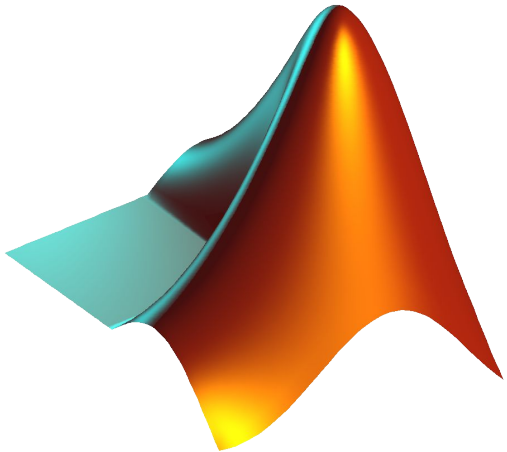
CS 1112 Introduction to Computing Using MATLAB



Instructor: Dominic Diaz

Website: https://www.cs.cornell.edu/courses/cs1112/2022fa/

Today: sorting/searching

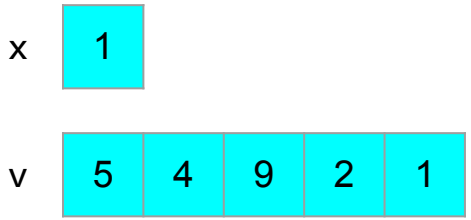
Agenda and announcements

- Last time
  - Recursion
- Today
  - An algorithm for sorting (merge sort)
  - An algorithm for searching (binary search)
- Announcements
  - Project 6 due Dec 5th
  - Code ‘til you drop session on Dec 14th
  - Last discussion exercise this week
    - Optional but can replace your lowest discussion score.
  - Final exam on Thursday, December 15th from 2 - 4:30 PM in Olin 155
    - Check your “final exam time and location” CMS assignment if you have an SDS letter. If you have 3+ finals in a 24 hour period, submit a regrade request and we can reschedule this exam.

Searching for an item in an unorganized collection?

- May need to look through the whole collection to find the target item
- Could use linear search

For example, find value x in vector v



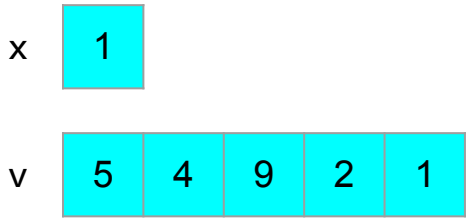

```
% Linear Search
% f is index of first occurrence
% of value x in vector v.
% f is -1 if x not found.
k= 1;
while k<=length(v) && v(k)~=x
     k= k + 1;
end

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

In the worst case, the `while` loop will be evaluated `n` times, assuming `n=length(v)`.

Searching for an item in an unorganized collection?

- May need to look through the whole collection to find the target item
- Could use linear search

For example, find value x in vector 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 f
else
     f= k;
end
```

In the worst case, the while loop will be evaluated n times, assuming n=length(v).

Searching in a sorted list should require less work!

An ordered (sorted) list

Manhattan phone book has 1,000,000+ entries

If the list were unsorted, you would have to look through every name to find someone.

wide at SuperPages.com 195 Car C

617 566-1282
781 447-4101
800 257-9981
617 566-1282
617 364-5188
361-0380
617 566-4548
617 628-8248
617 445-5116
617 822-2982
617 427-5712
617 569-2698
617 667-5190
617 569-1417
617 338-9110
617 825-9195
617 296-1593
617 670-2078
617 623-9001
617 296-4725
617 542-1521
617 364-5232
617 541-5649
617 739-2662
617 879-0030
617 541-3948
617 436-1513
617 569-4119
800 569-8782

Cartage New England Inc 26 Allen Ln Ipswich 01938 978 356-9960
Cartagena Lydia 18 Jewett Ros 02131 617 323-7639
Cartagena Avith 9 Bancroft Rox 02119 617 442-9780
B Hyd 02136 617 361-5253
Jessica 50 Decatur Cha 02129 617 241-0152
Lucilla 174 Harvard Cam 02139 617 491-5621
M 95 Rowe Ros 02131 617 323-9713
Melvin 501 Green Cam 02139 617 576-1061
Carte Nicholas 18 Appleton Boston 02116 617 695-6996
Cartegena O 4 Milford Bos 02138 617 338-8219
Carten Thos J Sr & Claire 1 Paradise Rd Mil 02186 617 698-6163
Thomas & Kathleen 50 Thompson Ln Mil 02186 617 696-6919
Carter A Ros 02131 617 327-2257
A Roxbury 617 442-5230
A 31 Bethune Wy Roxbury 02119 617 442-1219
A 260 Putnam Av Cambridge 02139 617 492-4174
A M 255 Maschsts Av Bos 02115 617 266-7153
Adams 361 Centre St Mil 02186 617 698-9074
Alice 108 Kilmarnock Bos 02215 617 425-0193
Alice 45 Market Cambridge 02139 617 945-2711
Andrew F 62 Vinal Av Som 02143 617 625-7623
Carter Anne MD 1101 Beacon Bro 02446 617 739-1022
Carter Athens 272 Newbury Boston 02116 617 536-6329
B E 68 Gladeside Av Mat 02126 617 296-6911
Carter Barbara L MD Tufts-New England Medical Center Bos 02111 Call 617 636-0051
Carter Becky Bos 02114 617 523-4368
Bernard J 112 Gladstone E Bos 02128 617 567-3430
Bithiah 25 Medway Dor 02124 617 298-8713
Blake 26 Mt Vernon Bos 02108 617 367-9931
Carter Broadcasting Co 20 Park Plz Bos 02116 617 423-0210
Carter & Burgess Consultants Inc 23 East St Cam 02141 617 225-0200
Carter C 2000 Commwth Av Bri 02135 617 782-2118
C 228 Faywood Av East Boston 02128 617 569-1545
C 359 Harvard Cam 02138 617 491-4822
C 610 Walk Hill Mat 02126 617 296-6392
C & M 43 Burroughs Jam 02130 617 524-9558

Carter F 24 Hillock Ros 02131 617 327-1105
Faye & Ricky 357 Columbus Av Bos 02116 617 437-7331
Francis S 134 Temple W Rox 02132 617 323-6781
Franklin & Anne 221 Mt Auburn Cam 02138 617 354-0798
Fred 42 Haverford Jam 02130 617 524-3078
Fred 96 Hinckley Rd Mil 02186 617 698-1343
G & R 8 Verdun Dor 02124 617 436-8906
G T 27 Franklin Av Som 02145 617 623-7121
Gayle 25 Fronteniac Dor 02124 617 825-0322
Geo S 115 Moss Hill Rd Jam 02130 617 522-3215
George 125 Nashua Bos 02114 617 367-9548
Carter Halliday Associate 187 S Street Bos 02111 617 456-1689
Carter Harry F 26 Runng Brk Rd W Rox 02132 617 325-5465
Carter Hide Co Inc 146 Summer Bos 02110 617 542-7987
Carter Hilary 61 Harvey Cam 02140 617 876-2750
Horace 241 Walnut Av Roxbury 02119 617 442-5307
Howard Jr 26 Notre Dme Rox 02119 617 445-5552
J Cam 617 354-2688
J 15 Chatham Bro 02446 617 232-7990
J 518 Harvard Bro 02446 617 730-9483
J 775 Vfw Pkwy West Roxbury 02132 617 323-5574
Carter J Jacques MD 1 Brookline Pl Bro 02446 617 735-8787
Carter J M 1410 Columbia Rd S Bos 02127 617 464-1040
Carter J M Ornamental Ironworks Call Pembroke TelNo 617 436-5353
Carter J Veal Co 48 Newmarket Sq Rox 02118 617 442-1775
Carter James 1573 Cambridge St Cam 02138 617 492-1214
James 182 Fisher Av Roxbury 02120 617 739-2193
James 37 Gold Star Rd Cambridge 02140 617 876-8841
Jas L 14 Roseberry Rd Mat 02126 617 361-0773
Jane 114 Adena Rd Newton 02465 617 964-0435
Jeffrey 41 Warren Av Bos 02116 617 426-5994
John 11 Mansfield Bri 02134 617 987-2163
John 327 Summer Bos 02210 617 423-4334
John 40 Westwind Rd Dor 02125 617 282-1235
June O 329 A Summit Av Bri 02135 617 734-6109
K 38 Browning Av Dorchester 02124 617 265-8456
K 17 Esmond Dorchester 02121 617 282-1593

Carter Nella E 333 Maschsts Av Bos 02115 617 267-6483
Nicholas S F 115 Randolph Av Mil 02186 617 698-5307
Nick 21 Fairfield Bos 02116 617 267-5222
Nick & Debbi 196 Herrick Rd Newton 02459 617 527-0480
Nicole 617 698-0713
Norman G 38 Chickatawbut Dor 02122 617 822-1203
P 94 Crestwood Pk Rox 02121 617 427-4754
P E 501 E Sixth S Bos 02127 617 268-4213
P L 44 Hutchings Rox 02121 617 427-9170
P R 91 Bynner Jam 02130 617 983-8692
Paul & Constance 114 Anawan Av W Rox 02132 617 325-2036
Paul E 501 E Sixth St S Bos 02127 617 268-4546
Paul M 27 Union Bri 02135 617 787-2115
Carter Pile Driving Inc 17 Beaver Ct Framingham 01702 Wellesley TelNo 781 235-8488
Carter Prudence 46 Franklin Watertown 02172 617 393-3782
Prudence 46 Franklin Watertown 02172 617 926-7063
Reginald 106 Brunswick Dorchester 02121 617 541-2843
Renee & Andrew 10 Walnut Bos 02108 617 720-3765
Carter Rice Dowd Bulkley Dunton Publishing 163 Main Wilmington 01887 Toll Free-Dial '1' & Then 800 638-1671 Cust Svc-Industrial Prod 613 Main Wilmington Toll Free-Dial '1' & Then 800 619-7447 Cust Svc-Printing 613 Main Wilmington Toll Free-Dial '1' & Then 800 648-7447 Headquarters 613 Main Wilmington 01887 Call 978 988-7447 Ingalls Cronin 163 Main Wilmington 01887 Toll Free-Dial '1' & Then 800 638-1673
Carter Richard 1079 Commwth Av Brighton 02215 617 987-0836
Richard A 97 Mt Vernon Bos 02108 617 566-7293
Carter Richard A MD 170 Commwth Av Bos 02116 617 267-0710
Carter Richard K 15 Mercer S Bos 02127 617 268-0448
Robert L 175 Richdale Av Cam 02140 617 864-1535
Roger 150 St Botolph Bos 02115 617 424-6148
Roy 44 Concord Av Cam 02138 617 491-6115
Royce 18 Seminary Cha 02129 617 241-0418

C

There are many algorithms for sorting

- Merge sort (discussed this lecture)
- Selection sort (exercise this week)
- Insertion sort (discussed next lecture)
- Bubble sort (read insight section 8.2)
- 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 to analyze the efficiency of programs and algorithms

Merge sort motivation

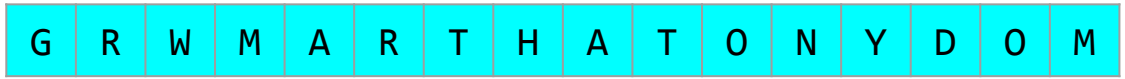
If I have two helpers, I would:

- Give each helper half the array to sort
- Then I get back the sorted subarrays and merge them

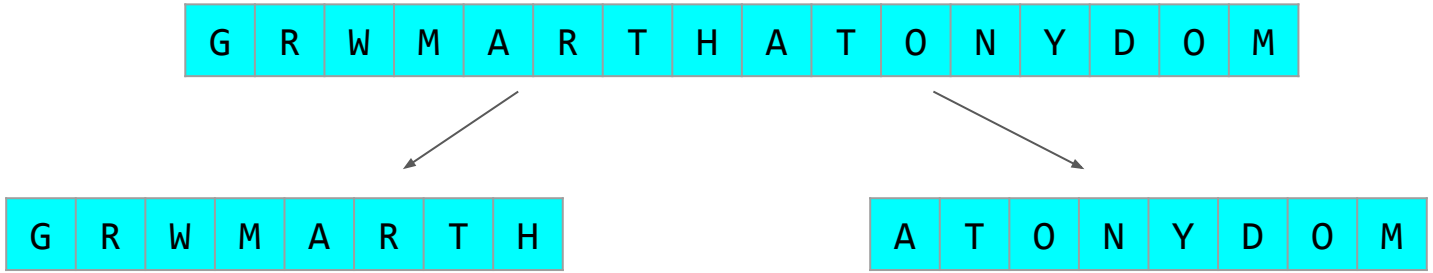
What if those two helpers each had two sub-helpers?

