Engenharia e Ciência da Computação – Estruturas de Dados

Trabalho 1 – Matrizes Esparsas

Informações Gerais

- **Data Limite de Entrega**: 21/05/2023 (23:59).
- **Pontuação**: 10 pontos (10% da nota do semestre) + 2 pontos extras.
- **Formato de Entrega**: Os arquivos produzidos no trabalho devem ser compactados em formato **.zip** e submetidos na tarefa do ambiente virtual de aprendizagem (AVA).
- Os trabalhos devem ser desenvolvidos individualmente.
- **Importante**: Trabalhos entregues após a data limite sem justificativa com comprovação documental (atestado médico, etc.), ou que não estiverem de acordo com a especificado receberão nota zero.

Contextualização

A solução de diversos problemas em computação, engenharia, física e outras áreas do conhecimento demandam a utilizam matrizes e suas operações. Matrizes são ditas esparsas quando são predominantemente preenchidas com zero. Matrizes diagonais como a identidade são exemplos de matrizes esparsas. A Figura 1 (a) ilustra uma matriz esparsa, enquanto a Figura 1 (b) ilustra uma matriz densa, na qual a maioria dos elementos é diferente de zero.

$$\begin{bmatrix} -1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 2 & 0 \end{bmatrix} \qquad \begin{bmatrix} 6 & 5 & 1 \\ 7 & 5 & 0 \\ 2 & 2 & 1 \end{bmatrix}$$

(a) Matriz esparsa (b) Matriz densa

Figura 1: Exemplos de matrizes esparsas e densas.

Nos casos em que as matrizes possuem altas dimensões, a utilização de uma representação especial para matrizes esparsas pode reduzir substancialmente o uso de memória e o custo computacional das operações. Existem diversas formas de representar matrizes esparsas, cada um buscando o ganho de performance em um subconjunto de operações. Contudo, todas as representações partem do pressuposto de que só precisamos armazenar e, sempre que possível, utilizar em operações, os valores diferentes de zero.

Tarefa

O objetivo deste trabalho é criar um tipo abstrato de dados para representar matrizes esparsas. Você deverá usar uma estrutura encadeada para armazenar elementos diferentes de zero da matriz:

- Cada célula não nula deverá armazenar sua posição (linha, coluna), o valor *float* da célula, um ponteiro para a próxima célula na linha e um ponteiro para a próxima célula

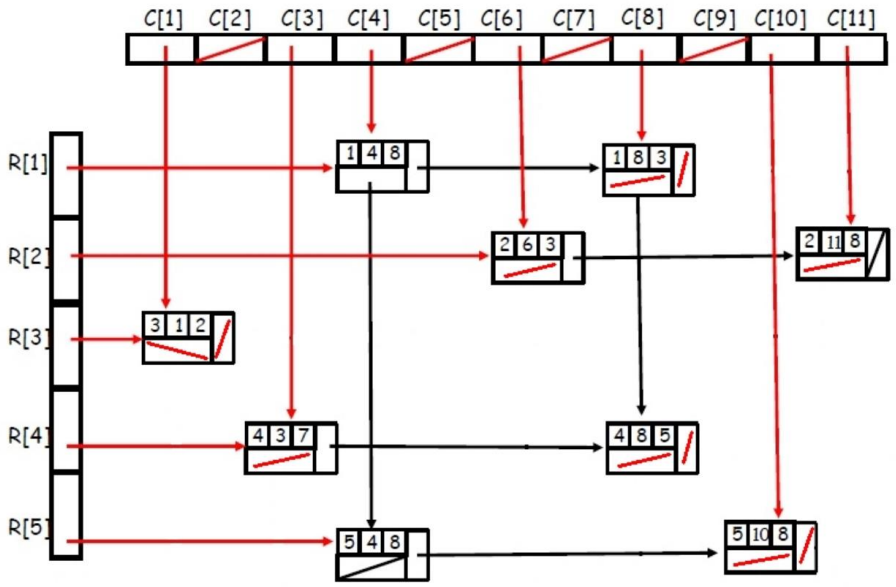
na coluna. Desta forma, cada linha e cada coluna da matriz formam uma lista com encadeamento simples.

