Matrices

CISC 2210 — CUNY Brooklyn College
Lecture Notes

Matrices: Intro

A **matrix** (plural: **matrices**) is a two-dimensional rectangular data structure. Examples:

1. $\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$
2. $\begin{bmatrix} \text{😀} \\ \text{😁} \\ \text{😎} \\ \text{😜} \end{bmatrix}$
3. $\begin{bmatrix} \text{Bla} \end{bmatrix}$
4. $\begin{bmatrix} a & b \\ c & d \\ e & f \\ g & h \\ i & j \end{bmatrix}$
5. $\begin{bmatrix} 65.7 & 62.4 & 73.8 & 76.6 & 75.0 \end{bmatrix}$
6. $\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

Notation: Just as for other data container we've seen (sets and sequences,) matrices will be named using capital letters, such as A (`$A$`), B, or M. Many textbooks also use boldfaced letters, such as $\mathbf{A}$ (`$\textbf{A}$`), $\mathbf{B}$, or $\mathbf{M}$. Example:

7. $A = \begin{bmatrix} \text{Foo} & \text{Bar} & \text{Baz} \end{bmatrix}$ (`$A = \begin{bmatrix} \text{Foo} & \text{Bar} & \text{Baz} \end{bmatrix}$`)

Matrices: Intro

$\LaTeX$ code for matrix examples 1 – 6 on the previous slide:

1. `$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$`
2. `$\begin{bmatrix} 😀 \\ 😁 \\ 😎 \\ 😜 \end{bmatrix}$`
3. `$\begin{bmatrix} \text{Bla} \end{bmatrix}$`
4. `$\begin{bmatrix} a & b \\ c & d \\ e & f \\ g & h \\ i & j \end{bmatrix}$`
5. `$\begin{bmatrix} 65.7 & 62.4 & 73.8 & 76.6 & 75.0 \end{bmatrix}$`
6. `$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$`

- $\LaTeX$ lets us create matrices using a matrix environment, e.g. `$\begin{bmatrix} ... \end{bmatrix}$`.
  - You must include the `\usepackage{amsmath}` package in your source code to be able to use this environment.
  - Various matrix environments / styles are displayed on the next slide.
- You will use `&` as separators between matrix elements on the same row, and `\\` to create another row of elements.

Matrices: Intro

Visual comparison between matrix styles:

Example #	8	9	10	11	12	13
Matrix	$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$	$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$	$\begin{Bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{Bmatrix}$	$\begin{vmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{vmatrix}$	$\begin{Vmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{Vmatrix}$	$\begin{matrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{matrix}$
Style Type	Square brackets	Parentheses	Curly braces	Vertical bars	Double vertical bars	None
LaTeX Matrix Environment Name	`bmatrix`	`pmatrix`	`Bmatrix`	`vmatrix`	`Vmatrix`	`matrix`

In these lecture notes, we'll use `bmatrix` matrices (due to stylistic reasons: bracket matrices look stable and solid.) The code for the `bmatrix` in example #8 is `$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$`.

Matrices: Intro

We could implement a matrix in Java as a two-dimensional (2D) array:

```
int[][] matrix = new int[n][m];
```

This matrix has `n` rows and `m` columns. To populate it with elements, we'll use a nested `for` loop:

```
for (int i = 0; i < n; i++)
    for (int j = 0; j < m; j++)
        matrix[i][j] = i + j;  // just an example
```

Example: 14. If we set `n = m = 3` and use the above code, we will get the matrix: $\begin{bmatrix} 0 & 1 & 2 \\ 1 & 2 & 3 \\ 2 & 3 & 4 \end{bmatrix}$.

Any questions so far?

Matrix Properties / Features

Now that we know what a matrix is, let's see what features matrices have, and get to know some special matrices.

Learning matrix features is important since when we learn about matrix operations in several slides from know, we'll need to verify whether a matrix has a certain feature before using it in an operation.

1. **Matrix size**. Each matrix is built from rows and columns of elements:
   - **Rows** are horizontal arrays of elements, and
   - **Columns** are vertical arrays of elements running throughout the matrix.

   Let n be equal to the number of rows and m equal to the number of columns in some matrix A.
   - The **size** (also called: **dimensions** or **order**) of A is denoted as $n \times m$ (`$n \times m$`).
   - The matrix $A_{n \times m}$ (`$A_{n \times m}$`) or $A_{n,m}$ (`$A_{n, m}$`) is a matrix with n rows and m columns.
     - A is then called an $n \times m$ matrix (read as "An n by m matrix").
   - Knowing what the size of a matrix is will help us decide whether a certain matrix operation is applicable or not.

Matrix Properties / Features

2. **Elements**. A **matrix element** or **entry** is the content of a matrix cell (the junction of a row and a column).
   - A matrix may include elements of various types (meaning, it can be heterogeneous), but usually it will contain elements of the same type (e.g., Boolean values, integers, or real numbers), and thus be homogeneous.
     - The rest of the lecture notes will focus mostly on matrices populated with numbers, since many matrix operations have significance only for numerical data.
   - We refer to an element on row i and column j as a_{ij} (`$a_{ij}$`), $a_{i,j}$ (`$a_{i,j}$`), $A(i,j)$ (`$A(i,j)$`), or $A[i,j]$ (`$A[i,j]$`).
     - In these lecture notes, we will use the notation $a_{i,j}$.
     - Examples: $a_{3,1}$, $b_{18,5}$, and $m_{6,8}$.
   - Using the element notation, can also denote matrix A as $A = [a_{i,j}]$.
   - ***Note***: Unlike Java, in these lecture notes, we start counting rows from 1 and columns from 1 (not from 0).
     - Example: In 5. $T = \begin{bmatrix} \mathbf{65.7} & 62.4 & 73.8 & 76.6 & 75.0 \end{bmatrix}$, the element 65.7 is $t_{1,1}$, not $t_{0,0}$.