And each of those sub-helpers each has two sub-sub-helpers? And…

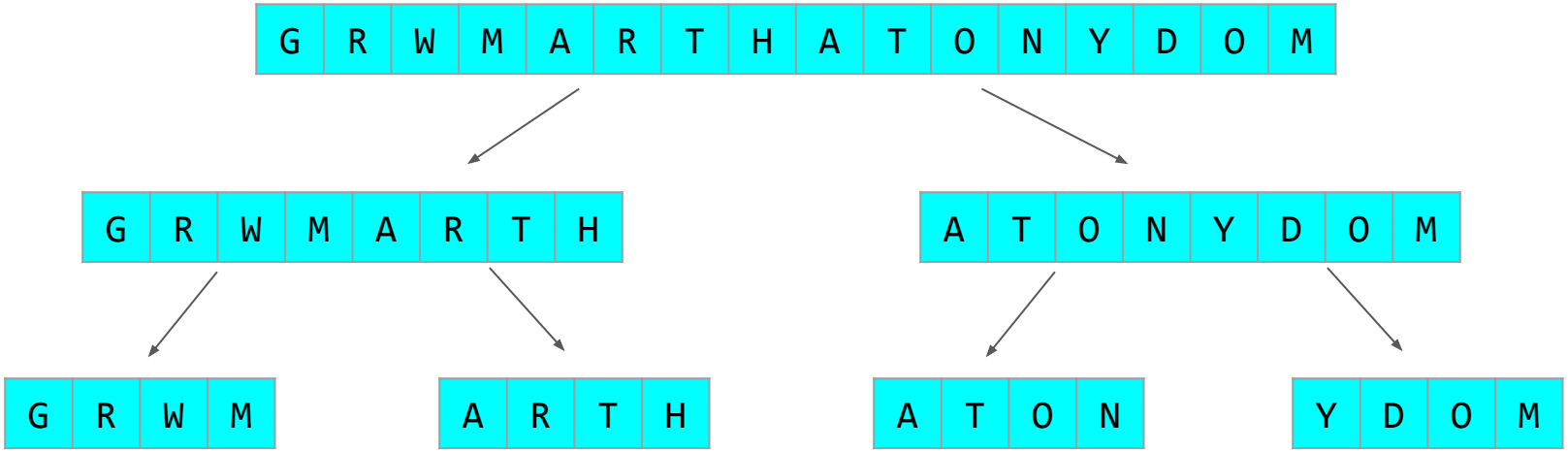
Let’s see how we can apply this idea to sort an unsorted list



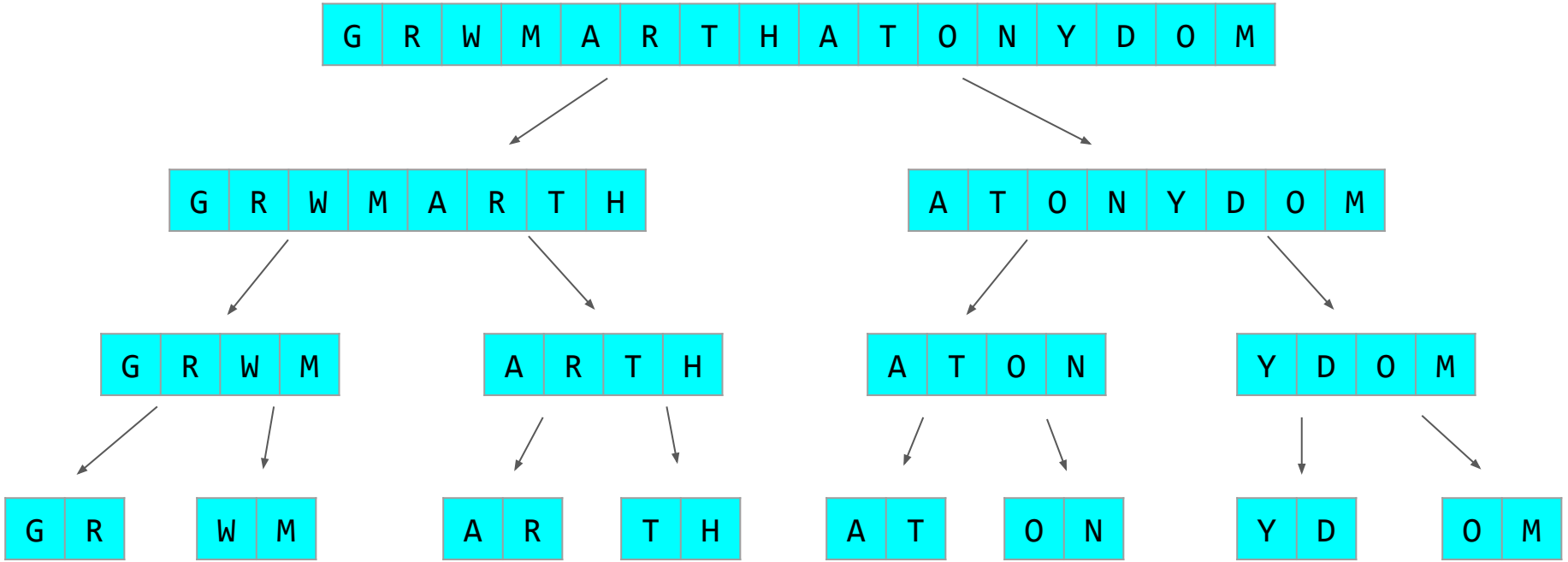
Let’s see how we can apply this idea to sort an unsorted list



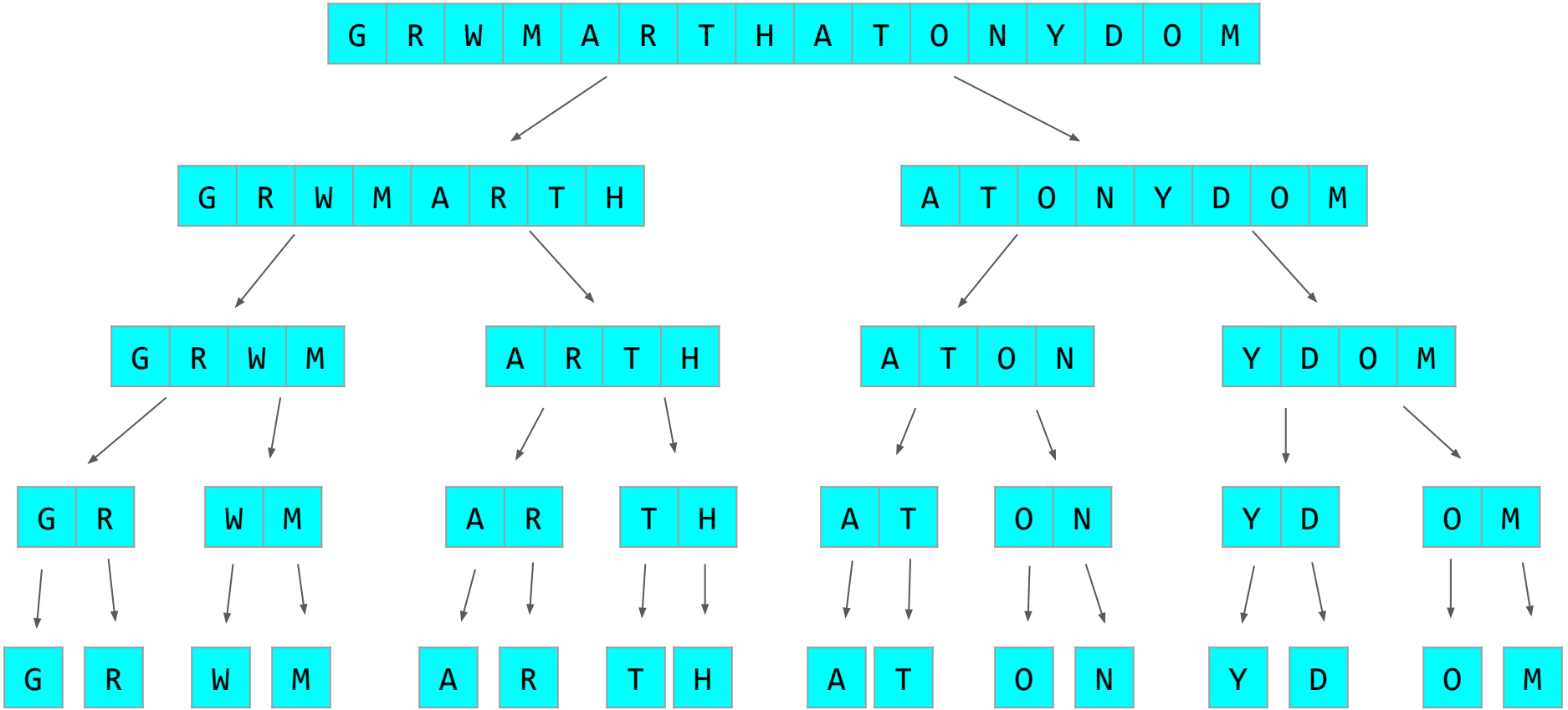
Let’s see how we can apply this idea to sort an unsorted list



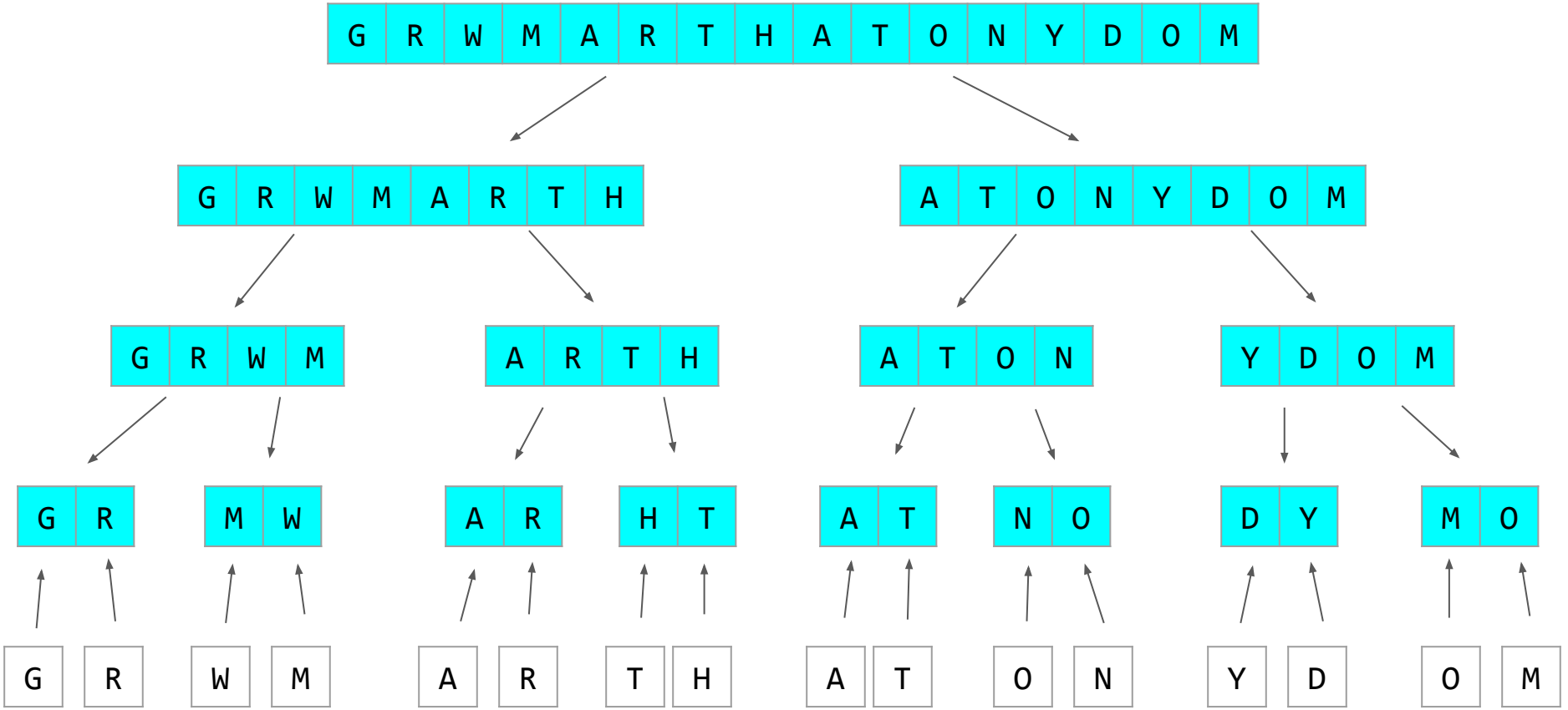
Let’s see how we can apply this idea to sort an unsorted list



