PDS LAB – 9 (Section-5) Date: 3rd April 2017
Searching & Sorting
Tutorial Problems

1. Write a C program to search an element in a sorted array using binary search method. Input the sorted array through keyboard and print the output as "Element is present" or "Element is not present" based on whether the element is present or not?

2. Assume that the sequence of data elements are present in linked list. First create a linked list and place the data elements in nodes. Write a C function to search the data elements present in the linked list and print the output as the given data element is present along with its position if the data element is present in the list, otherwise the output should indicate that the data element is not present in the list.

3. Write a C program to sort the given array of elements using selection sort technique.

Assignment Problems (For All Students)

1. A student record contains the following fields {int roll_no, char name[20], struct date dob}, where date has the following fields {int date, int month, int year}. Suppose, you have 10 student records, write a C program to search the students records and retrieve the specific student record based on the input date of birth in the given format dd/mm/yy.

2. Extend the Tut-1 problem using ternary search and interpolation search methods.

3. Write a C program to sort the given array of elements using Insertion sort technique.

4. Write a C program to sort the given array of elements using Merge sort technique.

5. Write a C program to sort the given array of elements using Quick sort technique.

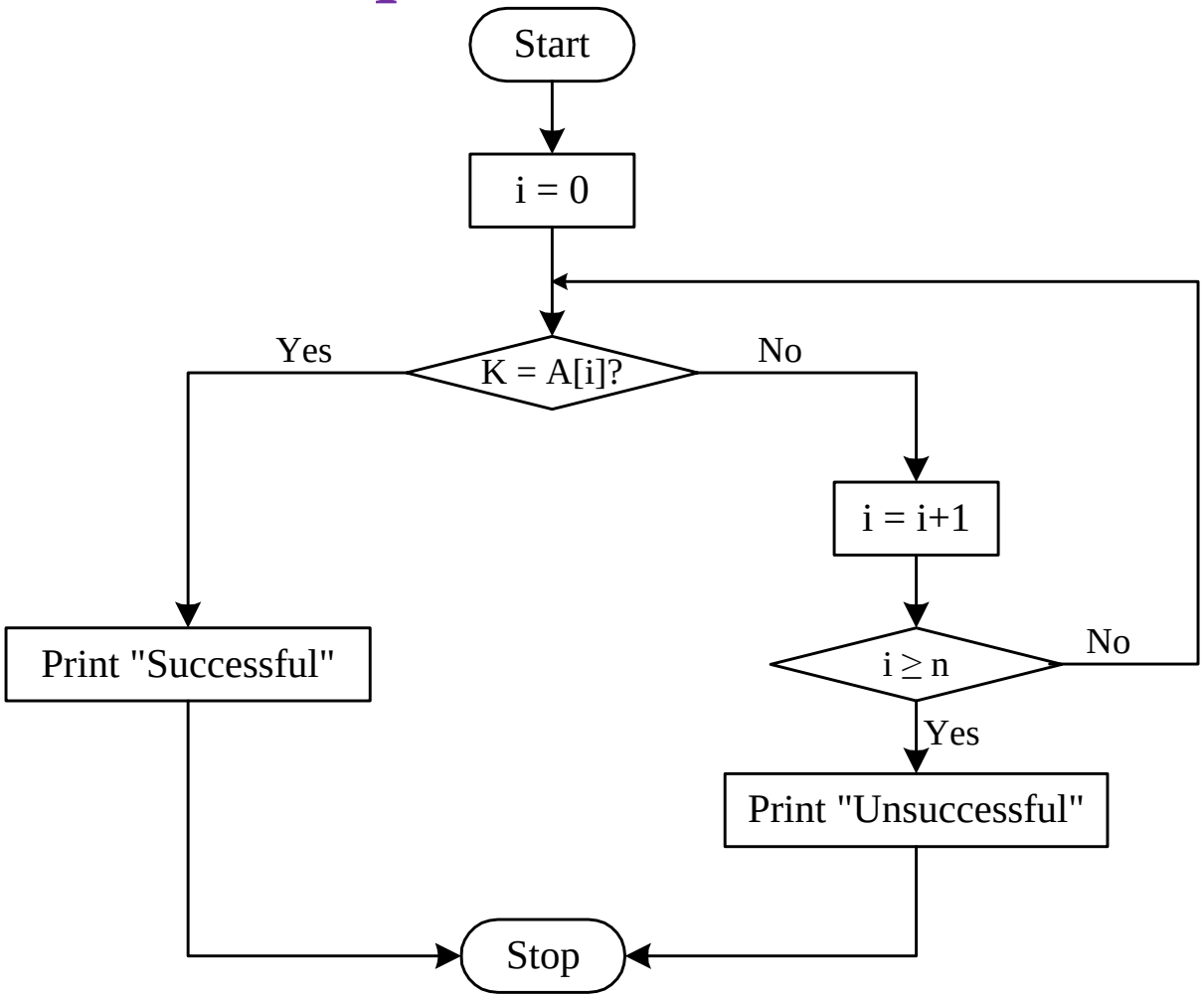
Assignment Problems (For those who completed 5 assignment problems)