Matrix Properties / Features

Examples:

#	1	2	3	4	5	6
Name	A	E	S	L	T	I
Matrix	$\begin{bmatrix} 1 & 2 & 3 \\ \mathbf{4} & 5 & 6 \end{bmatrix}$	$\begin{bmatrix} 😀 \\ 😁 \\ 😎 \\ 😜 \end{bmatrix}$	$\begin{bmatrix} \mathbf{Bla} \end{bmatrix}$	$\begin{bmatrix} a & b \\ c & d \\ e & f \\ g & h \\ i & \boldsymbol{j} \end{bmatrix}$	$\begin{bmatrix} 65.7 & 62.4 & 73.8 & \mathbf{76.6} & 75.0 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \mathbf{0} \\ 0 & 0 & 0 & 1 \end{bmatrix}$
Size	2×3	4×1	1×1	5×2	1×5	4×4
Sample Item	$a_{2,1} = 4$	$e_{3,1} = 😎$	$s_{1,1} = \text{Bla}$	$l_{5,2} = j$	$t_{1,4} = 76.6$	$i_{3,4} = 0$

Matrix Properties / Features

Generic matrices whose size is unspecified (or given by $n \times m$) can be created using ellipses (. . .). For example:

15. $$A = \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & \dots & a_{1,m} \\ a_{2,1} & a_{2,2} & a_{2,3} & \dots & a_{2,m} \\ a_{3,1} & a_{3,2} & a_{3,3} & \dots & a_{3,m} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & a_{n,3} & \dots & a_{n,m} \end{bmatrix}$$

or for $n \times n$:

16. $$S = \begin{bmatrix} s_{1,1} & s_{1,2} & s_{1,3} & \dots & s_{1,n} \\ s_{2,1} & s_{2,2} & s_{2,3} & \dots & s_{2,n} \\ s_{3,1} & s_{3,2} & s_{3,3} & \dots & s_{3,n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ s_{n,1} & s_{n,2} & s_{n,3} & \dots & s_{n,n} \end{bmatrix}$$

```
$A = \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & \dots & a_{1,m} \\ a_{2,1} & a_{2,2} & a_{2,3} & \dots & a_{2,m} \\ a_{3,1} & a_{3,2} & a_{3,3} & \dots & a_{3,m} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & a_{n,3} & \dots & a_{n,m} \end{bmatrix}$
```

Matrix Properties / Features

3. **Diagonals**. Being a grid of rows and columns, a matrix consists of multiple diagonals, but one diagonal has a special significance: the main diagonal.

   The **main diagonal** of the matrix $A_{n\times m} = [a_{i,j}]$ is the array of all elements of the form: $a_{i,i}$, where $1 \leq i \leq \min(n,m)$. That is, $\text{diag}(A) = [a_{1,1}, a_{2,2}, a_{3,3}, \ldots, a_{\min(n,m),\min(n,m)}]$. Examples:

Matrix	$\begin{bmatrix} \mathbf{1} & 2 & 3 \\ 4 & \mathbf{5} & 6 \end{bmatrix}$	$[\mathbf{Bla}]$	$\begin{bmatrix} \boldsymbol{a} & b \\ c & \boldsymbol{d} \\ e & f \\ g & h \\ i & j \end{bmatrix}$	$[\mathbf{65.7} \quad 62.4 \quad 73.8 \quad 76.6 \quad 75.0]$	$\begin{bmatrix} \mathbf{1} & 0 & 0 & 0 \\ 0 & \mathbf{1} & 0 & 0 \\ 0 & 0 & \mathbf{1} & 0 \\ 0 & 0 & 0 & \mathbf{1} \end{bmatrix}$
Name	$A_{2\times 3}$	$S_{1\times 1}$	$L_{5\times 2}$	$T_{1\times 5}$	$I_{4\times 4} = I_4$
Main Diag	$\text{diag}(A) = [1, 5]$	$\text{diag}(S) = [\text{Bla}]$	$\text{diag}(L) = [a, d]$	$\text{diag}(T) = [65.7]$	$\text{diag}(I) = [1, 1, 1, 1]$

Matrix Properties / Features

We often want to describe the **set of all matrices** of a given size and type.

General Notation: $\mathcal{M}_{n\times m}(S)$ (`$\mathcal{M}_{n \times m}(S)$`) or $\mathcal{M}_{n,m}(S)$ (`$\mathcal{M}_{n,m}(S)$`) denotes the set of all $n \times m$ matrices whose entries come from the set S. Examples:

- $\mathcal{M}_{n\times m}(\mathbb{R})$: all $n \times m$ matrices with real entries
- $\mathcal{M}_{n\times m}(\mathbb{Z})$: all $n \times m$ matrices with integer entries
- $\mathcal{M}_{n\times m}(\mathbb{Q})$: all $n \times m$ matrices with rational entries
- $\mathcal{M}_{n\times m}(\mathbb{C})$: all $n \times m$ matrices with complex entries

Alternative Notation: $S^{n\times m}$, meaning all $n \times m$ matrices over the set S.

Example: 1. $A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$ so $A \in \mathcal{M}_{2\times 3}(\mathbb{P})$ or $A \in \mathbb{P}^{2\times 3}$.

Matrix Properties / Features

Fun Class Activity:

1. Consider the matrix: 17. $M = [m_{i,j}] = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \\ j & k & l \end{bmatrix}$.
    1. What is the size of M?
    2. What is $m_{2,3}$ equal to?
    3. List the elements of $\mathrm{diag}(M)$.
2. On slide 8, we introduced multiple notations for a matrix element, but said that we will use only $a_{i,j}$.
    1. Explain one problem that one of the other notations, a_{ij}, might impose. [***Hint***: On what row and column is the element a_{132} located?]
    2. Provide one easy solution to this problem (other than using the $a_{i,j}$ notation.)

Matrix Properties / Features

Any questions so far?

Special Matrices

1. A **square matrix** has the same number of rows as columns.
   - This means that, if n is the number of rows (and columns), then the size of a square matrix is $n \times n$
   - We denote a square matrix A as $A_{n \times n}$. Morever, since the number of rows and columns is equal, we can conveniently use the shorthand notation: A_n (`$A_n$`).

