Sorting and Hashing

❖ 6.1 Sorting Methods:

1. Bubble sort
2. Selection sort
3. Quick sort(Partition Exchange sort)
4. Insertion sort
5. Merge sort
6. Radix sort

1. Explain Bubble Sort with algorithm.

- Bubble sort is easy to understand and simple sorting technique.
- During the first pass element 1 and 2 are compared and if they are out of order thenthey are interchanged. This process is repeated for elements 2 and 3 and so on.
- After the end of first pass the record with the largest value is placed at nth(last)position.

Algorithm:

Bubble_Sort(List,N)

Where, List-> Array of N Elements

N-> Size of array (Total No of Elements)

Step: 1 [Initialization]

i←0

Step: 2 while (i<N-1) repeat thru Step 7

Step: 3 j←0

Step: 4 while (j<N-i-1) repeat thru Step 6

Step: 5 if (List[j] > List [j+1])

(i) temp←List[j]

(ii) List[j]←List[j+1]

(iii) List[j+1]←temp

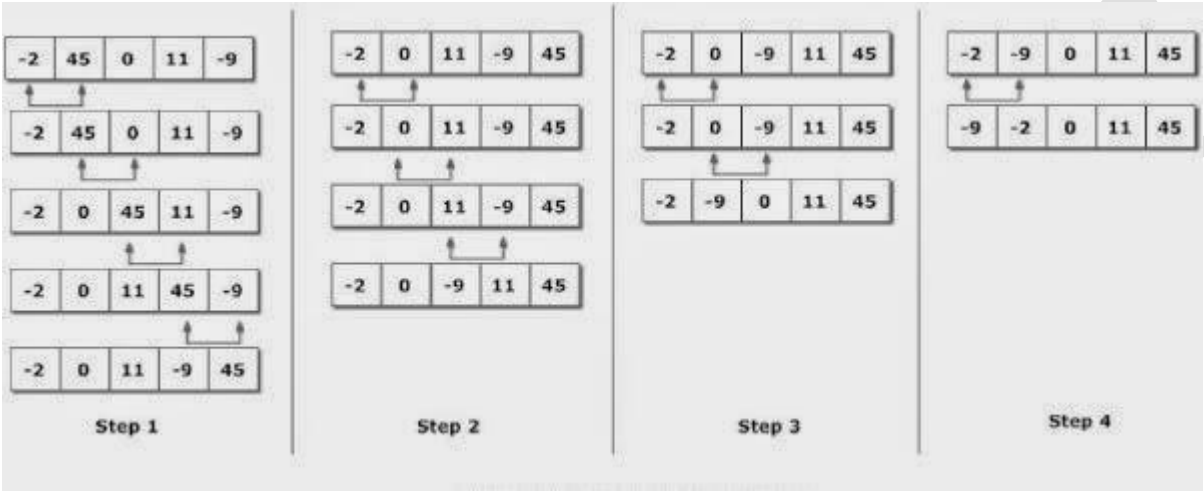
Step: 6 j←j+1

Step: 7 i←i+1

Step: 8 [Finished]

Exit.

Example:



2. Explain Selection Sort with algorithm.

- It starts from first element and searches the entire array until it find smallestelement. Then smallest value interchanges with the first element.
- Now select second element and searches for the second smallest element from thearray, if found then interchange with second element.
- So in this method, after pass 1 smallest value arranged at first position then afterpass 2 second minimum will arrange at second position and so on.
- This process continues until all the elements in the array are arranged in ascendingorder.

Algorithm Selection_Sort(List,N)

Where, List--> Array of N Elements

N-> Size of array (Total No of Elements)

Step: 1 [Initialization]

i←0

Step: 2 while (i <N-1) repeat thru Step 7

Step: 3 j←i+1

Step: 4 while (j<N) repeat thru Step 6

Step: 5 if (List[i] > List[j])

(i) temp←List[i]

(ii) List[i]←List[j]

(iii) List[j]←temp

Step: 6 j←j+1

Step: 7 i←i+1

Step: 8 [Finished]

Exit.

