Cell Arrays

Set-Up
Subscripting
Nested Loops
String Manipulations

A Small Cell Array…

```
C = { 'Alabama','New York','Utah'};
```

C:

'Alabama'	'New York'	'Utah'

A major use of cell arrays is to store collections of strings.

Syntax

Entries Separated by Commas

```
C = { 'Alabama','New York','Utah'};
```

Curly Brackets

Another Set-up Strategy

```
C = cell(1,3);
C{1} = ‘Alabama’;
C{2} = ‘New York’;
C{3} = ‘Utah’;
```

Is the same as. . .

```
C = { ‘Alabama’,’New York’,’Utah’};
```

"Vertical" vs "Horizontal"

```
C = cell(3,1);
C{1} = 'Alabama';
C{2} = 'New York';
C{3} = 'Utah';
```

```
C = cell(1,3);
C{1} = 'Alabama';
C{2} = 'New York';
C{3} = 'Utah';
```

Unlike arrays orientation is not important

More Flexible than Character Arrays

Unequal lengths make this illegal:

```
C = ['Alabama';'New York';'Utah'];
```

Padding with blanks is a major annoyance. Wasteful of storage too. Use cell arrays to house collections of arbitrary strings.

A Cell Array of Vectors…

```
C = { [1 2 3],[10;20], zeros(1,4)};
```

C:	`[1 2 3]`	`[10; 20]`	`zeros(1,4)`

Syntax

Entries Separated by Commas

```
C = { [1 2 3], [10;20], zeros(1,4)};
```

Curly Brackets

Synonym

```
C = { [1 2 3], [10;20], zeros(1,4)};
```

```
C = cell(1,3);
C{1} = [1 2 3];
C{2} = [10;20];
C{3} = zeros(1,4);
```

Great Flexibility

```
C = {'ABC', [1 2 3], zeros(5,6)}
```

The examples which follow will illustrate more typical uses:

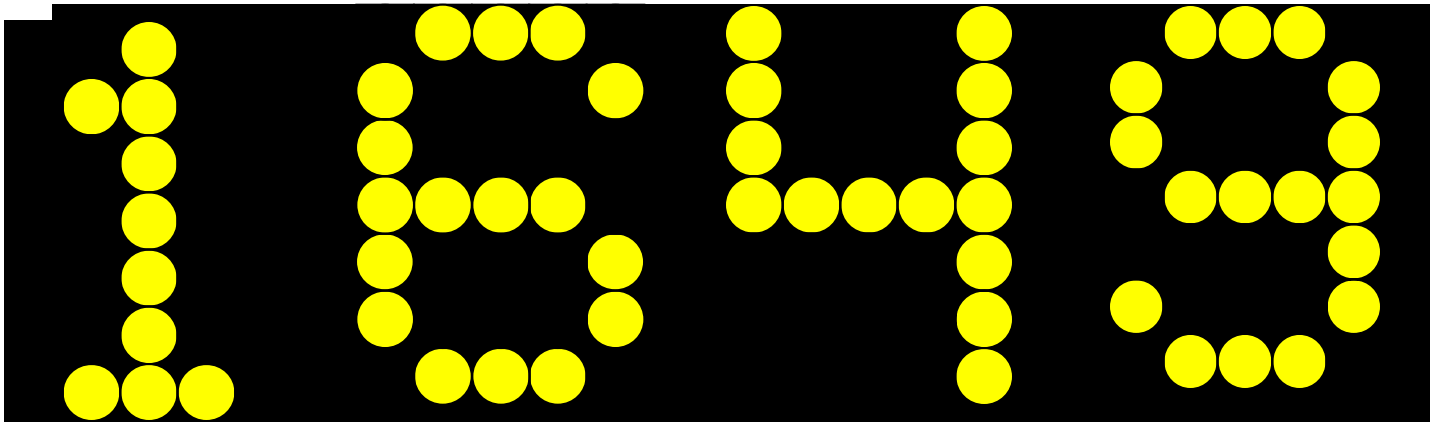
1. Cell arrays of strings
2. Cell arrays of vectors
3. Cell arrays of matrices

Example 1
A "Library" of Digitized Digits

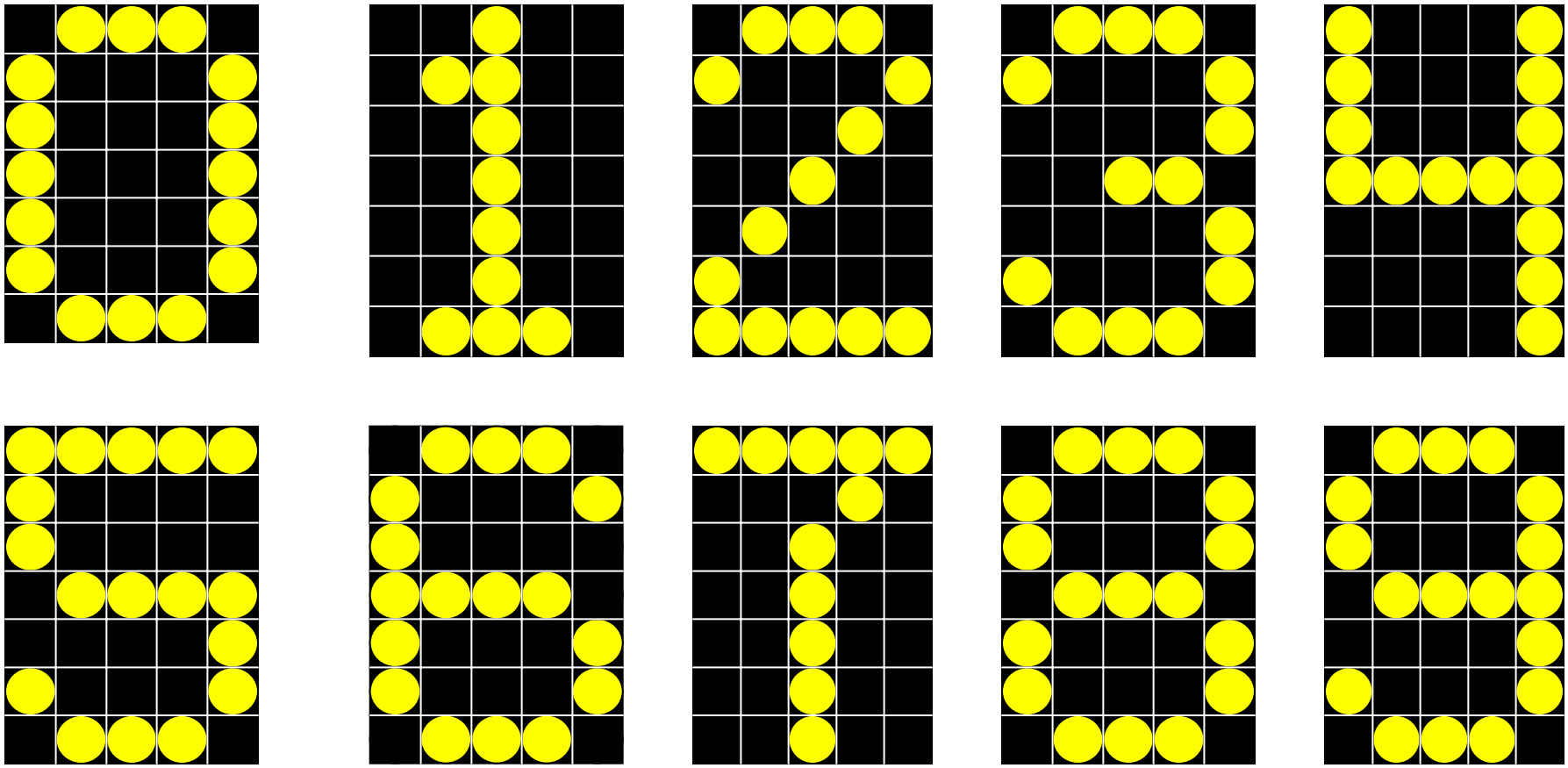
Theme: Use a Cell Array to House
a Collection of Matrices