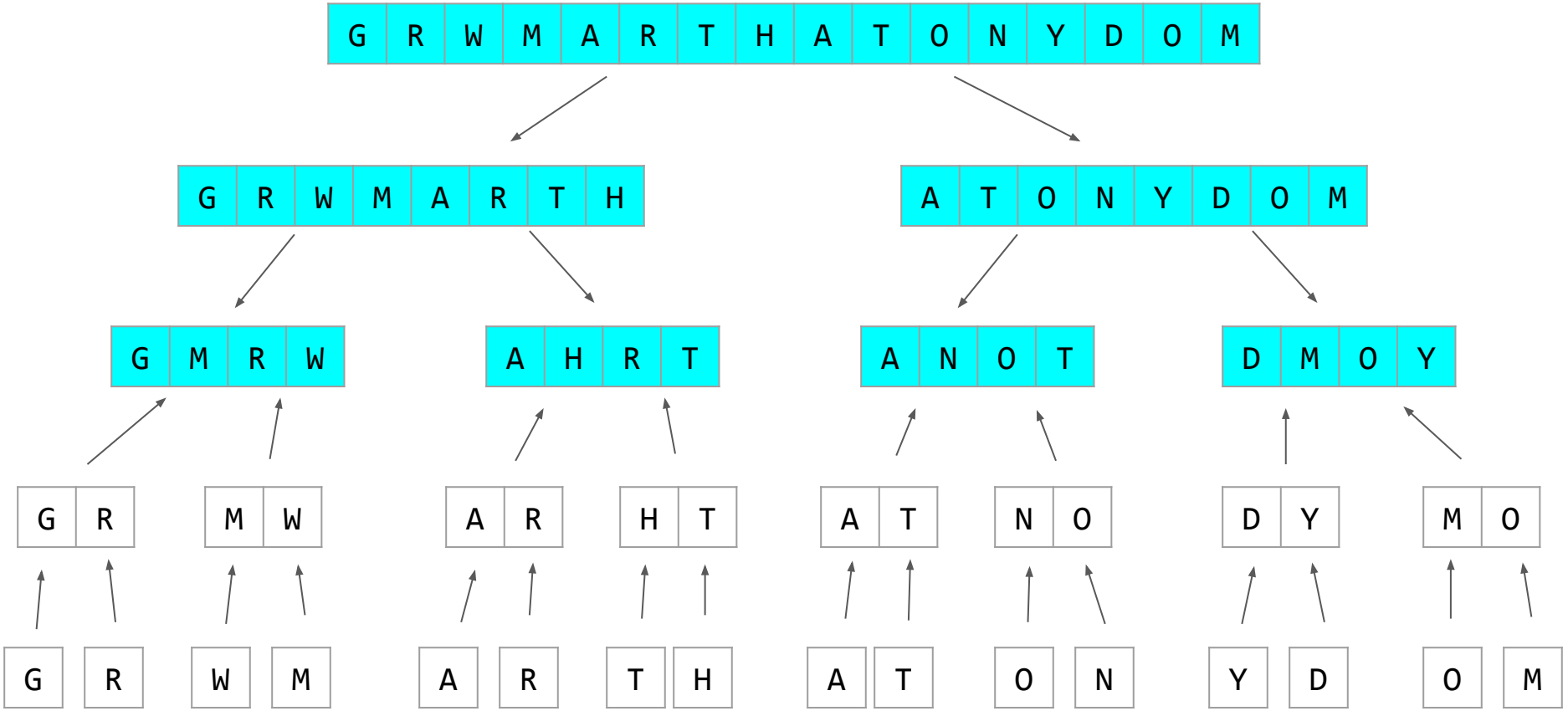
Let’s see how we can apply this idea to sort an unsorted list



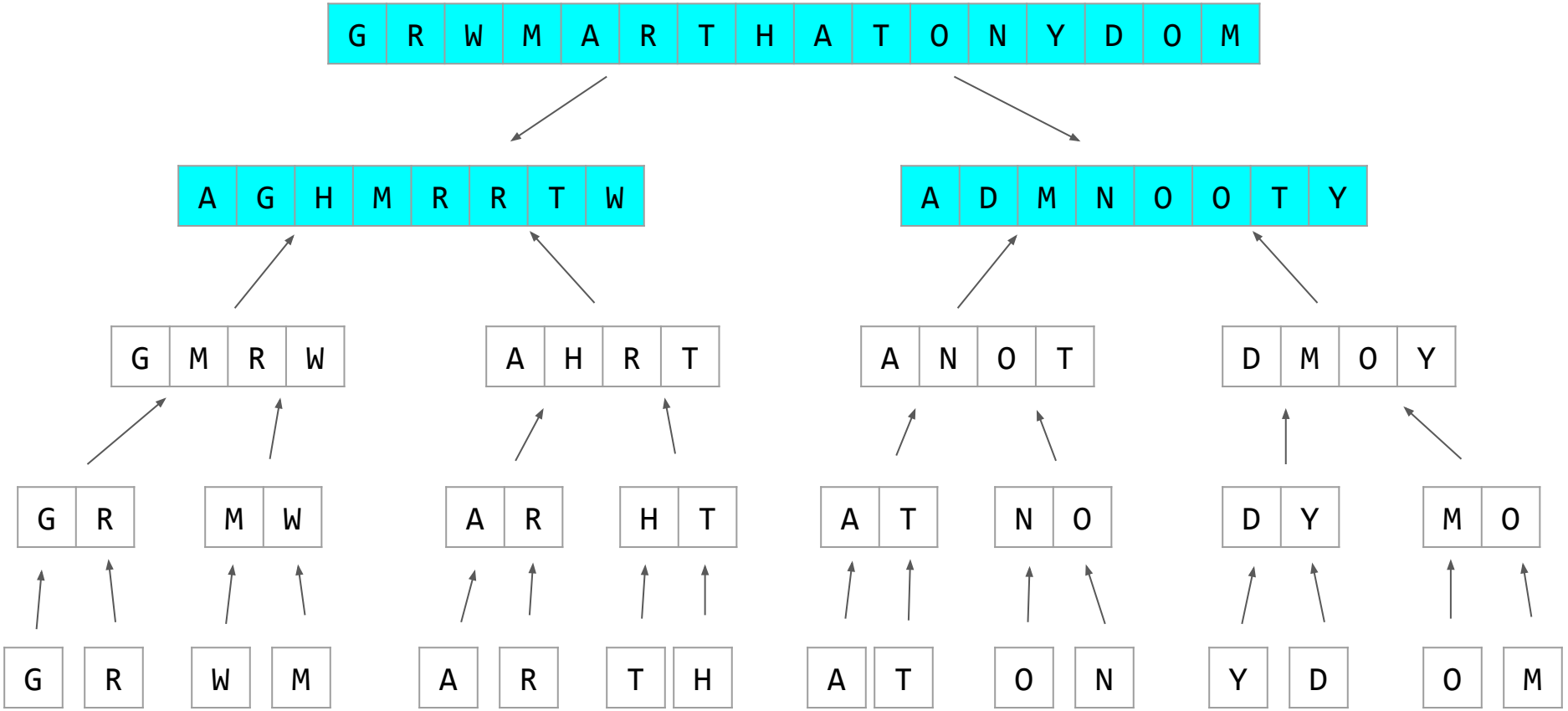
Let’s see how we can apply this idea to sort an unsorted list



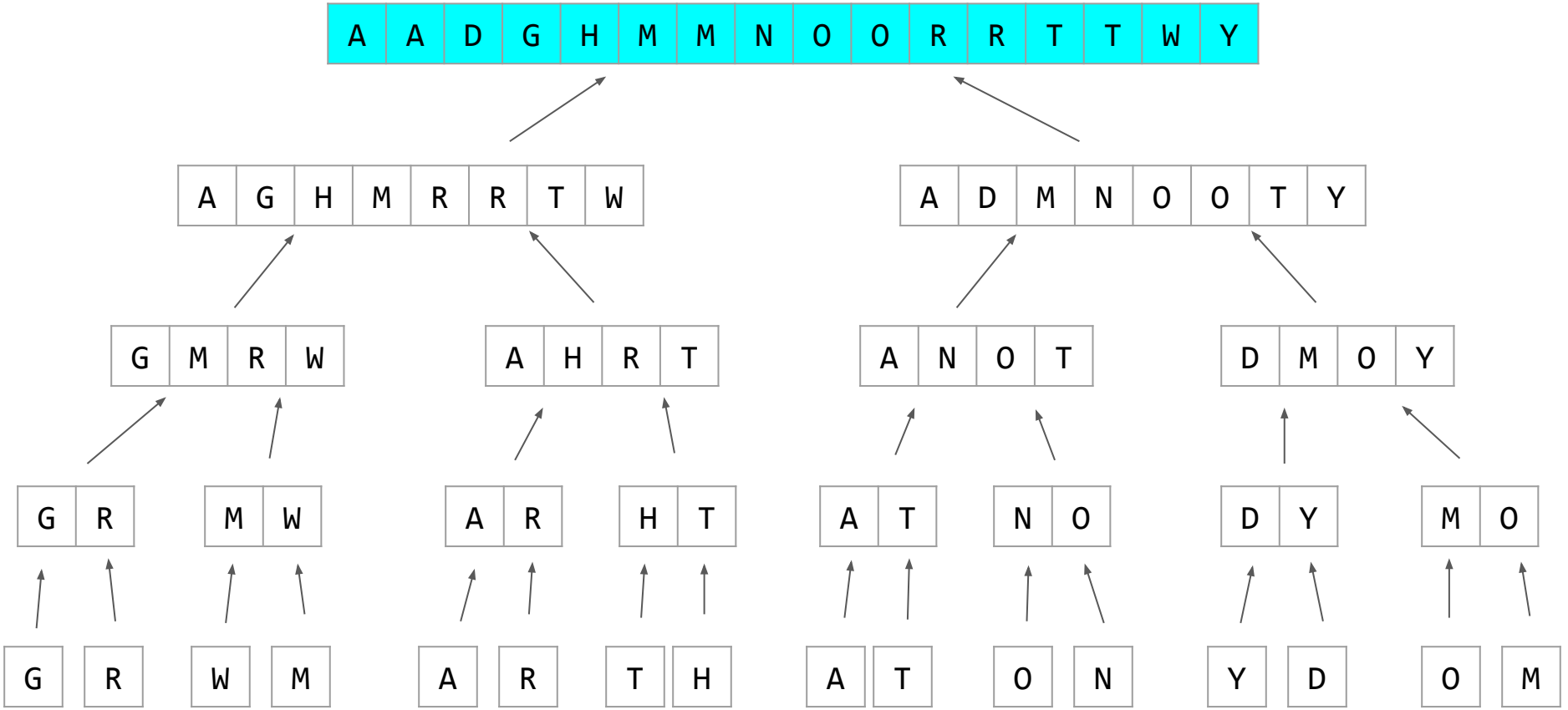
Let’s see how we can apply this idea to sort an unsorted list



Let’s see how we can apply this idea to sort an unsorted list



Let’s see how we can apply this idea to sort an unsorted list



Merge sort - writing the code

```
function y = mergeSort(x)
% x is a vector. y is a vector
% consisting of the values in x
% sorted from smallest to largest.
n = length(x);
if n == 1
    y = x;
else
    m = n/2;
    yL = mergeSort(x(1:m));
    yR = mergeSort(x(m+1:n));
    % merge sorted yL and yR
    y = merge(yL,yR);
end
```

Using recursion we need to identify base case(s) and make progress toward a base case.

```
% sort vector x
if length(x) is 1,
        Do nothing. x is already sorted.
else
        Cut x in half
        Sort the left half
        Sort the right half
        Merge the sorted left and right halves
```

Base case

Make progress towards base case

Writing the merge sort function

```
function y = mergeSort(x)
% x is a vector. y is a vector
% consisting of the values in x
% sorted from smallest to largest.
n = length(x);
if n == 1
    y = x;
else
    m = n/2;
    yL = mergeSort(x(1:m));
    yR = mergeSort(x(m+1:n));
    % merge sorted yL and yR
    y = merge(yL,yR);
end
```

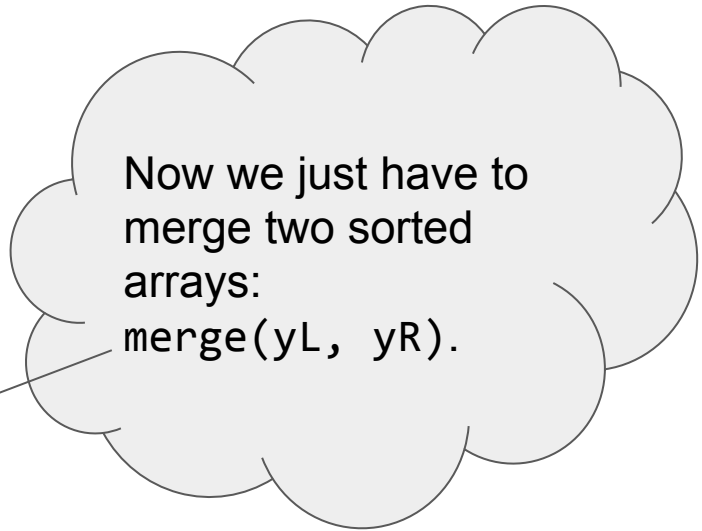
Assuming the function merge correctly merges two sorted arrays, are there any errors in this code?