Example:

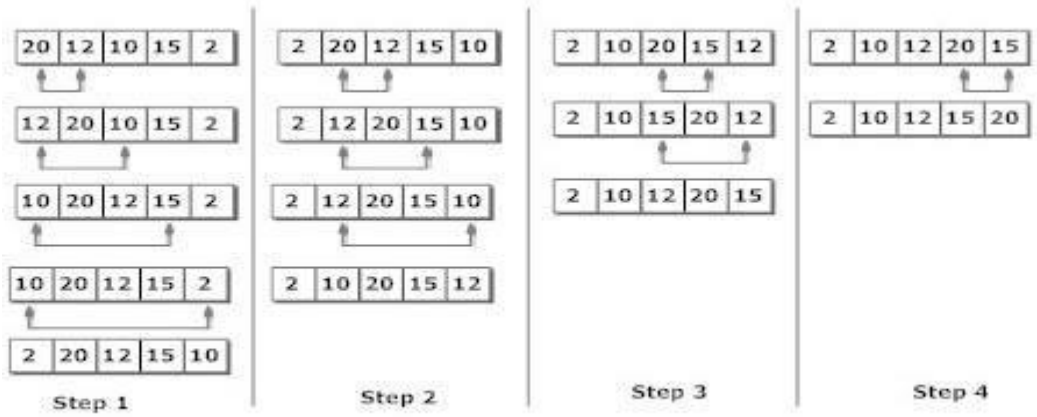
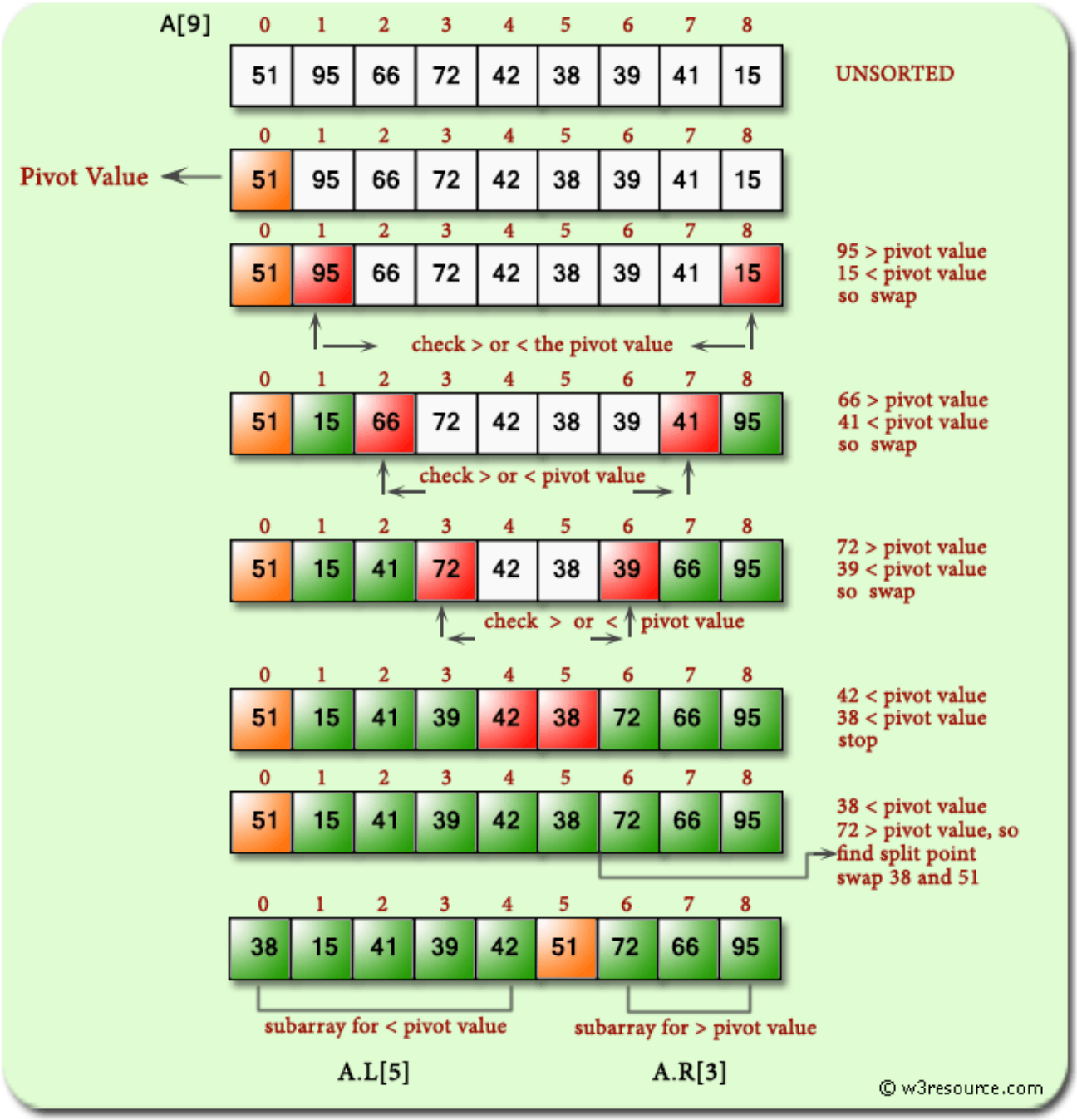


