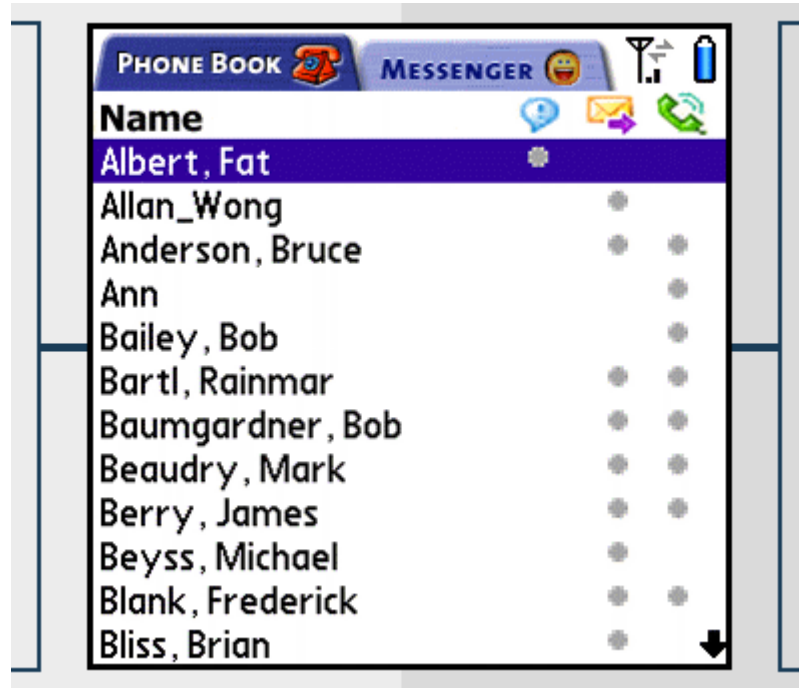


- Previous Lecture:
 - Recursion (Ch. 14)

- Today, Lecture 27:
 - Algorithms for **sorting** and efficiency analysis (Ch. 8)
 - Insertion Sort algorithm
 - See *Insight* §8.2 for the Bubble Sort algorithm
 - Algorithms for **searching** and analysis (Ch. 9)
 - Linear search (review)
 - Binary search

- Announcements:
 - Test 2B submissions due today, 4:30pm EDT
 - Since Tues 5/12 is the last day of classes, the Tues discussion sections will be converted to open office hrs. All students are welcome (Zoom links will be posted to Canvas).
 - Project 6 due Tues 11pm EDT. Remember **academic integrity!**
 - Regular office/consulting hours end on Tues. See Canvas and course website for Study period office/consulting hours
 - Final exam: “2hr” take-home, 48hr submission window. Mon, 5/18, 9am
 - Please complete course evaluations – worth extra point on Final

Sorting data allows us to search more easily



Boston Marathon Top Women Finishers

| Official Time | State | Country | Ctz |
|---------------|-------|---------|-----|
| 2:25:25 | | ETH | |
| 2:25:27 | | RUS | |
| 2:26:34 | | KEN | |
| 2:28:12 | | LAT | |
| 2:29:48 | | ETH | |
| 2:30:52 | | ITA | |
| 2:33:56 | | ROM | |
| 2:34:37 | | ETH | |
| 2:35:37 | | RUS | |
| 2:44:44 | IL | USA | CAN |
| 2:45:54 | NS | CAN | |
| 2:46:25 | | KEN | |
| 2:47:17 | FL | USA | RUS |
| 2:47:36 | | AUS | |
| 2:48:43 | MN | USA | |

| | | |
|---|------|----------------------------|
| 7 | F12 | Olaru, Nuta |
| 8 | F6 | Guta, Robe Tola |
| 9 | F1 | Grigoryeva, Lidiya |
| | F35 | Hood, Stephanie A. |
| | F14 | Robson, Denise C. |
| | F11 | Chemjor, Magdaline |
| | F101 | Sultanova-Zhdanova, Firaya |
| | F15 | Mayger, Eliza M. |
| | F24 | Anklam, Ashley A. |

| Name | Score | Grade |
|--------|-------|-------|
| Jorge | 92.1 | |
| Ahn | 91.5 | |
| Oluban | 90.6 | |
| Chi | 88.9 | |
| Minale | 88.1 | |
| Bell | 87.2 | |