1. Write C program to sort the English words based on alphabetic order.

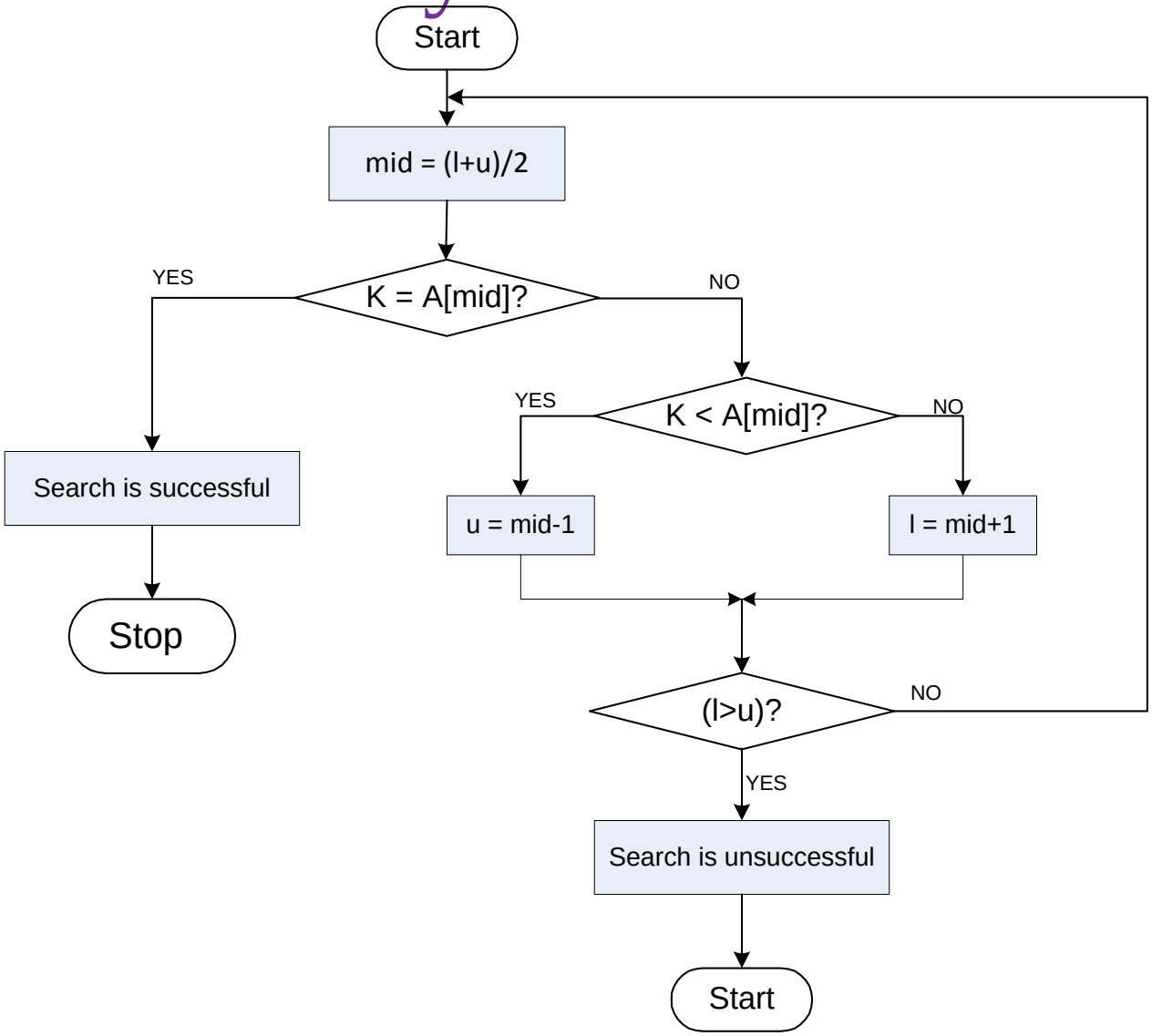
2. Write a C program to search the given word in the sorted list of words using binary search technique.