Figure: Selection Sort

3. Explain Quick Sort with algorithm.

- We pick the first element from the array and place it in it's proper position in the array such that all the elements preceding the element having smaller value and all the elements following the element having larger value.
- Thus the table is divided into two sub tables.
- The same process is then applied to each of the sub tables and repeated until all records are placed in their final position.
- **Example:**

Quick Sort



Algorithm

Quick_Sort(K,Start,End)

Where, K -> Array

Start-> Index of First ElementEnd->

Index of Last Element

Step: 1 If Start <End then

i←Start

j←End +1

Key←K[Start]

Step: 2 Repeat thru step 3 while I < j

Step: 3 i← i+1

while (K[i] < Key)

i++

j←j-1

While (K[j] > Key)

j - - if i< j

then

temp←K[i]

K[i]←K[j]

K[j]←temp

Step: 4 [Place the Key element to its Position]

temp←K[j]

K[j]←Key

Key←temp

Step: 5

Call Quick_Sort(K,Start,j-1)Call

Quick_Sort(K,j+1,End)

Step: 6 [Finished]

Exit.

4. Explain Insertion sort with algorithm.

- Suppose L is the list of n elements.
- The insertion sort technique scans the list L from L [1] to L[n].

- Inserting each element in to its proper position in the previously sorted sub list L[1], L [2]… L [i-1].

Algorithm

Insertion Sort (a,N)

Where, a -> Array

N-> Total no of elementsStep:1

Read N

Step: 2 Repeat thru step 2 for i=0,1,2,…N-1Read (a[i])

Step: 3 Repeat thru Step 6 for i= 1,2,…N-1

Step:4 index ← a[i]

j←i

Step: 5 Repeat while (j>0 and a[j-1]>index)

a[j]←a[j-1)

j← j-1

Step: 6 a[j] ←Index

Step :7 [Finished]

Exit.

Example:

We take an unsorted array for our example.

Insertion sort compares the first two elements.

It finds that both 14 and 33 are already in ascending order. For now, 14 is in sorted sub-list.

Insertion sort moves ahead and compares 33 with 27.

And finds that 33 is not in the correct position.

It swaps 33 with 27. It also checks with all the elements of sorted sub-list. Here we see that the sorted sub-list has only one element 14 and 27 is greater than 14. Hence, the sorted sub-list remains sorted after swapping.

By now we have 14 and 27 in the sorted sub-list. Next, it compares 33 with 10.

These values are not in a sorted order.

So we swap them.

However, swapping makes 27 and 10 unsorted.

Hence, we swap them too.