Examples:

3. $S_1 = [\text{Bla}]$

6. $I_4 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

8. $N_3 = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$

14. $M_3 = \begin{bmatrix} 0 & 1 & 2 \\ 1 & 2 & 3 \\ 2 & 3 & 4 \end{bmatrix}$

18. $D_2 = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$

Special Matrices

2. A **singleton matrix** matrix consists only of one element.
   - This is a square matrix of size 1×1
   - We denote a singleton matrix A as $A_{1\times 1} = A_1 = [a_{1,1}] = [a]$ (`A_{1 \times 1} = A_1 = [a_{1,1}] = [a]`).
     - Note that a singleton matrix that contains a number isn't a number: this is a still a matrix. That is, if $A = [a]$ is singleton, then $A \notin \mathbb{R}$; instead, $a \in \mathbb{R}$ and $A \in \mathcal{M}_{1\times 1}(\mathbb{R}) = \mathcal{M}_1(\mathbb{R})$.

Examples:

3. $S_1 = [\text{Bla}]$ 19. $I_1 = [1]$ 20. $Z_1 = [0]$ 21. $T_1 = [90.3]$ 22. $E_1 = [$  $]$

Special Matrices

3. An **identity matrix** is a square matrix that has 1s on its main diagonal, and 0s elsewhere.
   - We use the letter I to denote an identity matrix. For example, the identity matrix of size 3×3 is $I = I_{3\times 3} = I_3$.
     - For the identity $I_n = [i_{j,k}]$, we have $i_{j,j} = 1$, where $1 \leq j \leq n$, and $i_{j,k} = 0$, where $1 \leq j, k \leq n$ but $j \neq k$.

Examples:

19. $I_1 = \begin{bmatrix} 1 \end{bmatrix}$ 23. $I_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ 24. $I_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ 6. $I_4 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$ 25. $I_5 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$

Special Matrices

Any questions so far?

Special Matrices

4. A **diagonal matrix** is a square matrix that has 0s anywhere outside the main diagonal.
   - The elements on the diagonal could be anything, including any 0s.
   - We could use the letter D to denote a diagonal matrix. For example, $D = D_{2\times 2} = D_2$.
     - For the diagonal matrix $D_n = [d_{i,j}]$, we have $d_{i,j} = 0$, where $1 \leq i, j \leq n$ but $i \neq j$.
     - We can also say: $D_n = \text{diag}(d_{1,1}, d_{2,2}, \ldots, d_{n,n})$, where $[d_{1,1}, d_{2,2}, \ldots, d_{n,n}]$ are the diagonal elements.
       - ***Note***: $D_n = \text{diag}(d_{1,1}, d_{2,2}, \ldots, d_{n,n})$ is a matrix of size $n \times n$, while $\text{diag}(D_n)$ is a one dimensional array $(1 \times n)$ of the elements on the diagonal of D_n.
   - Singleton matrices and identity matrices are specific cases of diagonal matrices.

Examples:

3. $S_1 = [\text{Bla}]$ 22. $E_1 = [$  $]$ 23. $I_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ 26. $L_3 = \begin{bmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & c \end{bmatrix}$ 27. $Q_3 = \begin{bmatrix} -19 & 0 & 0 \\ 0 & 2.2 & 0 \\ 0 & 0 & \frac{1}{2} \end{bmatrix}$

Special Matrices

5. An **upper triangular matrix** is a square matrix that has 0s anywhere below the main diagonal.
   - The elements on the diagonal and above it could be anything, including any 0s.
   - We could use the letter U to denote an upper triangular matrix.
     - For the upper triangular matrix $U_n = [u_{i,j}]$, we have $u_{i,j} = 0$, where $1 \leq i, j \leq n$ and $i > j$.
   - A diagonal matrix is a specific case of an upper triangular matrix.

Examples:

22. $E_1 = [\ ]$ 

28. $U_2 = \begin{bmatrix} 1 & 2 \\ 0 & 1 \end{bmatrix}$

29. $U_3 = \begin{bmatrix} a & b & c \\ 0 & d & e \\ 0 & 0 & f \end{bmatrix}$

30. $U_4 = \begin{bmatrix} -19 & 0 & 5 & 4e3 \\ 0 & 1.11 & 7 & 0 \\ 0 & 0 & 2^{100} & \frac{1}{2} \\ 0 & 0 & 0 & 3 \end{bmatrix}$

Special Matrices

6. A **lower triangular matrix** is a square matrix that has 0s anywhere above the main diagonal.
   - The elements on the diagonal and below it could be anything, including any 0s.
   - We could use the letter L to denote a lower triangular matrix.
     - For the lower triangular matrix $L_n = [l_{i,j}]$, we have $l_{i,j} = 0$, where $1 \leq i, j \leq n$ and $i < j$.
   - A diagonal matrix is a specific case of a lower triangular matrix, too.

Examples:

21. $E_1 = [$  $]$

31. $L_2 = \begin{bmatrix} 1 & 0 \\ 2 & 1 \end{bmatrix}$

32. $L_3 = \begin{bmatrix} a & 0 & 0 \\ b & c & 0 \\ d & e & f \end{bmatrix}$

33. $L_4 = \begin{bmatrix} \log 5 & 0 & 0 & 0 \\ 7 & 1.11 & 0 & 0 \\ 0 & \frac{1}{2} & 2^{100} & 0 \\ 0.4 & 9e2 & 0 & 1 \end{bmatrix}$

Special Matrices

7. A **sparse matrix** is a matrix that has a relatively large number of **zero elements**: elements whose value is equal to 0 in it.
   - What "relatively large" means would depend on the algorithm. For instance, one algorithm would consider matrices with 70% or more zero elements as sparse, while another would require 90%. This is called the **sparsity threshold**.
     - In our lecture notes (and HW and exams,) we set the sparsity threshold to 50%.
   - The **sparsity ratio** for a matrix $A_{n\times m}$ is the following number: $\text{sparsity}(A) = \frac{\text{number of zero elements}}{n \cdot m}$. For A to be considered sparse in our course, the following must be true: $\text{sparsity}(A) \geq 0.5$.