If n is odd, m will be a decimal number.
Answer: Let's floor (or ceil) m

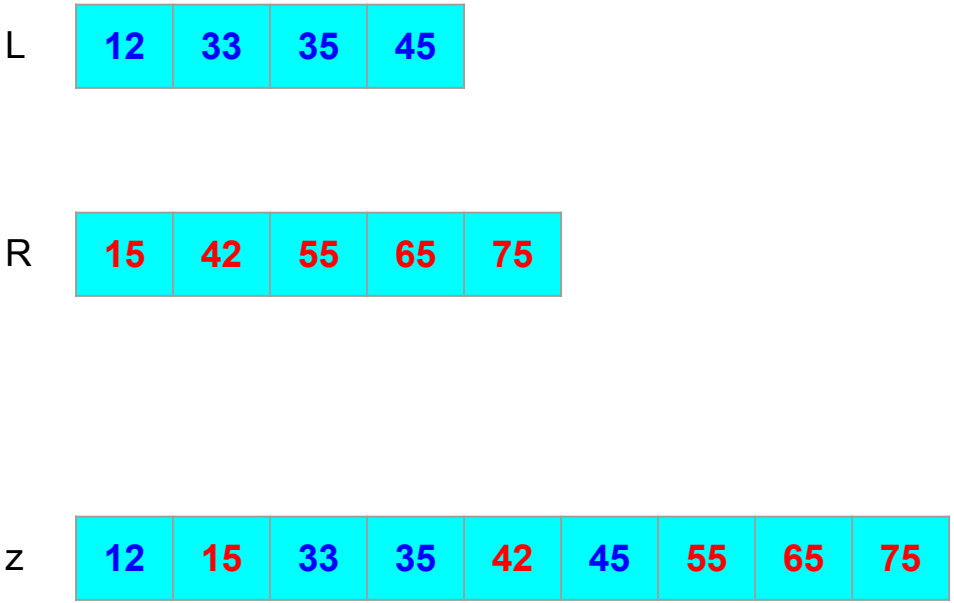
If x = [ ], your code enters infinite recursion.
Answer: Add to base cases

Writing the merge sort function

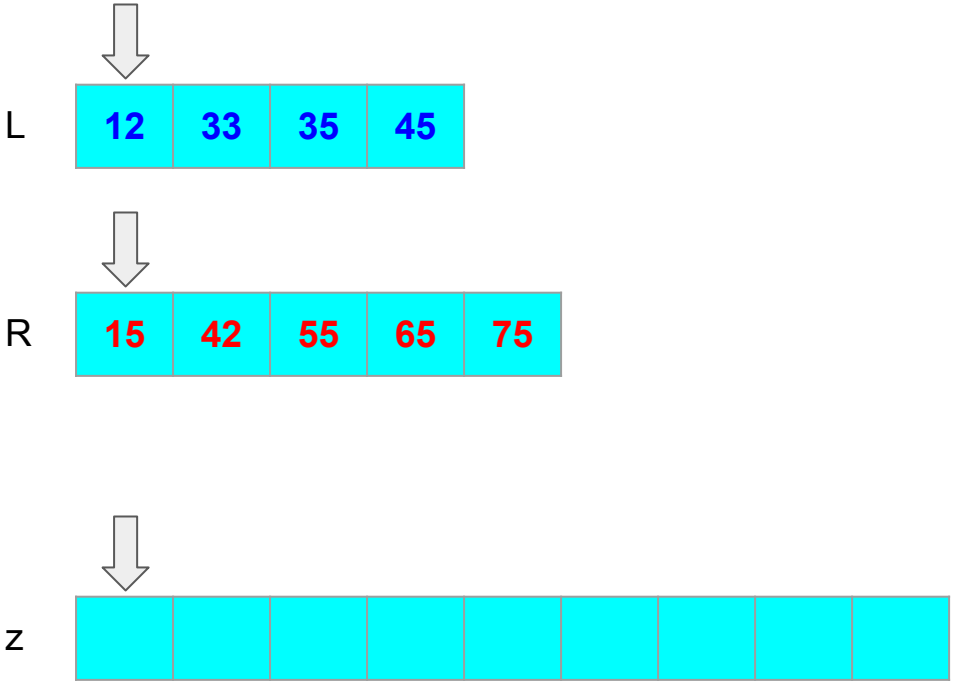
```
function y = mergeSort(x)
% x is a vector. y is a vector
% consisting of the values in x
% sorted from smallest to largest.
n = length(x);
if n == 1 || n == 0
    y = x;
else
    m = floor(n/2);
    yL = mergeSort(x(1:m));
    yR = mergeSort(x(m+1:n));
    % merge sorted yL and yR
    y = merge(yL,yR);
end
```

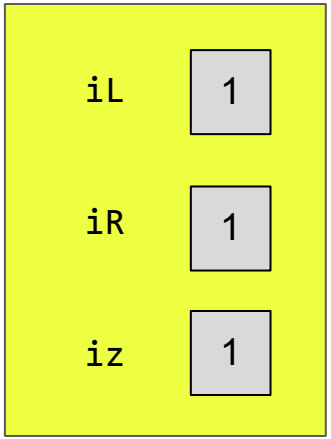


How can we merge two sorted arrays?



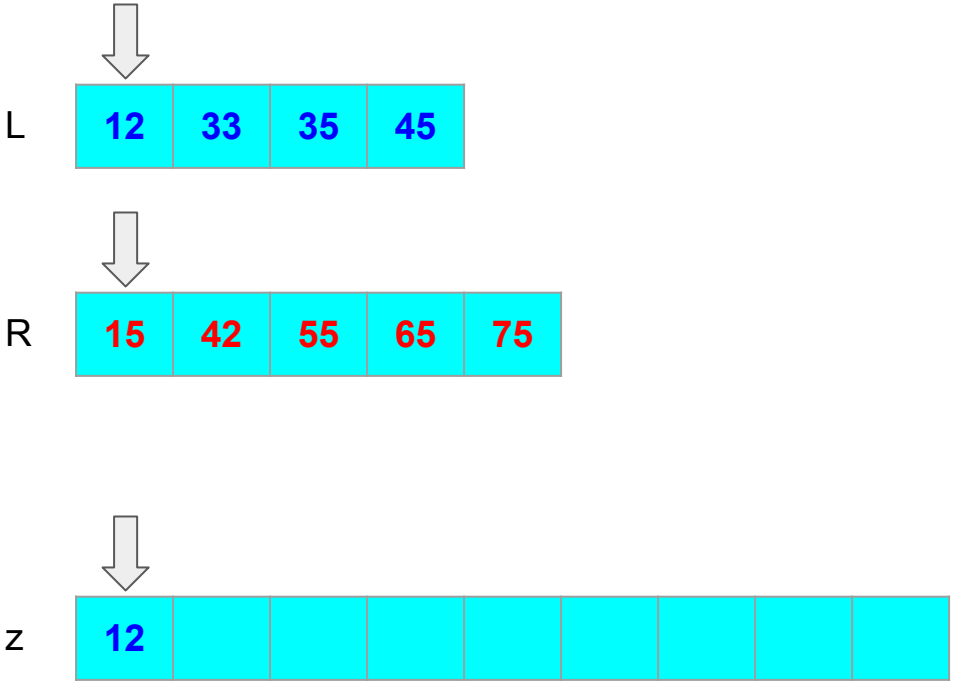
How can we merge two sorted arrays?

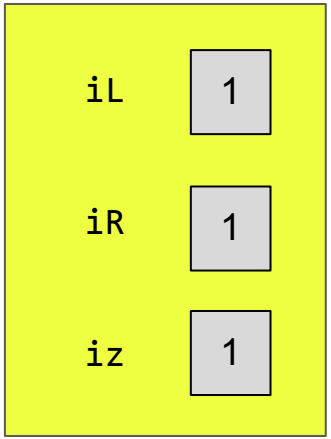





```
if L(iL) <= R(iR)
     z(iz) = L(iL);
     Increment iL and iz;
end
```

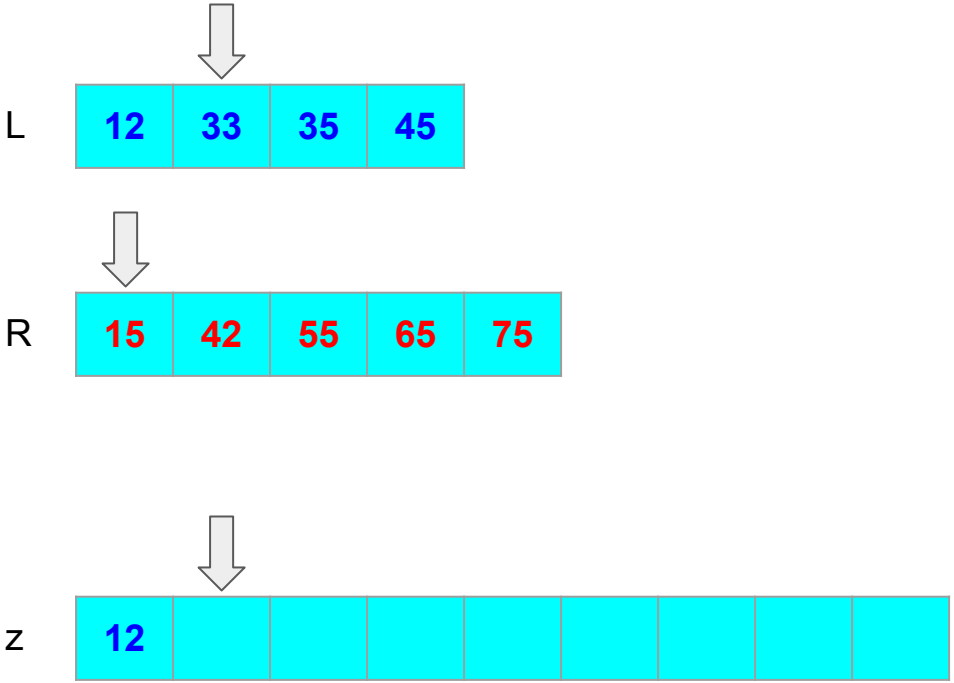
How can we merge two sorted arrays?