3. Write a C program to search an element in a sorted array using random search method.

Flowchart: Sequential Search with Array



Flowchart: Binary Search with Array

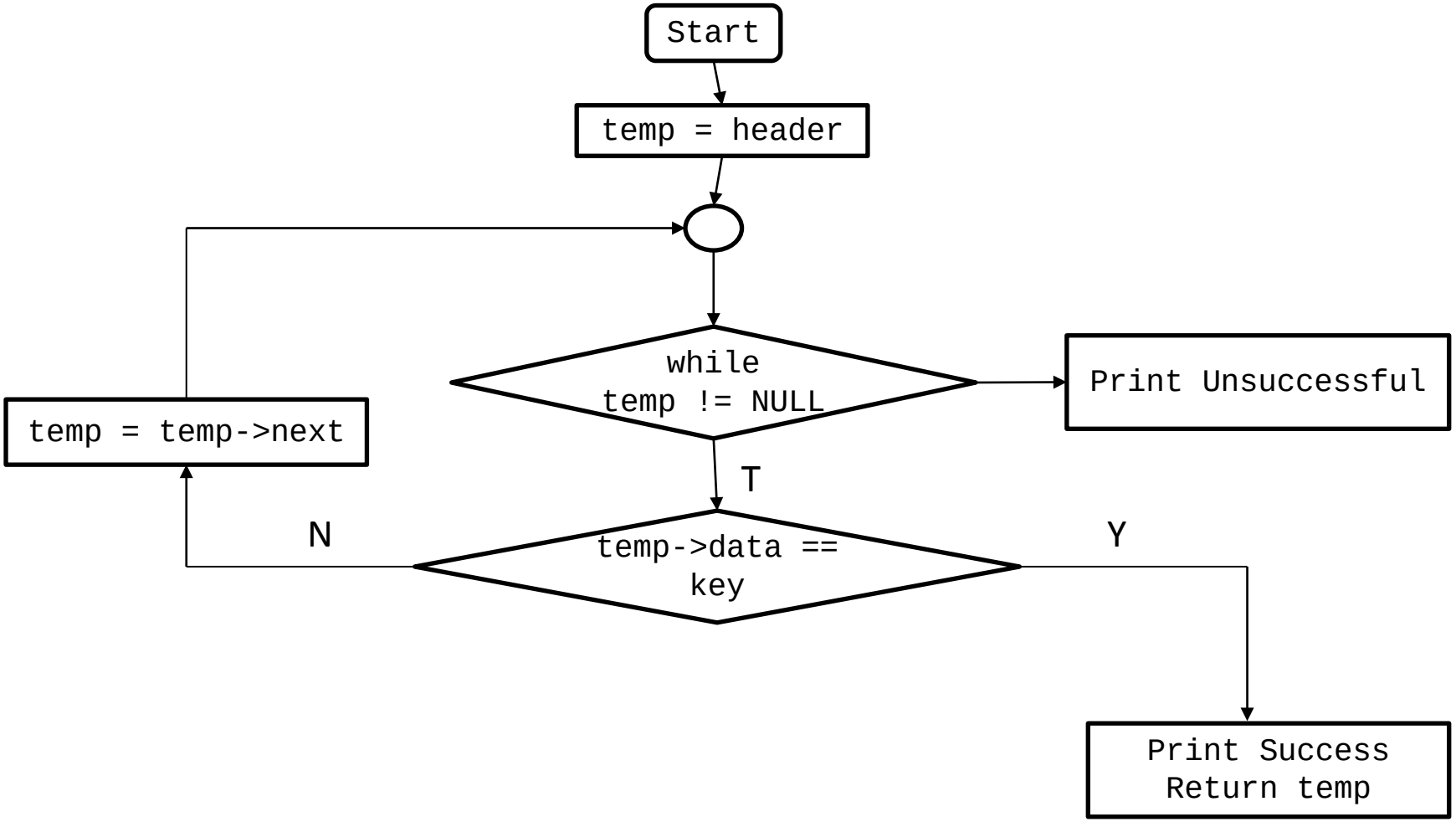


Interpolation Search

1.	$l = 1, u = n$				// Initialization: Range of searching
2.	flag = FALSE				// Hold the status of searching
3.	**While** (flag = FALSE) **do**				
4.		$loc = \left\lceil \frac{K - A[l]}{A[u] - A[l]} \right\rceil \times (u - l) + l$			
5.		**If** ($l \le loc \le u$) **then**			// If loc is within the range of the list
6.			**Case:** K < A[loc]		
7.				u = loc -1	
8.			**Case:** K = A[loc]		
9.				flag = TRUE	
10.			**Case:** K > A[loc]		
11.				l = loc +1	
12.		**Else**			
13.			**Exit**()		
14.		**EndIf**			
15.	**EndWhile**				
16.	**If** (flag) **then**				
17.		**Print** “Successful at” loc			
18.	**Else**				
19.		**Print** “Unsuccessful”			
20.	**EndIf**				
21.	**Stop**				