Examples of sparse matrices:

27. $Q_3 = \begin{bmatrix} -19 & 0 & 0 \\ 0 & 2.2 & 0 \\ 0 & 0 & \frac{1}{2} \end{bmatrix}$ 6. $\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$ 20. $Z_1 = [0]$ 34. $M = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 5 \end{bmatrix}$ 35. $N = \begin{bmatrix} 0 & 6 \\ 3 & 0 \\ 0 & 10 \end{bmatrix}$

Special Matrices

8. A **dense matrix** is a matrix that has a relatively small number of **zero elements** in it.
   - 'Dense' is the opposite of 'sparse': a sparse matrix is NOT dense, and vice versa.
   - The **density ratio** for a matrix $A_{n\times m}$ is the following number: $\text{density}(A) = \frac{\text{number of nonzero elements}}{n \cdot m}$.
     - Note that: $\text{density}(A) = 1 - \text{sparsity}(A)$.
     - For A to be considered dense in our course, the following must be true: $\text{density}(A) > 0.5$.

Examples of dense matrices:

19. $I_1 = \begin{bmatrix} 1 \end{bmatrix}$

14. $M_3 = \begin{bmatrix} 0 & 1 & 2 \\ 1 & 2 & 3 \\ 2 & 3 & 4 \end{bmatrix}$

5. $T = \begin{bmatrix} 65.7 & 62.4 & 73.8 & 76.6 & 75.0 \end{bmatrix}$

36. $B = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{bmatrix}$

Special Matrices

9. A **row vector** (also called a **row matrix**) is a matrix of the size $1 \times n$.
   - 'Vector' is a synonym of 'array'.
   - The number n in a $1 \times n$ row vector is called the **vector size** and stands for the number of elements in the vector.
   - Every $n \times m$ matrix consists of at least one row vector.
   - We denote a row vector using a lowercase letter, (usually 'v' for vector) as follows: $\vec{v}$ (`$\vec{v}$`) or $\mathbf{v}$ (`$\textbf{v}$`), usually with a subscript, like $\vec{v}_1$ (`$\vec{v}_1$`) to tell what matrix row it corresponds to or just to keep stack of vectors we are working with.
     - ***Note***: In our vector notation, the subscript isn't the vector size: it is just some serial number.

Examples:

19. $I_1 = \vec{v}_1 = \begin{bmatrix} 1 \end{bmatrix}$

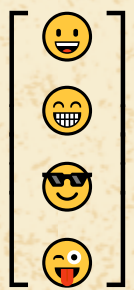
7. $A = \vec{v}_2 = \begin{bmatrix} \text{Foo} & \text{Bar} & \text{Baz} \end{bmatrix}$

5. $T = \vec{v}_3 = \begin{bmatrix} 65.7 & 62.4 & 73.8 & 76.6 & 75.0 \end{bmatrix}$

Special Matrices

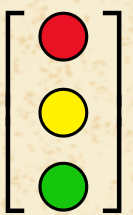
10. A **column vector** (also called a **column matrix**) is a matrix of the size $n \times 1$.
    - The number n in a $n \times 1$ column vector is the vector size of the column vector.
    - Every $n \times m$ matrix consists of at least one column vector.
    - We use the notation of row vectors to denote column vectors, too: $\vec{v}$ (`$\vec{v}$`) or $\mathbf{v}$ (`$\textbf{v}$`), usually with a subscript, like $\vec{v}_1$ (`$\vec{v}_1$`).

Examples:

2. $E = \vec{v}_4 =$ 

3. $S = \vec{v}_5 = [\text{Bla}]$

37. $\vec{v}_6 = \begin{bmatrix} x \\ y \end{bmatrix}$

38. $\vec{v}_7 =$ 

39. $\vec{v}_8 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$

Special Matrices

11. An **adjacency matrix** is a square matrix consisting of Boolean values only: 0s and 1s.
    - The adjacency matrix plays a great role in implementing various algorithms whose purpose is to answer a "yes" or "no" question, e.g., "Is there a path between two cities A and B," or "Can I get from country A to country B by taking 2 or fewer flights?"
    - We will get back to the adjacency matrix both at the end of this Topic 10 and in Topic 11: Graphs & Trees.

Examples:

19. $I_1 = \begin{bmatrix} 1 \end{bmatrix}$ 20. $Z_1 = \begin{bmatrix} 0 \end{bmatrix}$ 23. $I_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ 24. $I_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ 36. $B = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{bmatrix}$

Special Matrices

Fun Class Activity:

1. What do we call a matrix that is both upper and lower triangular?
2. Which of the special matrices we covered are always sparse (no matter what elements we put in them?)
3. We covered 11 special matrix types. Consider the matrix:

$$40.\ A = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Which of these types does A belong to? Specify all that apply (there could be more than one type.)

Special Matrices

Any questions so far?

Matrix Operations

Now that we got familiar with various matrix traits and with special types of matrices, we can introduce various **matrix operations**, which are computations we can perform using matrices.

1. **Matrix equality**. We compare two matrices to see if they are equal.

   Input: Two matrices, A and B.
   Output: A Boolean value (`true` ($=1$), or `false` ($=0$)).

   Algorithm (= description of the procedure of this matrix operation:)
   1. If the size of A isn't equal to the size of B, return `false`. Otherwise:
   2. Check if the respective elements of A and B are equal. That is, given the matrices $A_{n\times m}$ and $B_{n\times m}$ (of the same size), check if $a_{i,j} = b_{i,j}$ for every $1 \leq i \leq n$ and every $1 \leq j \leq m$. If, for at least one pair of respective elements we have $a_{i,j} \neq b_{i,j}$, then return `false`. Otherwise (if all respective elements are equal), return `true`.

Matrix Operations

1. Runtime analysis (= description of how much time it will take to run it on a computer:)
   We will compute the runtime based on how many comparisons we perform:
   1. In case the matrices aren't of the same size, we will discover this by noticing that, for A_{n_1,m_1} and B_{n_2,m_2}, either $n_1 \neq n_2$ or $m_1 \neq m_2$. Meaning, we will do either 1 or 2 comparisons. Since this is a constant number $c \leq 2$, we say that the runtime is $\mathcal{O}(1)$ (or, being specific, it is $\Theta(1)$) since, as we learned in Topic 8, $c = \mathcal{O}(1)$ no matter how large or small the constant c is.
   2. In case the matrices *are* of the same size, we will compare all their elements. Assuming the matrices have $n \times m$ elements, this will result in nm comparisons. This means that the runtime is $\mathcal{O}(nm)$ (or, being specific, it is $\Theta(nm)$).