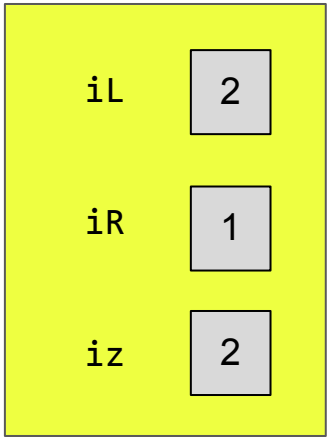





```
if L(iL) <= R(iR)
     z(iz) = L(iL);
     Increment iL and iz;
end
```

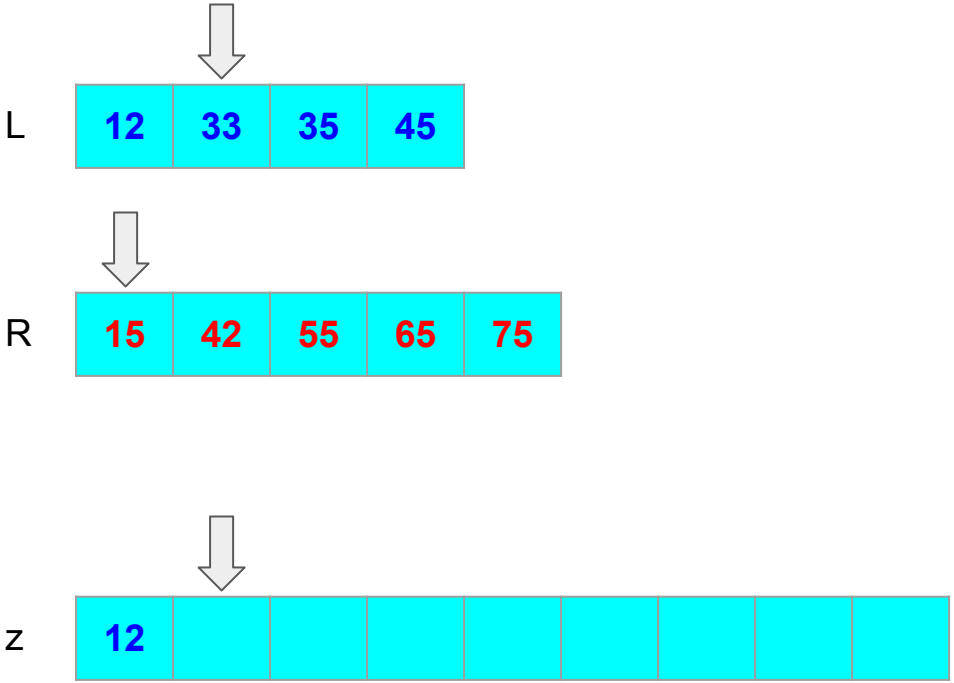
How can we merge two sorted arrays?

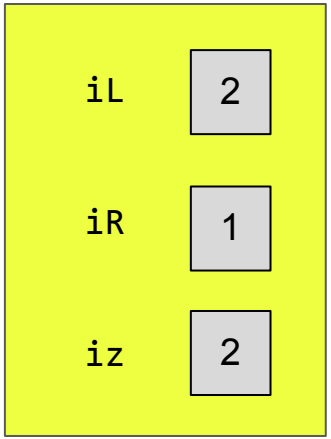





```
if L(iL) <= R(iR)
     z(iz) = L(iL);
     Increment iL and iz;
end
```

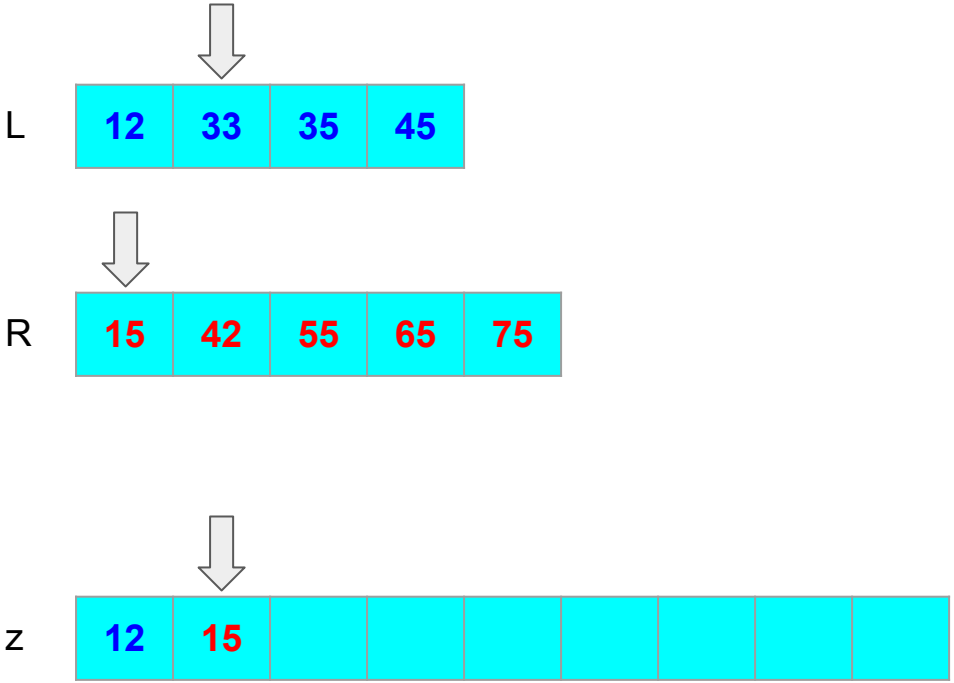
How can we merge two sorted arrays?






```
if L(iL) <= R(iR)
     z(iz) = L(iL);
     Increment iL and iz;
else
     z(iz) = R(iR);
     Increment iR and iz;
end
```

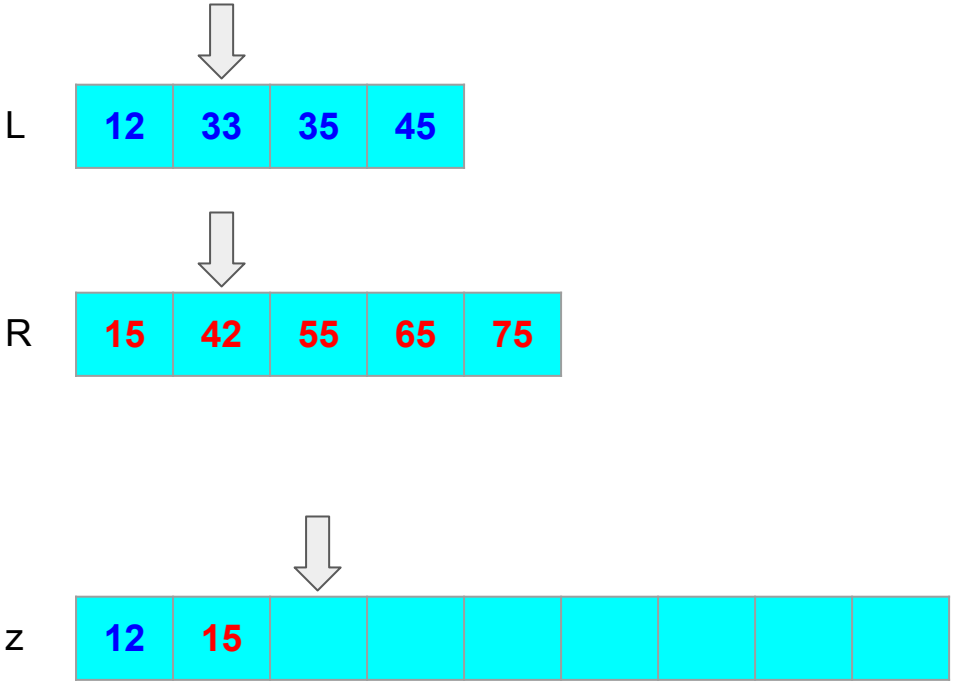
How can we merge two sorted arrays?



iL	2
iR	1
iz	2

```
if L(iL) <= R(iR)
     z(iz) = L(iL);
     Increment iL and iz;
else
     z(iz) = R(iR);
     Increment iR and iz;
end
```

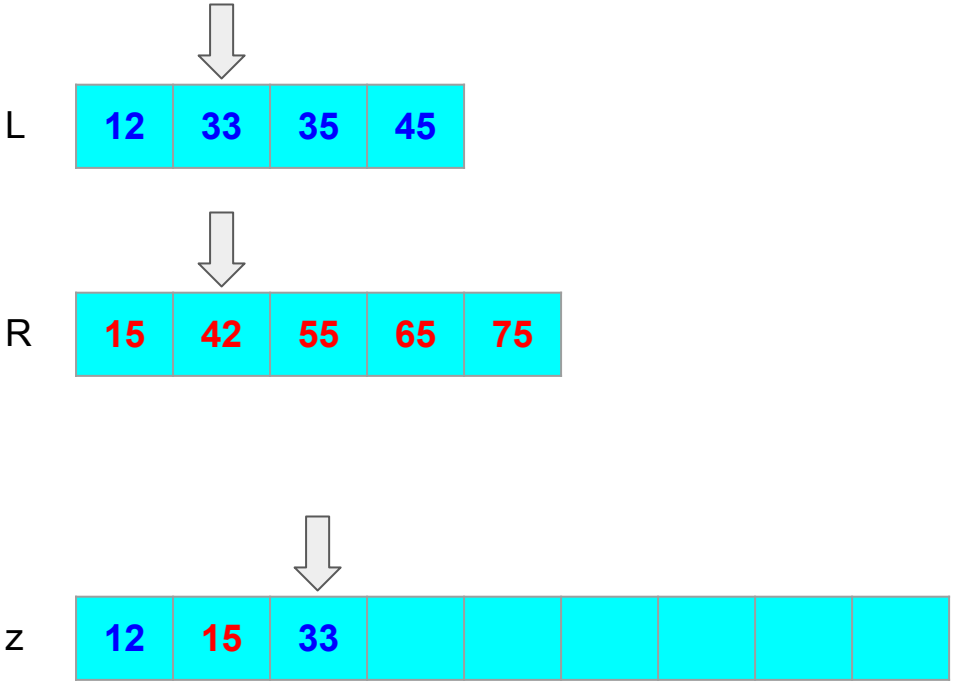
How can we merge two sorted arrays?

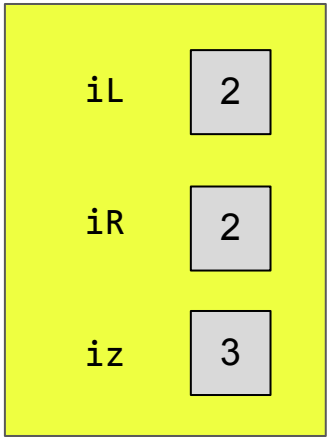


iL	2
iR	2
iz	3

```
if L(iL) <= R(iR)
     z(iz) = L(iL);
     Increment iL and iz;
else
     z(iz) = R(iR);
     Increment iR and iz;
end
```

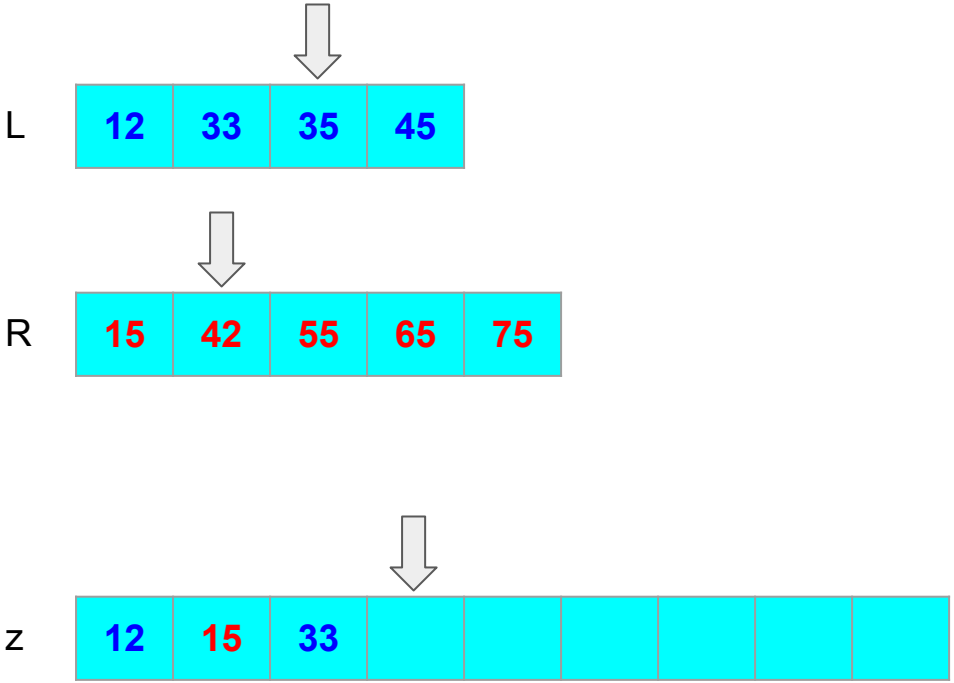
How can we merge two sorted arrays?

