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 \( i \).
- Generate a random link array and display the connectivity:
  - There is no link from a page to itself
  - If \( i \neq j \) then \( A(i,j) = 1 \) with probability \( \frac{1}{1+|i-j|} \)
  - There is more likely to be a link if \( i \) is close to \( j \).

function \( A = \text{RandomLinks}(n) \)

```matlab
function A = RandomLinks(n)
% A is n-by-n matrix of 1s and 0s
% representing n webpages
A = zeros(n,n);
for i=1:n
    for j=1:n
        r = rand;
        if i~=j && r <= 1/(1 + abs(i-j));
            A(i,j) = 1;
        end
    end
end
```

Represent the web pages graphically...

100 Web pages arranged in a circle.
Next display the links....

Somewhat inefficient: each blue line gets drawn twice.
See ShowRandomLinks.m

Bidirectional links are blue. Unidirectional link is black as it leaves page \( j \), red when it arrives at page \( i \).
% Given an n-by-m matrix A.
% What is this operation?
for g = 1: n
    for h = 1: floor(m/2)
        A(g,h) = A(g, m-h+1);
    end
end

A

B

Reflect the right half of A onto the left half
Reflect the bottom half of A onto the top half

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

Characters & strings

- We have used strings already:
  - n = input('Next number: ')
  - sprintf('Answer is %d', ans)
- A string is made up of individual characters, so a string is a 1-d array of characters
- 'CS1112 rocks!' is a character array of length 13; it has 7 letters, 4 digits, 1 space, and 1 symbol.

Some Matlab types: char, double, logical

- a is a 1-d array with type char components. We call a a "string" or "char array"
- b is a 1-d array with type double components. double is the default type for numbers in Matlab. We call b a "numeric array"
- d is a scalar of the type logical. We call d a "boolean value"
Strings are important in computation

Numerical data is often encoded in strings. E.g., a file containing Ithaca weather data begins with the string 

\[ \text{W07629N4226} \]

meaning

- **Longitude:** 76° 29’ West
- **Latitude:** 42° 26’ North

We may need to grab hold of the substring \[ \text{W07629} \], convert 076 and 29 to the numeric values 76 and 29, and do some computation

Comparison of genomic sequences is another example of string computation

- **E.g., looking for a pattern:** Given the sequence \[ \text{ATTCTGACCTCGATC...} \]
  Look for the pattern \[ \text{ACCT} \]
- **E.g., quantifying the difference between sequences:**

\[ \text{ATTCTGACCTCGATC} \]
\[ \text{ATTCGTGACCTCGAT} \]

Single quotes enclose strings in Matlab

Anything enclosed in single quotes is a string (even if it looks like something else)

- ‘100’ is a character array (string) of length 3
- 100 is a numeric value
- ‘pi’ is a character array of length 2
- pi is the built-in constant 3.1416...
- ‘x’ is a character (vector of length 1)
- x may be a variable name in your program

Example: removing all occurrences of a character

- From a genome bank we get a sequence
  \[ \text{ATTG CCG TA GCTA CGTACGC AACTGG AAATGGC CGTAT...} \]
- First step is to “clean it up” by removing all the blanks. Write this function:

```
function s = removeChar(c, s)
% Return string s with all occurrences
% of character c removed
```
The ASCII Table

<table>
<thead>
<tr>
<th>ASCII code</th>
<th>Character</th>
</tr>
</thead>
<tbody>
<tr>
<td>65</td>
<td>'A'</td>
</tr>
<tr>
<td>66</td>
<td>'B'</td>
</tr>
<tr>
<td>67</td>
<td>'C'</td>
</tr>
<tr>
<td>90</td>
<td>'Z'</td>
</tr>
</tbody>
</table>

ASCII characters
(American Standard Code for Information Interchange)

<table>
<thead>
<tr>
<th>ascii code</th>
<th>Character</th>
</tr>
</thead>
<tbody>
<tr>
<td>48</td>
<td>'0'</td>
</tr>
<tr>
<td>49</td>
<td>'1'</td>
</tr>
<tr>
<td>50</td>
<td>'2'</td>
</tr>
<tr>
<td>57</td>
<td>'9'</td>
</tr>
</tbody>
</table>

Character vs ASCII code

str= 'Age 19'
% a 1-d array of characters
code= double(str)
% convert chars to ascii values
str1= char(code)
% convert ascii values to chars

Arithmetic and relational ops on characters

- 'c' - 'a' gives 2
- '6' - '5' gives 1
- letter1='e'; letter2='f';
  letter1-letter2 gives -1
- 'c' > 'a' gives true
- letter1==letter2 gives false
- 'A' + 2 gives 67
- char('A' + 2) gives 'C'