Working with Images

Lecture 16 (Mar 13)
CS100M - Spring 2008

Announcements

- Prelim 2 is tonight: Thursday, March 13
  - Time: 7:30-9:00 pm
  - Location: Last names starting with
    - A-F in Kimball B11
    - G-Le in Olin 255
    - Li-Q in Upson B17
    - R-Z in Phillips 101
  - Includes material through Wednesday, March 5
    - User-defined functions
    - One-dimensional arrays (vectors)
    - Characters and strings (a string is a vector of characters)
    - Vectorized code
      - There is an document on the website about *vectorized code*
    - Simple plotting
    - No matrices on prelim 2
- Project 4 is due Thursday, March 27
  - Will be online Friday before break

A Cost/Inventory Problem

- A company has 3 factories that make 5 different products
  - The cost of making a product varies from factory to factory
  - The inventory varies from factory to factory

- A customer submits a purchase order that is to be filled by a single factory
  - Find the cheapest way to do this

Data

- Cost array

C:

10	36	22	15	62
12	35	20	12	66
13	37	21	16	59

- Inventory array

Inv:

38	5	99	34	42
82	19	83	12	42
51	29	21	56	87

- Purchase Order

PO:

1	0	12	29	5

Function to Find Cost using Factory i

```
  function  TheBill = iCost(i,C,PO)
% The cost when factory i fills the purchase order
  nProd = length(PO);
  TheBill = 0;
  for j=1:nProd
    TheBill = TheBill + C(i,j)*PO(j);
  end
```

Finding the Cheapest

C:

10	36	22	15	62	1019
12	35	20	12	66	930
13	37	21	16	59	1040

PO:

1	0	12	29	5

As computed by **`iCost`**

Finding Cheapest: Initialization

C:

10	36	22	15	62	1019
12	35	20	12	66	930
13	37	21	16	59	1040

PO:

1	0	12	29	5

iBest: 0

minBill: inf

A Note on “**`inf`**”

A special value that can be regarded as + infinity

`x = 10/0`	assigns inf to x
`y = 1+x`	assigns inf to y
`z = 1/x`	assigns zero to z
`w < inf`	is always true if w is numeric

Finding the Cheapest

```
iBest = 0; minBill = inf;
for i=1:nFact
   iBill = iCost(i,C,PO);
   if iBill < minBill
         % Found an Improvement
         iBest = i; minBill = iBill;
   end
end
```

Inventory Considerations

- What if a factory lacks the inventory to fill the purchase order?
- Such a factory should be excluded from the find-the-cheapest computation

Who Can Fill the Order?

Inv:

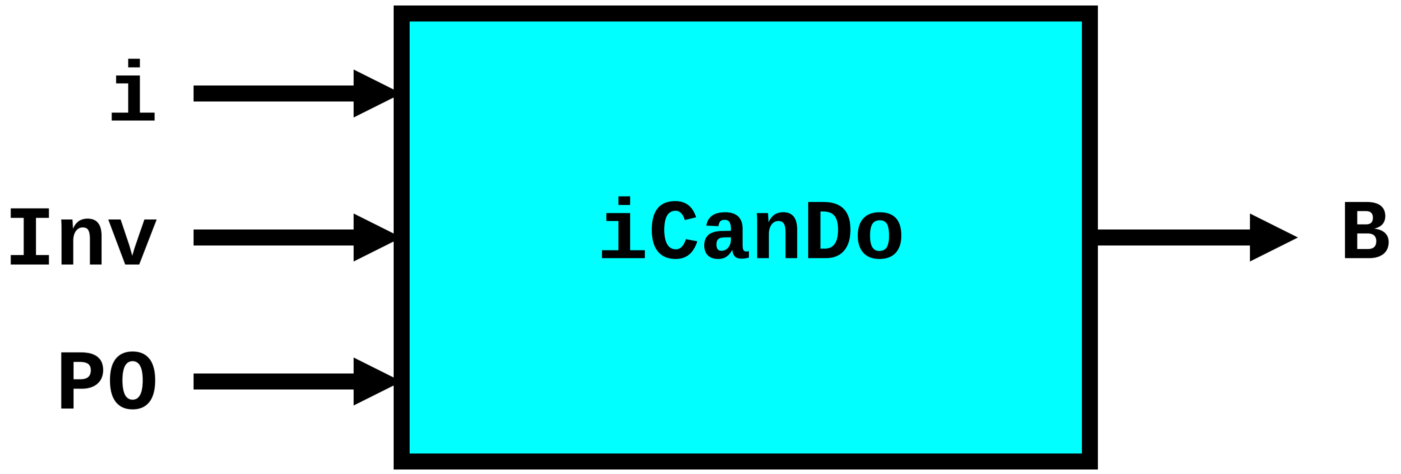
38	5	99	34	42	Yes
82	19	83	12	42	No
51	29	21	56	87	Yes

PO:

1	0	12	29	5

Because 12 < 29

Wanted: A True/False Function



B is "true" if factory i can fill the order.

B is "false" if factory i cannot fill the order.

Boolean Operations in Matlab

When discussing expressions like

```
a <= x  && x <= b

abs(y) > 10
```

we say that an expression is either true or false

The 0-1 Secret

In reality, expressions like

```
        a <= x   && x <= b

                abs(y) > 10
```

render the value "1" if they are TRUE and "0" if they are FALSE

Back to Inventory Problem

Inv:

38	5	99	34	42
82	19	83	12	42
51	29	21	56	87

PO:

1	0	12	29	5

Initialization

Inv:

38	5	99	34	42
82	19	83	12	42
51	29	21	56	87

B: 1

PO:

1	0	12	29	5

Still True...