   [**Note**: If we were to implement this algorithm in Java for example, besides doing the actual comparisons, we will of course also need to increment the row and columns indexes i and j, which also takes time to run in a computer. However, notice that, per each comparison, we will have 1 or 2 increments, which is a constant (call it c_2), and $1 + c_2$ is still a constant, so the increments still preserve the same runtime: $(1 + c_2)nm = \mathcal{O}(nm)$.]

Matrix Operations

1. Examples:

1. $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \neq \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

2. $\begin{bmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & c \end{bmatrix} \neq \begin{bmatrix} a & b & c \\ 0 & d & e \\ 0 & 0 & f \end{bmatrix}$

3. $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 5 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 5 \end{bmatrix}$

Fun Class Activity:

1. Explain why the matrices in examples 1 and 2 above aren't equal, but the ones in example 3 are.
2. Implement the matrix equality algorithm in Java (or any other computer language of your choice.)
3. State whether or not the following matrices are equal, and explain your reasoning:

$A = \begin{bmatrix} 2 & 7 & -1 & 5 \\ 3 & 4 & 8 & -3 \end{bmatrix}$ and $B = \begin{bmatrix} 2 & 7 & -1 & 5 \\ 3 & 4 & 6 & -3 \end{bmatrix}$.

Matrix Operations

2. **Matrix transpose**. We take matrix $A_{n \times m}$ and generate another matrix $A^T_{m \times n}$ (`A^T_{m \times n}`), called A's transpose, by exchanging rows with columns.

   Input: Any matrix $A_{n \times m} = [a_{i,j}]$.
   Output: The matrix $A^T_{m \times n} = [a^T_{j,i}]$.

   Algorithm:
   1. Create an empty matrix container $A^T = [a^T_{j,i}]$ of the size $m \times n$ (m rows and n columns).
   2. For every $1 \leq j \leq m$ and every $1 \leq i \leq n$, set $a^T_{j,i} = a_{i,j}$. Then, return A^T.

   Runtime analysis:
   In the 1st step, creating the matrix double array will take $\Theta(nm)$ operations. In the 2nd step, we will do $\Theta(nm)$ element assigments. In total, we will have $\Theta(nm) + \Theta(nm) = c_1 nm + c_2 nm = (c_1 + c_2)nm = \Theta(nm)$ operations.

Matrix Operations

2. Examples:

1. $\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}^T = \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix}$

2. $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}^T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

3. $\begin{bmatrix} \bullet \\ \bullet \\ \bullet \end{bmatrix}^T = [\ \bullet \quad \bullet \quad \bullet\ ]$

4. $\begin{bmatrix} a & b \\ 0 & c \end{bmatrix}^T = \begin{bmatrix} a & 0 \\ b & c \end{bmatrix}$.

Fun Class Activity:

1. What does example 2 tell you about the transpose of an identity matrix (or diagonal matrices)?
2. What does example 3 tell you about the connection between row vectors and column vectors?
3. What does example 3 tell you about the connection between upper triangular matrices and lower triangular matrices?
4. Compute the transpose of $A = \begin{bmatrix} 2 & 7 & -1 & 5 \\ 3 & 4 & 8 & -3 \end{bmatrix}$.

Matrix Operations

Any questions so far?

Matrix Operations

3. **Matrix trace**. Given a square matrix A, we find the sum of its diagonal elements, called the trace of the matrix, and denote it as $\text{tr}(A)$ (`\text{tr}(A)`).

   Input: A square matrix A_n.
   Output: $s = \text{tr}(A) = \sum_{i=1}^{n} a_{i,i}$ (usually, $s \in \mathbb{Z}$ or $s \in \mathbb{R}$.)

   Algorithm:
   1. Declare $s = 0$.
   2. For every $1 \leq i \leq n$, set $s = s + a_{i,i}$. Then, return s.

   Runtime analysis:
   The index i goes from 1 to n, so there are n additions and assignments (there is also a constant-time assignment $s = 0$ and increments of i, but they won't change the runtime). This results in an overall runtime of $\Theta(n)$.

Matrix Operations

3. Examples:

1. $\mathrm{tr}\left(\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}\right) = 1+1+1 = 3.$

2. $\mathrm{tr}\left(\begin{bmatrix} \log 5 & 0 & 0 & 0 \\ 7 & 1.11 & 0 & 0 \\ 0 & \frac{1}{2} & 2^{100} & 0 \\ 0.4 & 9e2 & 0 & 1 \end{bmatrix}\right) = \log 5 + 1.11 + 2^{100} + 1 \approx 2^{100}.$

Fun Class Activity:

1. We defined the trace operation on square matrices only. Suggest a trace definition for non-square matrices $A_{n\times m}$:
   - First, describe it shortly using words (still without math symbols).
   - Next, use math symbols to explain how the trace is computed.
   - Finally, analyze the runtime of your algorithm.

Matrix Operations

4. **Matrix addition**. Given two matrices A and B of the same size, we add them element-wise and denote the result as $A + B$.

   Input: Two matrices $A_{n \times m}$ and $B_{n \times m}$.
   Output: $C = A + B$, where $c_{i,j} = a_{i,j} + b_{i,j}$.

   Algorithm:
   1. Create a matrix C of size $n \times m$.
   2. For every $1 \leq i \leq n$, $1 \leq j \leq m$, set $c_{i,j} = a_{i,j} + b_{i,j}$. Then, return C.

   Runtime analysis: Counting additions and value assignments, we get a constant of nm, so the runtime is $\Theta(nm)$.