- Deverá ser criada uma estrutura externa para armazenar um vetor de ponteiros para a cabeça de cada linha e um vetor de ponteiros para a cabeça de cada coluna, além do tamanho destes vetores.

Esta forma de representar os dados permite que a iteração sobre linhas e colunas seja feita de forma eficiente. A Figura 0 (a) ilustra uma matriz 5 x 11 e a Figura 0 (b) ilustra sua representação em formato de matriz esparsa.

0	0	0	8	0	0	0	3	0	0
0	0	0	0	0	3	0	0	0	8
2	0	0	0	0	0	0	0	0	0
0	0	7	0	0	0	0	5	0	0
0	0	0	8	0	0	0	0	8	0

(a) Matriz Original



(b) Matriz Esparsa

Figura 0: Ilustração de como uma matriz será representada em formato de matriz esparsa neste trabalho.

Devem ser implementadas as operações listadas abaixo para manipulação das matrizes. Todas as operações devem assumir que os índices de linhas e colunas se iniciam em zero. As operações devem retornar novas matrizes, isto é, não devem ser realizadas *in-place*. Para cada função, deve ser adicionado um comentário no código com a complexidade de tempo da operação usando a notação O e justificando brevemente o porquê de você achar que esta é a complexidade. Podem ser implementadas funções auxiliares além das listadas sempre que julgar necessário.

Funções:

- Criar matriz e destruir matriz.
- Atribuir valores à uma célula a partir da linha e coluna.
- Ler valores de uma célula a partir da linha e coluna.

- Somar matrizes.
- Multiplicar matrizes por um escalar.
- Multiplicar matrizes.
- Multiplicação ponto a ponto entre matrizes: dadas duas matrizes M1 e M2, com as mesmas dimensões, a operação gera como saída uma matriz M3 com as mesmas dimensões das entradas e tal que M3(i, j) = M1(i, j) * M2(i, j).
- Trocar (swap) duas linhas.
- Trocar (swap) duas colunas.
- Slice: retornar a submatriz retangular definida por um ponto esquerdo superior (início) e um ponto direito inferior (fim). A Figura 2 ilustra a operação.

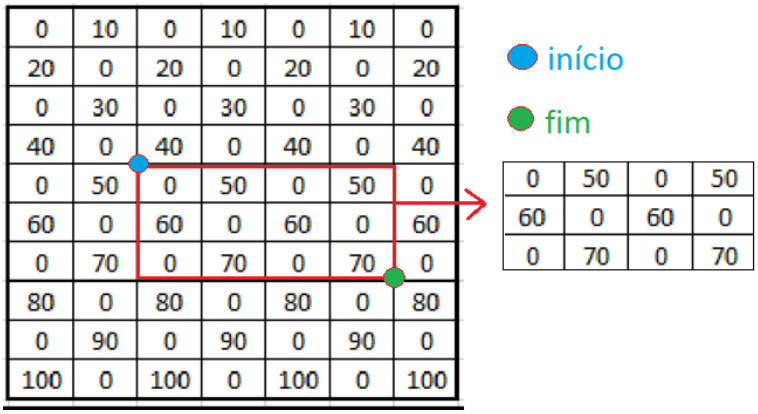


Figura 2: Visualização da operação de *slicing* que retorna uma submatriz a partir dos índices iniciais e finais.

- Transpor uma matriz.
- Realizar a convolução de uma matriz por outra matriz chamada de kernel. Vamos implementar uma versão da operação de convolução que gera como saída uma matriz do mesmo tamanho da entrada. O valor da célula (i, j) na matriz de saída será obtida fazendo:
    - Passo 1: a multiplicação ponto a ponto do kernel pela submatriz da entrada centrada na posição (i, j) e de mesmo tamanho do kernel;
    - Passo 2: a soma dos itens resultantes da operação anterior.

