Unit 6

Sorting and Hashing

Sorting

- **Sorting** is process of arranging the elements in a particular order. The order may be ascending order or descending order.
- The advantage of sorting is effective data accessing.

Types of sorting

1. Internal sorting
2. External sorting

- **Internal sorting**: If all the elements to be sorted are in the main memory then such a sorting is called **internal sorting.**
- **External sorting:** If some of the elements to be sorted are in the secondary storage or disk such a sorting is called **External sorting**.

Types of sorting techniques:

1. Bubble sort (Exchange sort or Sinking sort).
2. Selection sort.
3. Insertion sort.
4. Quick sort.
5. Merge sort.
6. Heap sort.
7. Shell sort.
8. Radix sort.
9. Couting sort

Selection Sort

- In first pass, First find the smallest value in the array and swap with the first element. Then, in second pass find the second smallest value in the array and swap with the second element. Repeat this procedure until the entire array is sorted

Selection Sort

SELECTION_SORT(K,N):- This algorithm is used to sort all N elements in ascending order. N is integer type variable indicate there are N elements in vector K. The variable PASS denoted the pass index and the position of the first elements in the vector. MIN_INDEX denoted the position of the smallest element. I denoted index elements.

1. [loop on pass index]

Repeat thru step 4 for pass = 1,2,…,N-1

2. [Initialize minimum index]

MIN_INDEX <- PASS

3. [Make a pass and obtain element with smallest value]

Repeat for I = PASS+1, PASS+2,….., N

if K[I] < K[MIN_INDEX]

then MIN_INDEX <- I

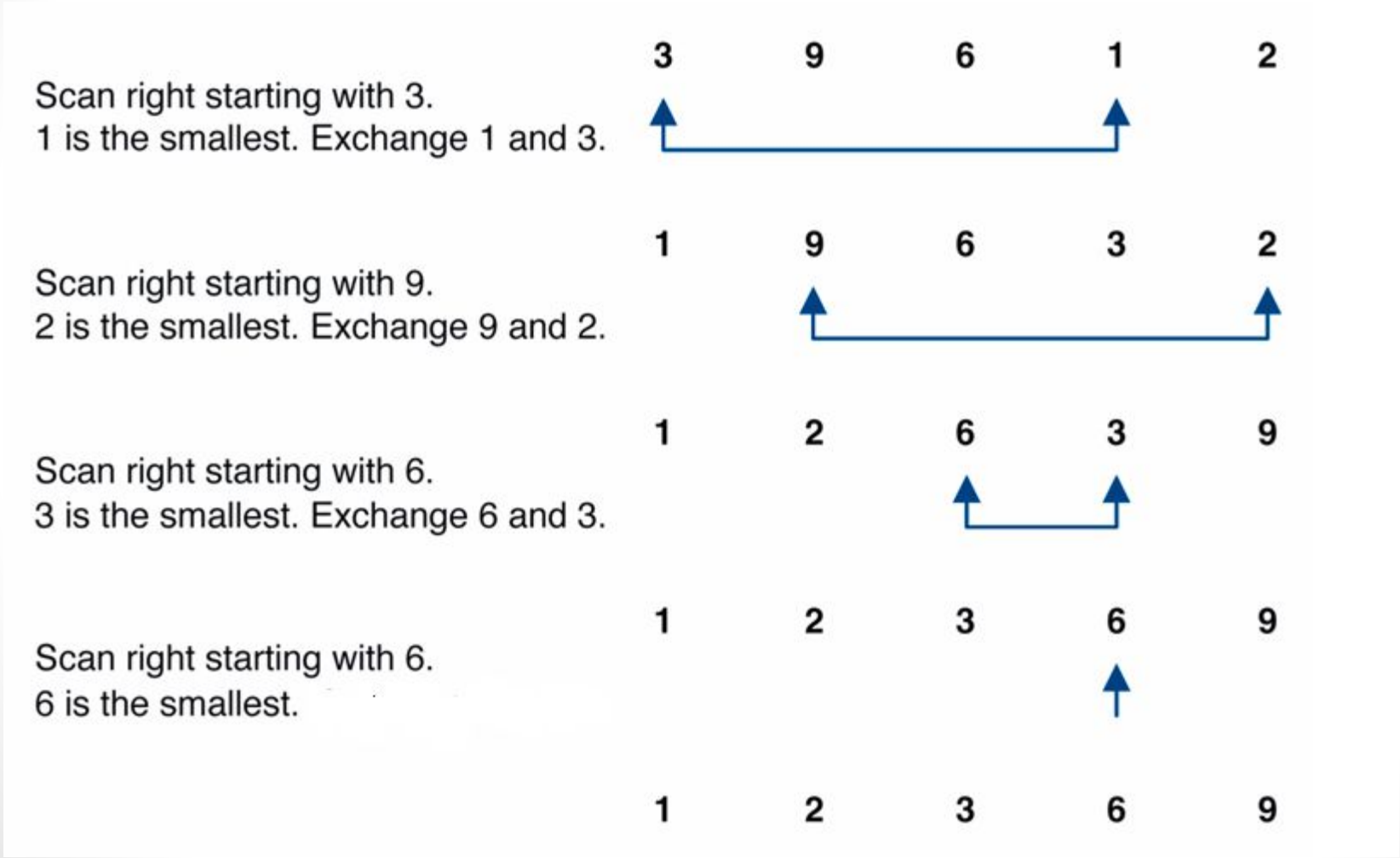
4. [Exchange elements]