(See Section 12.3)

Digital Displays



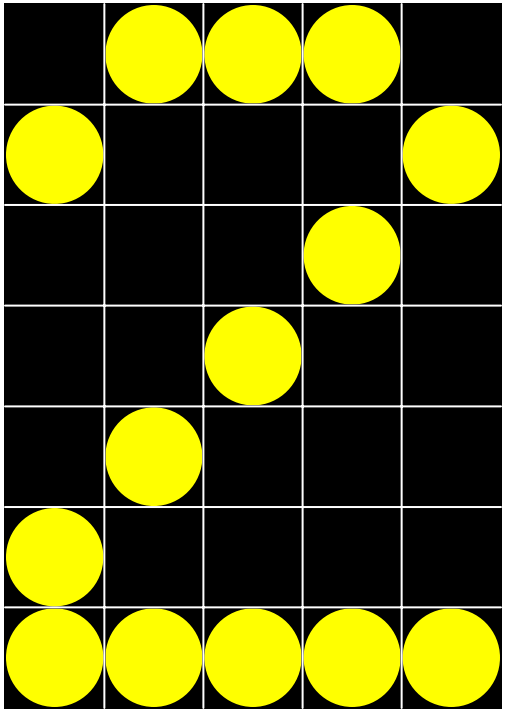
7-by-5 "Dot Matrices"



How can we store all these 0-1 patterns?

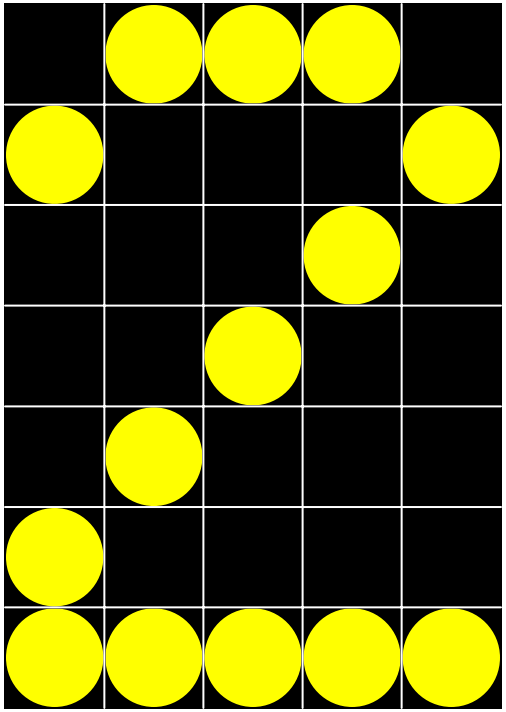
Can Use a Numerical Array For Each Digit

```
[ 0 1 1 1 0;…
  1 0 0 0 1;…
  0 0 0 1 0;…
  0 0 1 0 0;…
  0 1 0 0 0;…
  1 0 0 0 0;…
  1 1 1 1 1];
```



Use a Cell Array to House the Ten Digits

```
M = [ 0 1 1 1 0;…
      1 0 0 0 1;…
      0 0 0 1 0;…
      0 0 1 0 0;…
      0 1 0 0 0;…
      1 0 0 0 0;…
      1 1 1 1 1];
```



```
D{2} = M;
```

Each cell is a numerical matrix

Accessing the Data

Suppose D{1},…,D{10} are set up and k is initialized with 1 <= k <= 10

```
M = D{k};
if M(4,3)==1
    disp('Middle Light is On')
end
```

A Style Point

```
M = D{k};
if M(4,3)==1
    disp('Middle Light is On')
end

% Legal but confusing . . .
if D{k}(4,3)==1
    disp('Middle Light is On')
end
```

Example 2.
Set Up a Card Deck

Theme: Use a Cell Array to House
a Collection of Strings

Idea…

```
A{1} = ‘A Hearts’;
A{2} = ‘2 Hearts’;
                :
A{13} = ‘K Hearts’;
A{14} = ‘A Clubs’;
                :
A{52} = ‘K Diamonds’;
```

Initializations…

```
suit = {‘Hearts’, ‘Clubs’, …
        ‘Spades’, ‘Diamonds’};

rank = {‘A’,‘2’,’3’,’4’,’5’,’6’,…
    ’7’,’8’,’9’,’10’,’J’,’Q’,’K’};

A = cell(1,52);
```

Use Concatenation…

```
suit = {‘Hearts’, ‘Clubs’, …
        ‘Spades’, ‘Diamonds’};

rank = {‘A’,‘2’,’3’,’4’,’5’,’6’,…
    ’7’,’8’,’9’,’10’,’J’,’Q’,’K’};

A{16} = [rank{3} ‘ ‘ suit{2} ]
```