Inv:

38	5	99	34	42
🟥 82	19	83	12	42
51	29	21	56	87

B: 1

PO:

🟩 1	0	12	29	5

```
B = B && ( Inv(2,1) >= PO(1) )
```

Still True...

Inv:

38	5	99	34	42
82	19	83	12	42
51	29	21	56	87

B: 1

PO:

1	0	12	29	5

```
B = B && ( Inv(2,2) >= PO(2) )
```

Still True...

Inv:

38	5	99	34	42
82	19	83	12	42
51	29	21	56	87

B: 1

PO:

1	0	12	29	5

```
B = B && ( Inv(2,3) >= PO(3) )
```

No Longer True...

Inv:

38	5	99	34	42
82	19	83	12	42
51	29	21	56	87

B: 0

PO:

1	0	12	29	5

```
B = B && ( Inv(2,4) >= PO(4) )
```

Encapsulate…

```
function  B = iCanDo(i,Inv,PO)
% B is true if factory i can fill
% the purchase order. Otherwise, false
nProd = length(PO);
B = true;
for j = 1:nProd
    B = B && ( Inv(i,j) >= PO(j) );
end
```

Back To Finding the Cheapest

```
iBest = 0; minBill = inf;
for i=1:nFact
   iBill = iCost(i,C,PO);
   if iBill < minBill
         % Found an Improvement
         iBest = i; minBill = iBill;
   end
end
```

Problem: Can't be "best" if insufficient inventory

Back To Finding the Cheapest

```
iBest = 0; minBill = inf;
for i=1:nFact
   iBill = iCost(i,C,PO);
   if iBill < minBill && iCanDo(i, Inv, PO)
         % Found an Improvement
         iBest = i; minBill = iBill;
   end
end
```

Finding the Cheapest

C:

10	36	22	15	62
12	35	20	12	66
13	37	21	16	59

PO:

1	0	12	29	5

1019	Yes
930	No
1040	Yes
As computed by **iCost**	As computed by **iCanDo**

Images in Matlab

Pictures as Arrays

A black and white picture can be encoded as a 2D Array

Typical:

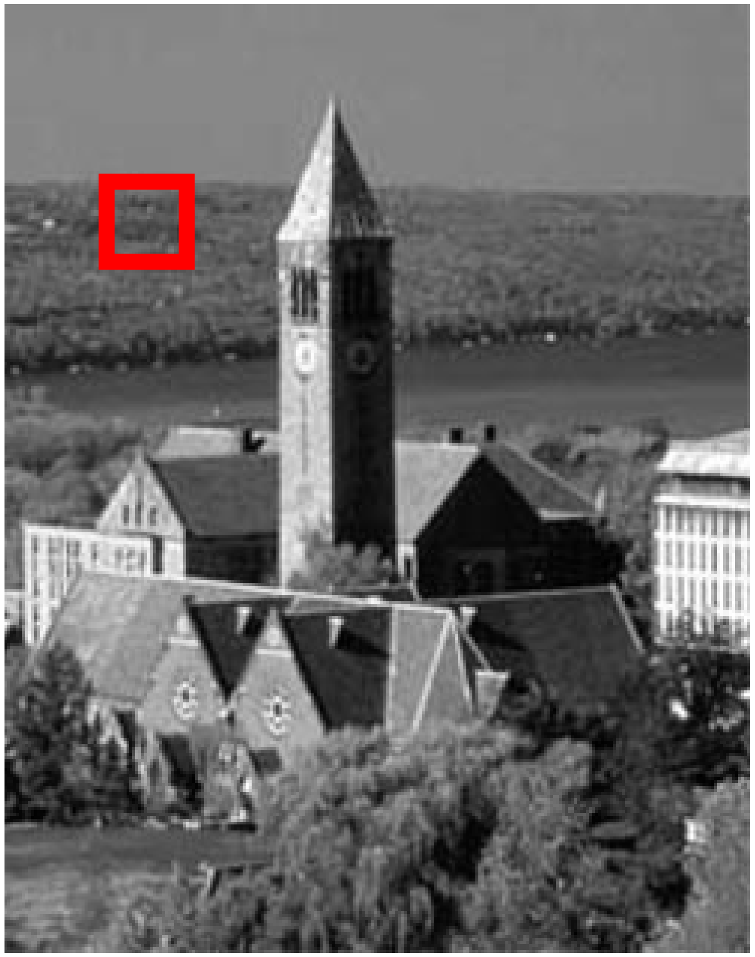
$$\underset{\text{(black)}}{0} \le A(i,j) \le \underset{\text{(white)}}{255}$$

Values in between correspond to different levels of grayness

Just a Bunch of Numbers

318-by-250

49	55	58	59	57	53
60	67	71	72	72	70
102	108	111	111	112	112
157	167	169	167	165	164
196	205	208	207	205	205
199	208	212	214	213	216
190	192	193	195	195	197
174	169	165	163	162	161



A Color Picture is Represented by 3 Arrays

Stack them in a single 3D array

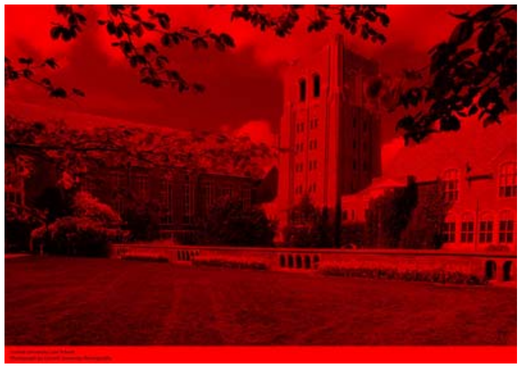
Typical:

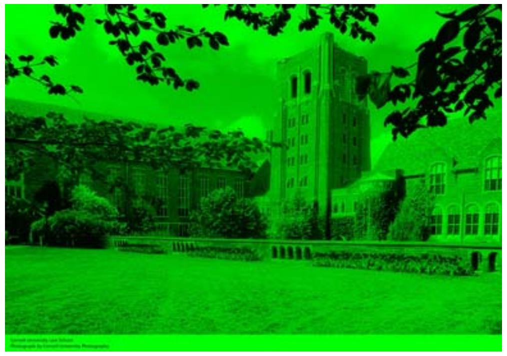
```
0 <= A(i,j,1) <= 255   (red)
0 <= A(i,j,2) <= 255   (green)
0 <= A(i,j,3) <= 255   (blue)
```

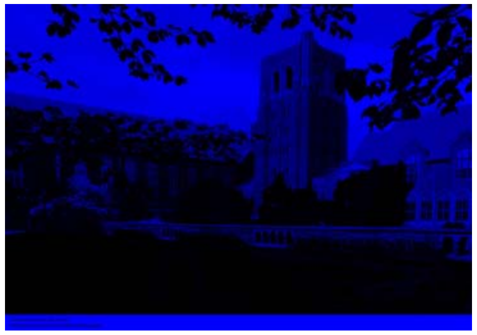
↑

Note 3rd Subscript

Cornell University Law School
Photograph by Cornell University Photography







Encoding Images

- There are a number of file formats for images

- Some common ones:

  - JPEG
    - Joint Photographic Experts Group

  - GIF
    - Graphics Interchange Format

Behind the scenes: compressing data

A Compression Idea

	1	2	3	4	5	6	7	8	9
1	1	2	3	4	5	6	7	8	9
2	2	4	6	8	10	12	14	16	18
3	3	6	9	12	15	18	21	24	27
4	4	8	12	16	20	24	28	32	36
5	5	10	15	20	25	30	35	40	45
6	6	12	18	24	30	36	42	48	54
7	7	14	21	28	35	42	49	56	63
8	8	16	24	32	40	48	56	64	72
9	9	18	27	36	45	54	63	72	81

Store the array (81 numbers) or the purple vectors (18 numbers)?

More Dramatic

- Suppose A is a 1000-by 2000 *multiplication table*
  - Do I store A (2,000,000 numbers)?

    or

  - Do I store the two 1-dimensional multiplier arrays (3000 numbers) and "reconstruct" A?

Storing an Image

- An image can be written as a sum of a relatively small number of times tables
- 1000-by-2000 picture might be well-approximated by the sum of 100 times tables

2,000,000 vs. (100×3000)

Operations on Images

- Image operations are operations on 2D Arrays
- A good place to practice "array" thinking

Two Problems

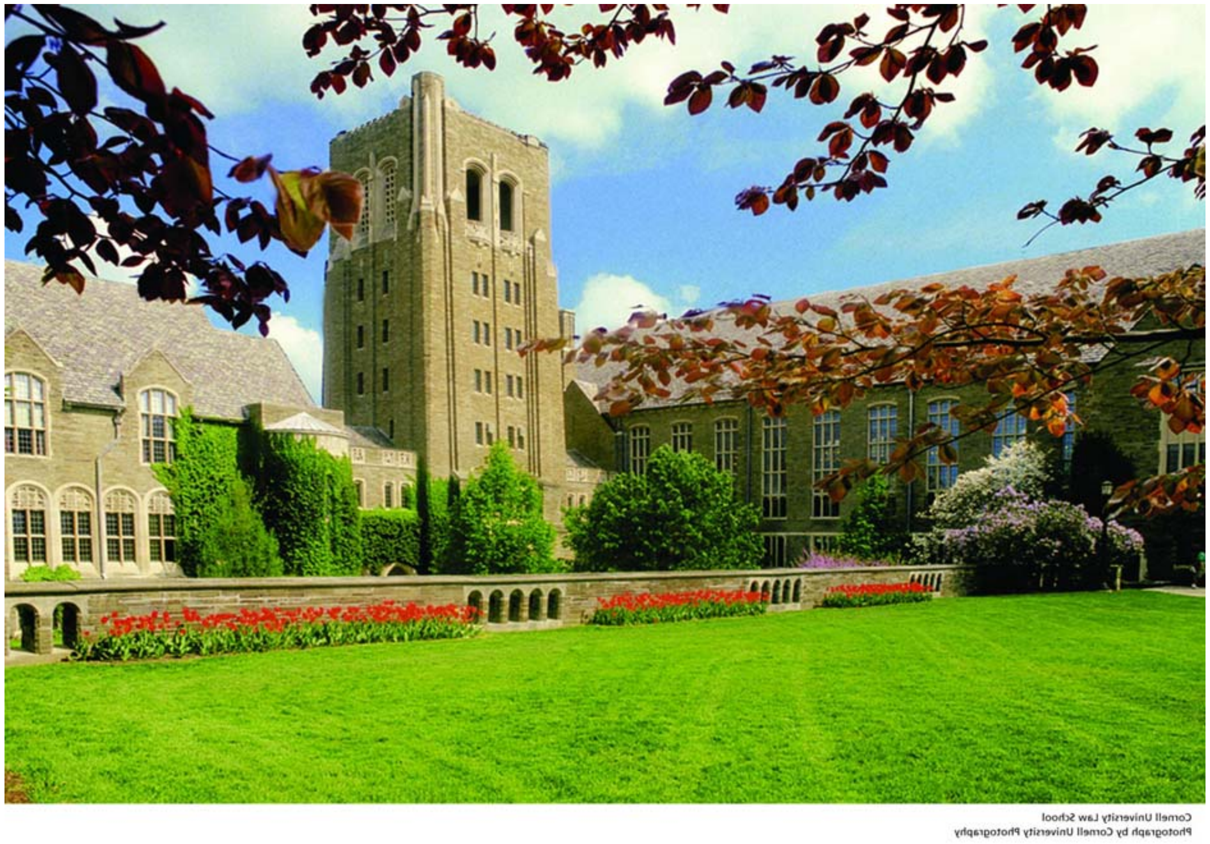
We have:

Cornell University Law School
Photograph by Cornell University Photography

LawSchool.jpg

Problem 1

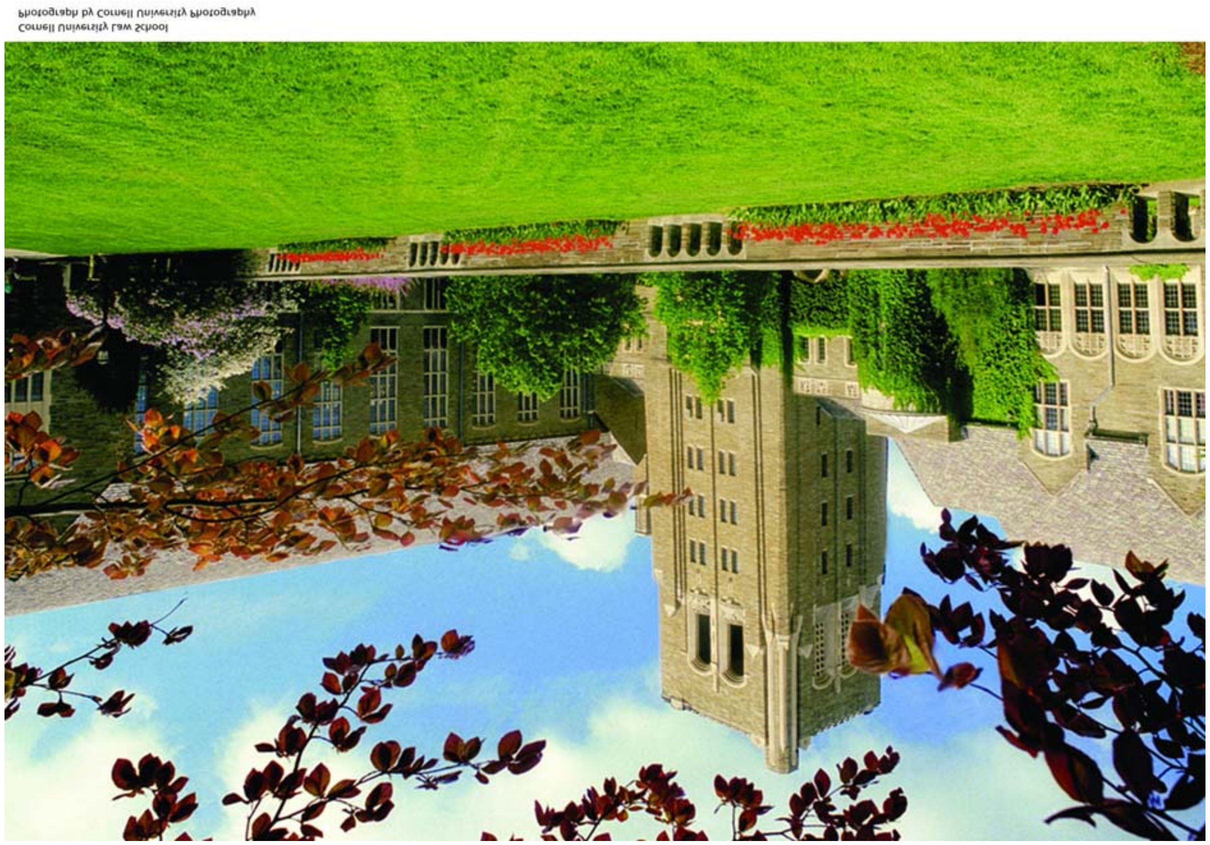
Want:


Cornell University Law School
Photograph by Cornell University Photography

`LawSchoolMirror.jpg`

Problem 2

Want:


Cornell University Law School
Photograph by Cornell University Photography

LawSchoolUpDown.jpg

Solution Framework

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

imread

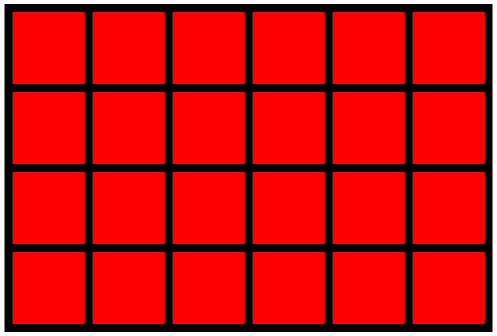
```
% Read image and convert to a 3D array...

>> A = imread('LawSchool.jpg');
>> [m,n,p] = size(A)

m =
      1458
n =
      2084
p =
         3
```