Matrix Operations

4. Examples:

1.
$$\begin{bmatrix} 0 & -1 \\ 2 & 3 \end{bmatrix} + \begin{bmatrix} 4 & 1 \\ -2 & 5 \end{bmatrix} = \begin{bmatrix} 0+4 & -1+1 \\ 2+(-2) & 3+5 \end{bmatrix} = \begin{bmatrix} 4 & 0 \\ 0 & 8 \end{bmatrix}$$

2.
$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \\ 0 & 1 & 2 \end{bmatrix} + \begin{bmatrix} 9 & 8 & 7 \\ 6 & 5 & 4 \\ 3 & 2 & 1 \\ 2 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 10 & 10 & 10 \\ 10 & 10 & 10 \\ 10 & 10 & 10 \\ 2 & 2 & 2 \end{bmatrix}$$

Fun Class Activity:

1. Find two different non-zero matrices A and B such that $A + B = \mathbf{0}$.
2. What must be true about A and B?

Matrix Operations

5. **Scalar multiplication**. Multiply every element of a matrix by a scalar c.

   Input: Matrix $A_{n\times m}$ and scalar c (usually, $c \in \mathbb{R}$.)
   Output: $B = cA$, where $b_{i,j} = c \cdot a_{i,j}$.

   Algorithm:
   1. Create space for matrix $B_{n\times m}$.
   2. For every $1 \leq i \leq n$ and $1 \leq j \leq n$, set $b_{i,j} = c \cdot a_{i,j}$. Then, return matrix B.

   Runtime analysis: $\Theta(nm)$.

Matrix Operations

5. Examples:

$$1.\ 2\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix} = 2\cdot\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix} = \begin{bmatrix}2\cdot 1 & 2\cdot 2\\ 2\cdot 3 & 2\cdot 4\end{bmatrix} = \begin{bmatrix}2 & 4\\ 6 & 8\end{bmatrix}$$

$$2.\ -1\begin{bmatrix}3 & -2\\ 0 & 5\end{bmatrix} = \begin{bmatrix}-3 & 2\\ 0 & -5\end{bmatrix}$$

Fun Class Activity:

1. Find a scalar $c \neq 0$ and matrix $A \neq 0$ such that $cA = A$.
2. What does this tell you about c?

Matrix Operations

6. **Inner product**. Given two vectors, compute the sum of pairwise products. We also call an inner product a **dot product** and a **linear combination**.

   Input: Vectors $u, v \in \mathbb{R}^n$ (meaning, each of u and v has n real elements in it.) Usually, u (the leftmost one) is a row vector, and v (the rightmost one) is a column vector.
   Output: $s = \sum_{i=1}^{n} (u_i \cdot v_i)$. The figure s is just a number/scalar.

   Algorithm:
   1. Set $s = 0$.
   2. For each $1 \leq i \leq n$, set $s = s + u_i \cdot v_i$. Then, return s.

   Runtime analysis: $\Theta(n)$. This is because we do n multiplications and $n - 1$ additions (and n index increments), which results in $n + (n - 1) + n = 3n - 1 = \Theta(n)$.

Matrix Operations

6. Examples:

1. $\begin{bmatrix}1 & 2 & 3\end{bmatrix} \cdot \begin{bmatrix}4\\5\\6\end{bmatrix} = 1\cdot 4 + 2\cdot 5 + 3\cdot 6 = 32$

2. $\begin{bmatrix}2 & -1\end{bmatrix} \cdot \begin{bmatrix}3\\4\end{bmatrix} = 6 - 4 = 2$

Fun Class Activity:

1. Find two non-zero vectors whose inner product is 0.
2. What geometric relationship do they have?

Matrix Operations

Any questions so far?

Matrix Operations

7. **Matrix multiplication**. Multiply rows of A with columns of B.

   Input: $A_{n\times k}$ and $B_{k\times m}$. Note that the number of columns in A must be equal to the number of rows in B; otherwise, the multiplication can't be done.
   Output: $C_{n\times m}$ where $c_{i,j} = \sum_{t=1}^{k} a_{i,t} \cdot b_{t,j}$.

   Algorithm:
   1. Create space for the matrix $C_{n\times m}$.
   2. For each $1 \le i \le n$ and $1 \le j \le n$, compute the inner product of row i and column j as follows: $c_{i,j} = \sum_{t=1}^{k} a_{i,t} \cdot b_{t,j}$. Then, return matrix C.

   Runtime analysis: For every value of i and j, we perform an inner product computation, which runs from 1 to k, hence takes $\Theta(k)$. We do this inner product nm times (remember the Multiplication Principle from the topic on Counting? That's the same idea!), so the runtime is $\Theta(k) \cdot \Theta(nm) = \Theta(knm)$. **Fun fact**: for square A_n and B_n, it is $\Theta(n^3)$.

Matrix Operations

7. Example (we will do this example in detail on the next slides:)

$$AB = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \\ 7 & 8 \end{bmatrix} \begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & 3 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 8 \\ 3 & 4 & 18 \\ 5 & 6 & 28 \\ 7 & 8 & 38 \end{bmatrix} = C$$

Fun Class Activity:

1. Find matrices A and B such that $AB \neq BA$.
2. Why does this happen?
3. What type of matrix do we get when multiplying:
    1. Two identity matrices?
    2. Two diagonal matrices?
    3. Two upper triangular (or tow lower triangular) matrices?

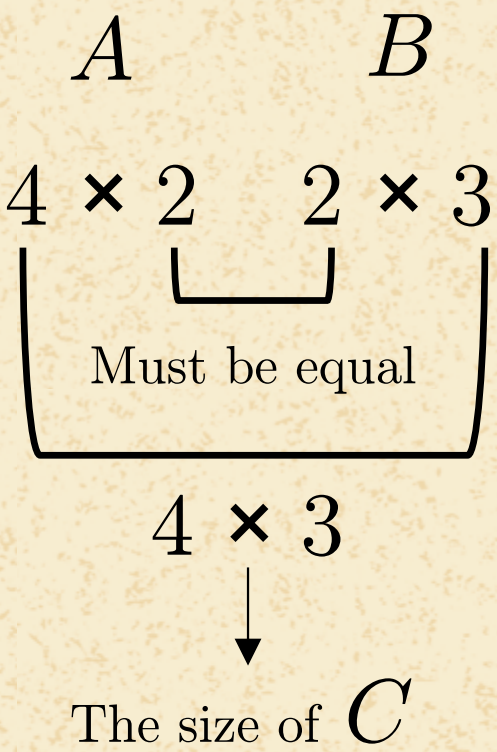
Matrix Operations: Step-By-Step Multiplication Example

7. Example:

$$AB = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \\ 7 & 8 \end{bmatrix} \begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & 3 \end{bmatrix} = \begin{bmatrix} c_{1,1} & c_{1,2} & c_{1,3} \\ c_{2,1} & c_{2,2} & c_{2,3} \\ c_{3,1} & c_{3,2} & c_{3,3} \\ c_{4,1} & c_{4,2} & c_{4,3} \end{bmatrix} = C$$

Explanation: First, we begin with an empty matrix $C_{4,3}$. We know that C will be of this size since A is of the size 4×2 and B is of the size 2×3.
The following slide contains an easy way of remembering what the dimensions of the resulting matrix are.