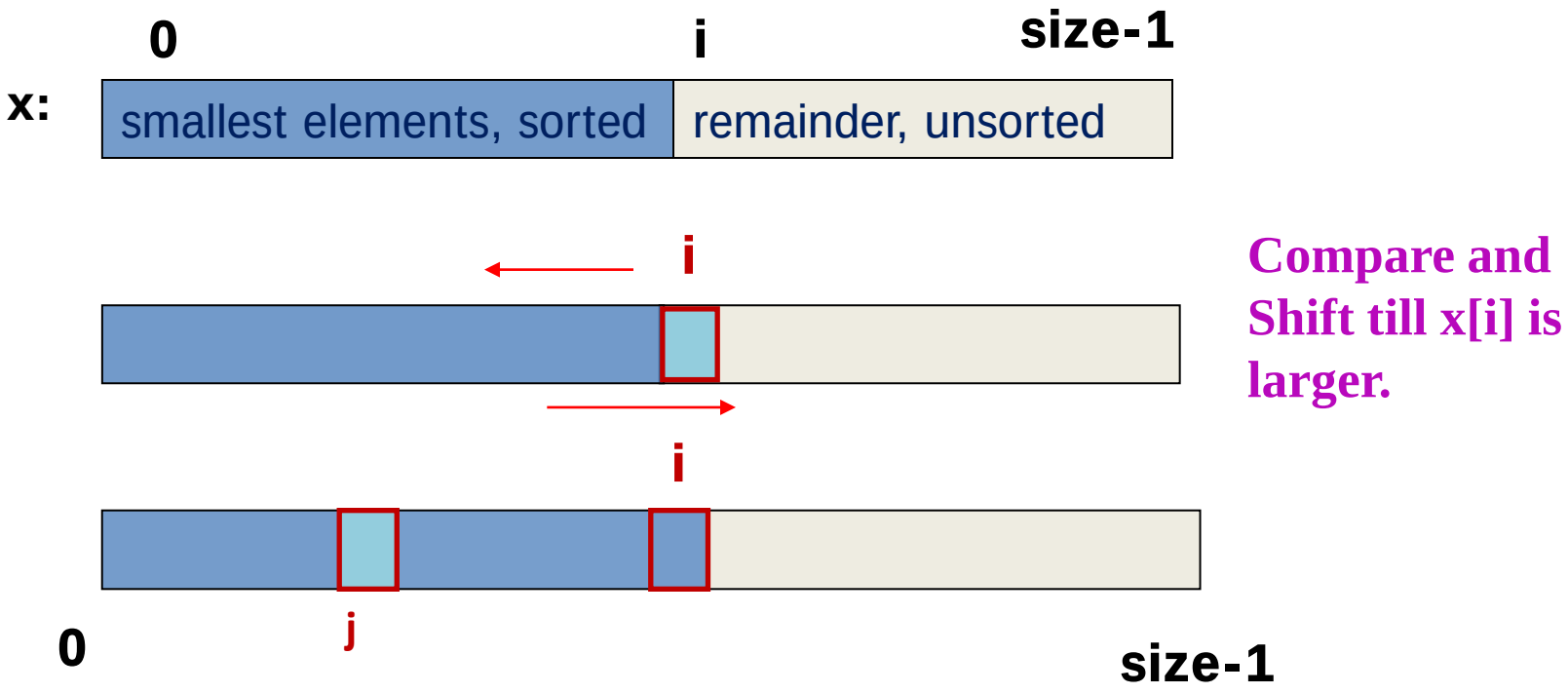
```
for i = 1 to length(A) j ← i while j > 0 and A[j-1] > A[j] swap A[j] and A[j-1] j ← j - 1 end while end for
```

Flow Chart: Sequential Search with LL



Insertion Sort

General situation :

