

22. Searching a List

Topics:

Linear Search

Binary Search

Measuring Execution Time

The Divide and Conquer Framework

Search

Examples:

Is this song in that playlist?

Is this number in that phone book?

Is this name in that phone book?

Is this fingerprint in that archive of fingerprints?

Is this photo in that yearbook?

More on Using Phone Books

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 |
|--|------------------------------|---|
| Cartage New England Inc 26 Allen Ln Jamaica 01088..... | 617 356-9960 | |
| Cartagena Lydia 38 Jewett Rte 02131..... | 617 323-7639 | |
| Cartagena Arith 9 Boscroft Rte 02119..... | 617 442-9780 | |
| B Hrd 02126..... | 617 361-5253 | |
| Jessica 48 Decatur Cha 02129..... | 617 241-0152 | |
| Lucilla 174 Harvard Cam 02136..... | 617 491-5621 | |
| M 95 Rowe Rte 02133..... | 617 323-9713 | |
| Melvin 501 Green Can 02139..... | 617 576-1061 | |
| Carte Nicholas 18 Appleton Boston 02136..... | 617 695-6996 | |
| Cartegena O 4 Millard Rte 02118..... | 617 338-8219 | |
| Carten Thos J Sr & Claire 1 Paradise Rd Mt 02186..... | 617 698-6163 | |
| Thomas & Kathleen 50 Thompson Ln Mt 02186..... | 617 696-6919 | |
| Cartier A Rte 02131..... | 617 327-2257 | |
| A Rosbury A 31 Bethune Wy Rosbury 02119..... | 617 442-1219 | |
| A 200 Putnam Av Cambridge 02139..... | 617 492-4174 | |
| A M 255 Mearns Av Rte 02135..... | 617 266-7153 | |
| Adams 361 Centre St Mt 02186..... | 617 698-9074 | |
| Alice 108 Kilmackock Bsc 02135..... | 617 425-0193 | |
| Alice 45 Market Cambridge 02139..... | 617 945-2711 | |
| Andrew F 62 Veal Av Son 02143..... | 617 625-7623 | |
| Cartier Anne MD 1101 Beacon Bro 02446..... | 617 739-1022 | |
| Cartier Athens 272 Newbury Boston 02116..... | 617 536-6329 | |
| B E 68 Gloucester Av Mt 02126..... | 617 296-6911 | |
| Cartier Barbara L MD Turbo-New England Medical Center Box 02111 Cam..... | 617 636-0051 | |
| Cartier Becky Ros 02114..... | 617 523-4368 | |
| Bernard J 112 Guilford E Rte 02128..... | 617 567-3430 | |
| Bithiah 25 Medway Der 02124..... | 617 298-8713 | |
| Blake 26 Mt Vernon Bsc 02108..... | 617 367-9931 | |
| Cartier Broadcasting Co 26 Park Pl Rte 02126..... | 617 423-0210 | |
| Cartier & Burgess Consultants Inc 23 East St Cam 02143..... | 617 225-0200 | |
| Cartier C 2000 Cornwell Av Br 02135..... | 617 782-2118 | |
| C 228 Faywood Av East Boston 02128..... | 617 569-1545 | |
| C 209 Harvard Cam 02136..... | 617 491-8822 | |
| C 610 Wale Hill Mt 02126..... | 617 296-6392 | |
| C & M 43 Burroughs Jan 02130..... | 617 524-9558 | |
| Cartier F 24 Hilluck Rte 02131..... | 617 327-1105 | |
| Faye & Ricky 357 Columbus Av Rte 02136..... | 617 437-7331 | |
| Francis S 134 Temple W Rte 02132..... | 617 323-6781 | |
| Franklin & Anne 221 Mt Auburn Cam 02138..... | 617 354-0798 | |
| Fred 40 Haverford Jan 02130..... | 617 524-3678 | |
| Fred 95 Hockley Rd Mt 02186..... | 617 698-1343 | |
| G & R 8 Vernon Der 02134..... | 617 436-8906 | |
| G T 27 Franklin Av Son 02145..... | 617 623-7121 | |
| Gayle 25 Frontenac Der 02124..... | 617 825-0322 | |
| Geo S 115 Morse Hill Rd Jan 02136..... | 617 522-3215 | |
| George 125 Nashua Rte 02134..... | 617 367-9548 | |
| Cartier Halliday Associate 107 S Street Rte 02111..... | 617 456-1689 | |
| Cartier Harry F 26 Spring Brd W Rte 02112..... | 617 325-5465 | |
| Cartier Hide Co Inc 146 Sumner Box 02110..... | 617 542-7987 | |