The value of A{16} is ‘3 Clubs’

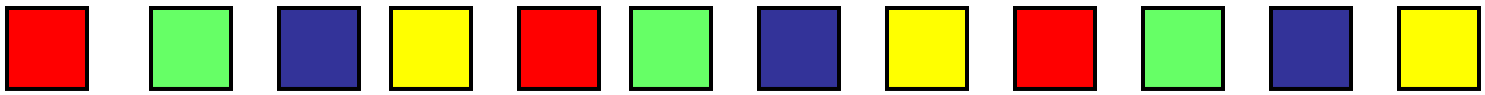
Use a Nested Loop to Get all Possible Combinations…

```
% i is index of next card…
  i = 1;
  for k=1:4
% Set up the cards in suit k
      for j=1:13
          A{i} = [ rank{j} ' ' suit{k} ];
          i = i+1
      end
  end
```

Example 3
Deal a Card Deck

Theme: Build four small cell arrays
from a single large cell array.

Deal a length-12 Card Deck

A: 

N:	1,5,9	4k-3
E:	2,6,10	4k-2
S:	3,7,11	4k-1
W:	4,8,12	4k

Idea: During the kth "pass", E gets card 4k-2 .

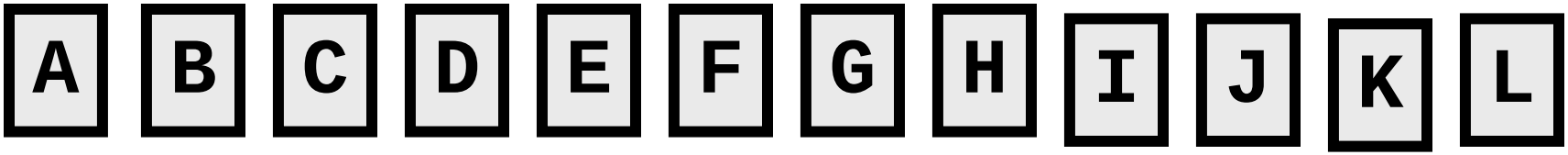
```
N = cell(1,13); E = cell(1,13);
S = cell(1,13); W = cell(1,13);

 for k=1:13
    N{k} = A{4*k-3};
    E{k} = A{4*k-2};
    S{k} = A{4*k-1};
    W{k} = A{4*k};
 end
```