  As Figuras 3 e 4 ilustram como as células de saída (0, 0) e (0, 3) seriam calculadas, respectivamente. **Importante:** Nestas figuras, a matriz de saída está perdendo tantas linhas e colunas quanto a metade do tamanho do kernel. Como indicado na Figura 5, para prevenir este efeito, **vamos realizar um tratamento de borda e assumir que os índices das células fora da matriz possuem valor zero**. Note que não é necessário de fato adicionar a borda na matriz, mas apenas assumir que se os índices da célula estiverem fora dos limites da matriz, deve ser retornado valor zero.
    - Veja https://youtu.be/JxTzVAF-grU?t=337 .
    - Veja https://towardsdatascience.com/intuitively-understanding-convolutions-for-deep-learning-1f6f42faee1 .
- Mostrar uma matriz na tela em formato esparso (apenas os itens diferentes de zero).
- Mostrar uma matriz na tela em formato denso.
- Salvar matriz esparsa em um arquivo binário. Apenas os valores não nulos devem ser salvos.
- Ler matriz esparsa de um arquivo binário.

Devem ser criados um ou mais programas para testar todas as funções implementadas. Para cada teste deve ser descrita a saída esperada em um comentário.

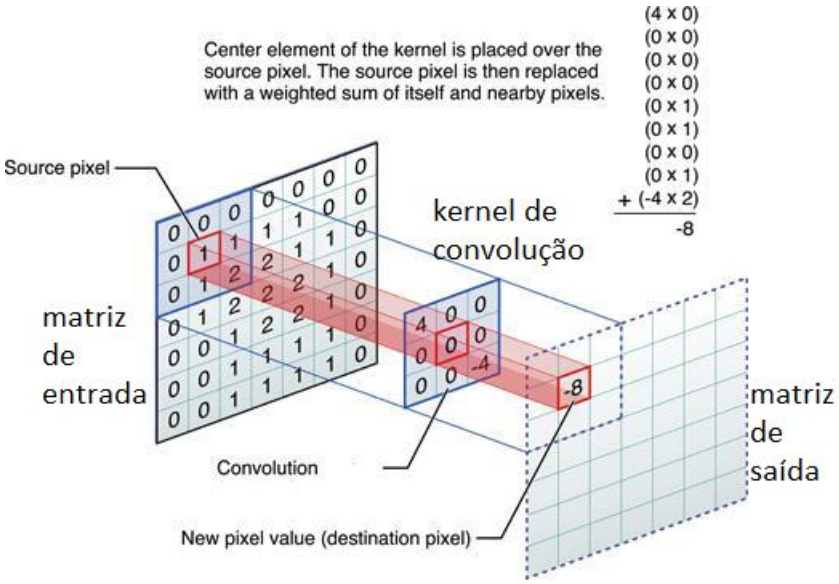


Figura 3: Ilustração de um passo da operação de convolução.
Imagem modificada de https://medium.com/@bdhuma/6-basic-things-to-know-about-convolution-daef5e1bc411

