Matrizes Esparsas como Aplicação de Listas Encadeadas: Um Projeto de Estrutura de Dados

Atílio G. Luiz[1], Calebe Mesquita da Silva[2], Francisco Emilson S. Souza Filho[2]

[1]Professor Associado Campus Quixadá (CE) – Universidade Federal do ceará (UFC)
Av.José de Freitas Queiroz, 5003 – Cedro – Quixadá – Ceará 63902-580 – Brazil

[2]Alunos do Campus Quixadá (CE) – Universidade Federal do ceará (UFC)
Av.José de Freitas Queiroz, 5003 – Cedro – Quixadá – Ceará 63902-580 – Brazil

gomes.atilio@ufc.br;

calebcamesquita@alu.ufc.br, francisco.filho24@alu.ufc.br

Abstract. *This work presents the implementation of a sparse matrix using circular plain linked lists for the Data Structure course at Universidade Federal do Ceará. It draws parallels to optimization solutions discussed in the literature and details the software development process.*

Resumo. *O presente trabalho apresenta a implementação de uma matriz esparsa usando listas circulares simplesmente encadeadas para o curso de Estrutura de Dados da Universidade Federal do Ceará. Além disso, ele traça paralelos com soluções de otimização discutidas na literatura e detalha o processo de desenvolvimento do software final.*

1. Introdução

Matrizes esparsas são definidas como aquelas em que a maioria de seus elementos é igual a zero, indicando que muitos valores não estão efetivamente presentes ou são desnecessários. A implementação desse tipo de matriz é especialmente útil em contextos onde as dimensões das matrizes são extremamente grandes, o que dificulta o processamento eficiente dos dados pelos processadores atuais, ou quando não todos os elementos da matriz serão utilizados. É relevante ressaltar que matrizes, em especial as multidimensionais, tornam-se inviáveis em software convencional ao atingirem grandes dimensões, pois demandam a alocação de espaço de memória para armazenar todos os elementos de uma matriz tradicional. Por exemplo, uma matriz de dimensões 30.000×30.000 requereria aproximadamente $9 \cdot 10^8$ espaços de memória, o que é excessivo, mesmo para computadores modernos.

Além disso, matrizes esparsas apresentam aplicações vantajosas em softwares destinados a cálculos científicos e engenharia. Outras aplicações diretas incluem a resolução de sistemas lineares na forma $Ax = b$ com múltiplas variáveis e a representação de grafos [Pissanetzki 1984], contribuindo para a resolução de problemas de álgebra linear. Uma forma clara de visualizar essa aplicação é em um programa de planilhas, onde muitos dados podem ser representados como uma matriz esparsa, em que uma célula não preenchida indica a ausência de um valor na matriz. Nesse contexto, é importante diferenciar entre matriz lógica e matriz física. A matriz física refere-se à representação da matriz como um todo; por exemplo, uma matriz A de dimensões 5×5 pode ser visualizada como uma estrutura completa.

Por outro lado, ao nos referirmos a matrizes lógicas, estamos tratando da interpretação da matriz na memória, sem a necessidade de armazenar os valores de todas as células que contêm zero, conforme mencionado anteriormente.

Embora este trabalho não trate diretamente de um software de planilhas, é possível estabelecer um paralelo com esse tipo de aplicação.

Existem diversas maneiras de implementar a ideia mencionada. Os autores [Pissanetzki 1984] e [Schildt 1997] apresentam algumas alternativas, como listas encadeadas, árvores binárias, *arrays* ou matrizes de ponteiros. Este artigo discutirá a abordagem que utiliza listas encadeadas circulares simplesmente encadeadas, que será detalhada posteriormente.

1.1. Justificativa

A justificativa para a escolha desta abordagem baseia-se na eficiência no uso da memória do computador. No entanto, ao acessar seus elementos, não é possível fazê-lo em tempo constante, como ocorre em matrizes convencionais, uma vez que é necessário percorrer os elementos para alcançar a célula ou nó desejado.

1.2. Objetivo

O objetivo deste trabalho é unificar o conhecimento adquirido em sala de aula com a pesquisa, visando elaborar um projeto que reforce o aprendizado dos alunos de forma prática, assegurando uma melhor fixação do conteúdo.