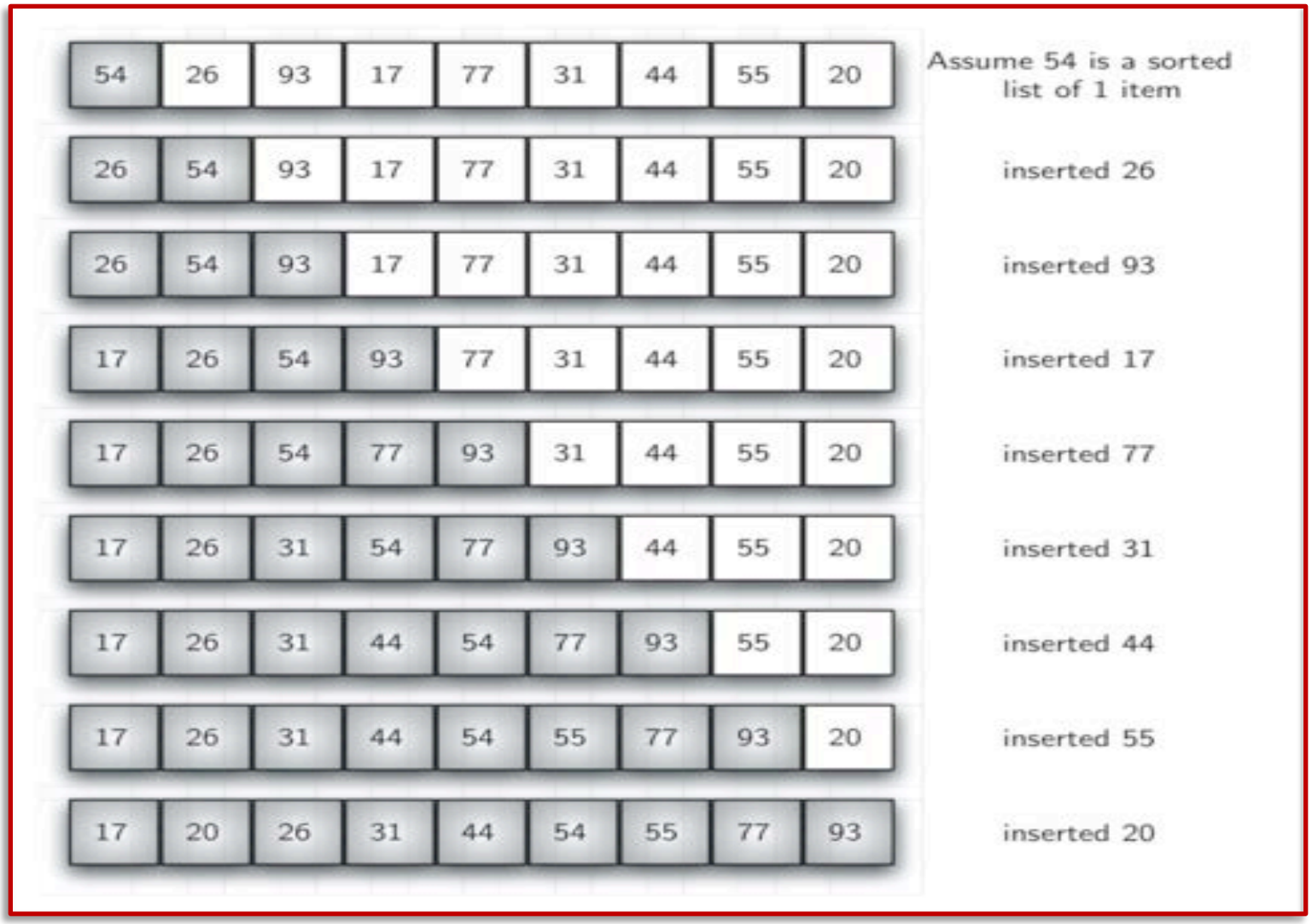


Compare and Shift till x[i] is larger.

Insersion Sort

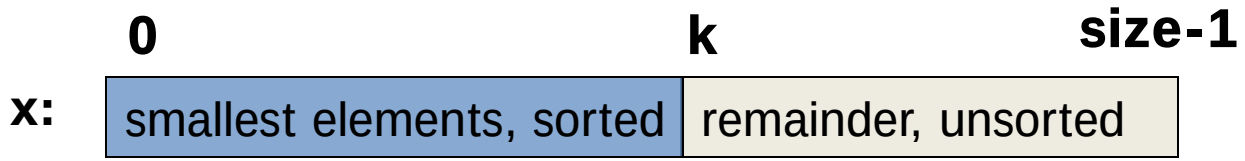
```
for i = 1 to length(A)
    j ← i
    while j > 0 and A[j-1] > A[j]
        swap A[j] and A[j-1]
        j ← j - 1
    end while
end for
```

Insertion Sort - Example



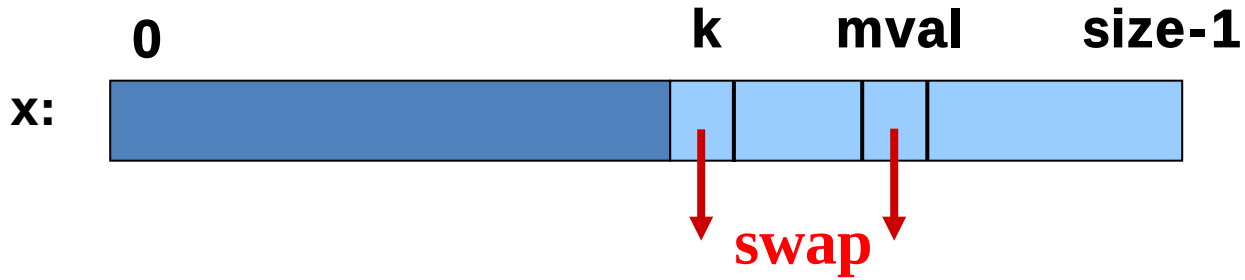
Selection Sort

General situation :



Steps :

- Find smallest element, mval, in x[k…size-1]
- Swap smallest element with x[k], then increase k.