Matrix Operations: Step-By-Step Multiplication Example



Easy way of understanding matrix multiplication: 'inner' vs. 'outer' dimensions. Miriam Briskman, CC BY-NC 4.0.

Matrix Operations: Step-By-Step Multiplication Example

7. Example:

$$AB = \begin{bmatrix} \mathbf{1} & \mathbf{2} \\ 3 & 4 \\ 5 & 6 \\ 7 & 8 \end{bmatrix} \begin{bmatrix} \mathbf{1} & 0 & 2 \\ \mathbf{0} & 1 & 3 \end{bmatrix} = \begin{bmatrix} \mathbf{1} & c_{1,2} & c_{1,3} \\ c_{2,1} & c_{2,2} & c_{2,3} \\ c_{3,1} & c_{3,2} & c_{3,3} \\ c_{4,1} & c_{4,2} & c_{4,3} \end{bmatrix} = C$$

Explanation: We do $c_{1,1} = 1 \cdot 1 + 2 \cdot 0 = 1 + 0 = 1$. This is the element on row 1 and column 1, so we used row 1 of matrix A and column 1 of matrix B.

Matrix Operations: Step-By-Step Multiplication Example

7. Example:

$$AB = \begin{bmatrix} \mathbf{1} & \mathbf{2} \\ 3 & 4 \\ 5 & 6 \\ 7 & 8 \end{bmatrix} \begin{bmatrix} 1 & \mathbf{0} & 2 \\ 0 & \mathbf{1} & 3 \end{bmatrix} = \begin{bmatrix} 1 & \mathbf{2} & c_{1,3} \\ c_{2,1} & c_{2,2} & c_{2,3} \\ c_{3,1} & c_{3,2} & c_{3,3} \\ c_{4,1} & c_{4,2} & c_{4,3} \end{bmatrix} = C$$

Explanation: We do $c_{1,2} = 1 \cdot 0 + 2 \cdot 1 = 0 + 2 = 2$. This is the element on row 1 and column 2, so we used row 1 of matrix A and column 2 of matrix B.

Matrix Operations: Step-By-Step Multiplication Example

7. Example:

$$AB = \begin{bmatrix} \mathbf{1} & \mathbf{2} \\ 3 & 4 \\ 5 & 6 \\ 7 & 8 \end{bmatrix} \begin{bmatrix} 1 & 0 & \mathbf{2} \\ 0 & 1 & \mathbf{3} \end{bmatrix} = \begin{bmatrix} 1 & 2 & \mathbf{8} \\ c_{2,1} & c_{2,2} & c_{2,3} \\ c_{3,1} & c_{3,2} & c_{3,3} \\ c_{4,1} & c_{4,2} & c_{4,3} \end{bmatrix} = C$$

Explanation: We do $c_{1,3} = 1 \cdot 2 + 2 \cdot 3 = 2 + 6 = 8$. This is the element on row 1 and column 3, so we used row 1 of matrix A and column 3 of matrix B.

Matrix Operations: Step-By-Step Multiplication Example

7. Example:

$$AB = \begin{bmatrix} 1 & 2 \\ \mathbf{3} & \mathbf{4} \\ 5 & 6 \\ 7 & 8 \end{bmatrix} \begin{bmatrix} \mathbf{1} & 0 & 2 \\ \mathbf{0} & 1 & 3 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 8 \\ \mathbf{3} & c_{2,2} & c_{2,3} \\ c_{3,1} & c_{3,2} & c_{3,3} \\ c_{4,1} & c_{4,2} & c_{4,3} \end{bmatrix} = C$$

Explanation: We do $c_{2,1} = 3 \cdot 1 + 4 \cdot 0 = 3 + 0 = 3$. This is the element on row 2 and column 1, so we used row 2 of matrix A and column 1 of matrix B.

Matrix Operations: Step-By-Step Multiplication Example

7. Example:

$$AB = \begin{bmatrix} 1 & 2 \\ \mathbf{3} & \mathbf{4} \\ 5 & 6 \\ 7 & 8 \end{bmatrix} \begin{bmatrix} 1 & \mathbf{0} & 2 \\ 0 & \mathbf{1} & 3 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 8 \\ 3 & \mathbf{4} & c_{2,3} \\ c_{3,1} & c_{3,2} & c_{3,3} \\ c_{4,1} & c_{4,2} & c_{4,3} \end{bmatrix} = C$$

Explanation: We do $c_{2,2} = 3 \cdot 0 + 4 \cdot 1 = 0 + 4 = 4$. This is the element on row 2 and column 2, so we used row 2 of matrix A and column 2 of matrix B.

Matrix Operations: Step-By-Step Multiplication Example

7. Example:

$$AB = \begin{bmatrix} 1 & 2 \\ \mathbf{3} & \mathbf{4} \\ 5 & 6 \\ 7 & 8 \end{bmatrix} \begin{bmatrix} 1 & 0 & \mathbf{2} \\ 0 & 1 & \mathbf{3} \end{bmatrix} = \begin{bmatrix} 1 & 2 & 8 \\ 3 & 4 & \mathbf{18} \\ c_{3,1} & c_{3,2} & c_{3,3} \\ c_{4,1} & c_{4,2} & c_{4,3} \end{bmatrix} = C$$

Explanation: We do $c_{2,3} = 3 \cdot 2 + 4 \cdot 3 = 6 + 12 = 18$. This is the element on row 2 and column 3, so we used row 2 of matrix A and column 3 of matrix B.