rows

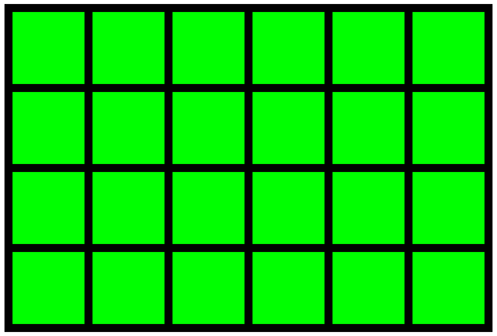
columns

layers

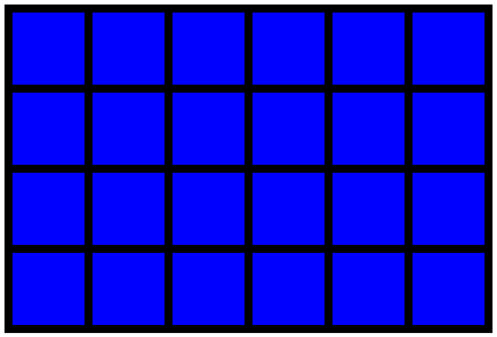
The Layers



`1458-by-2084` `A(:,:,1)`



`1458-by-2084` `A(:,:,2)`



`1458-by-2084` `A(:,:,3)`

Left-Right Mirror Image

```
A = imread('LawSchool.jpg')
[m,n,p] = size(A);
for j=1:n
        B(:,j,1) = A(:,n+1-j,1)
        B(:,j,2) = A(:,n+1-j,2)
        B(:,j,3) = A(:,n+1-j,3)
end
imwrite(B,'LawSchoolMirror.jpg')
```

Vectorized-Code Equivalent

```
for j=1:n
        B(:,j,1) = A(:,n+1-j,1)
        B(:,j,2) = A(:,n+1-j,2)
        B(:,j,3) = A(:,n+1-j,3)
end
```

```
B = A(:,n:-1:1,:);
```

The Upside Down Image

```
A = imread('LawSchool.jpg')
[m,n,p] = size(A);
for i=1:m
        C(i,:,1) = A(m+1-i,:,1)
        C(i,:,2) = A(m+1-i,:,2)
        C(i,:,3) = A(m+1-I,:,3)
end
imwrite(C,'LawSchoolUpDown.jpg')
```

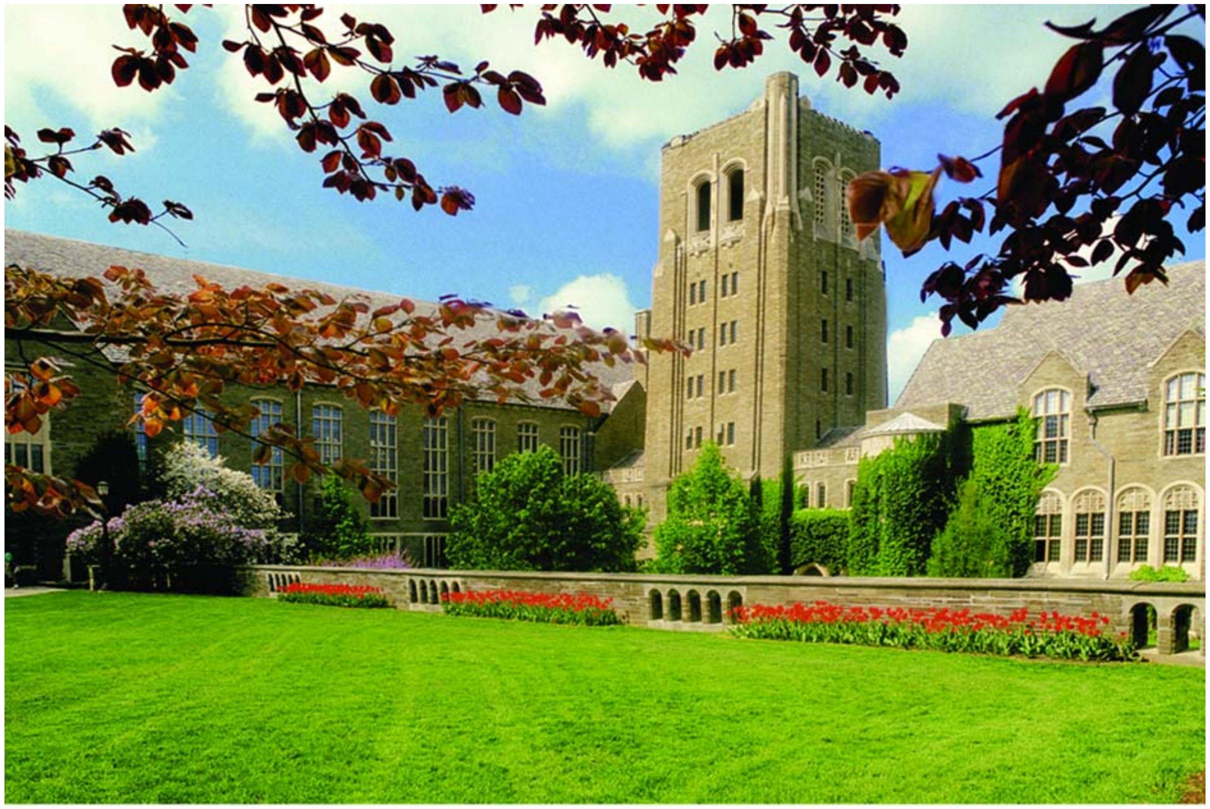
Vectorized-Code Equivalent

```
for j=1:n
        C(i,:,1) = A(m+1-i,:,1)
        C(i,:,2) = A(m+1-i,:,2)
        C(i,:,3) = A(m+1-i,:,3)
end
```

```
   C = A(m:-1:1,:,:);
```