2. Revisão Bibliográfica

2.1. Listas Encadeadas

2.1.1. Estrutura de uma Lista Encadeada

Uma lista encadeada é uma estrutura de dados empregada na programação para representar uma coleção de elementos. Nesta estrutura, os dados são armazenados de maneira dispersa na memória, o que requer a interconexão de cada elemento. Cada dado, denominado *nó*, deve conter o endereço de memória do próximo item da lista [Bhargava 2017], assegurando que os dados permaneçam interligados.

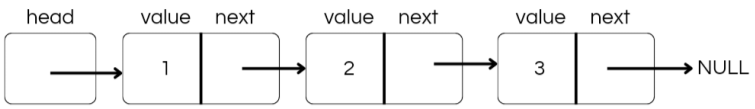

Figura 1. Representação de uma lista simplesmente encadeada

Em algumas abordagens, é possível optar por utilizar um recurso adicional: um nó sentinela no início da lista. Este nó consome alguns bytes adicionais de memória, mas sua utilização facilita as operações de inserção e remoção de elementos, uma vez que elimina a necessidade de verificações adicionais para casos especiais. Ademais, ele permanece sempre na posição inicial da lista encadeada, assegurando que a estrutura mantenha sua eficiência.

Cada nó na estrutura deve conter o valor armazenado e, em linguagens que suportam ponteiros, como C++, que é a linguagem adotada neste trabalho, o endereço do próximo nó da sequência. No último nó da lista, deve existir um ponteiro que aponta para um endereço nulo, indicando o término da lista. No código a seguir, apresenta-se uma definição concisa de um nó em C++.

```
struct Node {
 int data;
 Node* next;
};
```

2.1.2. Tipos de Listas Encadeadas

Além da estrutura básica, existem variações que oferecem distintas vantagens, dependendo da aplicação a qual se destinam. Uma dessas variações, que será utilizada neste estudo, é a lista encadeada circular. Nessa modificação, ao invés de o último nó da sequência apontar para um endereço de memória nulo (`nullptr`), o atributo `next` direciona-se para o endereço do primeiro nó da lista, conferindo-lhe a característica de circularidade. Abaixo, a Figura 2 ilustra o comportamento dessa disposição.

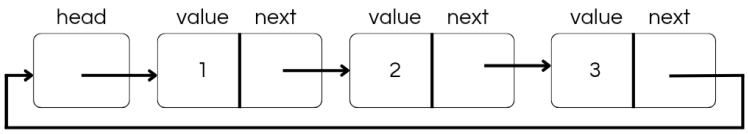

Figura 2. Representação de uma lista encadeada circular

Além desse formato, é possível adicionar um ponteiro adicional na configuração dos nós, com a finalidade de possibilitar o retorno ao nó anterior em tempo constante. Assim, por meio de distintas abordagens para a implementação de uma lista circular, é possível gerar diversas transformações a partir de uma mesma concepção.

2.1.3. Operações básicas

Conforme demonstrado na obra de [Bhargava 2017], nas listas encadeadas, há uma renúncia ao tempo de acesso constante dos *arrays* ($O(1)$) em prol de um acesso que apresenta complexidade linear ($O(n)$), uma vez que é necessário percorrer todos os elementos da lista até se chegar ao elemento desejado. Da mesma forma, a inserção de novos elementos na lista também possui complexidade $O(n)$, dado que é preciso percorrer todos os elementos até o final. Entretanto, existem variações, como as listas circulares duplamente encadeadas, nas quais a inserção pode ocorrer em tempo constante. Em contrapartida, a remoção dos primeiros elementos de uma lista é realizada em $O(1)$, pois o endereço do primeiro elemento é mantido na estrutura.

2.2. Matrizes Esparsas com Lista Encadeada

Neste artigo, trataremos apenas da aplicação de matrizes esparsas com lista encadeada, utilizando-se de nós sentinelas para auxiliar nas operações dentro da matriz.

Nesta configuração, há inicialmente um nó sentinela para cada linha e cada coluna, além de um nó sentinela que conecta ambas as partes, centralizando o início da estrutura. Os nós de cada linha estão interligados de forma circular, conforme explicitado anteriormente, formando uma organização semelhante à representada na Figura 3. Ademais, cada nó deve suportar, além de sua configuração inicial, os valores de linha e coluna da matriz física, mostrado posteriormente.

