Matrix example: Random Web
- N web pages can be represented by an N-by-N Link Array A.
- $A(i,j)$ is 1 if there is a link on webpage j to webpage i.

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

Triangular traversal
- $[nr, nc] = \text{size}(M);
- \text{for } A = B\cdot E
- \text{for } D = E\cdot F
- \text{disp(M(r,c))}
- \text{end}
- \text{end}$

What should be $A, B, \ldots, E$ in order to traverse the "triangular part" of a square matrix row-wise as in Case 1? How about traversing column-wise as in Case 2?

Represent the connectivity of the web pages graphically

Web pages arranged in a circle. Bidirectional links are blue. Unidirectional link is black as it leaves page j, red when it arrives at page i.

Pictures as matrices
- "512 x 384" image as 512 x 512 array

---

Graded Prelim 1 will be available on Announcements:
- Type as it week
- Today's Lecture:
- Complete matrix example from previous lecture
- Image processing
- Type uint8
- Vectorsized code for accessing subarrays
- Announcements:
  - P4 to be posted after today’s lectures
  - Graded Prelim 2 will be available on Gradescope next week
  - Read §12.4 of Insight—learn about arithmetic in type uint8
Image files & raster data

- **File formats**
  - JPEG: Photographs, lossy
  - PNG: Graphics, lossless
  - TIFF: Technical

- **Others**
  - WebP, GIF, DNG, OpenEXR, ...

**Properties**

- **Channels**
  - Greyscale, RGB(A), YCbCr
- **Bit depth, range**
  - 8-bit, 10-bit, HDR
- **Color space, “gamma”**
  - sRGB, DCI-P3, raw
- **Subsampling**
  - 4:4:4, 4:2:0

MATLAB image features

- % Read image file into matrix
  \[ \text{mat} = \text{imread}('filename') \]

- % Plot matrix as image
  \[ \text{imshow(mat)} \]

- % Write matrix to image file
  \[ \text{imwrite(mat, 'filename')} \]

Greyness: a value in [0..255]

- **New type: uint8**
  - Integer value between 0 and 255
  - Can see types of variables in Workspace panel

Let’s put a picture in a frame

**Things to do:**

1. Read `liftingbody.png` from disk and convert it into an array
2. Show the original picture
3. Assign a black value (frame color) to the “edge pixels”
4. Show the manipulated picture

Reading a PNG file and displaying the image

- % Read jpg image and convert to a type uint8 array \( P \)
  \[ P = \text{imread}('liftingbody.png'); \]

- % Show the data in array \( P \) as an image
  \[ \text{imshow(P)} \]

% Frame a grayscale picture

\[ P = \text{imread}('liftingbody.png'); \]

% Change the “frame” color

\[ \text{imshow(P)} \]
% Frame a grayscale picture
P = imread('liftingbody.png');
imshow(P)

% Change the "frame" color
width = 50;
frameColor = 20; % dark gray

% Frame a grayscale picture
P = imread('liftingbody.png');
imshow(P)
width = 50;
frameColor = 20; % dark gray

[nr, nc] = size(P);
for r = 1:nr
    for c = 1:nc
        % At pixel (r,c)
        if (r <= width) || (r > nr - width) || ... | (c <= width) || (c > nc - width)
            P(r,c) = frameColor;
        end
    end
end
imshow(P)

Things to consider:
1. What is the type of the values in P?
2. Can we be more efficient?

Type conversions
P = imread('liftingbody.png');
% (all of) P has type uint8
framecolor = 20;
% framecolor has type double
P(r,c) = framecolor;
% RHS value is implicitly converted to type of LHS var
P(r,c) = uint8(framecolor);
% RHS value is explicitly converted to uint8

Accessing a submatrix
M = [1 2 3 4; 5 6 7 8; 9 10 11 12; 13 14 15 16];
% M refers to the whole matrix
% M(3,5) refers to one component of M

- M refers to the whole matrix
- M(3,5) refers to one component of M

See pictureFrame*.m
Accessing a submatrix

- $M$ refers to the whole matrix
- $M(3,5)$ refers to one component of $M$
- $M(2:3,3:5)$ refers to a submatrix of $M$

```
2 -1 5 0 3
3 8 6 7 .
5 -3 5 9 14
52 81 .5 7 .2
```

```
row indices
```
```
column indices
```

Pictures as matrices

```
26 29 30 28 26
216 212 142 65 32
231 232 232 198 130
232 228 224 225 215
```

```
512 x 384 image ⇒ 384 x 512 array
```

Color

- 3 different cone cells in eye means color can be represented by 3 numbers (channels)
- Cameras, displays work with Red, Green, and Blue light: RGB
- Each channel (color) represented by its own matrix “plane”
- MATLAB: `pic(row, col, ch)`
  * `pic(:,1)`: Red
  * `pic(:,2)`: Green
  * `pic(:,3)`: Blue

```
E.g., color image data is stored in a 3-d array $A$
0 ≤ $A(i,j,1)$ ≤ 255
0 ≤ $A(i,j,2)$ ≤ 255
0 ≤ $A(i,j,3)$ ≤ 255
```

Example: Mirror Image

```
1. Read LawSchool.jpg from memory and convert it into an array.
2. Manipulate the Array.
3. Convert the array to a jpg file and write it to memory.
```

Reading and writing jpg files

```
% Read jpg image and convert to a 3D type uint8 array A
A = imread('LawSchool.jpg');

% Write 3D array B to memory as a jpg image
imwrite(B,'LawSchoolMirror.jpg')
```
A 3-d array as 3 matrices

\[
[nr, nc, np] = \text{size}(A) \quad \text{\% dimensions of 3-d array } A
\]

#rows  #columns  #layers (pages)