New Problem
Color → Black and White

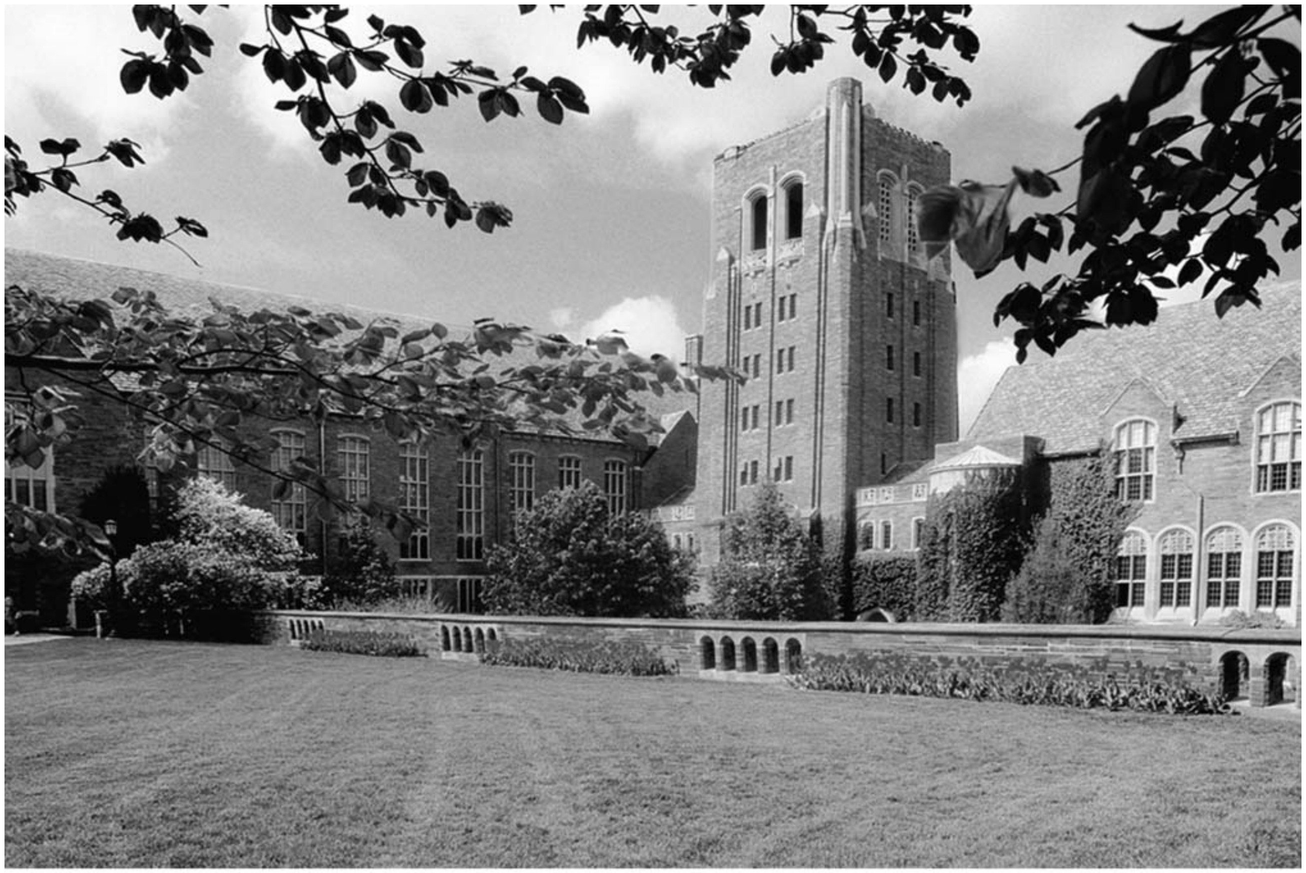
Have:



Cornell University Law School
Photograph by Cornell University Photography

New Problem
Color → Black and White

Want:



Cornell University Law School
Photograph by Cornell University Photography

rgb2gray

```
A = imread('LawSchool.jpg');
bwA = rgb2gray(A);
imwrite(bwA,'LawSchoolBW.jpg')
```

How Does the Conversion Work?

```
 r    g    b       gray
-------------------------
167  219  241       206
 66   35   15        42
 95   14   20        39
163  212  242       201
182  228  215       213
225  244  222       236
136  199  240       185
```

It's a complicated mapping

Why not take Average?

```
bwA = uint8(zeros(m,n));
for i=1:m
  for j = 1:n
     bwA(i,j) = ( A(i,j,1) + A(i,j,2) + A(i,j,3) )/3;
  end
end
imwrite(bwA,'LawSchoolBW.jpg')
```