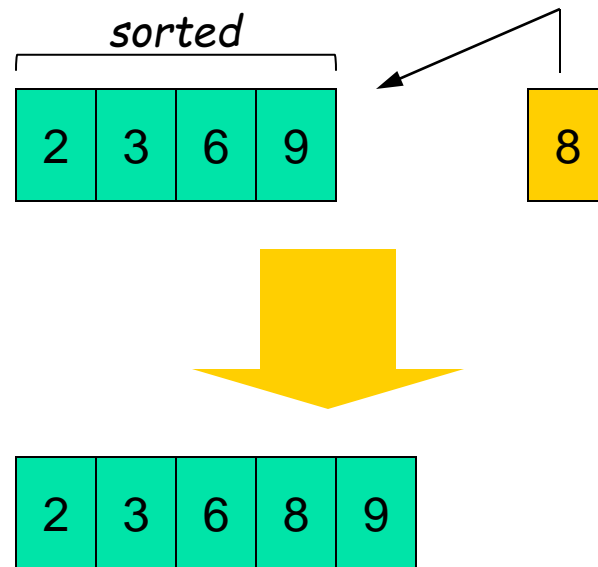
There are many algorithms for sorting

- Insertion Sort (to be discussed today)
- Bubble Sort (read *Insight* §8.2)
- Merge Sort (to be discussed next lecture)
- Quick Sort (a variant used by Matlab's built-in `sort` function)

- Each has advantages and disadvantages. Some algorithms are faster (**time-efficient**) while others are **memory-efficient**
- *Great opportunity for learning how to analyze programs and algorithms!*

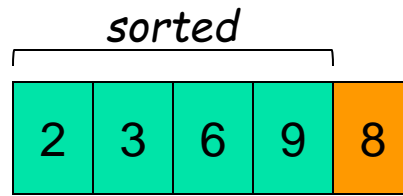
The Insertion Process

- Given a sorted array x , insert a number y such that the result is sorted



Insertion

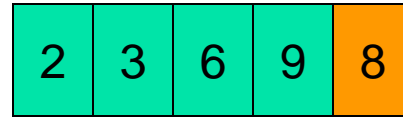
one insert
process



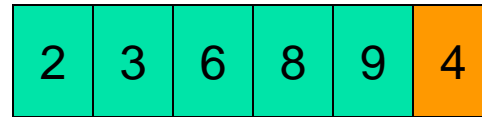
Insert 8 into the sorted segment

Just swap 8 & 9

Insertion

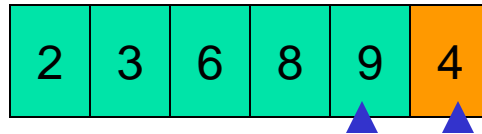
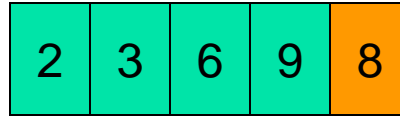


sorted



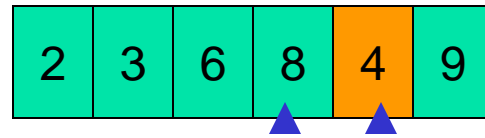
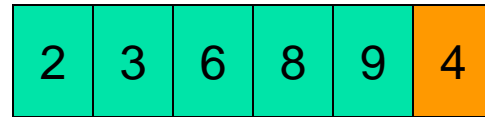
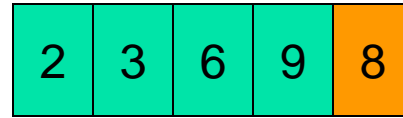
Insert 4 into the sorted segment

Insertion



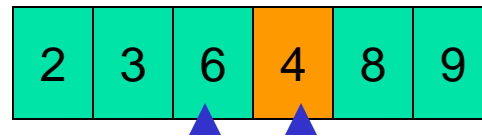
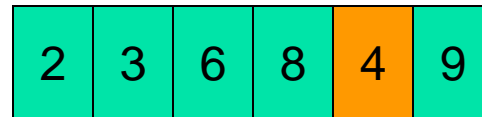
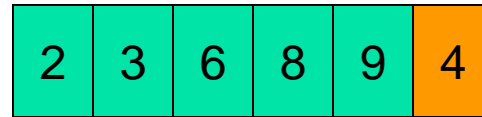
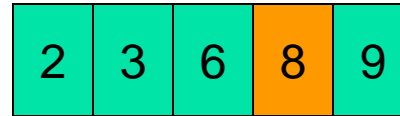
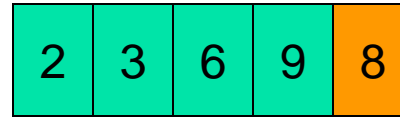
Compare adjacent components:
swap 9 & 4