| Cartier Hilary 61 Harvey Cam 02140..... | 617 876-2750 | |
| Horace 241 Walnut Av Rosbury 02119..... | 617 442-5307 | |
| Howard Jr 26 Notre Dame Rte 02119..... | 617 445-5552 | |
| J Cam J 15 Chatham Bro 02446..... | 617 232-7990 | |
| J 518 Harvard Bro 02446..... | 617 730-9483 | |
| J 775 Via Pines Wood Rosbury 02132..... | 617 323-5574 | |
| Cartier J Jacques MD 3 Brookline Pl Br 02446..... | 617 735-8787 | |
| Cartier J M 1410 Columbus Rd S Bsc 02127..... | 617 464-1040 | |
| Cartier J M Ornamental Ironworks Pembroke Telfo 617 436-5353 | | |
| Cartier J Veal Co 46 Newmarket Sq Rte 02118..... | 617 442-1775 | |
| Cartier James 1572 Cambridge St Cam 02138..... | 617 492-1214 | |
| James 182 Fisher Av Rosbury 02130..... | 617 739-2193 | |
| James 37 Gold Star Rd Cambridge 02140..... | 617 876-8841 | |
| Jas L 14 Rosbury Rd Mt 02126..... | 617 361-0773 | |
| Jane 134 Adams Rd Newton 02459..... | 617 564-0435 | |
| Jeffrey 41 Warren Av Rte 02116..... | 617 426-5964 | |
| John 11 Mansfield Br 02134..... | 617 987-2113 | |
| John 327 Sumner Box 02110..... | 617 423-4334 | |
| John 40 Westfield Rd Der 02125..... | 617 282-1235 | |
| June O 329 A Summit Av Br 02135..... | 617 734-6109 | |
| K 38 Bromfield Av Dorchester 02124..... | 617 265-8456 | |
| K 17 Exmouth Dorchester 02123..... | 617 282-1593 | |
| Cartier Nella E 323 Manchester Av Box 02115..... | 617 267-6483 | |
| Nicholas S F 115 Randolph Av Mt 02186..... | 617 698-5307 | |
| Nick & Debbie 116 Herrick Rd Newton 02459..... | 617 267-5222 | |
| Nicole 38 Chickatawbut Der 02122..... | 617 527-0480 | |
| Norman G 28 Chickatawbut Der 02122..... | 617 698-0713 | |
| P 46 Cranford Pl Rte 02123..... | 617 822-1203 | |
| P E 501 E South St Rte 02137..... | 617 427-4754 | |
| P L 44 Hockley Rte 02121..... | 617 268-4213 | |
| P R 91 Byrnes Jan 02136..... | 617 427-9170 | |
| Paul & Constance 114 Anawan Av W Rte 02118..... | 617 983-8692 | |
| Paul E 501 E South St Bsc 02137..... | 617 325-2036 | |
| Paul M 27 Union Br 02135..... | 617 268-4546 | |
| Cartier Pile Driving Inc 17 Beaver Ct Framingham 01702..... | 617 787-2115 | |
| Cartier Prudence 46 Franklin Watertown 02172..... | Wellesley Telfo 781 235-8488 | |
| Prudence 46 Franklin Watertown 02172..... | 617 393-3782 | |
| Reginald 46 Franklin Watertown 02172..... | 617 926-7063 | |
| Renee & Andrew 30 Walnut Box 02108..... | 617 541-2843 | |
| Cartier Rice Dowd Bulky Defton Publishing 163 Main Wilmington 01897 Toll Free-041 71 & Then..... | 617 720-3765 | |
| Cost Sec Industrial Prod 613 Main Wilmington Toll Free-041 71 & Then..... | 800 638-1671 | |
| Cost Sec-Printing 613 Main Wilmington Toll Free-041 71 & Then..... | 800 619-7447 | |
| Headquarters 613 Main Wilmington 01897 Call..... | 800 648-7447 | |
| Ingalls Cross 163 Main Wilmington 01897 Toll Free-041 71 & Then..... | 978 988-7447 | |
| Cartier Richard 1679 Cornwell Av Brighton 02215..... | 800 638-1673 | |
| Richard A 17 Mt Vernon Bsc 02108..... | 617 987-0836 | |
| Richard A MD 170 Cornwell Av Rte 02116..... | 617 566-7293 | |
| Cartier Richard K 15 Mercer S Bsc 02127..... | 617 267-0710 | |
| Robert L 175 Richards Av Cam 02140..... | 617 268-0448 | |
| Roger 150 St Raleigh Box 02115..... | 617 864-1535 | |
| Roy 44 Concord Av Cam 02136..... | 617 424-6148 | |
| Royce 18 Seminary Cha 02129..... | 617 491-6115 | |
| | 617 241-0418 | |