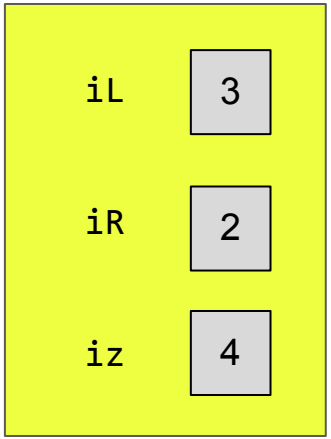





```
if L(iL) <= R(iR)
     z(iz) = L(iL);
     Increment iL and iz;
else
     z(iz) = R(iR);
     Increment iR and iz;
end
```

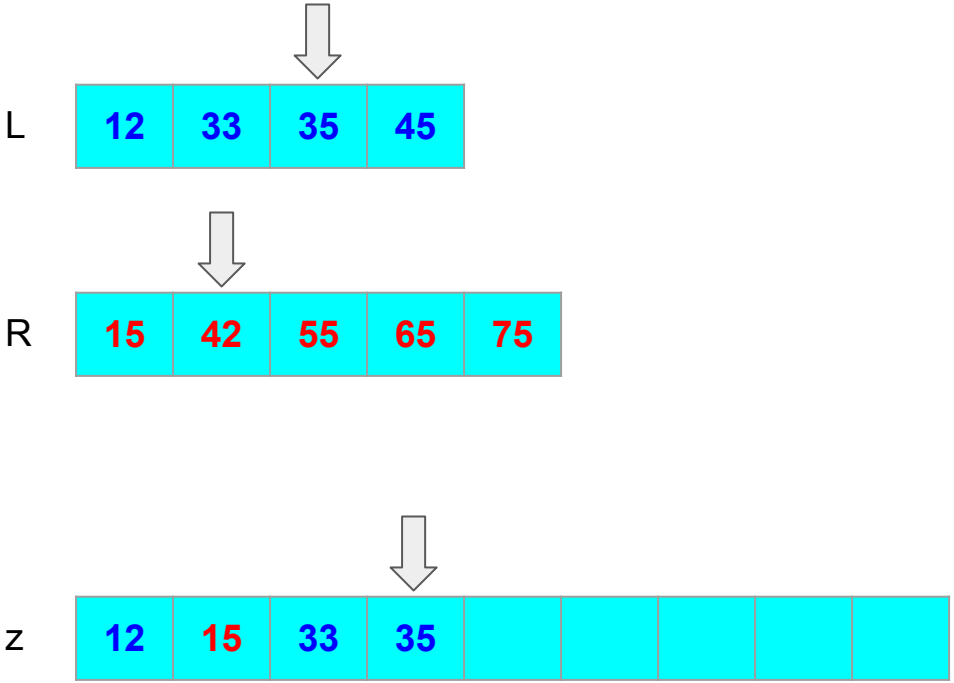
How can we merge two sorted arrays?

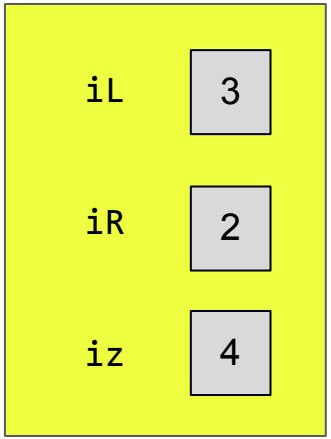





```
if L(iL) <= R(iR)
     z(iz) = L(iL);
     Increment iL and iz;
else
     z(iz) = R(iR);
     Increment iR and iz;
end
```

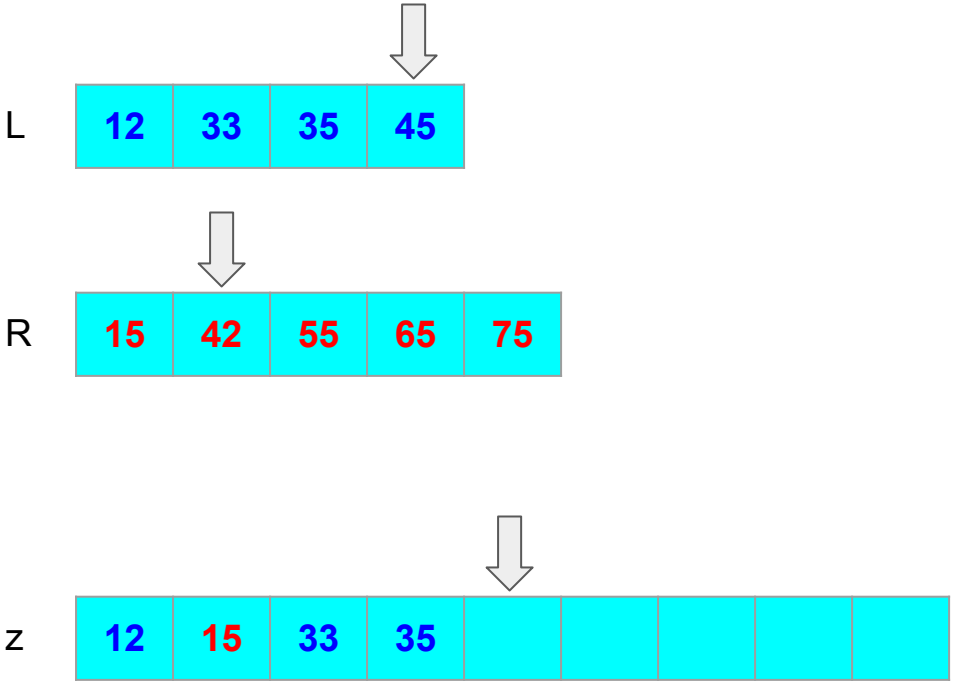
How can we merge two sorted arrays?






```
if L(iL) <= R(iR)
     z(iz) = L(iL);
     Increment iL and iz;
else
     z(iz) = R(iR);
     Increment iR and iz;
end
```

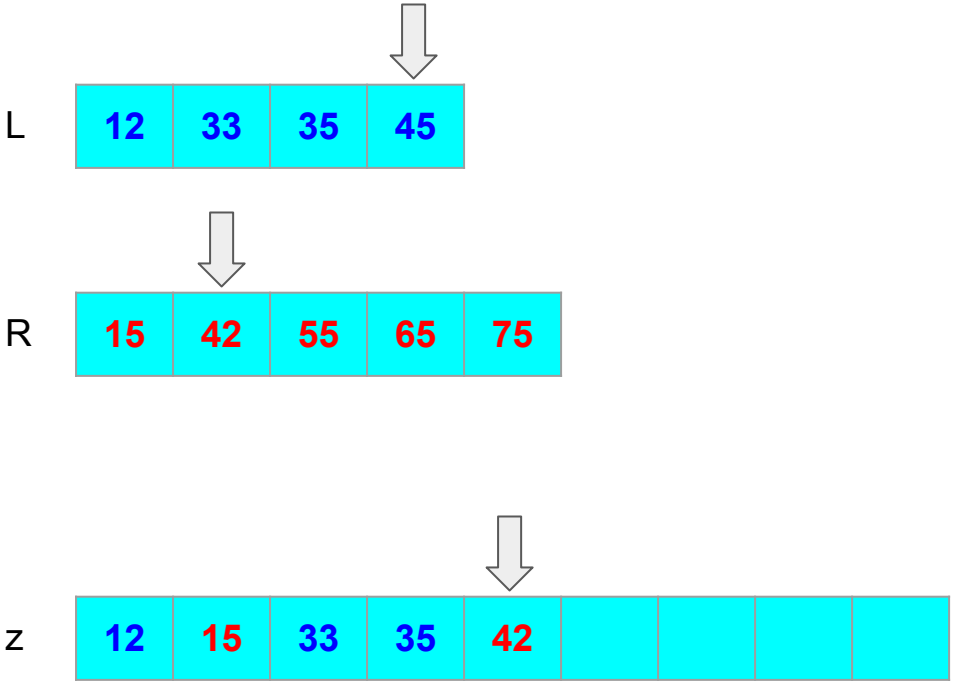
How can we merge two sorted arrays?

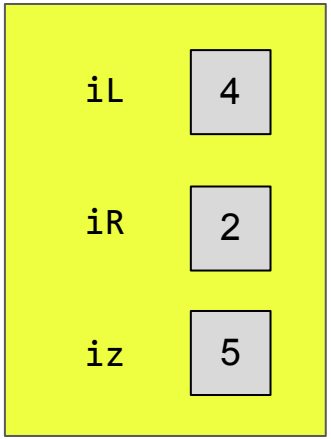


iL	4
iR	2
iz	5

```
if L(iL) <= R(iR)
     z(iz) = L(iL);
     Increment iL and iz;
else
     z(iz) = R(iR);
     Increment iR and iz;
end
```

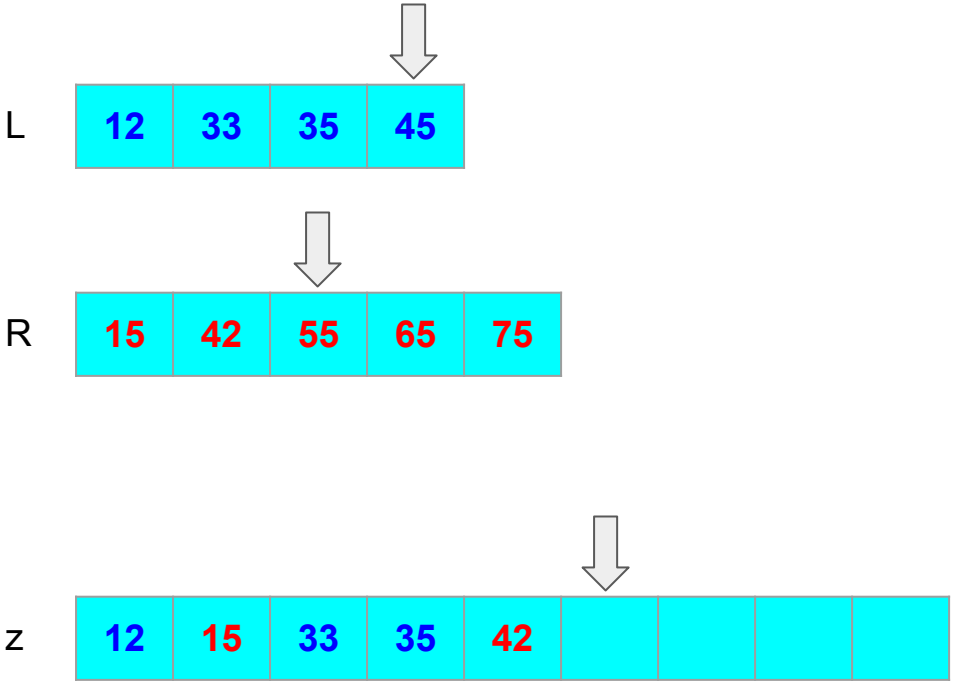
How can we merge two sorted arrays?

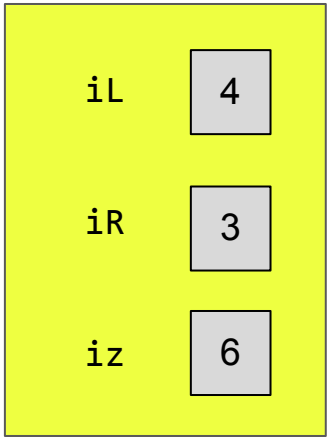





```
if L(iL) <= R(iR)
     z(iz) = L(iL);
     Increment iL and iz;
else
     z(iz) = R(iR);
     Increment iR and iz;
end
```

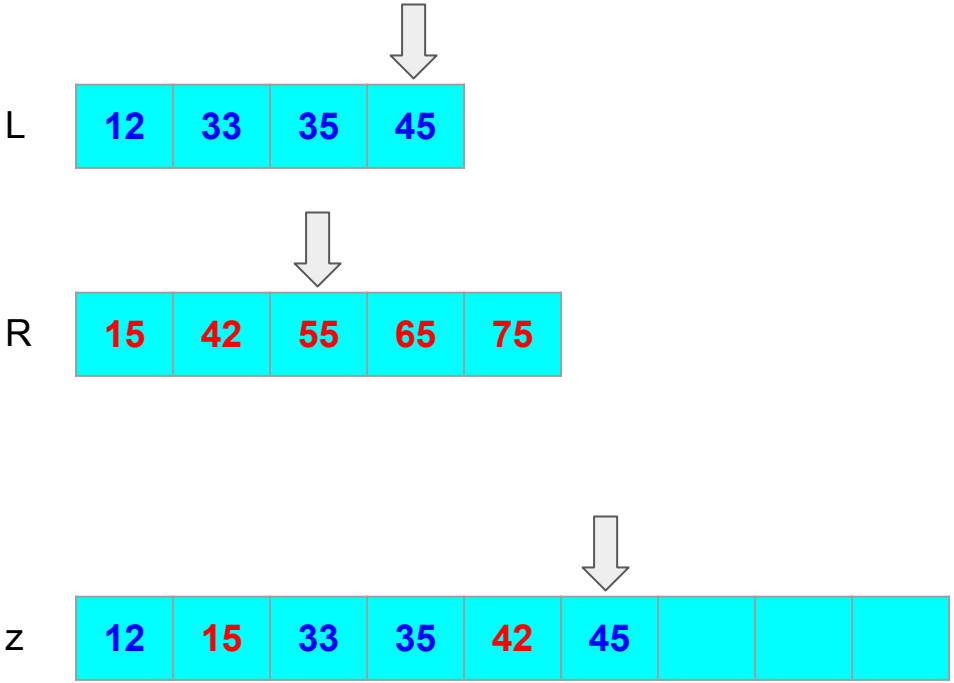
How can we merge two sorted arrays?