Insertion



Compare adjacent components:
swap 8 & 4

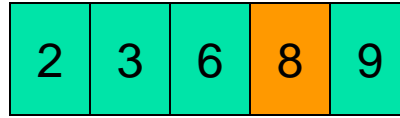
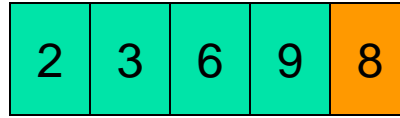
Insertion



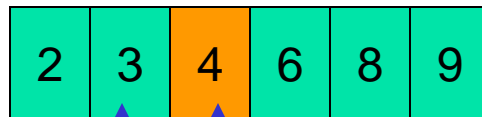
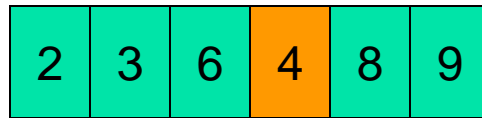
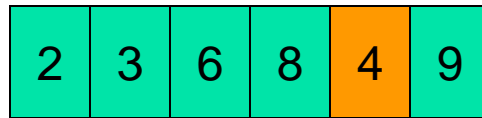
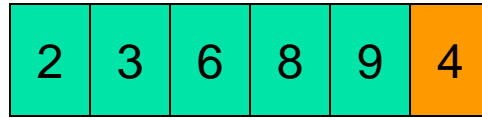
Compare adjacent components:
swap 6 & 4

Insertion

one insert
process



one insert
process

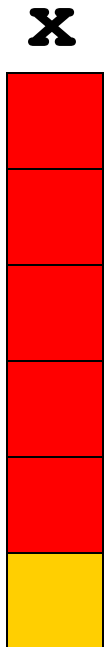


Compare adjacent components:
DONE! No more swaps.

See function `Insert` for the insert process

Sort vector \mathbf{x} using the **Insertion Sort** algorithm

Need to start with a *sorted* subvector. How do you find one?



Length 1 subvector is “sorted”

Insert $\mathbf{x}(2) : \mathbf{x}(1:2) = \text{Insert}(\mathbf{x}(1:2))$

Insert $\mathbf{x}(3) : \mathbf{x}(1:3) = \text{Insert}(\mathbf{x}(1:3))$

Insert $\mathbf{x}(4) : \mathbf{x}(1:4) = \text{Insert}(\mathbf{x}(1:4))$

Insert $\mathbf{x}(5) : \mathbf{x}(1:5) = \text{Insert}(\mathbf{x}(1:5))$

Insert $\mathbf{x}(6) : \mathbf{x}(1:6) = \text{Insert}(\mathbf{x}(1:6))$

`insertionSortSimple.m`

Contract between Insert and InsertionSort

Insert

- Assumes all but the last element of x is already sorted
- Returns a fully-sorted array (one more element sorted than given)

therefore

InsertionSort (driver)

- Must only call Insert() on a subarray with a pre-sorted prefix
- Has a bigger pre-sorted subarray to pass to Insert() next time – progress is made each iteration

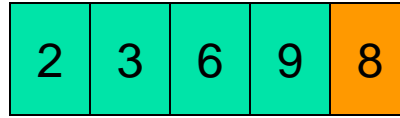
Size of sorted prefix grows each time.
When it equals the size of the original
array, the task is done

How much “work” is insertion sort?

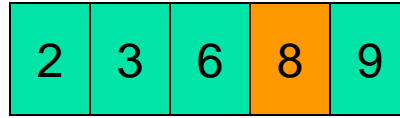
- In the worst case, make k comparisons to insert an element in a sorted array of k elements.

Insertion

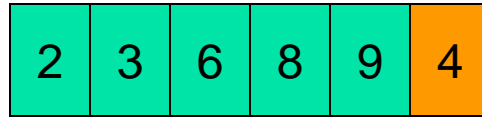
one insert
process



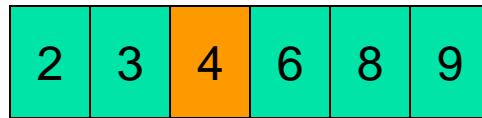
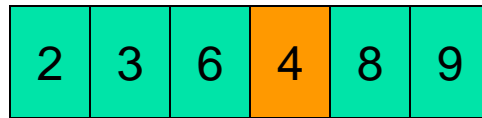
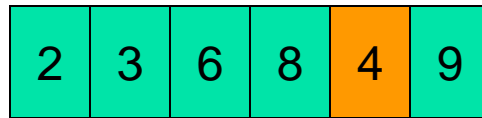
Insert into sorted array of
length 4



one insert
process



Insert into sorted array of
length 5



How much “work” is insertion sort?