Assim, ao inserir um elemento de forma semelhante à apresentada na subseção anterior (ver Seção 2.1), é necessário configurar as referências de linha e coluna para manter organizadas as listas encadeadas de linhas e colunas. Na Figura 4, apresenta-se o esquema lógico para a matriz física abaixo.

$$A = \begin{bmatrix} 50 & 0 & 0 & 0 \\ 10 & 0 & 20 & 0 \\ 0 & 0 & 0 & 0 \\ -30 & 0 & -60 & 5 \end{bmatrix}$$

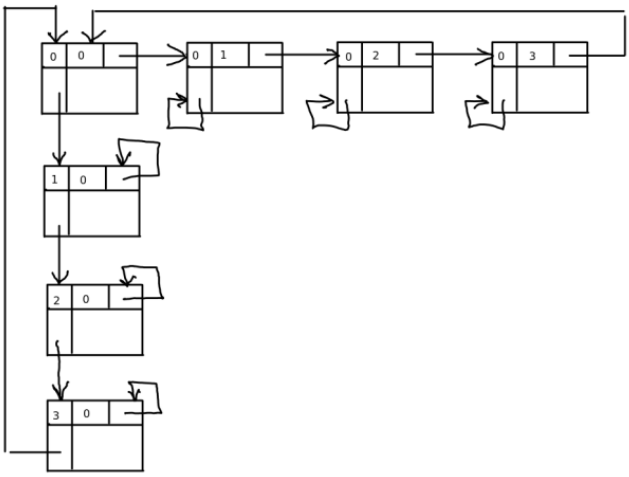

Figura 3. Esquema de uma matriz esparsa vazia

De certa forma, essa disposição pode ser adaptada para diminuir a complexidade das operações, aproximando sua execução de um tempo constante.

2.3. Padrão de Projeto Command

Em um programa computacional, é comum a interação com o usuário por meio de diversas abordagens, como interfaces gráficas ou linhas de comando. Nesse contexto, é necessário que o código seja modular e suficientemente flexível para acomodar alterações futuras. Desse modo, surgem diferentes estratégias para organizar a programação de um software a fim de atender a essas exigências.

No campo da computação, diversos padrões são estabelecidos na literatura [Gamma et al. 1995] e categorizados em várias formas. A presente pesquisa se concentrará no padrão de projeto comportamental denominado *Command*. De maneira geral, esse padrão possibilita a unificação das ações realizadas dentro do sistema em um conjunto exclusivo de componentes.

2.3.1. Estrutura

O primeiro desses componentes é uma classe remetente (também conhecida como invocadora) [Guru 2025], responsável por chamar o comando correspondente. Por esse motivo, esse tipo de dado deve conter um conjunto de referências aos comandos necessários para invocar seus métodos. Outro constituinte dessa perspectiva é a definição de uma interface comum a todos os comandos, denominados *comandos concretos*.

É importante salientar que a classe remetente não é responsável por criar um comando; geralmente, ele é pré-criado por meio de um construtor no cliente. No contexto da Programação Orientada a Objetos (POO), um cliente pode ser uma classe externa que chama o remetente. Essa estrutura também é responsável por enviar todos os dados necessários para o funcionamento correto do comando a ser executado. A Figura 5 ilustra a estrutura básica do padrão de projeto estabelecido.