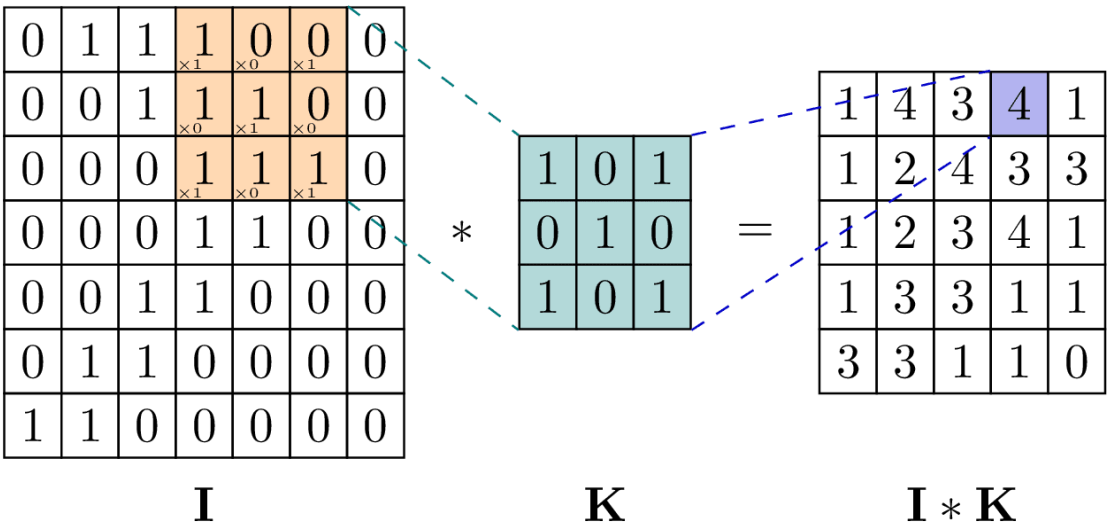


Figura 4: Terceiro passo da operação de convolução.
Figura extraída de https://tikz.net/conv2d/ .

Pontos Extras

- +1 ponto extra: resolver um sistema linear representado em formato de matriz usando a eliminação de Gauss.
- +0.5 pontos extras: adicionar uma função para cálculo do determinante de matrizes quadradas de qualquer tamanho.
- +0.5 pontos extras: calcular a inversa de matrizes quadradas de qualquer tamanho.

Punições

- Erros no *valgrind* e não liberar espaços alocados resultarão em nota zero.
- O objetivo de utilizar o formato de matriz esparsa é reduzir a quantidade de computação e de memória em relação ao uso de matrizes densas. Portanto, realizar as operações como se as matrizes fossem densas desnecessariamente resultará em punições graves na nota.

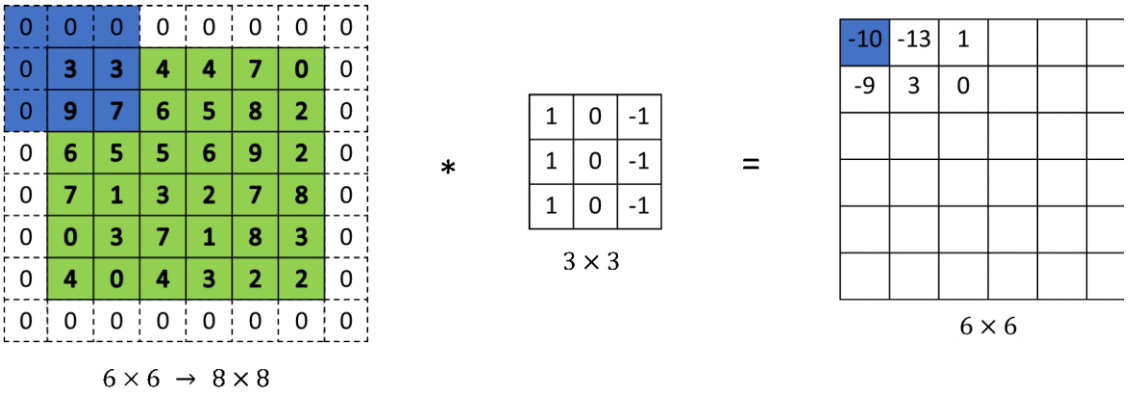


Figura 5: Para gerar uma matriz de saída do mesmo tamanho da entrada na operação de convolução, vamos assumir que as células exteriores à matriz possuem valor zero. Isto equivale a adicionar linhas e colunas com zeros ao redor da matriz original (em verde na imagem).