Selection Sort (Algorithm)

Step 1 − Set MIN to location 0

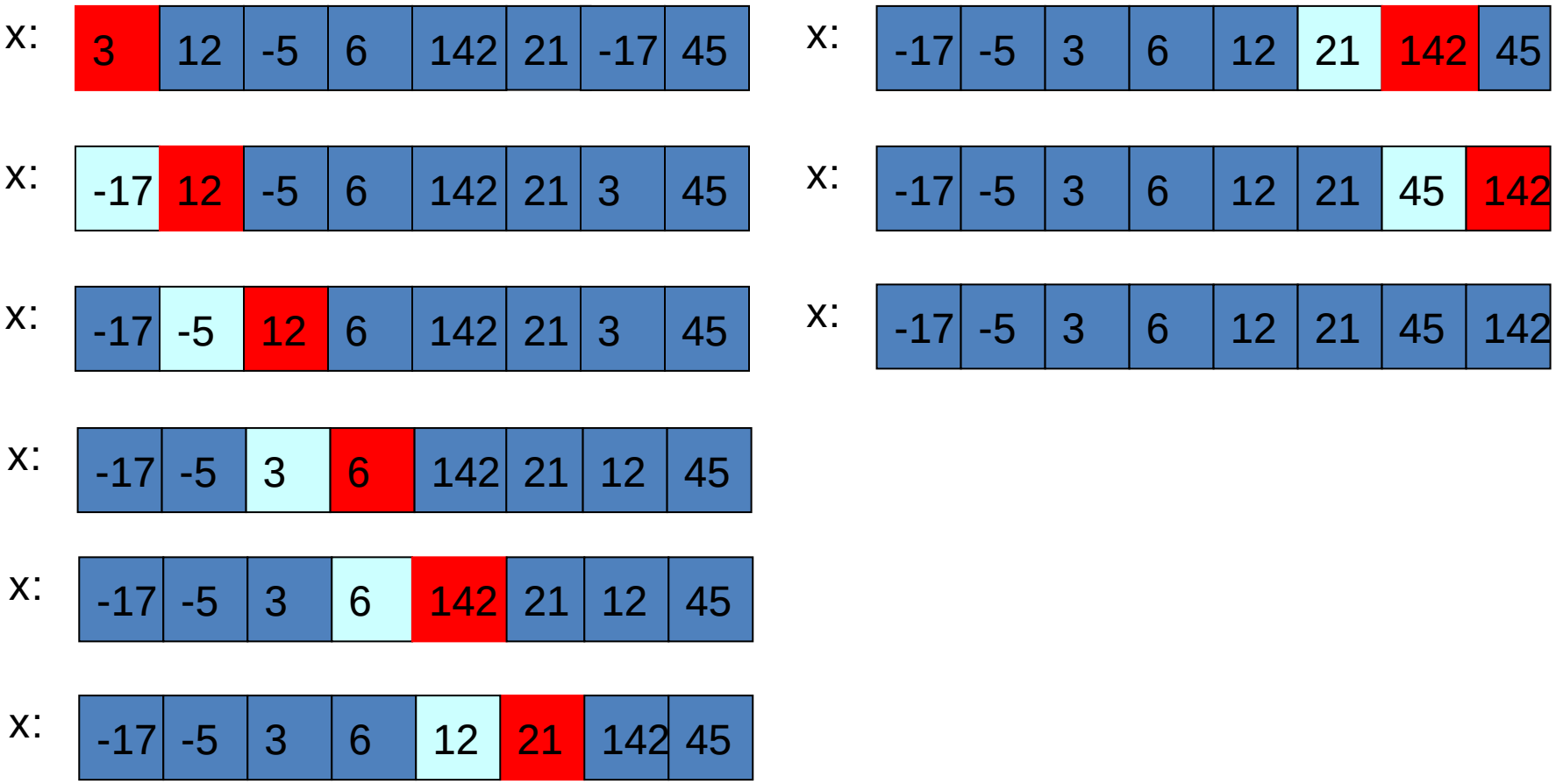
Step 2 − Search the minimum element in the list

Step 3 − Swap with value at location MIN

Step 4 − Increment MIN to point to next element

Step 5 − Repeat until list is sorted

Selection Sort - Example



Quick Sort (Algorithm)

```
algorithm quicksort(A, lo, hi)
    if lo < hi then
        p := partition(A, lo, hi)
        quicksort(A, lo, p)
        quicksort(A, p + 1, hi)
```

```
algorithm partition(A, lo, hi) is
    pivot := A[lo]
    i := lo - 1
    j := hi + 1
    loop forever
        do
            i := i + 1
        while A[i] < pivot

        do
            j := j - 1
        while A[j] > pivot

        if i >= j then
            return j

        swap A[i] with A[j]
```