There must be a great
search algorithm behind
the scenes.

Linear Search

LinSearch: The Spec

```
def LinSearch(x,a) :  
    """ Returns an int k with the  
    property that a[k]==x is True.  
    If no such k exists, then  
    k==-1.  
  
    PreC: a is a nonempty list of  
    ints and x is an int.  
    """
```

Could also apply the same ideas for searching a list of strings.

Linear Search

0 1 2 3 4 5 6 7 8 9 10 11

a-→

| | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|
| 86 | 73 | 43 | 35 | 23 | 45 | 42 | 62 | 15 | 25 | 51 | 35 |
|----|----|----|----|----|----|----|----|----|----|----|----|

k-→

| |
|---|
| 0 |
|---|

x-→

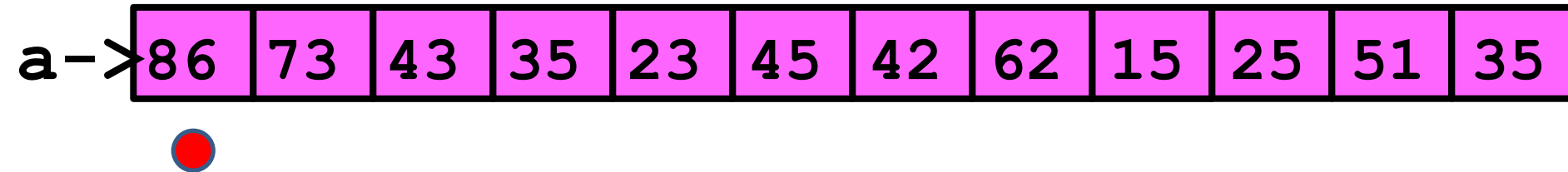
| |
|----|
| 23 |
|----|

```
def LinSearch(x,a):  
    for k in range(len(a)):  
        if x == a[k]:  
            return k  
    return -1
```

Walk down the list looking for a match

Linear Search

0 1 2 3 4 5 6 7 8 9 10 11



k-→

| |
|---|
| 0 |
|---|

x-→

| |
|----|
| 23 |
|----|

```
def LinSearch(x,a):  
    for k in range(len(a)):  
        if x == a[k]:  
            return k  
    return -1
```

Walk down the list looking for a match

Linear Search

0 1 2 3 4 5 6 7 8 9 10 11

a-→

| | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|
| 86 | 73 | 43 | 35 | 23 | 45 | 42 | 62 | 15 | 25 | 51 | 35 |
|----|----|----|----|----|----|----|----|----|----|----|----|



k-→

| |
|---|
| 1 |
|---|

x-→

| |
|----|
| 23 |
|----|

```
def LinSearch(x,a):  
    for k in range(len(a)):  
        if x == a[k]:  
            return k  
    return -1
```

Walk down the list looking for a match

Linear Search

0 1 2 3 4 5 6 7 8 9 10 11

a-→

| | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|
| 86 | 73 | 43 | 35 | 23 | 45 | 42 | 62 | 15 | 25 | 51 | 35 |
|----|----|----|----|----|----|----|----|----|----|----|----|



k-→

| |
|---|
| 2 |
|---|

x-→

| |
|----|
| 23 |
|----|

```
def LinSearch(x,a):  
    for k in range(len(a)):  
        if x == a[k]:  
            return k  
    return -1
```

Walk down the list looking for a match

Linear Search

0 1 2 3 4 5 6 7 8 9 10 11

a-→

| | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|
| 86 | 73 | 43 | 35 | 23 | 45 | 42 | 62 | 15 | 25 | 51 | 35 |
|----|----|----|----|----|----|----|----|----|----|----|----|



k-→

| |
|---|
| 3 |
|---|

x-→

| |
|----|
| 23 |
|----|

```
def LinSearch(x,a):  
    for k in range(len(a)):  
        if x == a[k]:  
            return k  
    return -1
```

Walk down the list looking for a match

Linear Search

0 1 2 3 4 5 6 7 8 9 10 11

a-→

| | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|
| 86 | 73 | 43 | 35 | 23 | 45 | 42 | 62 | 15 | 25 | 51 | 35 |
|----|----|----|----|----|----|----|----|----|----|----|----|



k-→

| |
|---|
| 4 |
|---|

x-→

| |
|----|
| 23 |
|----|

```
def LinSearch(x,a):  
    for k in range(len(a)):  
        if x == a[k]: Yup  
            return k  
    return -1
```

Walk down the list looking for a match

Linear Search

0 1 2 3 4 5 6 7 8 9 10 11

a-→

| | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|
| 86 | 73 | 43 | 35 | 23 | 45 | 42 | 62 | 15 | 25 | 51 | 35 |
|----|----|----|----|----|----|----|----|----|----|----|----|



k-→

| |
|---|
| 4 |
|---|

x-→

| |
|----|
| 23 |
|----|

```
def LinSearch(x,a):  
    for k in range(len(a)):  
        if x == a[k]:  
            return k  
    return -1
```

All done

Walk down the list looking for a match

Linear Search: No Match Case

0 1 2 3 4 5 6 7 8 9 10 11

a-→

| | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|
| 86 | 73 | 43 | 35 | 23 | 45 | 42 | 62 | 15 | 25 | 51 | 35 |
|----|----|----|----|----|----|----|----|----|----|----|----|



k-→

| |
|----|
| 11 |
|----|

x-→

| |
|---|
| 7 |
|---|

```
def LinSearch(x,a):  
    for k in range(len(a)):  
        if x == a[k]: Nope  
        return k  
    return -1
```