- In the worst case, make k comparisons to insert an element in a sorted array of k elements. For an array of length N :

$$1 + 2 + \dots + (N-1) = \frac{N(N-1)}{2}, \text{ say } N^2 \text{ for big } N$$

`InsertionSort.m`

Checkpoint question: N^2 performance

Suppose it takes 5ms to sort an array with 100 elements using Insertion Sort. How long would you expect sorting 1000 elements to take?

A. 25ms

C. 500ms

E. $1e6$ ms

B. 50ms

D. 5000ms

Efficiency considerations

- Worst case, best case, average case
- Use of subfunction incurs an “overhead”
- Memory use and access

- Example: Rather than directing the *insert* process to a subfunction, have it done “**in-line.**”
- Also, Insertion sort can be done “**in-place,**” i.e., using “only” the memory space of the original vector.

```
function x = InsertionSortInplace(x)
% Sort vector x in ascending order with insertion sort

n = length(x);
for i= 1:n-1
    % Sort x(1:i+1) given that x(1:i) is sorted

end
```

```
function x = InsertionSortInplace(x)
% Sort vector x in ascending order with insertion sort

n = length(x);
for i= 1:n-1
    % Sort x(1:i+1) given that x(1:i) is sorted
    j= i;

    while

        % swap x(j+1) and x(j)

        j= j-1;

    end
end
end
```

Do this for review later!

A note on optimization

- “Inlining” multiple pieces of an algorithm should *not* be your go-to strategy
 - It’s easier to understand (and verify) small pieces that do a simple task than monolithic code that does a complicated task
 - Better communication, less buggy
 - Hard to predict when it will actually be faster
 - Large code has a performance cost in addition to a maintenance cost
 - Measuring performance not as easy as it sounds
 - Compilers can do this automatically
 - Auto-inlining will reveal opportunities for in-place array edits

Sort an array of objects

- Given `x`, a 1-d array of `Interval` references, sort `x` according to the widths of the `Intervals` from narrowest to widest
- Use the insertion sort algorithm
- How much of our code needs to be changed?

A. No change

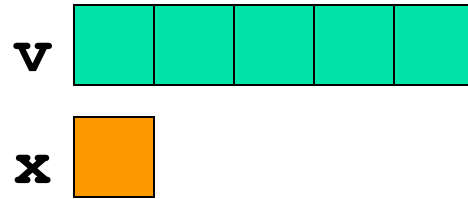
B. One statement

C. About half the code

D. Most of the code

Searching for an item in an unorganized collection?

- May need to look through the whole collection to find the target item
- E.g., find value x in vector v



- Linear search

```
% Linear Search
% f is index of first occurrence
%   of value x in vector v.
% f is -1 if x not found.

k= 1;
while k<=length(v) && v(k)~=x
    k= k + 1;
end
if k>length(v)
    f= -1; % signal for x not found
else
    f= k;
end
```

| | | | | | | |
|----------|----|----|----|----|----|----|
| v | 12 | 35 | 33 | 15 | 42 | 45 |
| x | 31 | | | | | |

```
% Linear Search
% f is index of first occurrence of value x in vector v.
% f is -1 if x not found.

k= 1;
while k<=length(v) && v(k)~=x
    k= k + 1;
end
if k>length(v)
    f= -1; % signal for x not found
else
    f= k;
end
```

n comparisons against the target
are needed in worst case,
n=length(v) .


```
% Linear Search
% f is index of first occurrence
%   of value x in vector v.
% f is -1 if x not found.
```

```
k= 1;
while k<=length(v) && v(k)~=x
    k= k + 1;
end
if k>length(v)
    f= -1; % signal for x not found
else
    f= k;
end
```

Searching in
a sorted list should
require less work

| | | | | | | |
|----------|----|----|----|----|----|----|
| v | 12 | 15 | 33 | 35 | 42 | 45 |
| x | 31 | | | | | |

What if **v** is sorted?

An ordered (sorted) list

The Manhattan phone book has 1,000,000+ entries.

How is it possible to locate a name by examining just a tiny, tiny fraction of those entries?