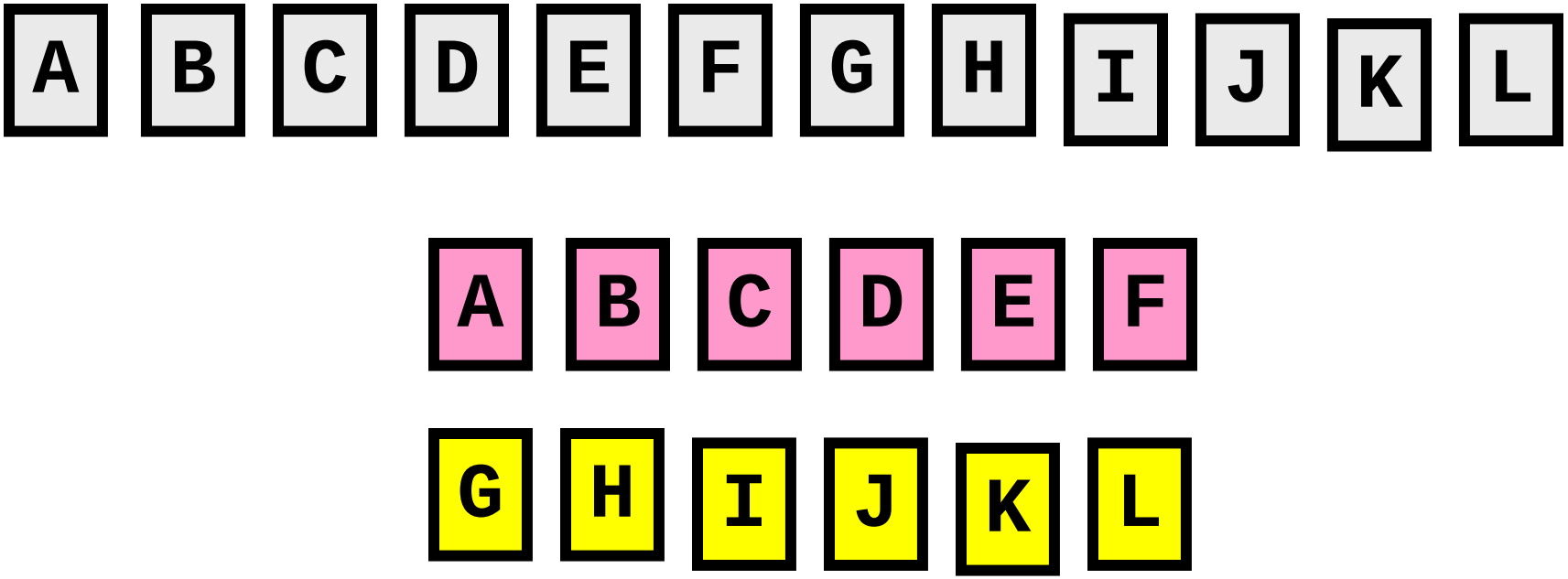
Example 4
Shuffle a Card Deck

Theme: Permute the cells in a cell array

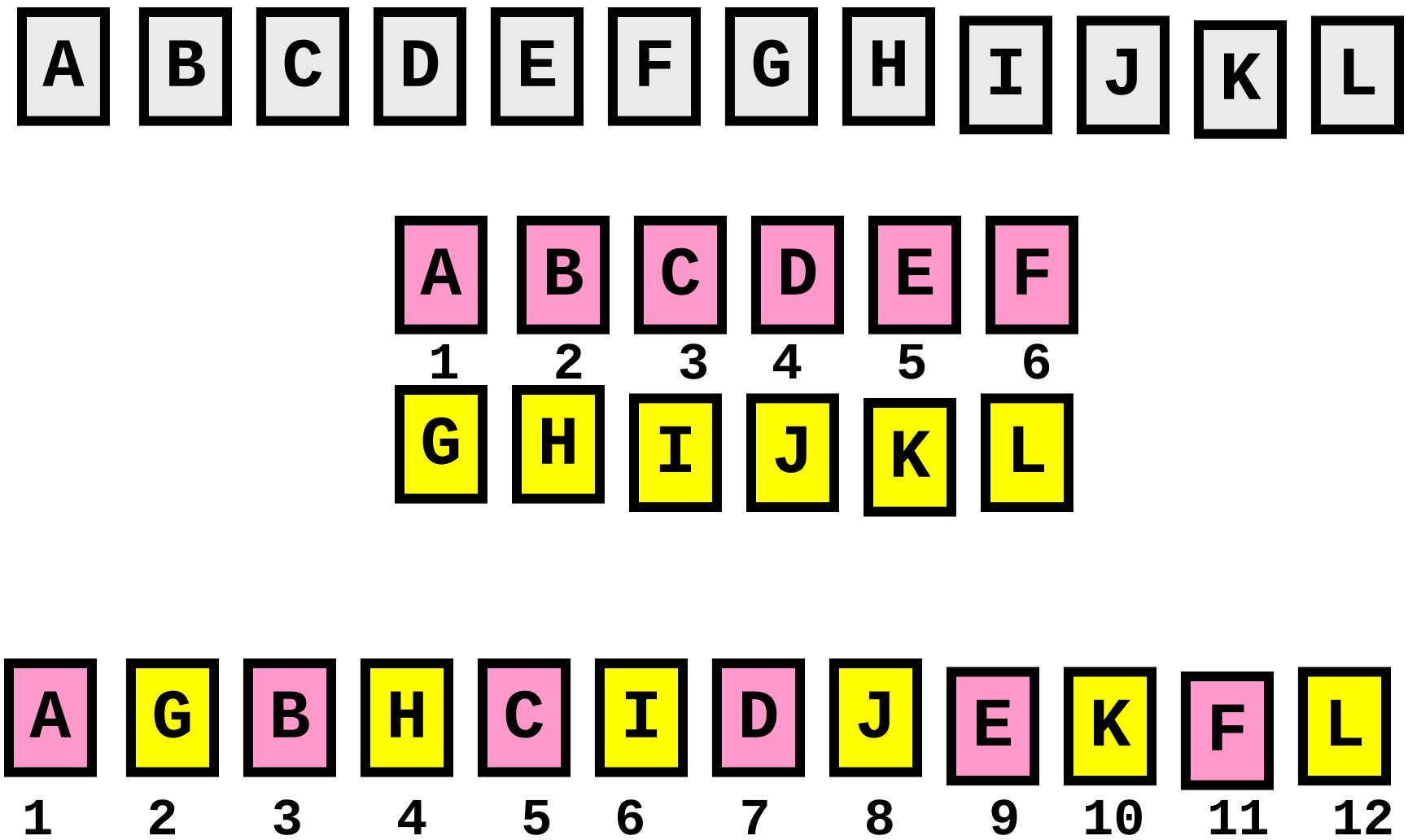
Shuffle a length-12 Card Deck



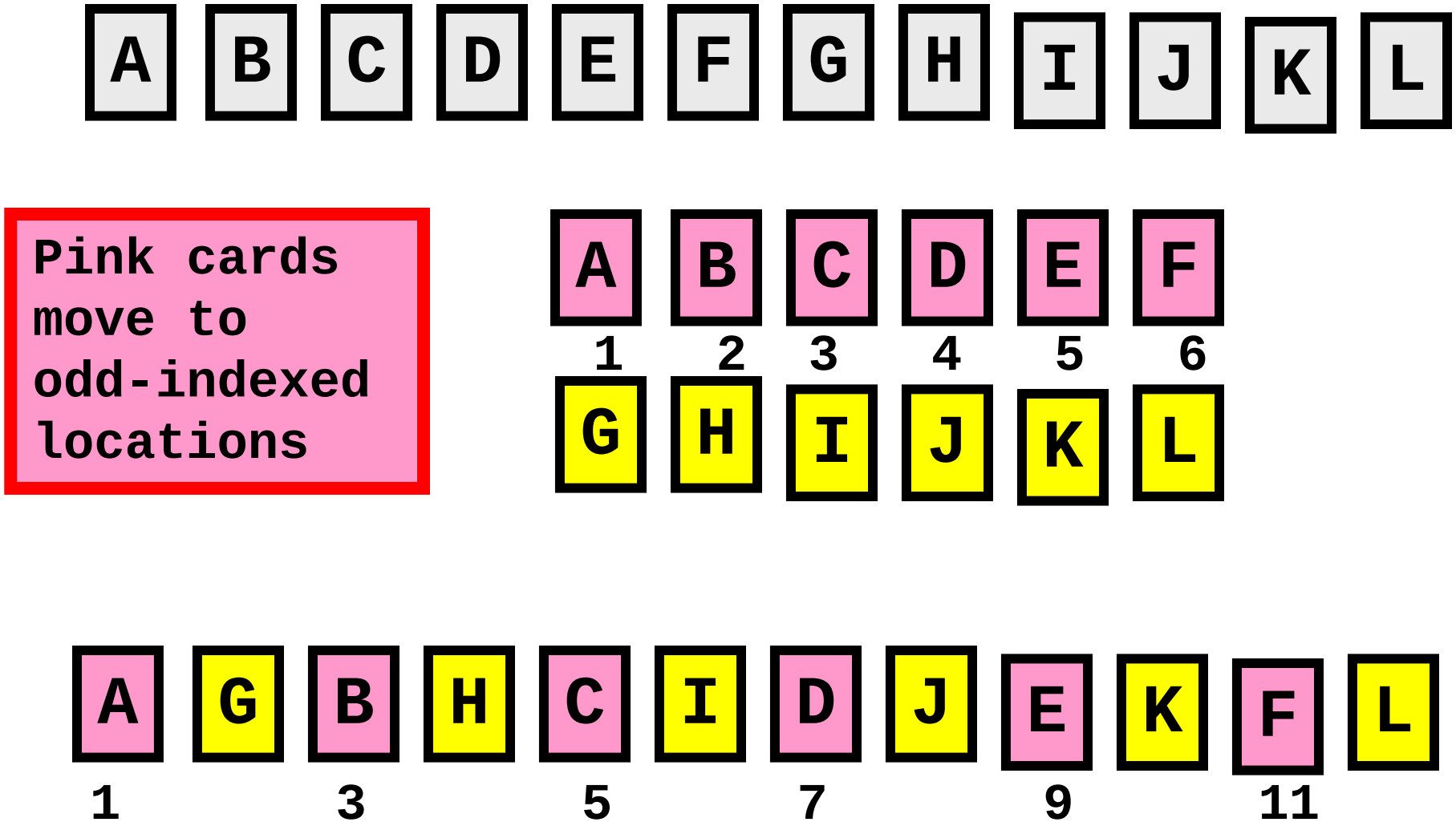
Step 1: Cut the Deck



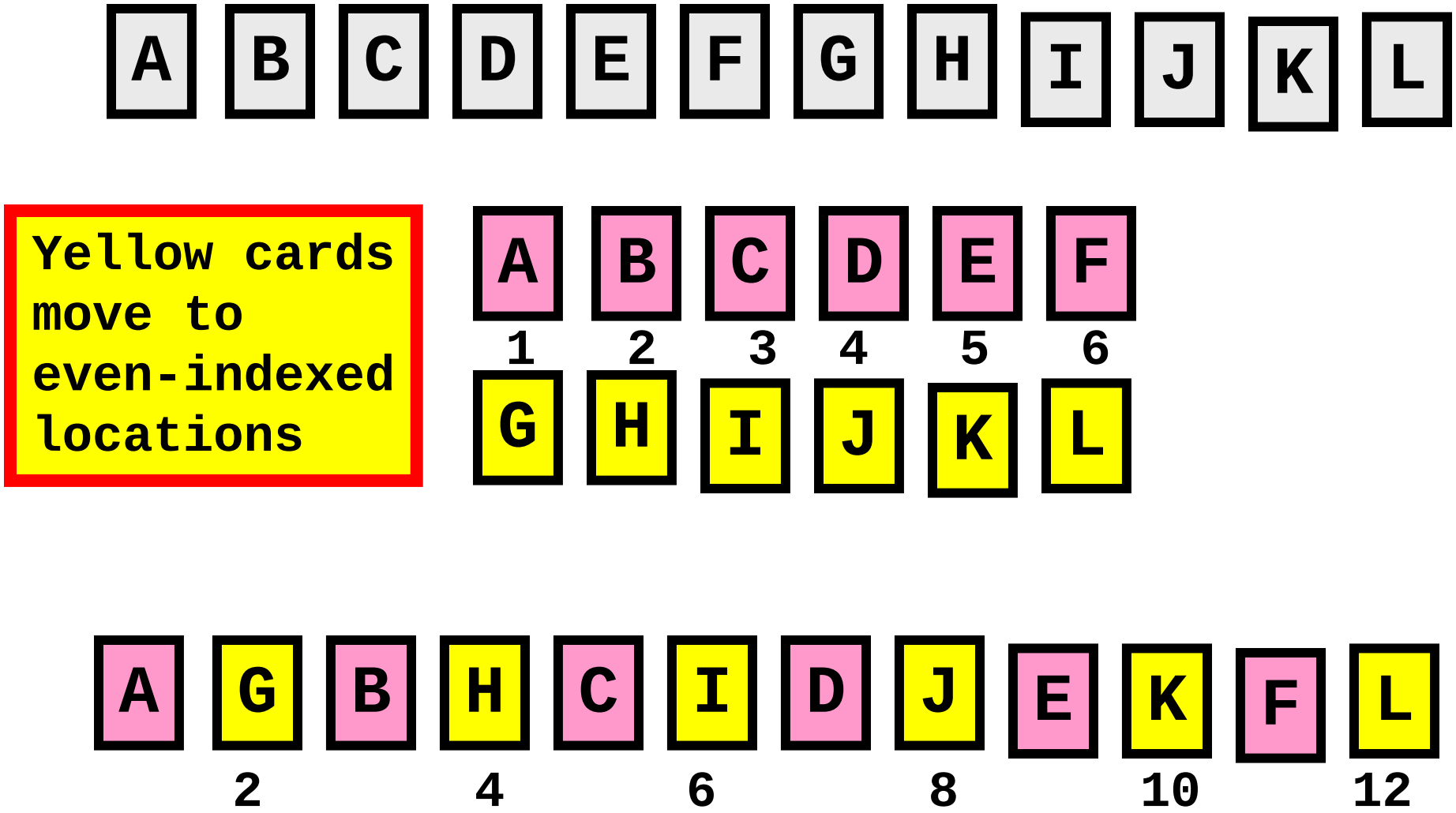
Step 2: Alternate



Step 2: Alternate



Step 2: Alternate



```
  function  T = Shuffle(S)
% S is 1-by-n cell array and
    n is even.
% T is 1-by-n cell array that is a
    perfect shuffle of S.

% Initializations..

% Cut the deck…

% Alternate…
```

```
function  T = Shuffle(S)
% S is 1-by-n cell array and n is
    even.
% T is 1-by-n cell array that is
    a perfect shuffle of S.

% Initializations..
    n=length(S);
    T=cell(1,n);
    mid=n/2;
    Top=cell(1,mid);
    Bot=cell(1,mid);
```

```
  function  T = Shuffle(S)
% S is 1-by-n cell array and n is even.
% T is 1-by-n cell array that is a
  perfect shuffle of S.

            :
% Cut the deck…
  for k= 1:mid
     Top{k}=S{k};        % “pink” cards
     Bot{k}=S{mid+k};    % “yellow cards”
  end
            :
```

```
  function  T = Shuffle(S)
% S is 1-by-n cell array and n is even.
% T is 1-by-n cell array that is a
  perfect shuffle of S.

          :
% Alternate…
  for k= 1:mid
    T{2*k-1} = Top{k}; % pink to odd
    T{2*k}   = Bot{k}; % yellow to even
  end
```

Interesting Fact…

Eight perfect shuffles of an ordinary card deck reinstate the original order.

```
% Assume T = cell(1,52) is
% a deck of cards

for k=1:8
   T = Shuffle(T);
end
```