if MIN_INDEX ≠ PASS

then K[PASS] < ____ > K[MIN_INDEX]

5.[finish] Exit

Selection Sort



Time Complexity

- BestCase :$O(n^2)$
- AverageCase: $O(n^2)$
- WortCase: $O(n^2)$

Selection Sort

- Simple and easy to implement.
- It can be used for small data sets.

Bubble Sort

- Bubble sort is a very simple method that sorts the array elements by repeatedly moving the largest element to the highest index position of the array segment (in case of arranging elements in ascending order).
- In bubble sorting, consecutive adjacent pairs of elements in the array are compared with each other.
- If the element at the lower index is greater than the element at the higher index, the two elements are interchanged so that the element is placed before the bigger one.
- This process will continue till the list of unsorted elements exhausts.
- This procedure of sorting is called bubble sorting because elements ‘bubble’ to the top of the list. Note that at the end of the first pass, the largest element in the list will be placed at its proper position

Bubble Sort

BUBBLE_SORT(K,N):- This algorithm is used to sort all N elements in ascending order. N is integer type variable indicate there are N elements in vector K. The variable PASS and LAST denote the pass counter and position of the last element, respectively. EXCHS is used to count the number of exchange made on any pass. I denoted index elements.

1. [Initialize]

LAST <- N

2. [loop on pass index]

Repeat thru step 5 for pass = 1,2,…,N-1

3. [Initialize exchanges counter for this pass]

EXCHS <- 0

4. [Perform pair wise comparisons on unsorted elements]

Repeat for I = 1, 2,….., LAST - 1

if K[I] > K[I + 1]

then K[I] < > K[I + 1]

EXCHS < - EXCHS + 1

5. [Were any exchanges made on this pass ?]