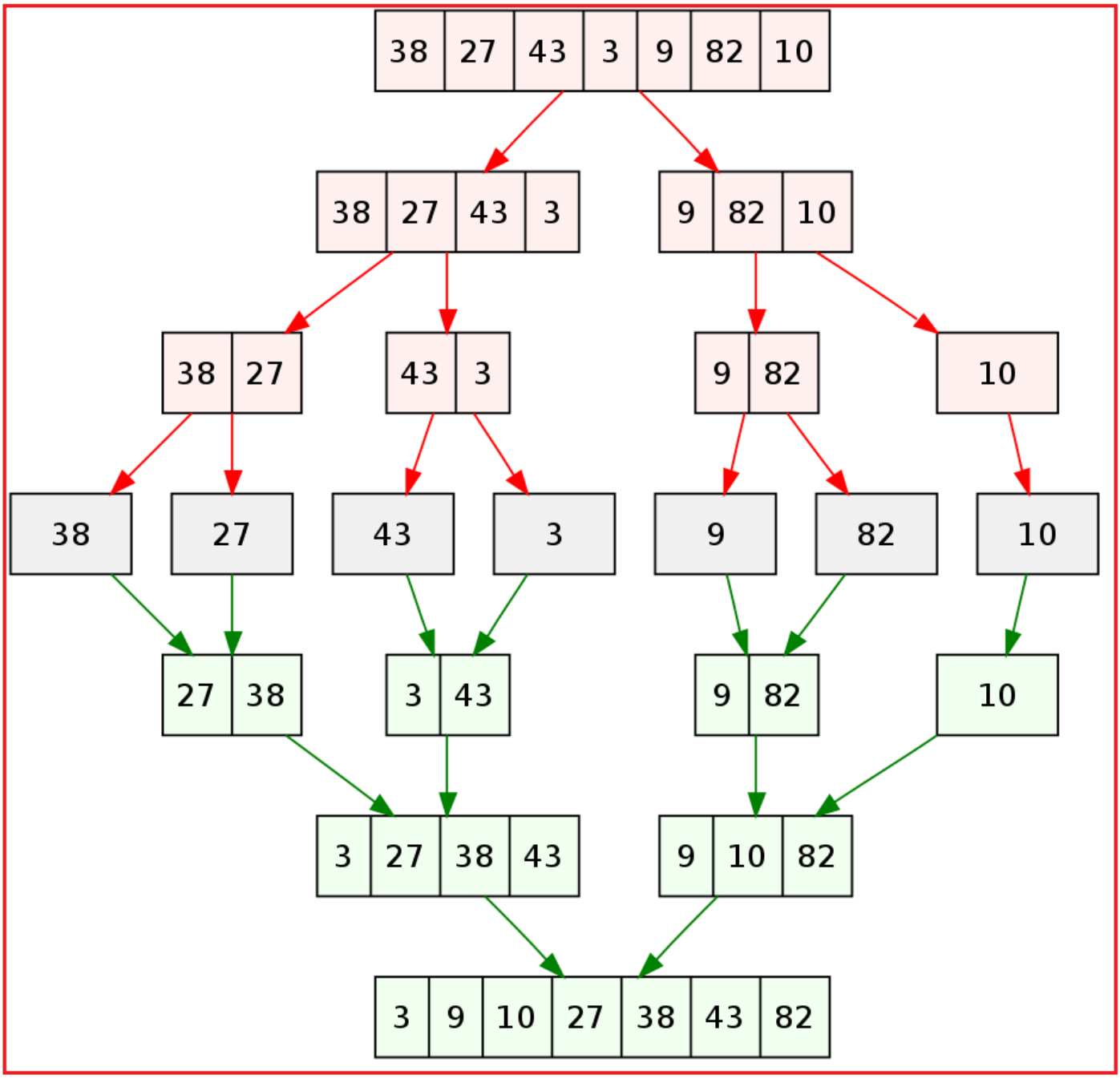
Again we find 14 and 10 in an unsorted order.

We swap them again. By the end of third iteration, we have a sorted sub-list of 4items.

This process goes on until all the unsorted values are covered in a sorted sub-list.