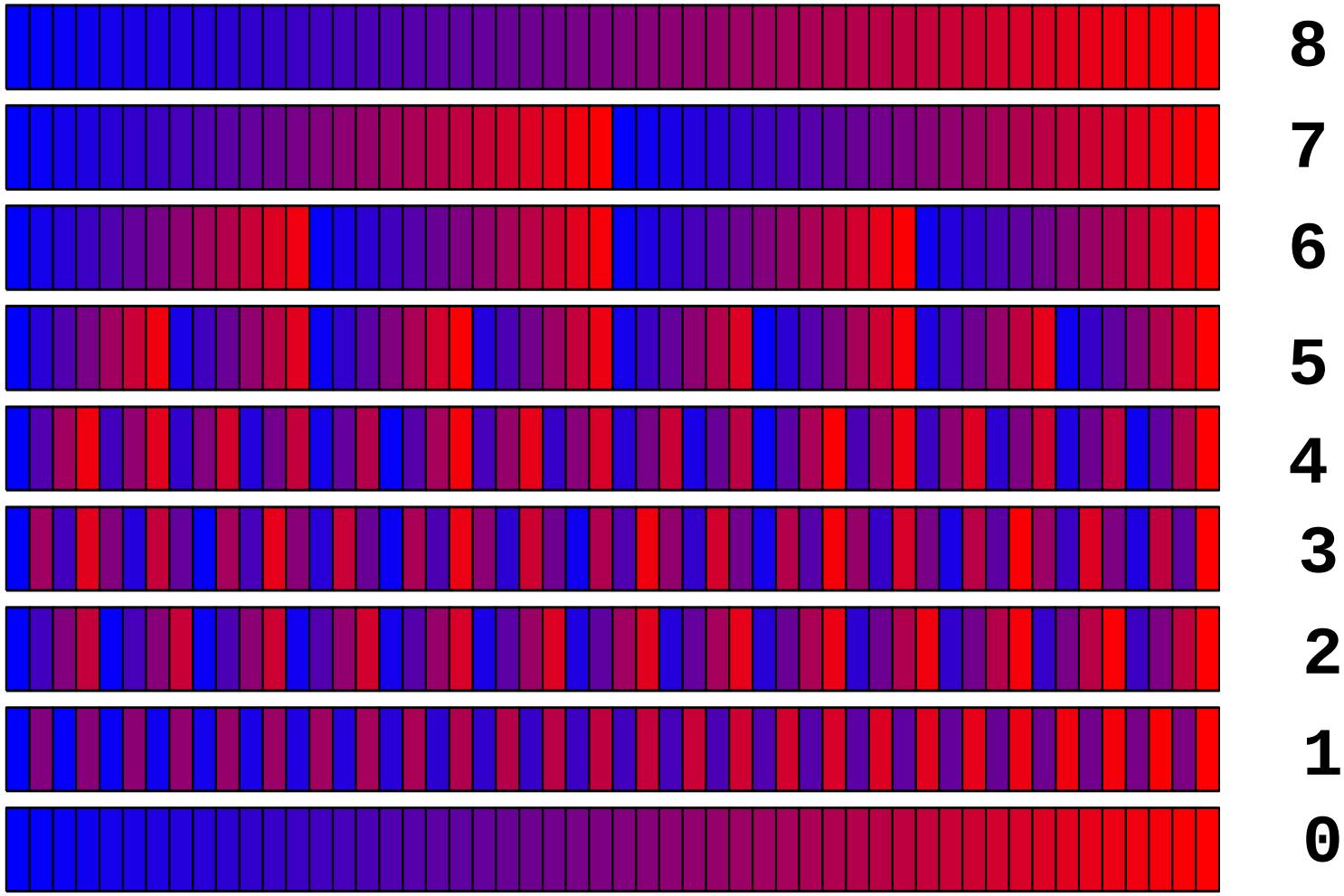
Illustrate with Color

```
% Set up a 52-color spectrum

C = cell(52,1);
for k=1:52
    f = (k-1)/51;
    C{k} = [f 0 1-f];
end
```