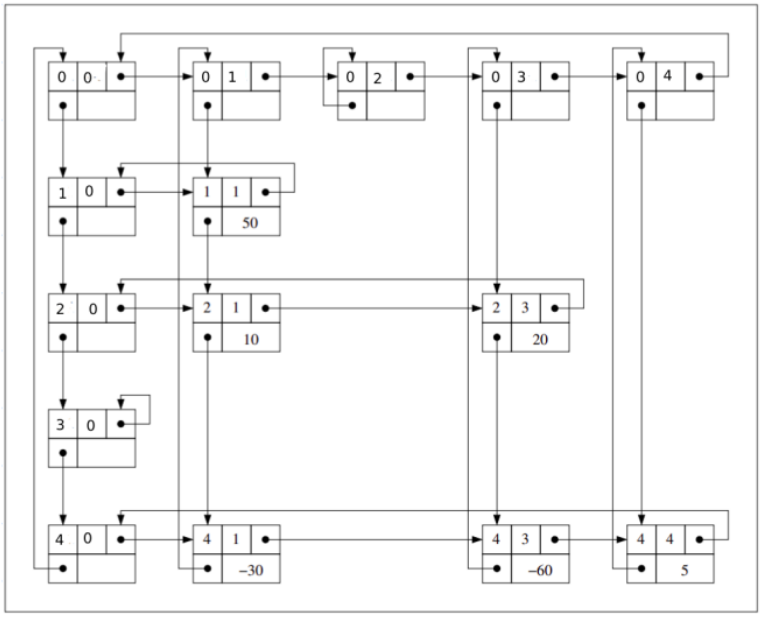

Figura 4. Exemplo de matriz esparsa

2.3.2. Aplicações

Uma vez compreendido o funcionamento desse padrão, é possível descrever suas possíveis aplicações. O padrão Command pode ser aplicado em sistemas que implementam mecânicas de desfazer e refazer ações (Undo/Redo), além de facilitar a criação de filas de operações. No contexto deste projeto, tal abordagem contribui para a melhoria da legibilidade e da evolução do código ao longo do tempo.

3. Implementação da Estrutura de Dados

4. Decisões de Implementação

5. Divisão do Trabalho

6. Dificuldades Encontradas

7. Testes e Validação

8. Análise de Complexidade

9. Conclusão

10. Referências

All full papers and posters (short papers) submitted to some SBC conference, including any supporting documents, should be written in English or in Portuguese. The format paper should be A4 with single column, 3.5 cm for upper margin, 2.5 cm for bottom margin and 3.0 cm for lateral margins, without headers or footers. The main font must be Times, 12 point nominal size, with 6 points of space before each paragraph. Page numbers must be suppressed.

Full papers must respect the page limits defined by the conference. Conferences that publish just abstracts ask for **one**-page texts.

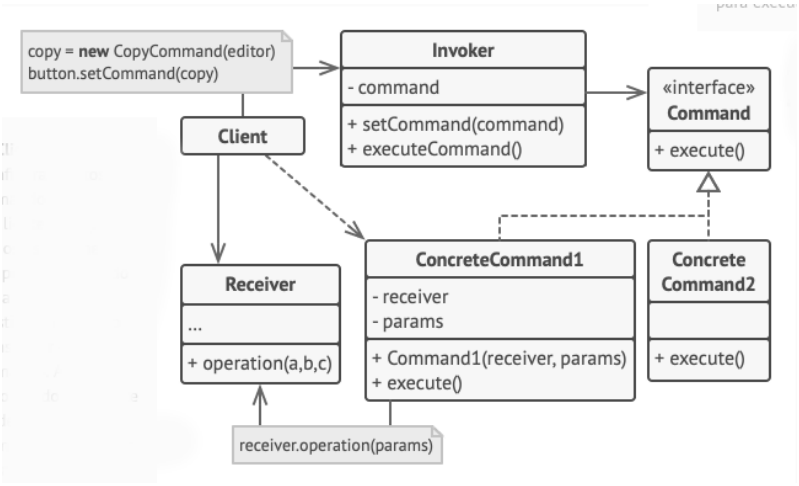

Figura 5. Ilustração da organização do Command

11. First Page

The first page must display the paper title, the name and address of the authors, the abstract in English and "resumo" in Portuguese ("resumos" are required only for papers written in Portuguese). The title must be centered over the whole page, in 16 point boldface font and with 12 points of space before itself. Author names must be centered in 12 point font, bold, all of them disposed in the same line, separated by commas and with 12 points of space after the title. Addresses must be centered in 12 point font, also with 12 points of space after the authors' names. E-mail addresses should be written using font Courier New, 10 point nominal size, with 6 points of space before and 6 points of space after.

The abstract and "resumo" (if is the case) must be in 12 point Times font, indented 0.8cm on both sides. The word **Abstract** and **Resumo**, should be written in boldface and must precede the text.

12. CD-ROMs and Printed Proceedings

In some conferences, the papers are published on CD-ROM while only the abstract is published in the printed Proceedings. In this case, authors are invited to prepare two final versions of the paper. One, complete, to be published on the CD and the other, containing only the first page, with abstract and "resumo" (for papers in Portuguese).