We'll stop here: try computing the other 6 elements in a similar fashion.

Matrix Operations

Any questions so far?

A Few Matrix Applications

Solving Systems of Linear Equations Using Matrices. A system of linear equations can be written compactly as a matrix equation:

$$Ax = b$$

Here, A is a matrix of coefficients, x is a vector of variables, and b is the result vector.

There are several ways to solve this system:

1. **Gaussian elimination**: Transform the augmented matrix $[A|b]$ into row-echelon form.
2. **Matrix inverse** (if A is invertible): $x = A^{-1}b$.

A Few Matrix Applications

Example 1 (Gaussian elimination):

Solve:

$$\begin{cases} x + y = 3 \\ 2x + 3y = 7 \end{cases}$$

Augmented matrix:

$$\left[\begin{array}{cc|c} 1 & 1 & 3 \\ 2 & 3 & 7 \end{array}\right]$$

A Few Matrix Applications

Example 1 (Gaussian elimination):

Row operation: $R_2 \leftarrow R_2 - 2R_1$

$$\left[\begin{array}{cc|c} 1 & 1 & 3 \\ 0 & 1 & 1 \end{array}\right]$$

Which corresponds to

$$\begin{cases} x + y = 3 \\ 0 + y = 1 \end{cases}$$

Back substitution gives $y = 1$, $x = 2$.

A Few Matrix Applications

Example 2 (Matrix inverse):

$$A = \begin{bmatrix} 1 & 1 \\ 2 & 3 \end{bmatrix}, b = \begin{bmatrix} 3 \\ 7 \end{bmatrix}$$

First, we'll compute A^{-1}: the inverse of A. To do so, we use the matrix inverse formula for a 2×2 matrix:

$$A^{-1} = \frac{1}{\det(A)} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

where $A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$.

A Few Matrix Applications

Step 1: Compute the determinant

$$\det(A) = (1)(3) - (1)(2) = 3 - 2 = 1$$

Step 2: Swap and negate entries

$$\begin{bmatrix} d & -b \\ -c & a \end{bmatrix} = \begin{bmatrix} 3 & -1 \\ -2 & 1 \end{bmatrix}$$

Step 3: Multiply by $\frac{1}{\det(A)}$

Since $\det(A) = 1$, we get:

$$A^{-1} = \begin{bmatrix} 3 & -1 \\ -2 & 1 \end{bmatrix}$$

Verification (optional but insightful). Check that $AA^{-1} = I$:

$$\begin{bmatrix} 1 & 1 \\ 2 & 3 \end{bmatrix} \begin{bmatrix} 3 & -1 \\ -2 & 1 \end{bmatrix} = \begin{bmatrix} (3-2) & (-1+1) \\ (6-6) & (-2+3) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

So the inverse is correct.

A Few Matrix Applications

Example 2 (Matrix inverse):

Final step (solving the system):

If $b = \begin{bmatrix} 3 \\ 7 \end{bmatrix}$, then:

$$x = A^{-1}b = \begin{bmatrix} 3 & -1 \\ -2 & 1 \end{bmatrix} \begin{bmatrix} 3 \\ 7 \end{bmatrix} = \begin{bmatrix} 9-7 \\ -6+7 \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$$

A Few Matrix Applications

Matrix Eigenvalues. Eigenvalues are special scalars that describe how a matrix transforms space. "Eigen" means "its own" in German, meaning, the matrix's own value. Given a square matrix A, an eigenvalue λ is a number such that there exists a non-zero vector v satisfying:

$$Av = \lambda v$$

This means that multiplying v by A only stretches or shrinks it (by λ), without changing its direction.

To compute eigenvalues, we solve the characteristic equation:

$$\det(A - \lambda I) = 0$$

This produces a polynomial in λ, whose roots are the eigenvalues.

A Few Matrix Applications

Example 1:

Let $A = \begin{bmatrix} 2 & 0 \\ 0 & 3 \end{bmatrix}$. Then:

$$\det(A - \lambda I) = \det \begin{bmatrix} 2-\lambda & 0 \\ 0 & 3-\lambda \end{bmatrix} = (2-\lambda)(3-\lambda)$$

Setting this equal to 0, we get $\lambda = 2, 3$.

A Few Matrix Applications

Example 2:

Let $A = \begin{bmatrix} 1 & 1 \\ 0 & 2 \end{bmatrix}$. Then:

$$\det(A - \lambda I) = \det \begin{bmatrix} 1-\lambda & 1 \\ 0 & 2-\lambda \end{bmatrix} = (1-\lambda)(2-\lambda) - 1 \cdot 0 = (1-\lambda)(2-\lambda)$$

So the eigenvalues are $\lambda = 1, 2$.

Eigenvalues are widely used in computer science, including graph analysis, machine learning, and stability analysis of systems.

A Few Matrix Applications

Using Adjacency Matrices to Detect Paths Between Cities. Graphs can represent connections between cities. An adjacency matrix A encodes these connections:

$a_{i,j} = 1$ if there is a direct road from city i to city j, and 0 otherwise.

Matrix multiplication reveals paths of longer lengths:

- A^2 tells us whether there is a path of length 2.
- A^k tells us whether there is a path of length k.

If $(A^k)_{i,j} > 0$, then there exists a path of length k from city i to city j.

A Few Matrix Applications

Example:

Suppose we have 3 cities with adjacency matrix:

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

This means:
City 1 $\rightarrow$ City 2
City 2 $\rightarrow$ City 3

Compute A^2:

A Few Matrix Applications

Example:

$$A^2 = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Since $(A^2)_{1,3} = 1$, there is a path of length 2 from City 1 to City 3.

This technique is widely used in network analysis, routing, and social network connectivity.

A Few Matrix Applications

Any questions?

Matrices: Sources

Doerr, Al and Levasseur, Ken. (2024). Chapter 5. *Applied Discrete Structures*, https://discretemath.org/. License: CC BY-NC-SA: Attribution-NonCommercial-ShareAlike.

Lumen Learning. College Algebra | Matrices and Matrix Operations. License: CC BY: Attribution.