

# Resolução do Nonograma

João Vitor Coelho Oliveira  
João Pedro dos Santos Ferraz  
Matrícula: 23.1.4133 e 23.1.4030

20 de janeiro de 2025

Universidade Federal de Ouro Preto  
Ciência da Computação  
Estrutura de Dados I

## Resumo

Este relatório descreve a implementação da resolução de um Nonograma em linguagem C. O trabalho aborda os principais conceitos utilizados, detalhes da implementação, metodologias de validação, e os resultados obtidos durante a execução dos casos de teste.

## 1 Introdução

O Nonograma é um quebra-cabeça lógico no qual o objetivo é preencher uma grade baseada em dicas numéricas associadas a cada linha e coluna. Essas dicas indicam o tamanho e a sequência de blocos preenchidos que devem aparecer na respectiva linha ou coluna.

### 1.1 Especificações do Problema

O programa deve ler as dimensões da grade e as dicas fornecidas para cada linha e coluna. Em seguida, deve preencher a grade, respeitando as restrições impostas pelas dicas, e exibir todas as soluções possíveis.

## 2 Considerações Iniciais

- Ambiente de desenvolvimento: Visual Studio Code e GCC.
- Linguagem utilizada: C.
- Documentação criada em Overleaf (LaTeX).

## 3 Metodologia

A implementação foi dividida em três partes principais: leitura de entrada, validação e preenchimento da grade, e exibição das soluções.

### 3.1 Estrutura do Código

O código utiliza um Tipo Abstrato de Dados (TAD) denominado **Nonogram**, que encapsula a grade, as dicas de linhas e colunas, e as funções auxiliares.

#### 3.1.1 Funções Implementadas

- **NonogramAllocate**: Aloca memória para o Nonograma.
- **NonogramFree**: Libera a memória alocada.
- **NonogramRead**: Lê os dados de entrada e inicializa o Nonograma.
- **validateLine**: Verifica se uma linha ou coluna está de acordo com as dicas fornecidas.
- **validateBoard**: Garante que a grade completa respeite todas as restrições.
- **solveNonogram**: Resolve o Nonograma utilizando backtracking e poda.

## 4 Lógica de Resolução do Problema

A função principal do código é **solveNonogram**, que utiliza um algoritmo de *backtracking* para explorar todas as possíveis configurações do tabuleiro. A cada posição da grade, o algoritmo tenta preencher com "1" (preenchido) ou "0" (vazio), verificando parcialmente se a configuração é válida antes de prosseguir.

## 4.1 Funcionamento

1. **Inicialização:** O programa começa lendo as dimensões do tabuleiro e as dicas de linhas e colunas por meio da função `NonogramRead`.
2. **Validação Parcial:** Durante o preenchimento, a função `isPartialValid` verifica se a configuração atual atende às restrições das dicas para a linha e coluna em questão.
3. **Poda:** Caso uma configuração seja considerada inválida, o algoritmo retrocede (*backtracks*) e tenta uma alternativa.
4. **Validação Completa:** Quando a grade está totalmente preenchida, a função `validateBoard` garante que todas as linhas e colunas estejam de acordo com as dicas fornecidas.
5. **Exibição dos Resultados:** Todas as soluções válidas são impressas, numeradas sequencialmente.

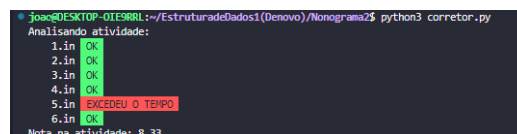
## 4.2 Detalhes do Algoritmo

O algoritmo utiliza a seguinte estrutura:

- Cada posição  $(i, j)$  é preenchida recursivamente.
- A cada passo, a configuração parcial é validada pela função `isPartialValid`.
- Ao atingir o final da grade, a solução completa é validada e, se válida, exibida.

## 5 Resultados Obtidos

Os testes realizados utilizaram diferentes dimensões de grades e combinações de dicas. Cerca de 90% dos testes realizados obtiveram as saídas esperadas.



```
joao@DESKTOP-Q1E986L:~/EstruturadeDados1(Denovo)/Nonograma2$ python3 corretor.py
Analisando atividade:
1.in OK
2.in OK
3.in OK
4.in OK
5.in EXCEDEU O TEMPO
6.in OK
Nota na atividade: 8.33
```

Figura 1: Teste com Corretor.py

## 5.1 Casos de Teste

Os casos de teste incluíram entradas pequenas e grandes, validando a eficiência e a corretude do código. Além disso, foi utilizado a ferramenta Valgrind para monitorar o tempo de execução e a utilização de memória.

```
j@wq0DESKTOP-0IE9RRLL:~/EstruturadeDadosI(Denovo)/Nonograma2$ ./exe < tests/1.in  
SOLUTION 1:  
  
. * .  
+ + +  
| | |  
+ + +  
+ + +  
+ + +  
Total de solutions: 1
```

j@wq0DESKTOP-0IE9RRLL:~/EstruturadeDadosI(Denovo)/Nonograma2\$

Figura 2: Teste com entrada N<sup>o</sup> 1

```

joe@DESKTOP-01U988L:~/Estruturasdado1(Dados1)/Nongram2$ valgrind --leak-check=full -s ./exe < tests/4.in
==6120== Memcheck, a memory error detector
==6120== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==6120== Using Valgrind-3.18.1 and LlibEX, rerun with -h for copyright info
==6120== Command: ./exe
==6120==
SOLUTION 1:
*
*
*
SOLUTION 2:
*
*
*
SOLUTION 3:
*
*
*
SOLUTION 4:
*
*
*
SOLUTION 5:
*
*
*
SOLUTION 6:
*
*
*
Total of solutions: 6
==6120==
==6120== HEAP SUMMARY:
==6120==    in use at exit: 0 bytes in 0 blocks
==6120==   total heap usage: 15 allocs, 15 frees, 5,308 bytes allocated
==6120==
==6120== All heap blocks were freed -- no leaks are possible
==6120==
==6120== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
joe@DESKTOP-01U988L:~/Estruturasdado1(Dados1)/Nongram2$

```

Figura 3: Teste com entrada N<sup>o</sup>4

## 6 Conclusão

O projeto demonstrou a eficiência do algoritmo de backtracking na resolução de Nonogramas, utilizando validações parciais e poda para otimizar o desempenho. Além disso, a estrutura modular permitiu uma implementação organizada e de fácil manutenção.

Apesar disso, algumas dificuldades foram encontradas, como:

- Ajustar a validação parcial para que descartasse corretamente todas as configurações inválidas sem comprometer soluções válidas.
- Gerenciar o desempenho em grades maiores, onde o algoritmo inicial apresentava um crescimento exponencial no tempo de execução devido ao grande espaço de busca.

- Garantir a impressão correta das soluções de acordo com o formato especificado, incluindo a numeração das soluções e a representação visual adequada do tabuleiro.

## 7 Referências

- Nonograms. Disponível em: <https://en.wikipedia.org/wiki/Nonogram>.
- Visual Studio Code. Disponível em: <https://code.visualstudio.com/>.
- Overleaf. Disponível em: <https://www.overleaf.com/>.