Walk down the list looking for a match

Linear Search: No Match Case

0 1 2 3 4 5 6 7 8 9 10 11

a →

| | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|
| 86 | 73 | 43 | 35 | 23 | 45 | 42 | 62 | 15 | 25 | 51 | 35 |
|----|----|----|----|----|----|----|----|----|----|----|----|



x →

| |
|---|
| 7 |
|---|

```
def LinSearch(x,a):  
    for k in range(len(a)):  
        if x == a[k]:  
            return k  
    return -1
```

Yup

Return -1 if no match

Linear Search: While Implementation

```
def LinSearchW(x,a):  
    k=0  
    while k<len(a) and a[k]!=x:  
        k+=1  
    if k==len(a):  
        return -1  
    else:  
        return k
```

Binary Search

Now we assume that the list
to be searched is sorted
from little to big.

```
a = [10,20,40,60,90]
```

```
a = ['brown', 'dog', 'fox', 'lazy', 'quick', 'the']
```


Back to Using Phone Books

The Ithaca
phone book has
10,000+ entries.

The Manhattan phone book has 1,000,000+ entries. But it does not take 100 x longer to look something up. Why?

wide at **SuperPages.com**

195

Car **C**

17 566-1282
Cortage New England Inc.
28 Allen Ln Easton 02145.....978 356-9960

100 447-4101
Cortagema Lydia
38 Jewett Rd 02131.....978 323-7639

100 257-9981
Cortagema Arith
9 Racefort Rd 02119.....617 442-9780
B Hyde 02136.....617 361-5253

17 566-1282
17 364-5180
Cortage 50 Decatur Ch 02129.....617 241-0152
Luna 144 Waverhill Cam 02139.....617 481-5621

361-0380
Melvin 101 Green Cam 02139.....617 576-1061

17 566-4548
Corte Nicholas
18 Appleton Boston 02114.....617 695-6996

17 628-8248
Cortegena O
10000 W 1st St 02118.....617 338-9219

17 445-5116
Corten Thos Jr & Claire
1 Paradise Rd Mt 02138.....617 698-6163

17 622-2982
Thomas & Kathleen
50 Thompson Ln Mt 02136.....617 696-6919

17 427-5712
Carter A R
A Roanoke
31 Bethune Wy Roanoke 02119.....617 442-5230

17 569-2698
A 260 Pitman Av Cambridge 02139.....617 492-4174

17 667-5190
A 255 Wachusett Av Mt 02115.....617 266-7153

17 569-1417
Adams 361 Center St Mt 02136.....617 698-7074

17 338-9110
Alice 105 Kilmack Rd 02215.....617 425-0193

17 825-9195
Alice 45 Kilmack Rd 02139.....617 945-2711

17 296-1959
Andrew F 62 Wilm Av Son 02141.....617 625-7623

17 296-1959
1101 Essex Rd 02046.....617 739-1022

17 670-2078
Carter Athens
277 Newton Boston 02116.....617 536-6329

17 623-9071
B E 48 Garfield Av Mt 02126.....617 296-6911

17 296-4725
Carter Barbara L M
Toby-New England Medical Center 02111
Call.....617 636-0051

17 542-1521
Carter Becky R
82 Mt 02134.....617 532-4368

17 364-5232
Bernard J
112 Graduate E Boston 02215.....617 567-3430

17 541-5649
Bibbith 25 Midway Dr 02124.....617 298-8713

17 739-2662
Blake 35 Mt Vernon St 02108.....617 367-9931

17 879-0030
Carter Broadcasting Co
70 Park Pl St 02136.....617 423-0210

17 431-3948
Carter C Burgess Consultants Inc
23 East St Son 02141.....617 225-0200