wide at SuperPages.com

195 Car C

| | | | | | | |
|--------------|--|--------------|--|--------------|--|--------------|
| 17 566-1282 | Cartage New England Inc 26 Allen Ln Ipswich 01938..... | 978 356-9960 | Carter F 24 Hillock Ros 02131..... | 617 327-1105 | Carter Nella E 333 Maschts Av Bos 02115..... | 617 267-6483 |
| 81 447-4101 | Cartagema Lydia 18 Jewett Ros 02131..... | 617 323-7639 | Faye & Ricky 357 Columbus Av Bos 02116..... | 617 437-7331 | Nicholas S F 115 Randolph Av Mil 02186..... | 617 698-5307 |
| 800 257-9981 | Cartagema Avith 9 Bancroft Rox 02119..... | 617 442-9780 | Francis S 134 Temple W Rox 02132.. | 617 323-6781 | Nick 21 Fairfield Bos 02116..... | 617 267-5222 |
| 17 566-1282 | B Hyd 02136..... | 617 361-5253 | Franklin & Anne 221 Mt Auburn Cam 02138..... | 617 354-0798 | Nick & Debbi 196 Herrick Rd Newton 02459..... | 617 527-0480 |
| 17 364-5188 | Jessica 50 Decatur Cha 02129..... | 617 241-0152 | Fred 42 Haverford Jam 02130..... | 617 524-3078 | Nicole..... | 617 698-0713 |
| 361-0380 | Lucilla 174 Harvard Cam 02139..... | 617 491-5621 | Fred 96 Hinckley Rd Mil 02186..... | 617 698-1343 | Norman G 38 Chickatawbut Dor 02122..... | 617 822-1203 |
| 17 566-4548 | M 95 Rowe Ros 02131..... | 617 323-9713 | G & R 8 Verdun Dor 02124..... | 617 436-8906 | P 94 Crestwood Pk Rox 02121..... | 617 427-4754 |
| 17 628-8248 | Melvin 501 Green Cam 02139..... | 617 576-1061 | G T 27 Franklin Av Som 02145..... | 617 623-7121 | P E 501 E Sixth S Bos 02127..... | 617 268-4213 |
| 17 445-5116 | Carte Nicholas 18 Appleton Boston 02116..... | 617 695-6996 | Gayle 25 Frontenac Dor 02124..... | 617 825-0322 | P L 44 Hutchings Rox 02121..... | 617 427-9170 |
| 17 822-2982 | Cartegena O 4 Millford Bos 02118..... | 617 338-8219 | George 125 Nashua Bos 02114..... | 617 367-9548 | P R 91 Byrner Jam 02130..... | 617 983-8692 |
| 17 569-2698 | Carten Thos J Sr & Claire 1 Paradise Rd Mil 02186..... | 617 698-6163 | Carter Halliday Associate 107 S Street Bos 02111..... | 617 456-1689 | Paul & Constance 114 Anawan Av W Rox 02132..... | 617 325-2036 |
| 17 667-5190 | Thomas & Kathleen 50 Thompson Ln Mil 02186..... | 617 696-6919 | Carter Harry F 26 Runng Bk Rd W Rox 02132..... | 617 325-5465 | Paul E 501 E Sixth St S Bos 02127.. | 617 268-4546 |
| 17 569-1417 | Carter A Ros 02131..... | 617 327-2257 | Carter Hide Co Inc 146 Summer Bos 02110..... | 617 542-7987 | Paul M 27 Union Bri 02135..... | 617 787-2115 |
| 17 338-9110 | A Roxbury..... | 617 442-5230 | Carter Hilary 61 Harvey Cam 02140.. | 617 876-2750 | Carter Pile Driving Inc 17 Beaver Ct Frammingham 01702..... | 617 235-8488 |
| 17 825-9195 | A 31 Bethune Wy Roxbury 02119..... | 617 442-1219 | Horace 241 Walnut Av Roxbury 02119..... | 617 442-5307 | Carter Prudence 46 Franklin Watertown 02172..... | 617 393-3782 |
| 17 296-1593 | A 260 Putnam Av Cambridge 02139..... | 617 492-4174 | Howard Jr 26 Notre Dme Rox 02119.. | 617 445-5552 | Prudence 46 Franklin Watertown 02172..... | 617 926-7063 |
| 17 670-2078 | A M 255 Maschts Av Bos 02115..... | 617 266-7153 | J Cam..... | 617 354-2688 | Reginald 106 Brunswick Dorchester 02121..... | 617 541-2843 |