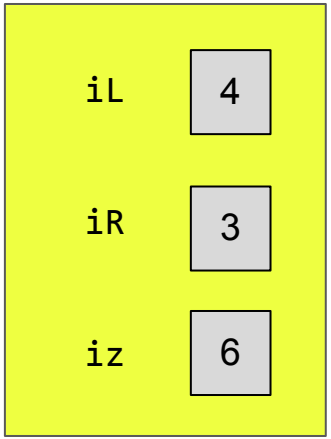





```
if L(iL) <= R(iR)
     z(iz) = L(iL);
     Increment iL and iz;
else
     z(iz) = R(iR);
     Increment iR and iz;
end
```

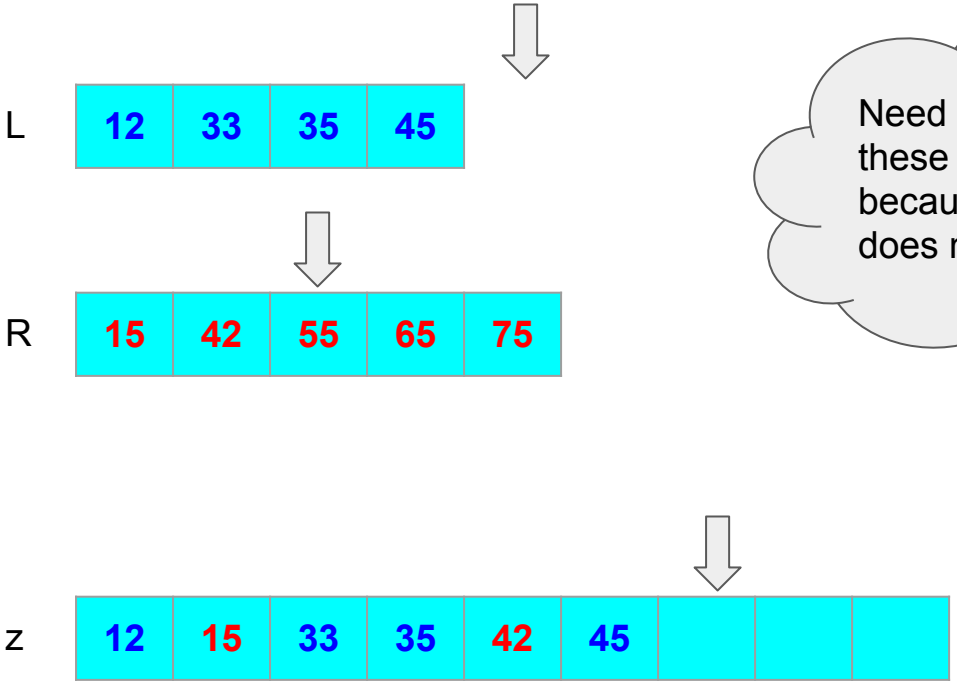
How can we merge two sorted arrays?






```
if L(iL) <= R(iR)
     z(iz) = L(iL);
     Increment iL and iz;
else
     z(iz) = R(iR);
     Increment iR and iz;
end
```

How can we merge two sorted arrays?

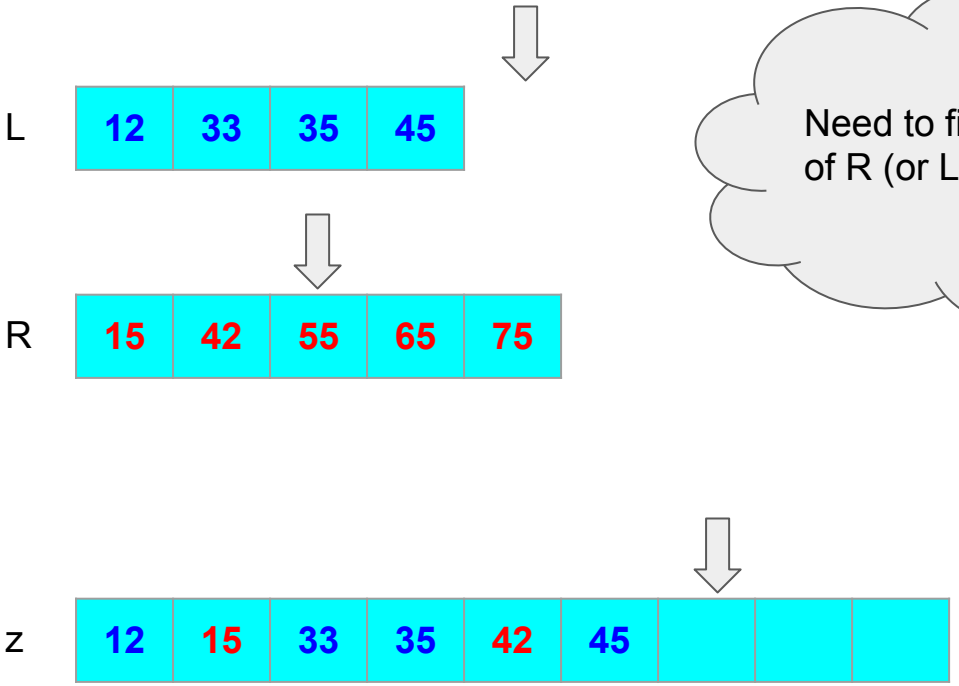


Need to stop doing these comparisons because `L(iL)=L(5)` does not exist.

iL	5
iR	3
iz	7

```
if L(iL) <= R(iR)
     z(iz) = L(iL);
     Increment iL and iz;
else
     z(iz) = R(iR);
     Increment iR and iz;
end
```

How can we merge two sorted arrays?

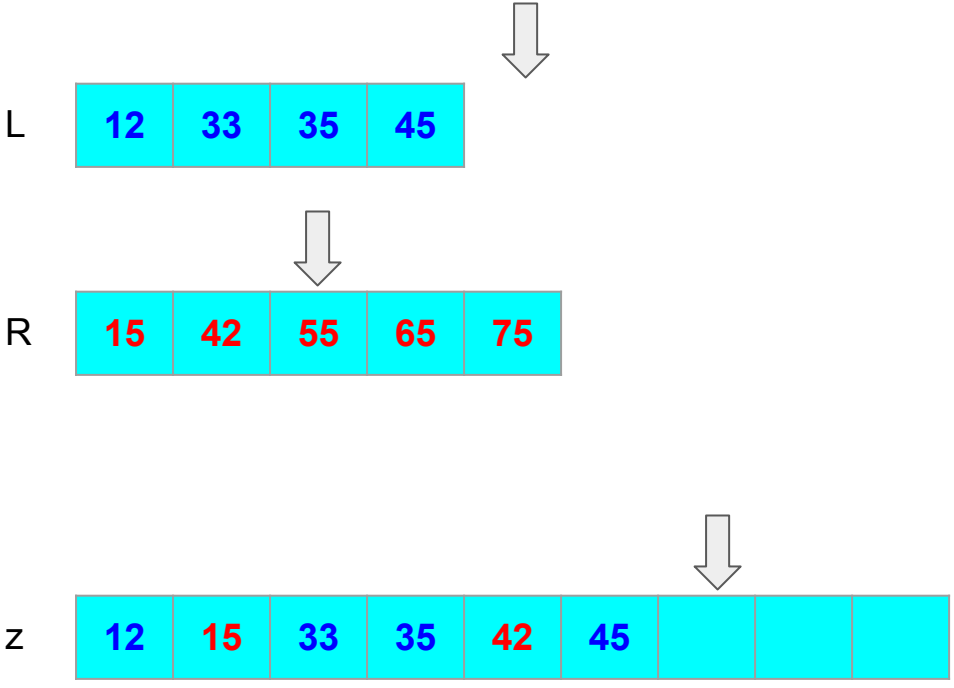


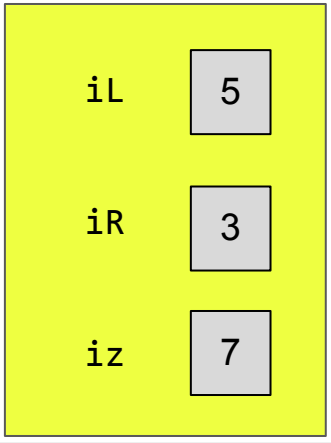
Need to fill in the rest of R (or L) into z

iL	5
iR	3
iz	7

```
while iL<=length(L) && iR<=length(R)
     if L(iL) <= R(iR)
          z(iz) = L(iL);
          Increment iL and iz;
     else
          z(iz) = R(iR);
          Increment iR and iz;
     end
end
```

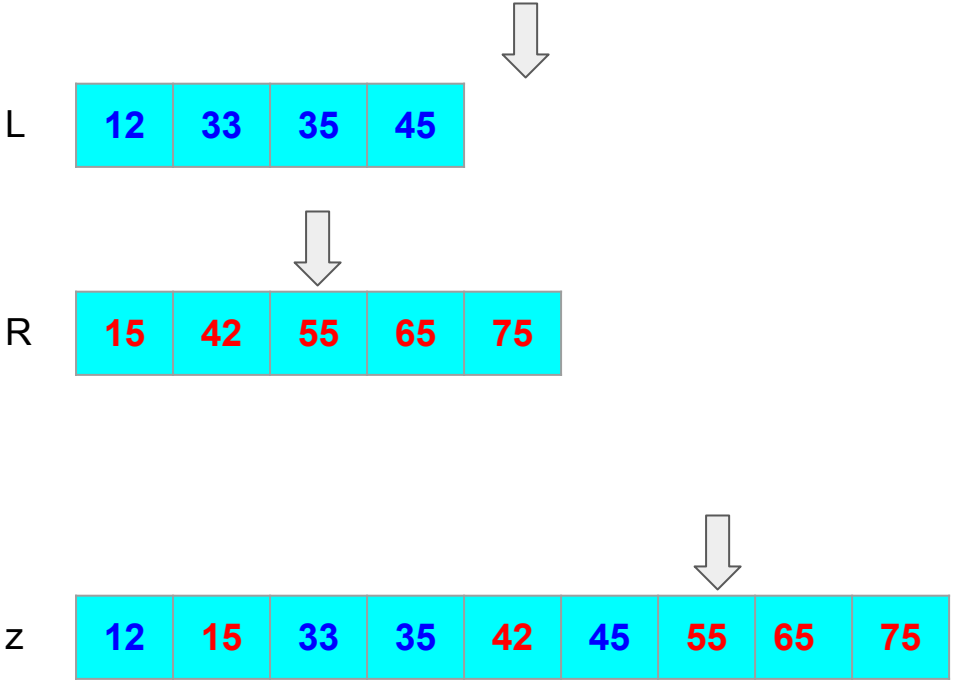
How can we merge two sorted arrays?