17 569-4119
Carter C 2000 Commonwealth Av Mt 02135.....617 782-2118

17 569-4119
C 228 Faywood Av East Boston 02215.....617 569-1545

17 569-4119
C 630 Mt Vernon St 02108.....617 296-6932

17 569-4119
C 610 W 1st St 02136.....617 524-9558

17 327-1105
Carter Faye & Ricky
257 Columbus Av Son 02136.....617 437-7331

17 327-1105
Francis S 124 Temple W Son 02132.....617 323-6781

17 327-1105
Franklin & Anne
221 Mt Auburn Cam 02138.....617 354-0798

17 327-1105
Fred 42 Howland Jm 02136.....617 524-0306

17 327-1105
Fred 46 Howland Jm 02136.....617 524-0306

17 327-1105
G & R Vernon Dr 02134.....617 438-8906

17 327-1105
G T 27 Franklin Av Son 02145.....617 623-7121

17 327-1105
Gayles 25 Frontenac Dr 02134.....617 825-0322

17 327-1105
Geo S 115 Mass Hs Rd Jm 02136.....617 522-3215

17 327-1105
George 125 Nantux Rd 02134.....617 367-9548

17 327-1105
Carter Halliday Association
107 S Street Son 02111.....617 456-1689

17 327-1105
Carter Harry F
36 Runny Rd Rd W Son 02132.....617 325-5465

17 327-1105
Carter Hilda Co
146 Summer Son 02119.....617 542-7987

17 327-1105
Carter Hilary
61 Harvey Cam 02140.....617 876-2750

17 327-1105
Horace
24 Walnut Av Roanoke 02119.....617 442-5307

17 327-1105
Howard Jr 26 Ndre Dne Son 02119.....617 445-5552

17 327-1105
J Cam
15 Chatham Bn 02046.....617 232-7990

17 327-1105
J 518 Harvard Bn 02046.....617 730-9483

17 327-1105
J 775 Wy Hwy West Rd 02132.....617 323-5574

17 327-1105
J Jackson 40
3 Brookline Pl Bn 02046.....617 735-8787

17 327-1105
Carter J M
1410 Columbia Rd S Son 02127.....617 644-1040

17 327-1105
Carter J M Ornamental Ironworks
Pembroke Telld 02173.....617 436-5353

17 327-1105
Carter J Veal Co
48 Neenontest Sq Son 02118.....617 442-1775

17 327-1105
Carter James
1571 Cambridge St Cam 02138.....617 492-1214

17 327-1105
James
382 Fisher Av Roanoke 02138.....617 739-2193

17 327-1105
J 37 Gold St Rd Cambridge 02140.....617 876-8841

17 327-1105
Jas L 14 Roanoke Rd Mt 02126.....617 361-0773

17 327-1105
John 124 Adams Rd Newton 02045.....617 564-0642

17 327-1105
Jeffrey 43 Wilmers Av Son 02116.....617 426-5994

17 327-1105
John 11 Mansfield Bn 02134.....617 987-2163

17 327-1105
John 327 Summer Son 02119.....617 423-4334

17 327-1105
John 40 Westford Rd Dr 02125.....617 282-1225

17 327-1105
Joseph 129 A Summit Av Son 02119.....617 734-6139

17 327-1105
K 38 Broadway Av East Boston 02215.....617 265-8456

17 327-1105
K 12 Esmond Dorchester 02124.....617 282-8891

17 327-1105
Carter Nella E
233 Massachusetts Av Son 02115.....617 267-6483

17 327-1105
Nicholas S F
115 Randolph Av Mt 02136.....617 698-5307

17 327-1105
Nick 21 Farfield Son 02116.....617 267-5222

17 327-1105
Nick & Debbie
106 Herrick Rd Newton 02049.....617 527-0480

17 327-1105
Norman G
115 Randolph Av Mt 02136.....617 698-5307

17 327-1105
P 34 Christened Pk Son 02127.....617 822-1203

Key Idea: Repeated Halving

To find Derek Jeter's number...

B = phone book

while (B is longer than 1 page):

1. P = middle page of B

2. Let Q be the first name on P

3. **if** 'Jeter' comes before Q:

 Rip away the 2nd half of B

else:

 Rip away the 1st half of B.

Scan remaining page P line-by-line for 'Jeter'

What Happens to 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

The idea of 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 approximately $\log_2 n$ comparisons.

$$\log_2 8 = 3 \quad \log_2 64 = 6 \quad \log_2 2^{**k} = k$$

What is $\log_2(n)$?

| n | $\text{ceil}(\log_2(n))$ |
|---------|--------------------------|
| <hr/> | |
| 10 | 4 |
| 100 | 7 |
| 1000 | 10 |
| 10000 | 14 |
| 100000 | 17 |
| 1000000 | 20 |

BinSearch: The Spec

```
def BinSearch(x,a):  
    """ Returns an int k with the  
    property that a[k]==x is True.  
    If no such k exists, then  
    k==-1.  
  
    PreC: a is a nonempty list of  
    ints that is sorted from smallest  
    to largest. x is an int.  
    """
```

Example:
Does this List have an Element
With Value Equal to 70?

0 1 2 3 4 5 6 7 8 9 10 11

| | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|
| 12 | 15 | 33 | 35 | 42 | 45 | 51 | 62 | 73 | 75 | 86 | 98 |
|----|----|----|----|----|----|----|----|----|----|----|----|

Let's Look For x in a

0 1 2 3 4 5 6 7 8 9 10 11

a →

| | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|
| 12 | 15 | 33 | 35 | 42 | 45 | 51 | 62 | 73 | 75 | 86 | 98 |
|----|----|----|----|----|----|----|----|----|----|----|----|



L:

0

a[Mid] <= x ????

Mid:

5

R:

11

x:

70

Mid = (L+R) / 2

The Midpoint Computations

| L | R | $(L+R) / 2$ |
|-------|-----|-------------|
| <hr/> | | |
| 0 | 11 | 5 |
| 2 | 6 | 4 |
| 1 | 100 | 50 |

Let's Look For x in a

0 1 2 3 4 5 6 7 8 9 10 11

a →

| | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|
| 12 | 15 | 33 | 35 | 42 | 45 | 51 | 62 | 73 | 75 | 86 | 98 |
|----|----|----|----|----|----|----|----|----|----|----|----|



L:

0

a[Mid] <= x ????