| 17 623-9001 | Adams 361 Centre St Mil 02186..... | 617 698-9074 | J 15 Chatham Bro 02446..... | 617 232-7990 | Renee & Andrew 10 Walnut Bos 02108..... | 617 720-3765 |
| 17 296-4725 | Alice 108 Kilmarnock Bos 02215..... | 617 425-0193 | J 518 Harvard Bro 02446..... | 617 730-9483 | Carter Rice Dowd Bulky Dutton Publishing 163 Main Wilmingt 01887 | 800 638-1671 |
| 17 542-1521 | Alice 45 Market Cambridge 02139..... | 617 945-2711 | J 775 Yw Pkwy West Roxbury 02132.. | 617 323-5574 | Toll Free-Dial '1' & Then..... | 800 638-1671 |
| 17 364-5232 | Andrew F 62 Vinal Av Som 02143..... | 617 625-7623 | Carter J Jacques MD 1 Brookline Pl Bro 02446..... | 617 735-8787 | Cust Svc-Industrial Prod 613 Main Wilmingt Toll Free-Dial '1' & Then..... | 800 619-7447 |
| 17 541-5649 | Carter Anne MD 1101 Beacon Bro 02446..... | 617 739-1022 | Carter J M 1410 Columbia Rd S Bos 02127..... | 617 464-1040 | Cust Svc-Printing 613 Main Wilmingt Toll Free-Dial '1' & Then..... | 800 648-7447 |
| 17 739-2662 | Carter Athens 272 Newbury Boston 02116..... | 617 536-6329 | Carter J M Ornamental Ironworks Call..... | 617 436-5353 | Headquarters 613 Main Wilmingt 01887 Call..... | 978 988-7447 |
| 17 879-0030 | B E 48 Gladeside Av Mat 02126..... | 617 296-6911 | Carter J Veal Co 48 Newmarket Sq Rox 02118..... | 617 442-1775 | Ingalls Cronin 163 Main Wilmingt 01880 Toll Free-Dial '1' & Then..... | 800 638-1673 |
| 17 541-3948 | Carter Barbara L MD Tufts-New England Medical Center Bos 02111 Call..... | 617 636-0051 | Carter James 1573 Cambridge St Cam 02138..... | 617 492-1214 | Carter Richard 1079 Commwllth Av Brighton 02215.... | 617 987-0836 |
| 17 569-4119 | Carter Becky Bos 02114..... | 617 523-4368 | James 182 Fisher Av Roxbury 02120..... | 617 739-2193 | Richard A 97 Mt Vernon Bos 02108.... | 617 566-7293 |
| 800 569-8782 | Bernard J 112 Gladstone E Bos 02128..... | 617 567-3430 | James 37 Gold Star Rd Cambridge 02140.... | 617 876-8841 | Carter Richard A MD 170 Commwllth Av Bos 02116..... | 617 267-0710 |
| | Bithiah 25 Medway Dor 02124..... | 617 298-8713 | Jas L 14 Roseberry Rd Mat 02126..... | 617 361-0773 | Carter Richard K 15 Mercer S Bos 02127..... | 617 268-0448 |
| | Blake 26 Mt Vernon Bos 02108..... | 617 367-9931 | Jane 114 Adena Rd Newton 02465..... | 617 964-0435 | Robert L 175 Richtale Av Cam 02140.. | 617 864-1535 |
| | Carter Broadcasting Co 20 Park Pkz Bos 02116..... | 617 423-0210 | Jeffrey 41 Warren Av Bos 02116..... | 617 426-5994 | Roger 150 St Botolph Bos 02115..... | 617 424-6148 |
| | Carter & Burgess Consultants Inc 23 East St Cam 02141..... | 617 225-0200 | John 11 Mansfield Bri 02134..... | 617 987-2163 | Roy 44 Concord Av Cam 02138..... | 617 491-6115 |
| | Carter C 2000 Commwllth Av Bri 02135.... | 617 782-2118 | John 327 Summer Bos 02210..... | 617 423-4334 | Royce 18 Seminary Cha 02129..... | 617 241-0418 |
| | C 228 Faywood Av East Boston 02128.... | 617 569-1545 | John 40 Westwind Rd Dor 02125..... | 617 282-1235 | | |
| | C 359 Harvard Cam 02138..... | 617 491-4822 | June O 329 A Summit Av Bri 02135.... | 617 734-6109 | | |
| | C 610 Walk Hill Mat 02126..... | 617 296-6392 | K 38 Browning Av Dorchester 02124.... | 617 265-8456 | | |
| | C & M 43 Burroughs Jam 02130..... | 617 524-9558 | K 17 Esmond Dorchester 02121..... | 617 282-1593 | | |