4-by-6  M1 = A(:,:,1)
4-by-6  M2 = A(:,:,2)
4-by-6  M3 = A(:,:,3)

% Store mirror image of A in array B

\[
[nr, nc, np] = \text{size}(A);
\]

\[
\text{for } r = 1:nr \\
\quad \text{for } c = 1:nc \\
\quad \quad \text{for } p = 1:np \\
\quad \quad \quad B(r, c, p) = A(r, nc-c+1, p);
\]

end

end

end

% Make mirror image of A -- the whole thing

A = imread(‘LawSchool.jpg’);

\[
[nr, nc, np] = \text{size}(A);
\]

\[
\text{for } r = 1:nr \\
\quad \text{for } c = 1:nc \\
\quad \quad \text{for } p = 1:np \\
\quad \quad \quad B(r, c, p) = A(r, nc-c+1, p);
\]

end

end

end

imshow(B) \quad \text{\% Show 3-d array data as an image}

imwrite(B, ‘LawSchoolMirror.jpg’)

% Make mirror image of A -- the whole thing

A = imread(‘LawSchool.jpg’);

\[
[nr, nc, np] = \text{size}(A);
\]

B = zeros(nr, nc, np);

\[
B = \text{uint8}(B); \quad \text{\% Type for image color values}
\]

\[
\text{for } r = 1:nr \\
\quad \text{for } c = 1:nc \\
\quad \quad \text{for } p = 1:np \\
\quad \quad \quad B(r, c, p) = A(r, nc-c+1, p);
\]

end

end

end

imshow(B) \quad \text{\% Show 3-d array data as an image}

imwrite(B, ‘LawSchoolMirror.jpg’)

% Store mirror image of A in array B

\[
[nr, nc, np] = \text{size}(A);
\]

\[
\text{for } r = 1:nr \\
\quad \text{for } c = 1:nc \\
\quad \quad \text{for } p = 1:np \\
\quad \quad \quad B(r, c, p) = A(r, nc-c+1, p);
\]

end

end

end

% Store mirror image of A in array B

\[
[nr, nc, np] = \text{size}(A);
\]

\[
\text{for } r = 1:nr \\
\quad \text{for } c = 1:nc \\
\quad \quad \text{for } p = 1:np \\
\quad \quad \quad B(r, c, p) = A(r, nc-c+1, p);
\]

end

end

end

A = imread(‘LawSchool.jpg’);

\[
[nr, nc, np] = \text{size}(A);
\]

\[
\text{for } r = 1:nr \\
\quad \text{for } c = 1:nc \\
\quad \quad \text{for } p = 1:np \\
\quad \quad \quad B(r, c, p) = A(r, nc-c+1, p);
\]

end

end

end

imshow(B) \quad \text{\% Show 3-d array data as an image}

imwrite(B, ‘LawSchoolMirror.jpg’)

% Store mirror image of A in array B

\[
[nr, nc, np] = \text{size}(A);
\]

\[
\text{for } r = 1:nr \\
\quad \text{for } c = 1:nc \\
\quad \quad \text{for } p = 1:np \\
\quad \quad \quad B(r, c, p) = A(r, nc-c+1, p);
\]

end

end

end

% Make mirror image of A -- the whole thing

A = imread(‘LawSchool.jpg’);

\[
[nr, nc, np] = \text{size}(A);
\]

\[
\text{for } r = 1:nr \\
\quad \text{for } c = 1:nc \\
\quad \quad \text{for } p = 1:np \\
\quad \quad \quad B(r, c, p) = A(r, nc-c+1, p);
\]

end

end

end

imshow(B) \quad \text{\% Show 3-d array data as an image}

imwrite(B, ‘LawSchoolMirror.jpg’)

% Make mirror image of A -- the whole thing

A = imread(‘LawSchool.jpg’);

\[
[nr, nc, np] = \text{size}(A);
\]

\[
\text{for } r = 1:nr \\
\quad \text{for } c = 1:nc \\
\quad \quad \text{for } p = 1:np \\
\quad \quad \quad B(r, c, p) = A(r, nc-c+1, p);
\]

end

end

end

imshow(B) \quad \text{\% Show 3-d array data as an image}

imwrite(B, ‘LawSchoolMirror.jpg’)

% Make mirror image of A -- the whole thing

A = imread(‘LawSchool.jpg’);

\[
[nr, nc, np] = \text{size}(A);
\]

\[
\text{for } r = 1:nr \\
\quad \text{for } c = 1:nc \\
\quad \quad \text{for } p = 1:np \\
\quad \quad \quad B(r, c, p) = A(r, nc-c+1, p);
\]

end

end

end

imshow(B) \quad \text{\% Show 3-d array data as an image}

imwrite(B, ‘LawSchoolMirror.jpg’)

% Make mirror image of A -- the whole thing

A = imread(‘LawSchool.jpg’);

\[
[nr, nc, np] = \text{size}(A);
\]

\[
\text{for } r = 1:nr \\
\quad \text{for } c = 1:nc \\
\quad \quad \text{for } p = 1:np \\
\quad \quad \quad B(r, c, p) = A(r, nc-c+1, p);
\]

end

end

end

imshow(B) \quad \text{\% Show 3-d array data as an image}

imwrite(B, ‘LawSchoolMirror.jpg’)

% Make mirror image of A -- the whole thing

A = imread(‘LawSchool.jpg’);

\[
[nr, nc, np] = \text{size}(A);
\]

\[
\text{for } r = 1:nr \\
\quad \text{for } c = 1:nc \\
\quad \quad \text{for } p = 1:np \\
\quad \quad \quad B(r, c, p) = A(r, nc-c+1, p);
\]

end

end

end

imshow(B) \quad \text{\% Show 3-d array data as an image}

imwrite(B, ‘LawSchoolMirror.jpg’)