Mid:

5

R:

11

x:

70

Let's Look For x in a

0 1 2 3 4 5 6 7 8 9 10 11

a →

| | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|
| 12 | 15 | 33 | 35 | 42 | 45 | 51 | 62 | 73 | 75 | 86 | 98 |
|----|----|----|----|----|----|----|----|----|----|----|----|



L:

0

Mid:

5

R:

11

x:

70

$a[\text{Mid}] \leq x$

Yes!

So throw away
The "left half"

Let's Look For x in a

0 1 2 3 4 5 6 7 8 9 10 11

a-→

| | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|
| 12 | 15 | 33 | 35 | 42 | 45 | 51 | 62 | 73 | 75 | 86 | 98 |
|----|----|----|----|----|----|----|----|----|----|----|----|



L:

0

Mid:

5

R:

11

$a[\text{Mid}] \leq x$

x:

70

Yes!

So throw away
The "left half"

Let's Look For x in a

0 1 2 3 4 5 6 7 8 9 10 11

a →

| | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|
| 12 | 15 | 33 | 35 | 42 | 45 | 51 | 62 | 73 | 75 | 86 | 98 |
|----|----|----|----|----|----|----|----|----|----|----|----|



L:

0

Mid:

5

R:

11

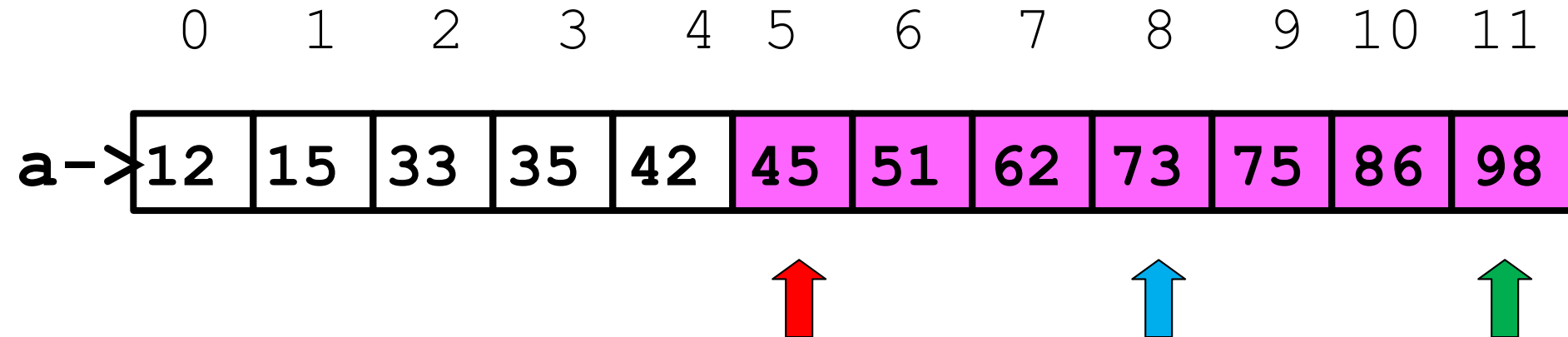
x:

70

$a[\text{Mid}] \leq x$

Revise L and Mid

Let's Look For x in a



L:

5

a[Mid] <= x ???

Mid:

8

R:

11

x:

70

Let's Look For x in a

0 1 2 3 4 5 6 7 8 9 10 11

a-→

| | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|
| 12 | 15 | 33 | 35 | 42 | 45 | 51 | 62 | 73 | 75 | 86 | 98 |
|----|----|----|----|----|----|----|----|----|----|----|----|



L:

5

Mid:

8

R:

11

$a[\text{Mid}] \leq x$

No

x:

70

So throw away the
"right half"

Let's Look For $x = 70$

0 1 2 3 4 5 6 7 8 9 10 11

a-→

| | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|
| 12 | 15 | 33 | 35 | 42 | 45 | 51 | 62 | 73 | 75 | 86 | 98 |
|----|----|----|----|----|----|----|----|----|----|----|----|



L:

5

Mid:

8

R:

11

x:

70

$a[\text{Mid}] \leq x$

Revise R and Mid

Let's Look For $x = 70$

0 1 2 3 4 5 6 7 8 9 10 11

a-→

| | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|
| 12 | 15 | 33 | 35 | 42 | 45 | 51 | 62 | 73 | 75 | 86 | 98 |
|----|----|----|----|----|----|----|----|----|----|----|----|



L:

5

Mid:

6

R:

8

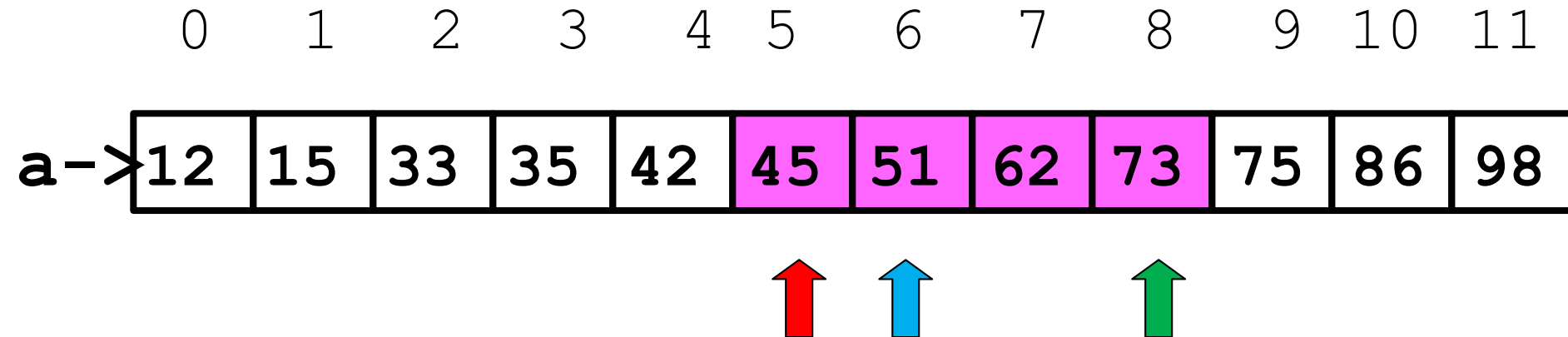
x:

70

$a[\text{Mid}] \leq x$

Revise R and Mid

Let's Look For x in a



L:

5

Mid:

6

R:

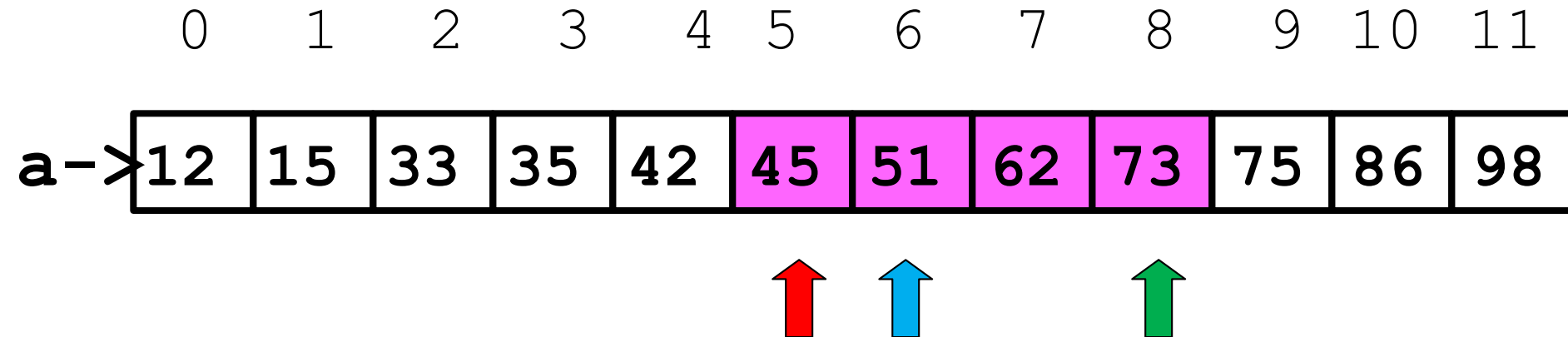
8

x:

70

a[Mid] <= x ????

Let's Look For x in a



L:

5

$a[\text{Mid}] \leq x$

Mid:

6

Yes

R:

8

x:

70

Throw away the
Left half

Let's Look For x in a

0 1 2 3 4 5 6 7 8 9 10 11

a →

| | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|
| 12 | 15 | 33 | 35 | 42 | 45 | 51 | 62 | 73 | 75 | 86 | 98 |
|----|----|----|----|----|----|----|----|----|----|----|----|



L:

6

$a[\text{Mid}] \leq x$

Mid:

7

Yes

R:

8

x:

70

Let's Look For x in a

0 1 2 3 4 5 6 7 8 9 10 11

a →

| | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|
| 12 | 15 | 33 | 35 | 42 | 45 | 51 | 62 | 73 | 75 | 86 | 98 |
|----|----|----|----|----|----|----|----|----|----|----|----|



L:

6

$a[\text{Mid}] \leq x$

Mid:

7

Throw away the
left half

R:

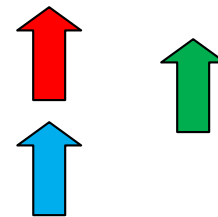
8

x:

70

Let's Look For x in a

| | | | | | | | | | | | | |
|-----------------|----|----|----|----|----|----|----|----|----|----|----|----|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| $a \rightarrow$ | 12 | 15 | 33 | 35 | 42 | 45 | 51 | 62 | 73 | 75 | 86 | 98 |



$L:$

6

$Mid:$

7

$R:$

8

$x:$

70

Done! At this point we just compare x with $a[L]$ and $a[L+1]$.