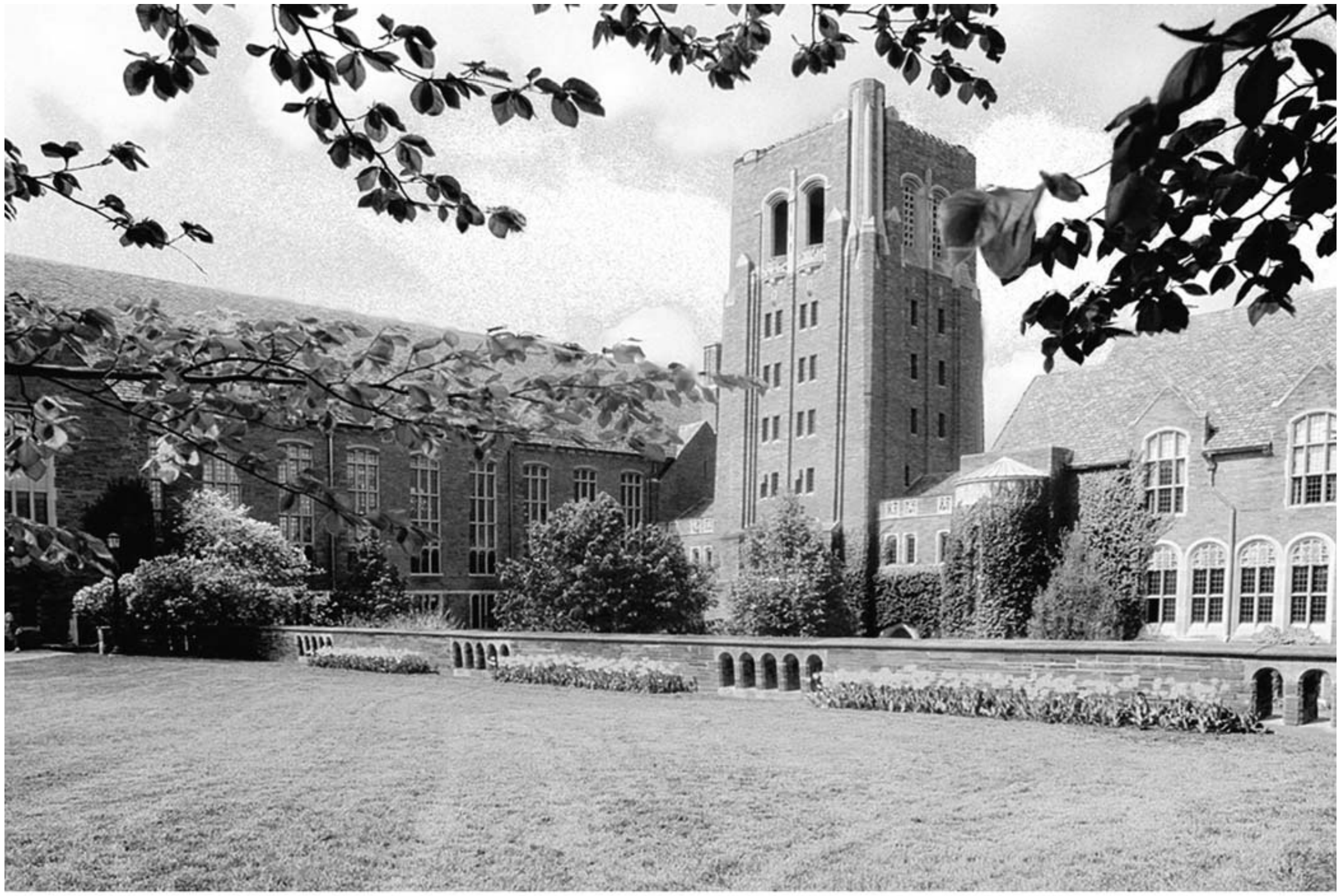
uint8 : unsigned 8-bit integer

Cornell University Law School
Photograph by Cornell University Photography

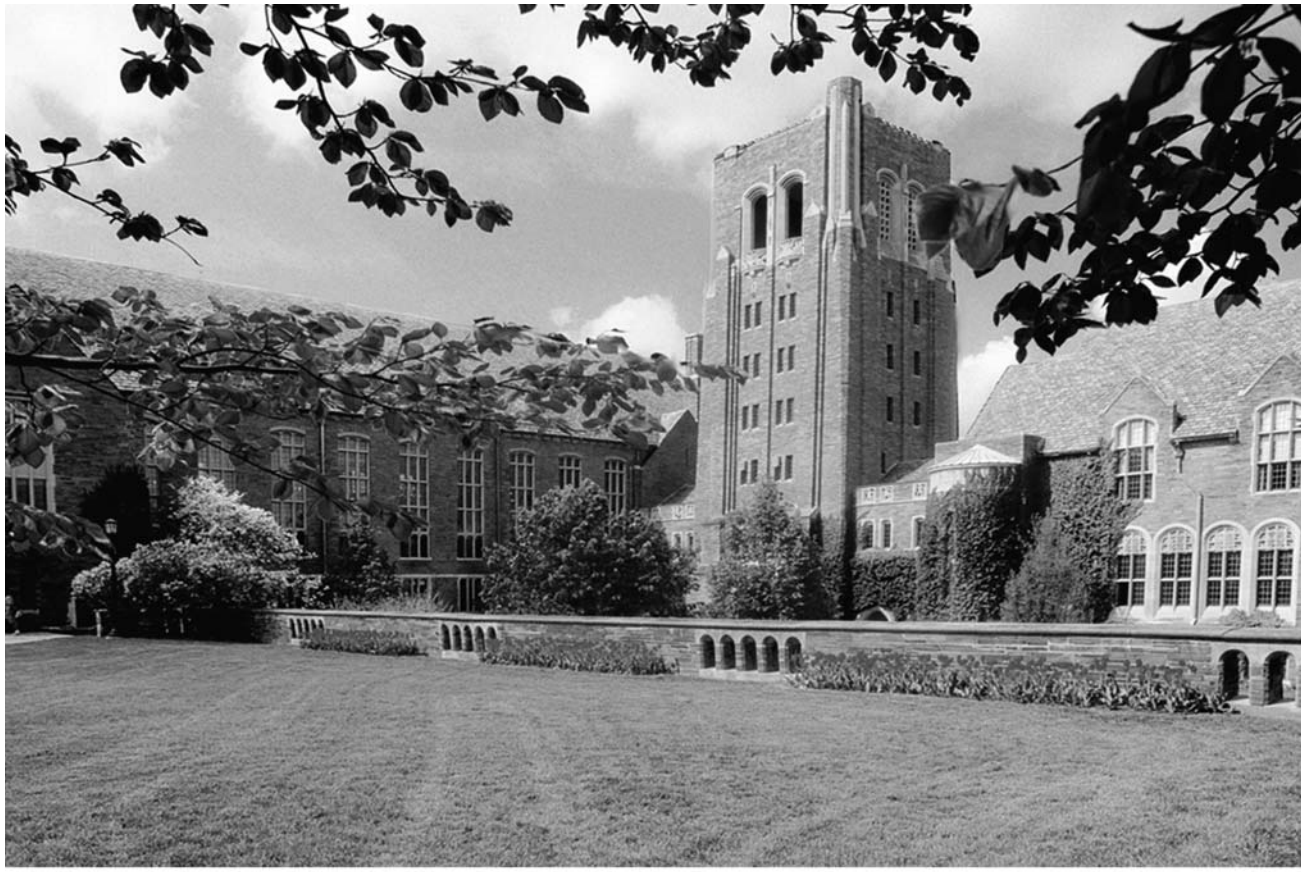
Why not take Max?

```
bwA = uint8(zeros(m,n));
for i=1:m
  for j = 1:n
     bwA(i,j) = max([A(i,j,1)  A(i,j,2)  A(i,j,3)]);
  end
end
imwrite(bwA,'LawSchoolBW.jpg')
```