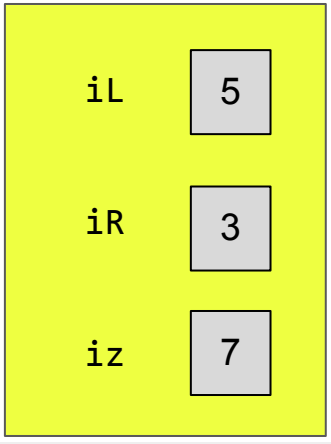





```
while iL<=length(L) && iR<=length(R)
     if L(iL) <= R(iR)
          z(iz) = L(iL);
          Increment iL and iz;
     else
          z(iz) = R(iR);
          Increment iR and iz;
     end
end
% if iR <= length(R), put in z
% if iL <= length(L), put in z
```

How can we merge two sorted arrays?






```
while iL<=length(L) && iR<=length(R)
     if L(iL) <= R(iR)
          z(iz) = L(iL);
          Increment iL and iz;
     else
          z(iz) = R(iR);
          Increment iR and iz;
     end
end
% if iR <= length(R), put in z
% if iL <= length(L), put in z
```

How can we merge two sorted arrays

```
function z = merge(L,R)
% Merge two sorted arrays L and R
nL = length(L); nR = length(R);
z = zeros(1, nL+nR);
iL = 1; iR = 1; iz = 1;
while iL<=nL && iR<=nR
    if L(iL) <= R(iR)
        z(iz)= L(iL); iL=iL+1; iz=iz+1;
    else
        z(iz)= R(iR); iR=iR+1; iz=iz+1;
    end
end

% if iL <= nL, put rest of L in z


% if iR <= nR, put rest of R in z
```

How can we merge two sorted arrays

```
function z = merge(L,R)
% Merge two sorted arrays L and R
nL = length(L); nR = length(R);
z = zeros(1, nL+nR);
iL = 1; iR = 1; iz = 1;
while iL<=nL && iR<=nR
    if L(iL) <= R(iR)
        z(iz)= L(iL); iL=iL+1; iz=iz+1;
    else
        z(iz)= R(iR); iR=iR+1; iz=iz+1;
    end
end
while iL<=nL % copy remaining L-values
    z(iz)= L(iL); iL=iL+1; iz=iz+1;
end
while iR<=nR % copy remaining R-values
    z(iz)= R(iR); iR=iR+1; iz=iz+1;
end
```

Merge sort: both codes together

```
function y = mergeSort(x)
% x is a vector. y is a vector
% consisting of the values in x
% sorted from smallest to largest.
n = length(x);
if n == 1 || n == 0
    y = x;
else
    m = floor(n/2);
    yL = mergeSort(x(1:m));
    yR = mergeSort(x(m+1:n));
    % merge sorted yL and yR
    y = merge(yL,yR);
end
```

```
function z = merge(L,R)
% Merge two sorted arrays L and R
nL = length(L); nR = length(R);
z = zeros(1, nL+nR);
iL = 1; iR = 1; iz = 1;
while iL<=nL && iR<=nR
    if L(iL) <= R(iR)
        z(iz)= L(iL); iL=iL+1; iz=iz+1;
    else
        z(iz)= R(iR); iR=iR+1; iz=iz+1;
    end
end
while iL<=nL % copy remaining L-values
    z(iz)= L(iL); iL=iL+1; iz=iz+1;
end
while iR<=nR % copy remaining R-values
    z(iz)= R(iR); iR=iR+1; iz=iz+1;
end
```

Merge sort: both codes together

```
function y = mergeSort(x)
% x is a vector. y is a vector
% consisting of the values i
% sorted from smallest to l
n = length(x);
if n == 1 || n == 0
    y = x;
else
    m = floor(n/2);
    yL = mergeSort(x(1:m));
    yR = mergeSort(x(m+1:n));
    % merge sorted yL and yR
    y = merge(yL,yR);
end
```

```
function z = merge(L,R)
% M    e two sorted arrays L and R
    L); nR = length(R);
    L+nR);
     iz = 1;
     iR<=nR
     R(iR)
     (iL); iL=iL+1; iz=iz+1;

     R(iR); iR=iR+1; iz=iz+1;

    L % copy remaining L-values
    (iz)= L(iL); iL=iL+1; iz=iz+1;

while iR<=nR % copy remaining R-values
    z(iz)= R(iR); iR=iR+1; iz=iz+1;
end
```

Challenge question:

When I call mergeSort(x) when x has length 17, what is the maximum number of mergeSort call frames that will be open at one time?

We have a sorted array, now what?

6	10	15	23	29	31	32	45	47	52

We can still use linear search to search for items in a sorted list.

When you search through a phone book, you do something a little more efficient than linear search…

Key idea of “phone book search”: repeated halving

To find the page containing Benito Antonio Martínez Ocasio’s number…

While (phone book is longer than 1 page)
    Open to the middle page
    If “Ocasio” comes before the first entry on the current page
        Rip and throw away the second half
    Else
        Rip and throw away the first 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

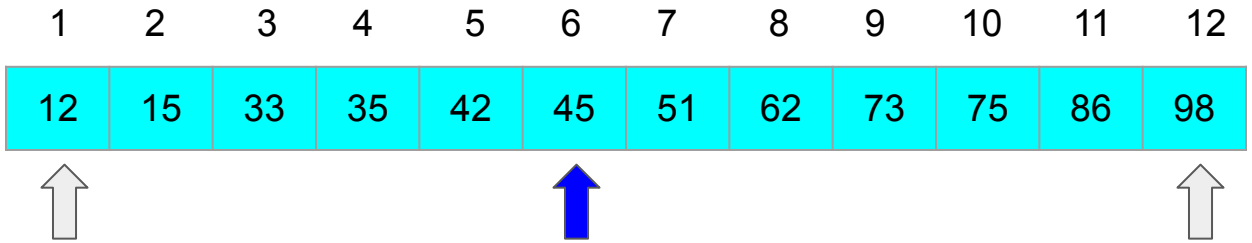
- Repeatedly halving the size of the “search space” is the main idea behind the method of binary search

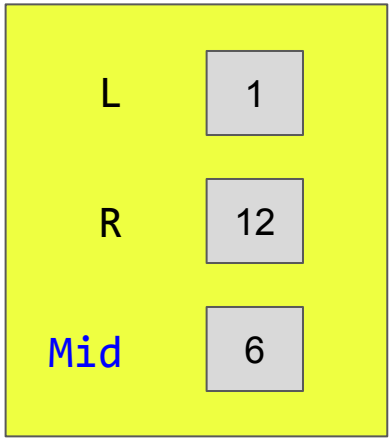
- An item in a sorted array of length `n` can be located with just $\log_2 n$ comparisons
  - Where does this `log` come from? Mathematics! Take a data structures and algorithms class to find out how!

- Time savings are significant!

n	$\log_2 n$
100	7
1000	10
10000	13

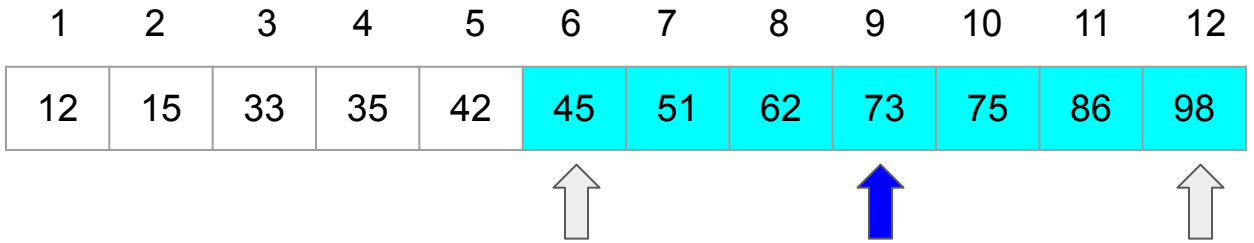
Binary search: target = 70

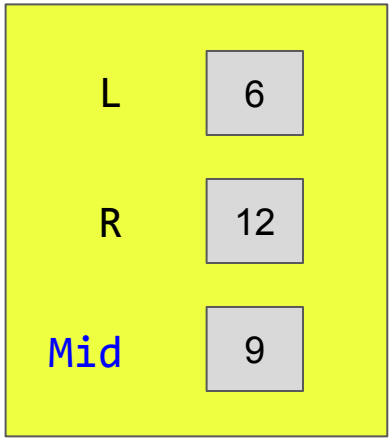




Since `v(mid) < target`, we know that we can throw away everything before index `mid`

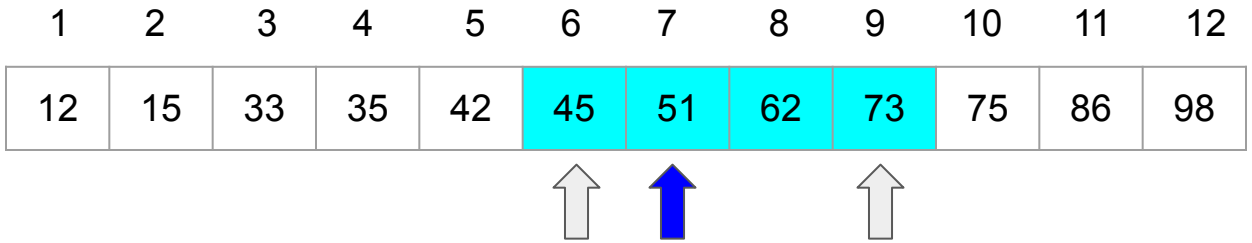
Binary search: target = 70

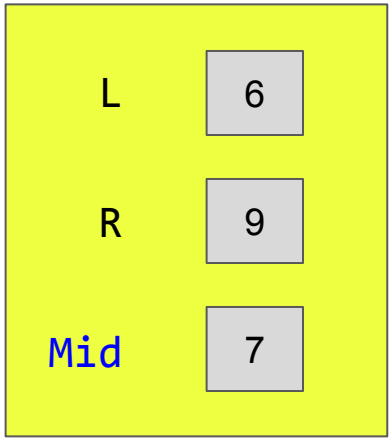




Since `v(mid) > target`, we know that we can throw away everything before the index `mid`

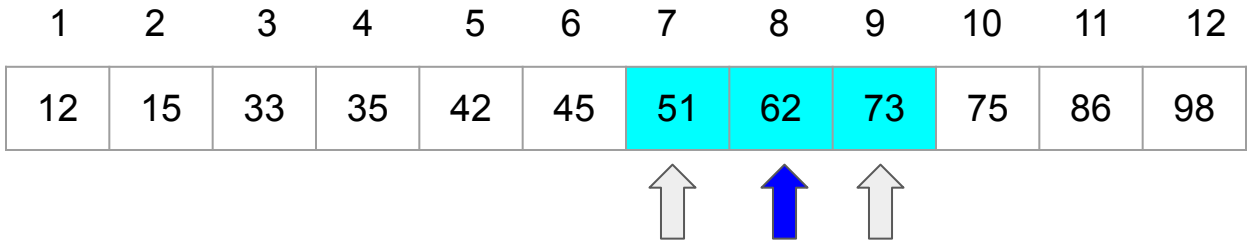
Binary search: target = 70





Since `v(mid) < target`, we know that we can throw away everything before the index `mid`

Binary search: target = 70



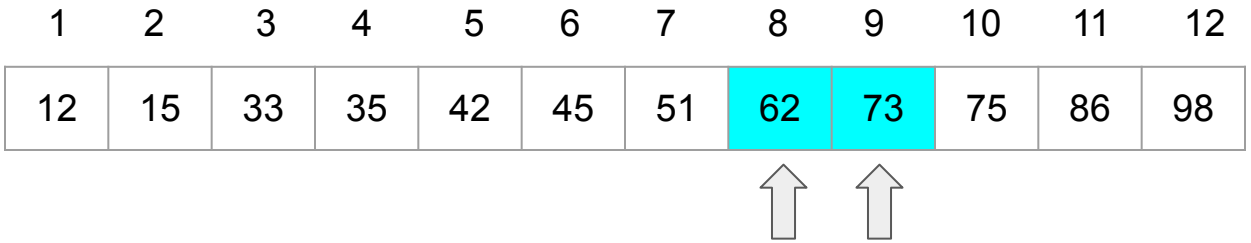
L 7

R 9

Mid 8

Since `v(mid) < target`, we know that we can throw away everything before the index `mid`

Binary search: target = 70



L 8

R 9

Mid 8

Done because `L-R == 1`

```
function L = binarySearch(x, v)
% Find position after which to insert x. v is sorted in ascending
% order. 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).
```

```
function L = binarySearch(x, v)
% Find position after which to insert x. v is sorted in ascending
% order. 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).




% Keep halving your search space [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
        % get rid of everything in left half
    else
        % get rid of everything in the right half
    end
end
```

```
function L = binarySearch(x, v)
% Find position after which to insert x. v is sorted in ascending
% order. 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).




% Keep halving your search space [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
```

```
function L = binarySearch(x, v)
% Find position after which to insert x. v is sorted in ascending
% order. 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 be outside of the range of v, initially set ...
L = 0;  R = length(v)+1;

% Keep halving your search space [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
```

Check out `binarySearch.m` and `showBinarySearch.m` on the course website!