What We Just Did

```
L = 0
R = len(a) - 1
while R - L > 1:
    # a[L] <= x <= a[R]
    Mid = (L + R) / 2
    if x <= a[mid]:
        R = Mid
    else:
        L = Mid
```

A Loop
Invariant

Note that $a[L] \leq x \leq a[R]$ remains True throughout the loop

What We Just Did

```
L = 0
R = len(a) - 1
while R-L > 1:
    # a[L] <= x <= a[R]
    Mid = (L+R) / 2
    if x <= a[mid]:
        R = Mid
    else:
        L = Mid
```

What is the situation when the loop terminates?

What We Just Did

```
L = 0
R = len(a) - 1
while R - L > 1:
    # a[L] <= x <= a[R]
    Mid = (L + R) / 2
    if x <= a[mid]:
        R = Mid
    else:
        L = Mid
```

$R - L \leq 1$ implies $R = L + 1$

After the Loop Ends



This is True: $a[L] \leq x \leq a[L+1]$

After the Loop Ends



```
if x==a[L]:  
    return L  
elif x==a[L+1]:  
    return L+1  
else:  
    return -1
```

Measuring Execution Time

We now have two ways to search a list:

`LinSearch(x, a)`

`BinSearch(x, a)`

Intuition: BinSearch much faster.

Can we quantify this with a “stop watch”?

The `timeit` Module

This module can be used to time how long it takes to execute a chunk of code.

Typical chunk = some function of interest.

This is called benchmarking.

Benchmarking

Let's benchmark `LinSearch(x, a)` and `BinSearch(x, a)`.

Compare how long it takes when `len(a)` equals 1000, 10000, 100000, and 1000000.

Our intuition tells us that as `len(a)` increases, `BinSearch` will be dramatically faster.

BinSearch vs LinSearch

| n | tBin | tLin | tLinW |
|---------|--------|---------|---------|
| ----- | | | |
| 1000 | 0.0007 | 0.0064 | 0.0119 |
| 10000 | 0.0009 | 0.0668 | 0.1203 |
| 100000 | 0.0011 | 0.8296 | 1.2082 |
| 1000000 | 0.0015 | 17.7388 | 13.9341 |

tBin = time for BinSearch

tLin = time for LinSearch (for loop version)

tLinW = time for LinSearch (while-loop version)

BinSearch vs LinSearch

| n | tLin/tBin |
|---------|-----------|
| ----- | |
| 1000 | 9 |
| 10000 | 74 |
| 100000 | 754 |
| 1000000 | 7095 |

Reporting ratios is more illuminating since we do not really care about the time units in this informal comparison

Using the `timeit` Module

We show how this module was used to get the results on the previous slides.

Our `LinSearch` vs `BinSearch` example is very typical: is one function faster than another?

A Benchmarking Framework

```
from timeit import *
```

```
S = """
```

Set-up code

```
"""
```

```
B = """
```

Code to Benchmark

```
"""
```

```
p = 10; m = 100
```

```
t = min(Timer(B, setup=S).repeat(p, m))
```

Yes, these are doc strings.

The Set-Up and Bench Codes

```
from random import randint as randi
from ShowSearch import BinSearch
n = 10000
s = [randi(0,10*n) for i in range(n)]
s.sort()
x = s[n/2]
```

```
k=BinSearch(x,s)
```

The set-up code is run once.

It is not timed.

It just sets up the code to be timed.

A Benchmarking Framework

```
from timeit import *
```

```
S = """
```

```
    Set-up code
```

```
    """
```

```
B = """
```

```
    Code to Benchmark
```

```
    """
```

```
p = 10; m = 100
```

```
t = min(Timer(B, setup=S).repeat(p, m))
```

An “experiment” consists of running the blue code m times.

The stopwatch will time how long it takes to do one experiment

Larger values necessary if the blue code executes very quickly

A Benchmarking Framework

```
from timeit import *
```

```
S = """
```

Set-up code

```
"""
```

```
B = """
```

Code to Benchmark

```
"""
```

```
p = 10; m = 100
```

```
t = min(Timer(B, setup=S) . repeat(p, m) )
```

Timer returns a length-p list. Each element is the stopwatch time for 1 experiment

This helps control for other stuff that may be running on your computer.

A Benchmarking Framework

```
from timeit import *
```

```
S = """
```

Set-up code

```
"""
```

```
B = """
```

Code to Benchmark

```
"""
```

```
p = 10; m = 100
```

```
t = min(Timer(B, setup=S).repeat(p, m))
```

In general, it is best to take the minimum as the most reliable. The benchmark time is assigned to t

This helps control for other stuff that may be running on your computer.

Why Benchmarking is Important

Confirms/refutes what our intuition might say about efficiency.

Makes us sensitive to the various issues that affect efficiency.

Steers us away from simplistic comparisons of different methods that can be used on the same problem.

Demonstration!

- Download and run the python script `ShowBenchmarking.py`
- TODO:
 - Run the above script for the following values of “n” (line 55)
 - 1000, 10000, 100000, 500000, 1000000, 10000000
 - Of the three search methods—binary search, linear search with for loop, linear search with while—which seems the fastest?
 - Which is faster of the following two—linear search with for loop, linear search with while loop?