.
- [2] A. L. Cervo, P. A. Bervian, e R. da Silva, Metodologia Científica. Pearson Universidades, 2006.
- [3] R. Wazlawick, 2017, Metodologia de Pesquisa para Ciência da Computação. Elsevier Brasil.
- [4] J. Zobel, 2015, Writing for Computer Science. Springer.