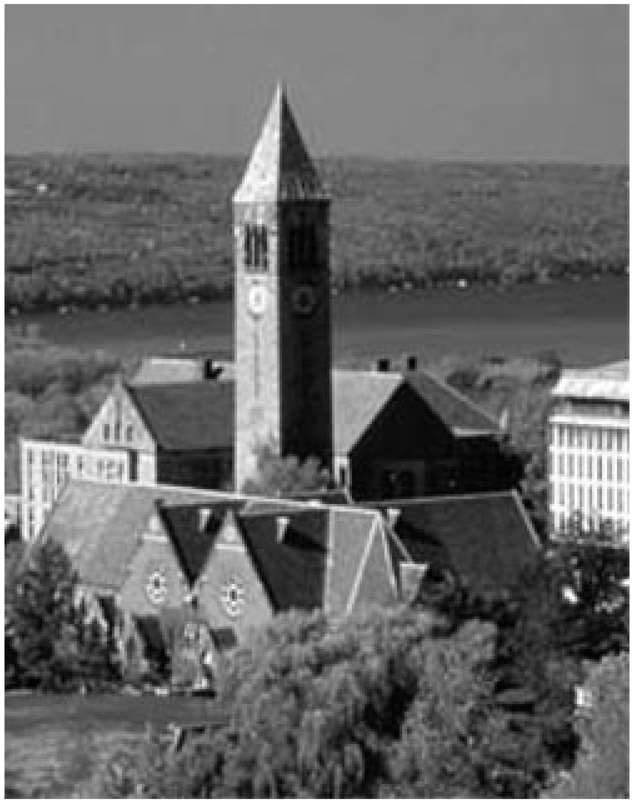
Max:

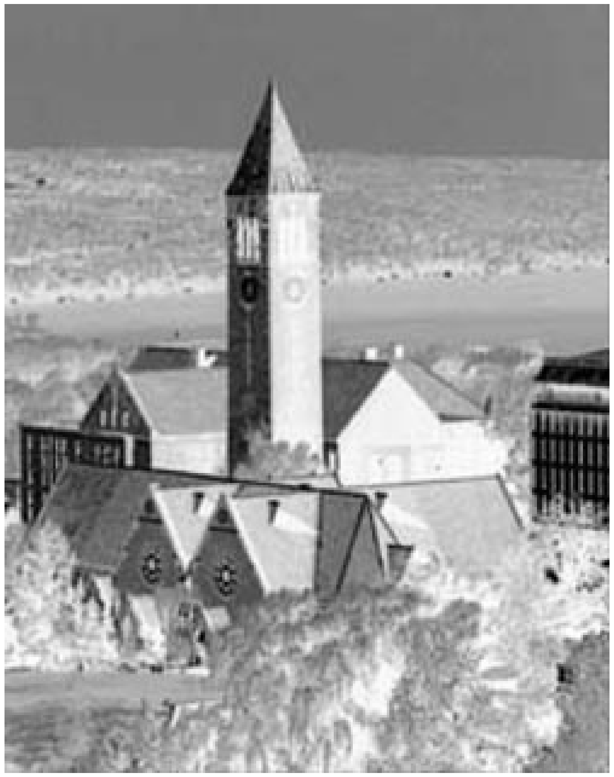

Cornell University Law School
Photograph by Cornell University Photography

Matlab:


Cornell University Law School
Photograph by Cornell University Photography

Problem: Produce a Negative





Idea

If matrix A represents the image and

$$B(i,j) = 255 - A(i,j)$$

for all i and j, then B will represent the negative