These are colors

Using **fill(, , C{k})…**



Example 5

A Cell Array of Roman Numerals

Idea…

```
C{1} = 'I'
C{2} = 'II'
C{3} = 'III'
   :
C{2007} = 'MMVII'
   :
C{3999} = 'MMMXMXCIX'
```

A Conversion Problem

1904 = 1*1000 + 9*100 + 0*10 + 4*1

= M CM IV

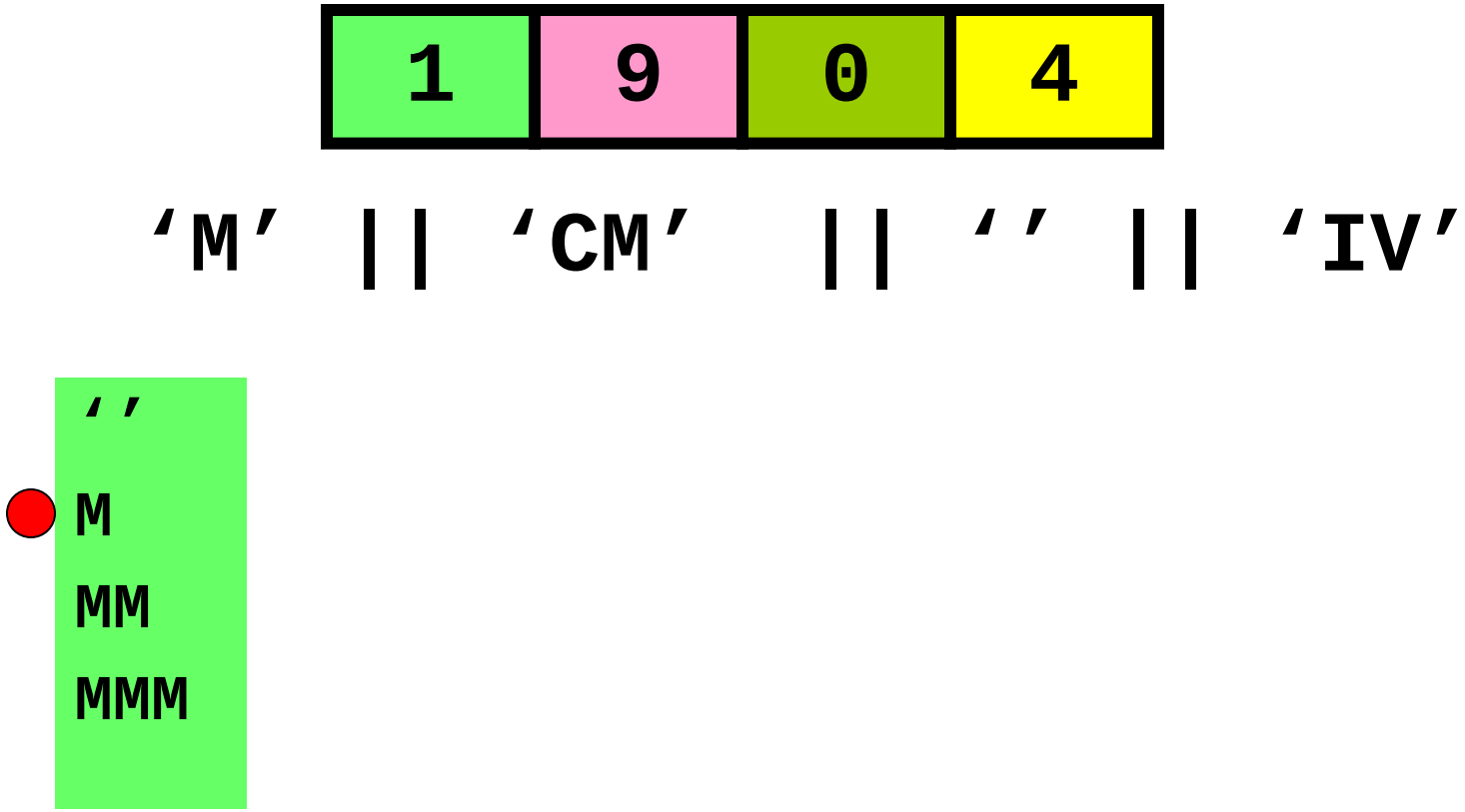
= MCMIV

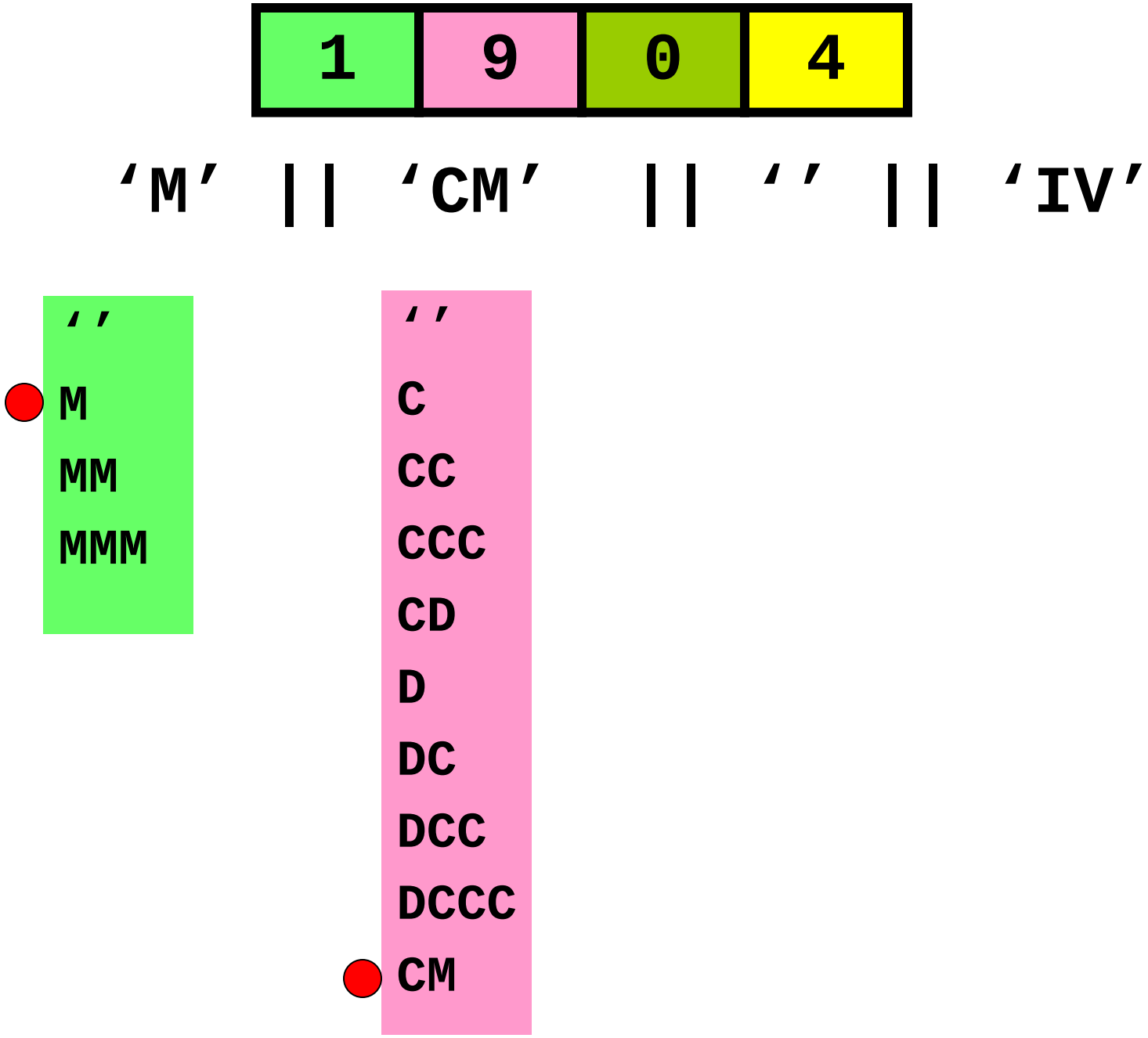
1	9	0	4

MCMIV

1	9	0	4

‘M’ || ‘CM’ || ‘’ || ‘IV’