13. Sections and Paragraphs

Section titles must be in boldface, 13pt, flush left. There should be an extra 12 pt of space before each title. Section numbering is optional. The first paragraph of each section should not be indented, while the first lines of subsequent paragraphs should be indented by 1.27 cm.

13.1. Subsections

The subsection titles must be in boldface, 12pt, flush left.

14. Figures and Captions

Figure and table captions should be centered if less than one line (Figure 6), otherwise justified and indented by 0.8cm on both margins, as shown in Figure 7. The caption font must be Helvetica, 10 point, boldface, with 6 points of space before and after each caption.

Figura 6. A typical figure

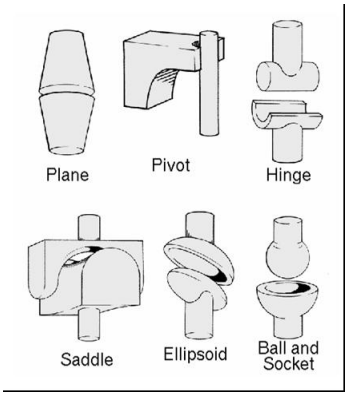

Figura 7. This figure is an example of a figure caption taking more than one line and justified considering margins mentioned in Section 14.

In tables, try to avoid the use of colored or shaded backgrounds, and avoid thick, doubled, or unnecessary framing lines. When reporting empirical data, do not use more decimal digits than warranted by their precision and reproducibility. Table caption must be placed before the table (see Table 1) and the font used must also be Helvetica, 10 point, boldface, with 6 points of space before and after each caption.

15. Images

All images and illustrations should be in black-and-white, or gray tones, excepting for the papers that will be electronically available (on CD-ROMs, internet, etc.). The image resolution on paper should be about 600 dpi for black-and-white images, and 150-300 dpi for grayscale images. Do not include images with excessive resolution, as they may take hours to print, without any visible difference in the result.

Tabela 1. Variables to be considered on the evaluation of interaction techniques

	Chessboard top view	Chessboard perspective view
Selection with side movements	6.02 ± 5.22	7.01±6.84
Selection with in-depth movements	6.29±4.99	12.22±11.33
Manipulation with side movements	4.66± 4.94	3.47±2.20
Manipulation with in-depth movements	5.71 ±4.55	5.37 ±3.28

16. References

Bibliographic references must be unambiguous and uniform. We recommend giving the author names references in brackets, e.g., [Boulic and Renault 1991], and [Smith and Jones 1999].

The references must be listed using 12 point font size, with 6 points of space before each reference. The first line of each reference should not be indented, while the subsequent should be indented by 0.5 cm.

Referências

Bhargava, A. Y. (2017). *Entendendo algoritmos: um guia ilustrado para programadores e outros curiosos*. Novatec, 1st edition. p. 46-49.

Boulic, R. and Renault, O. (1991). 3d hierarchies for animation. In Magnenat-Thalmann, N. and Thalmann, D., editors, *New Trends in Animation and Visualization*. John Wiley & Sons ltd.

Gamma, E., Helm, R., Johnson, R., and Vlissides, J. (1995). *Design Pattens: Elements of Reusable Object-Oriented Software*. Addison-Wesley. Disponível em: http://www.javier8a.com/itc/bd1/articulo.pdf. Acesso em: 17 jan. 2025. p.233-242.

Guru, R. (©2014-2025). Command. Disponível em: https://refactoring.guru/pt-br/design-patterns/command. Acesso em: 19 jan. 2025.

Pissanetzki, S. (1984). *Sparse Matrix Technology*. Academic Press, online edition. Disponível em: https://encurtador.com.br/jb5rn. Acesso em: 17 jan. 2025. p.1-34.

Schildt, H. (1997). *C Completo e total*. Makkron Books, 3th edition. Disponível em: https://www.inf.ufpr.br/lesoliveira/download/c-completo-total.pdf. Acesso em 16 jan. 2025. p. 566-571.

Smith, A. and Jones, B. (1999). On the complexity of computing. In Smith-Jones, A. B., editor, *Advances in Computer Science*, pages 555–566. Publishing Press.