Key idea of “phone book search”: repeated halving

To find the page containing Pat Reef’s number...

```
while (Phone book is longer than 1 page)
  Open to the middle page.
  if “Reef” comes before the first entry,
    Rip and throw away the 2nd half.
  else
    Rip and throw away the 1st half.
  end
end
```

What happens to the phone book length?

| | | |
|-----------------------|-------------|--------------|
| Original: | 3000 | pages |
| After 1 rip: | 1500 | pages |
| After 2 rips: | 750 | pages |
| After 3 rips: | 375 | pages |
| After 4 rips: | 188 | pages |
| After 5 rips: | 94 | pages |
| : | | |
| After 12 rips: | 1 | page |

Binary Search

Repeatedly halving the size of the “search space” is the main idea behind the method of **binary search**.

An item in a sorted array of length n can be located with just $\log_2 n$ comparisons.

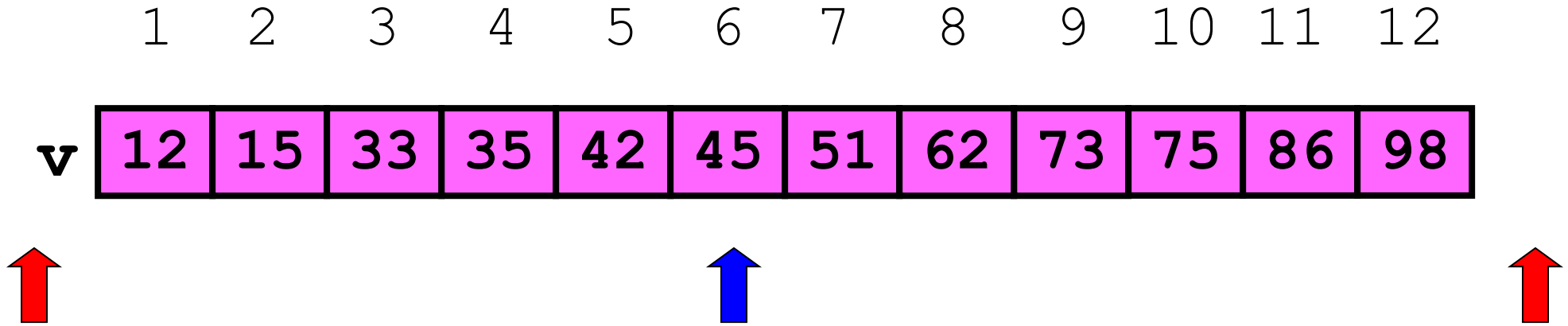
“Savings” is significant!

| n | $\log_2(n)$ |
|-------|-------------|
| 100 | 7 |
| 1000 | 10 |
| 10000 | 13 |

What is true of the half we keep?

- Let L be the leftmost page we keep (may be 0, aka front cover)
- Let R be the page after the last one we keep (might be $\text{length}(v)+1$, aka back cover)
- Then the name we are looking for is \geq the first name on page L , and $<$ the first name on page R
- When only one page left ($R = L+1$),
 - If name is in book, it will be on page L
 - If name is not in book, it should be inserted after some names already on page L

Binary search: target $x = 70$



L: 0

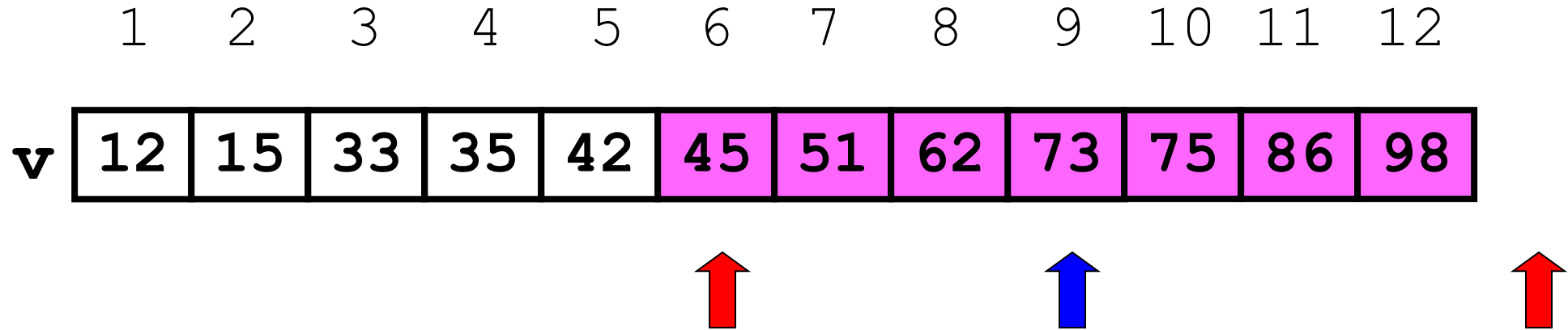
Mid: 6

R: 13

$$v(\text{Mid}) \leq x$$

So throw away the left half...

Binary search: target $x = 70$



L: 6

Mid: 9

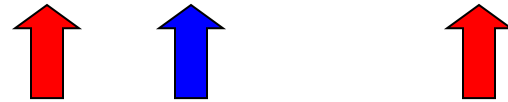
R: 13

$x < v(\text{Mid})$

So throw away the right half...