5. Explain Merge Sort with algorithm.

- Merge Sort is the process of combining two lists into third list.
- For example, Two sorted arrays are A and B.
- If we have to combine both arrays into another sorted array C, then a merge sort method is applied.
- In this method, a record is read from both arrays and compared it, the smaller record is inserted in to array C, then another record is read from same array and doingsame process.
- Whenever any one of the array is completed, the other remaining array records simply copied in to array C.
- **Example:**



Algorithm

Merge Sort (List1, n, List2, m, List)Where,

List1 ← First List of n elements

List2 ← Second List of m elements

List ← Merge elements are stored in this list

Step: 1 [Initialization]

i←0

j←0

k←0

Step: 2 while ((i<n) and (j<m)) repeat thru step3

Step 3: If (List1 [i] < List2 [j]) then

List[k] ←List1 [i]

k←k+1

i←i+1

Else If (List1 [i] > List2 [j]) then

List[k] ←List2 [j]

k←k+1j←j+1

Else

List[k] ←List1 [i]

k←k+1

i← i+1

j← j+1

Step: 4 if (i<n)

Repeat for x=i,i+1,…n-1

List[k] ← List1[x]

k←k+1

Step: 5 if (j < m)

Repeat for y=j,j+1,…m-1

List[k] ← List2[y]

k←k+1

Step: 6 [Finished]

Exit.

6. Explain Radix Sort with Example.

- We use ten pockets for the digits 0 to 9.
- Consider the following set of data:

  42 23 74 11 65 57 94 36 99 87 70 81 61
- During first pass we separate the unit digit of the number and place the number in toappropriate pocket according to its unit digit.
- For example the first number is 42 so we separate the unit digit of the number 42 which is 2 so we place the number in pocket 2. Same procedure is repeated for remaining numbers.
- After first pass the numbers in each pockets are as follow:

		61							
		81			94			87	
		70	11	42	23	74	65	36	57	99
Pocket:	0	1	2	3	4	5	6	7	8	9

- Now arrange the number according to their pocket. The numbers after firstpass are as follows:

  70 11 81 61 42 23 74 94 65 36 57 87 99
- During second pass we separate the next higher digit of the number andplace the number in to appropriate pocket according to its digit.
- After second pass the numbers in each pockets are as follow:

						65	74	87	99
		11	23	36	42	57	61	70	81	94
Pocket:	0	1	2	3	4	5	6	7	8	9

- Now arrange the number according to their pocket. The numbers aftersecond pass are as follows:
- 11 23 36 42 57 61 65 70 74 81 87 94 99

- The same process is repeated until all the elements are sorted.

❖ 6.2 Hashing Concepts

- Define Hashing: Hashing provides direct access of records from the file no matter where the record is in the file.
- This technique uses the hashing function say H which maps the Key to theparticular address of the record.
- Hashing function provides key to address transformation.
- Example: suppose there are 60 students in the classroom and we want to maintain their record. We have to store the record in an array. In this case the array index ranges from 0-59. We may use this index as the registration number of student. For example if a student has a registration number S15280531 then the record is storedat address 31.

❖ 6.3 Hash functions

- A hashing function H maps the Key space K into an address space.

✓ **Some of the most widely used hashing functions are :**

1) Division Method
2) The Mid square Method
3) The Folding Method
4) Multiplicative Hashing

1) Division Method

- Hashing function defined as: **H(x) = x mod m + 1**
- For example: H (35) = 35 mod 11 + 1 = 2 + 1 = 3
- The Division method generates a key value which belongs to the set {1, 2… m}

2) The Mid square Method

- A key is multiplied by itself and the address is obtained by selecting an appropriate number of digits from the middle of the square.
- For example: consider a six digit key 123456.
- Squaring this key result in the value 5241383936. if a three digit address is

required then position 5 to 7 could be chosen which gives 138.

3) The Folding Method

- A key is partitioned into parts. The length of each part is similar to thelength of the address required. These parts are added together and final carry is ignored to produce the address.
- For example if a key 35678943 is transformed into 2 digit addressthen we have:
  35 + 67 + 89 + 43 = 234 = 34.
- This method is known as **fold shifting** method

4) Multiplicative Hashing

- H(x)= [m (cx mod 1 + 1)] + 1
- Where, x=Key
- c= Constant(0<c<1)
- m= Hash Table Size