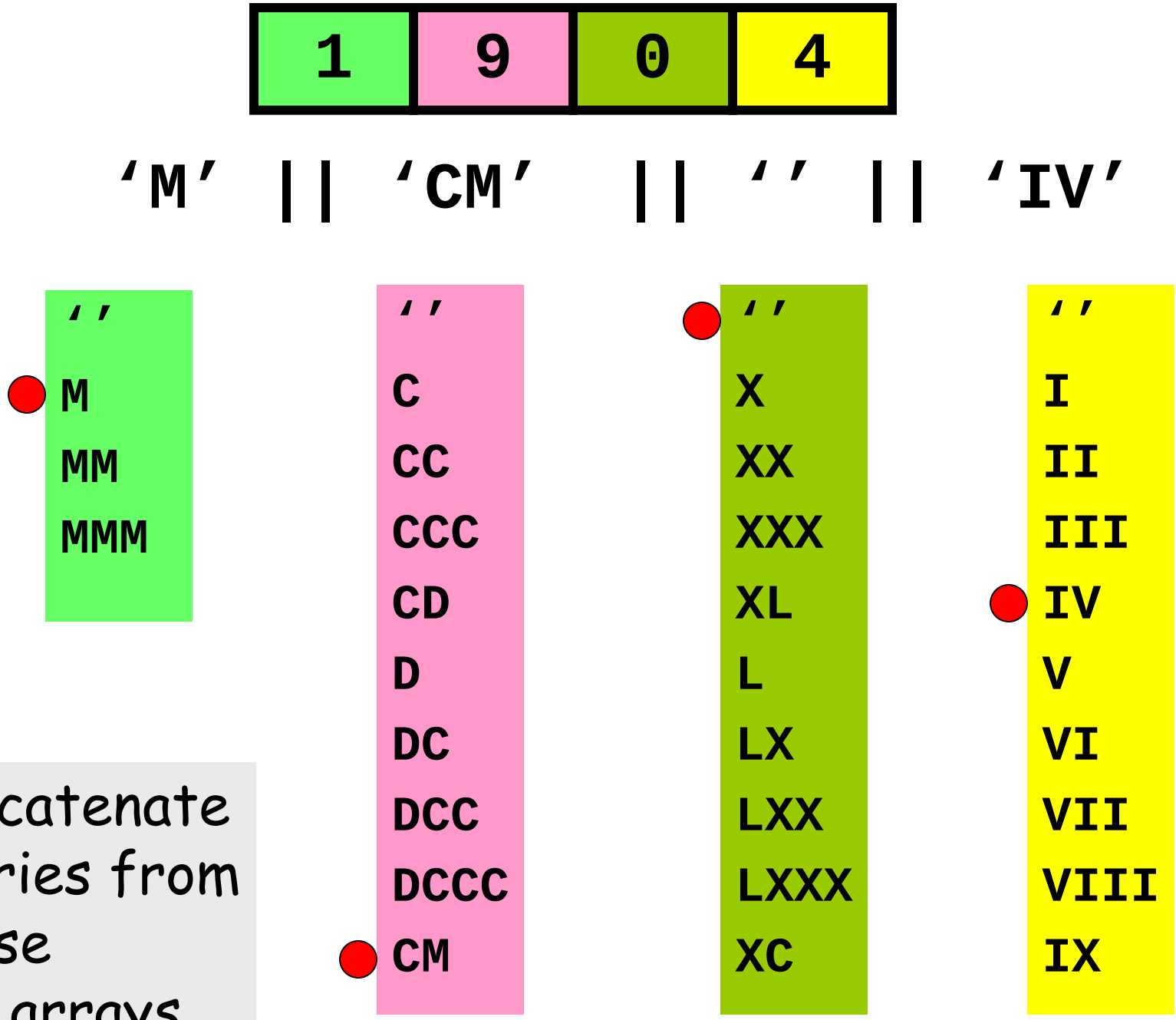

1
9
0
4
'M' || 'CM' || '' || 'IV'
''
M
MM
MMM


1
9
0
4
'M' || 'CM' || '' || 'IV'
''
M
MM
MMM
''
C
CC
CCC
CD
D
DC
DCC
DCCC
CM

1	9	0	4

‘M’ || ‘CM’ || ‘’ || ‘IV’

‘’	‘’	‘’
M	C	X
MM	CC	XX
MMM	CCC	XXX
	CD	XL
	D	L
	DC	LX
	DCC	LXX
	DCCC	LXXX
	CM	XC



Concatenate entries from these cell arrays

Ones-Place Conversion

```
  function r = Ones2R(x)
% x is an integer that satisfies
%       0 <= x <= 9
% r is the Roman numeral with value x.

Ones = {'I', 'II', 'III', 'IV',…
     'V', 'VI','VII', 'VIII', 'IX'};

if x==0
     r = '';
else
     r = Ones{x};
end
```

Tens-Place Conversion

```
  function r = Tens2R(x)
% x is an integer that satisfies
%       0 <= x <= 9
% r is the Roman numeral with value 10x.

Tens = {‘X', ‘XX', ‘XXX', ‘XL',…
     ‘L', ‘LX',‘LXX', ‘LXXX', ‘XC'};

if x==0
     r = '';
else
     r = Tens{x};
end
```

Hundreds-Place Conversion

```
  function r = Hund2R(x)
% d is an integer that satisfies
%       0 <= x <= 9
% r is the Roman numeral with value 100x.

Hund = {‘C', ‘CC', ‘CCC', ‘CD',…
     ‘D', ‘DC',‘DCC', ‘DCCC', ‘CM'};

if x==0
     r = '';
else
     r = Hund{x};
end
```

Thousands-Place Conversion

```
  function r = Thou2R(x)
% d is an integer that satisfies
%       0 <= x <=3
% r is the Roman numeral with value 1000x.

Thou = {‘M', ‘MM', ‘MMM'};

if x==0
     r = '';
else
     r = Thou{x};
end
```

Back to Our Problem

```
C{1} = 'I'
C{2} = 'II'
C{3} = 'III'
   :
C{2007} = 'MMVII'
   :
C{3999} = 'MMMXMXCIX'
```

Generate 1,…,3999

a	b	c	d

$0 <= a <= 3$

$0 <= b <= 9$

$0 <= c <= 9$

$0 <= d <= 9$

This Prints 0,…,3999

```
for a = 0:3
  for b = 0:9
    for c = 0:9
      for d = 0:9

        n = a*1000 + b*100 + c*10 + d

      end
    end
  end
end
```

```
n = a*1000 + b*100 + c*10 + d;
if n > 0

    C{n} = [Thou(a) Hund(b)…
                Tens(c) Ones(d)];

end
```

Reverse Problem

Given Roman Numeral, compute its value.

Assume cell array C(3999,1) available:

```
C{1} = 'I'
   :
C{3999} = 'MMMCMXCIX'
```

```
  function k = RN2Int(r)
% r is a string that represents
%       Roman numeral
% k is its value

  C = RomanNum;
  k=1;
  while ~strcmp(r,C{k})
     k=k+1;
  end
```