Quick Sort - Example

Input: 45 -56 78 90 -3 -6 123 0 -3 45 69 68

45 -56 78 90 -3 -6 123 0 -3 45 69 68

-6 -56 -3 0 -3 45 123 90 78 45 69 68

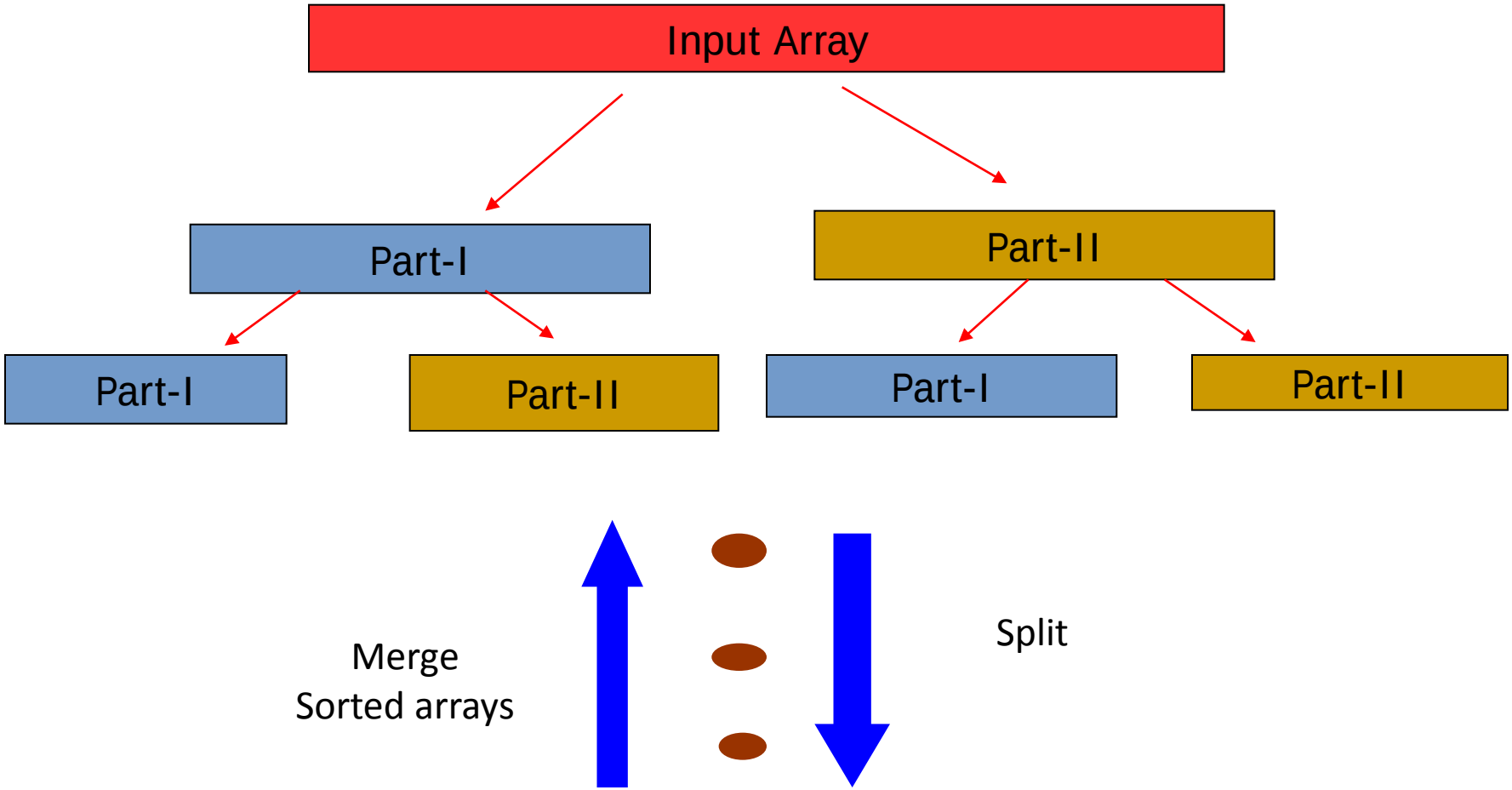
-56 -6 -3 0 -3 68 90 78 45 69 123

-3 0 -3 45 68 78 90 69

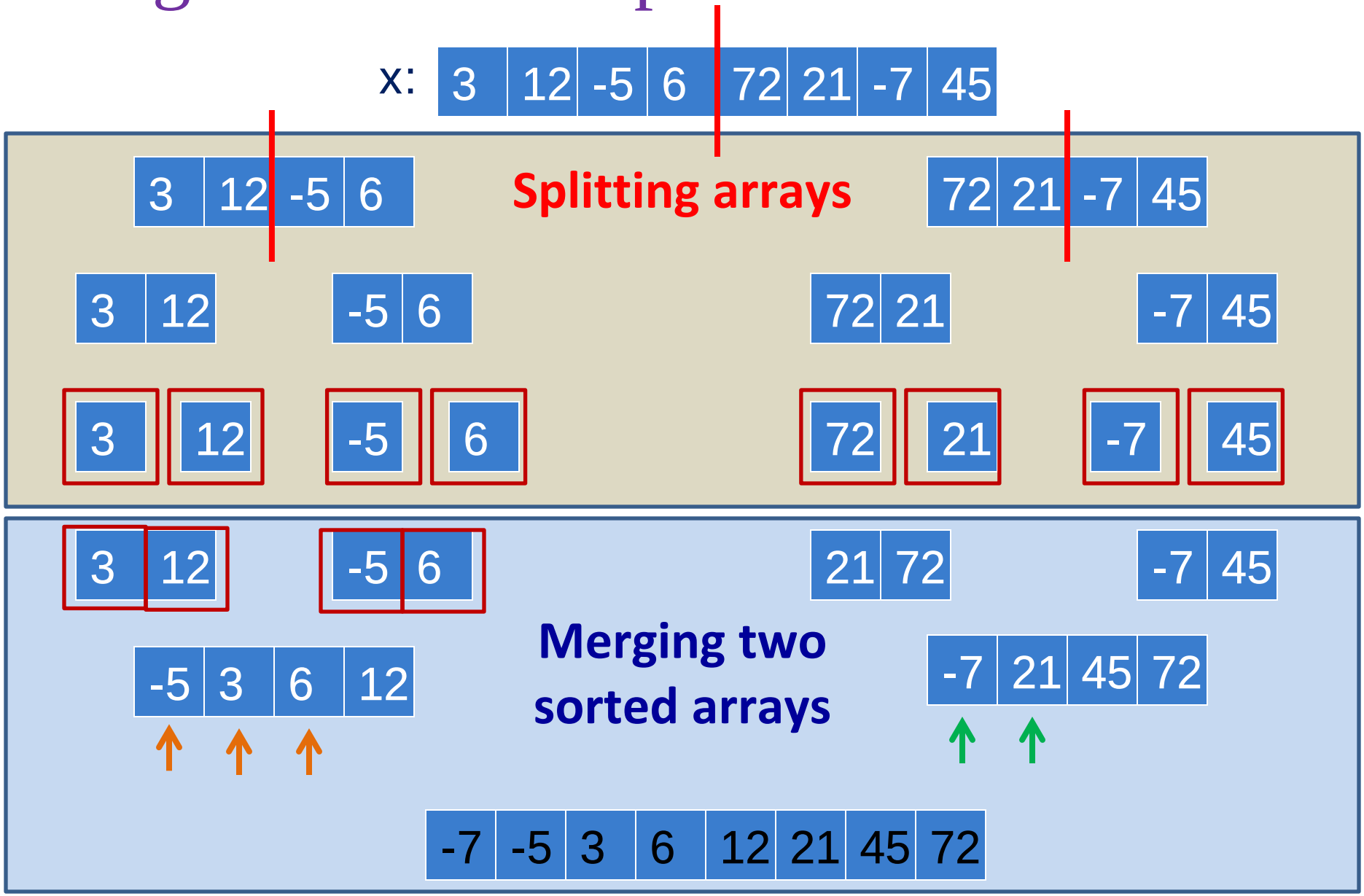
-3 0 69 78 90

Output: -56 -6 -3 -3 0 45 45 68 69 78 90 123

Merge Sort – How it Works?



Merge Sort – Example



Merge Sort (Agorithm)

```
procedure mergesort( var a as array )
  if ( n == 1 ) return a

  var l1 as array = a[0] ... a[n/2]
  var l2 as array = a[n/2+1] ... a[n]

  l1 = mergesort( l1 )
  l2 = mergesort( l2 )

  return merge( l1, l2 )
end procedure
```

Merge Sort (Algorithm cont..)

```
procedure merge( var a as array, var b as array )
  var c as array
  while ( a and b have elements )
    if ( a[0] > b[0] )
      add b[0] to the end of c
      remove b[0] from b
    else
      add a[0] to the end of c
      remove a[0] from a
    end if
  end while
    while ( a has elements )
    add a[0] to the end of c
    remove a[0] from a
  end while

  while ( b has elements )
    add b[0] to the end of c
    remove b[0] from b
  end while
  return c
end procedure
```