Binary search: target $x = 70$

| | | | | | | | | | | | | |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| v | 12 | 15 | 33 | 35 | 42 | 45 | 51 | 62 | 73 | 75 | 86 | 98 |



L: 6

Mid: 7

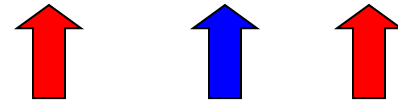
R: 9

$$v(\text{Mid}) \leq x$$

So throw away the left half...

Binary search: target $x = 70$

| | | | | | | | | | | | | |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| v | 12 | 15 | 33 | 35 | 42 | 45 | 51 | 62 | 73 | 75 | 86 | 98 |



L: 7

Mid: 8

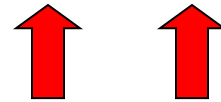
R: 9

$$v(\text{Mid}) \leq x$$

So throw away the left half...

Binary search: target $x = 70$

| | | | | | | | | | | | | |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| v | 12 | 15 | 33 | 35 | 42 | 45 | 51 | 62 | 73 | 75 | 86 | 98 |



L:

8

Mid:

8

R:

9

Done because

$$R - L = 1$$

```
function L = binarySearch(x, v)
% Find position after which to insert x. v(1)<...<v(end) .
% L is the index such that v(L) <= x < v(L+1);
% L=0 if x<v(1).  If x>v(end), L=length(v) but x~=v(L) .

% Maintain a search window [L..R] such that v(L)<=x<v(R) .
% Since x may not be in v, initially set ...
L=0;  R=length(v)+1;
```

```
% Keep halving [L..R] until R-L is 1,
% always keeping v(L) <= x < v(R)
while R ~= L+1
    m= floor((L+R)/2);  % middle of search window
    if
        v(m) <= x < v(m+1)
            R=m;
        else
            L=m;
        end
    end
end
```

```

function L = binarySearch(x, v)
% Find position after which to insert x. v(1)<...<v(end) .
% L is the index such that v(L) <= x < v(L+1) ;
% L=0 if x<v(1).  If x>v(end), L=length(v) but x~v(L) .

% Maintain a search window [L..R] such that v(L)<=x<v(R) .
% Since x may not be in v, initially set ...
L=0;  R=length(v)+1;

% Keep halving [L..R] until R-L is 1,
%   always keeping  v(L) <= x < v(R)
while  R ~= L+1
    m= floor((L+R)/2);  % middle of search window
    if  v(m) <= x

        L= m;
    else

        R= m;
    end
end
end

```

This version is different
from that in *Insight*

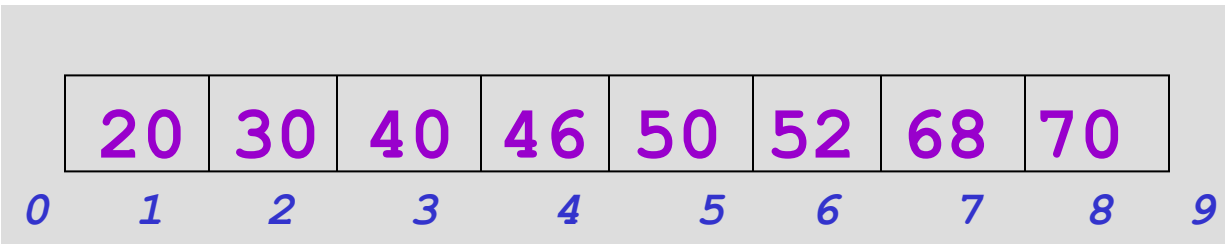
```

function L = binarySearch(x, v)
% Find position after which to insert x. v(1)<...<v(end) .
% L is the index such that v(L) <= x < v(L+1) ;
% L=0 if x<v(1).  If x>v(end), L=length(v) but x~v(L) .

% Maintain a search window [L..R] such that v(L)<=x<v(R) .
% Since x may not be in v, initially set ...
L=0;  R=length(v)+1;

% Keep halving [L..R] until R-L is 1,
% always keeping v(L) <= x < v(R)
while R ~= L+1
    m= floor((L+R)/2);  % middle of search window
    if v(m) <= x
        L= m;
    else
        R= m;
    end
end
end

```



Play with `showBinarySearch.m`

What happens if the values in the sorted vector are not unique? Say, the target value is in the vector and that value appears in the vector multiple times...

A. The first occurrence is identified

B. The last occurrence is identified

C. Any one of the occurrences may be identified

D. Binary search doesn't work

Binary search is efficient, but we need to sort the vector in the first place so that we can use binary search

- Many different algorithms out there...
- We saw insertion sort (and read about bubble sort)
- Let's look at **merge sort**
- An example of the “divide and conquer” approach using recursion