if EXCHS = 0

then Return

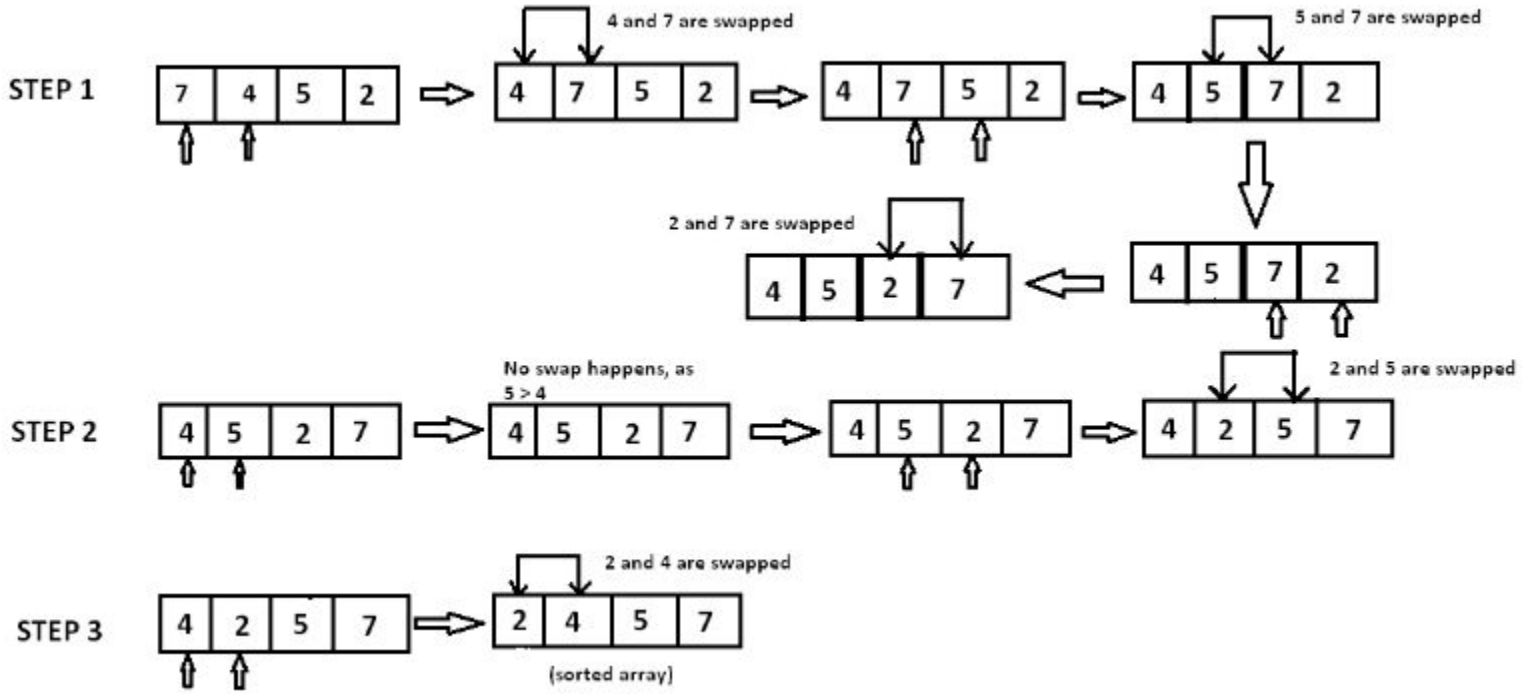
else

LAST < - LAST - 1

6.**[finish]**

Exit

Bubble Sort


STEP 1
7 4 5 2
4 and 7 are swapped
4 7 5 2
4 7 5 2
5 and 7 are swapped
4 5 7 2
2 and 7 are swapped
4 5 2 7
4 5 7 2
STEP 2
4 5 2 7
No swap happens, as 5 > 4
4 5 2 7
4 5 2 7
2 and 5 are swapped
4 2 5 7
STEP 3
4 2 5 7
2 and 4 are swapped
2 4 5 7
(sorted array)

Time Complexity

- BestCase :O(n)
- AverageCase: $O(n^2)$
- WortCase: $O(n^2)$

Insertion Sort

- **Insertion sort** is a simple sorting algorithm that works similar to the way you sort playing cards in your hands.
- The array is virtually split into a sorted and an unsorted part. Values from the unsorted part are picked and placed at the correct position in the sorted part.
- This algorithm sorts an array of items by repeatedly taking an element from the unsorted portion of the array and inserting it into its correct position in the sorted portion of the array.
- The sorting algorithm will proceed until there are elements in the unsorted set.
-

Insertion Sort

- Suppose there are n elements in the array. Initially, the element with index 0 is in the sorted set. Rest of the elements are in the unsorted set.
- The first element of the unsorted partition is known as key.
- Compare the key element with the previous elements. If the previous elements are greater than the key element, then you move the previous element to the next position.

Insertion Sort

INSERTION_SORT(K,N):- This algorithm is used to sort all N elements in ascending order. N is integer type variable indicate there are N elements in vector K. The variable TEMP denotes the temporary variable. I and J denotes index elements.

1. [loop on index]

Repeat thru step 4 for I = 2,…,N-1

2. [Initialize variable]

Key <- K[I]

J <- I -1

3. [Perform comparisons]

Repeat while J>=1 && Key < K[J]

K[J+1] <- K[J]

J <- J – 1

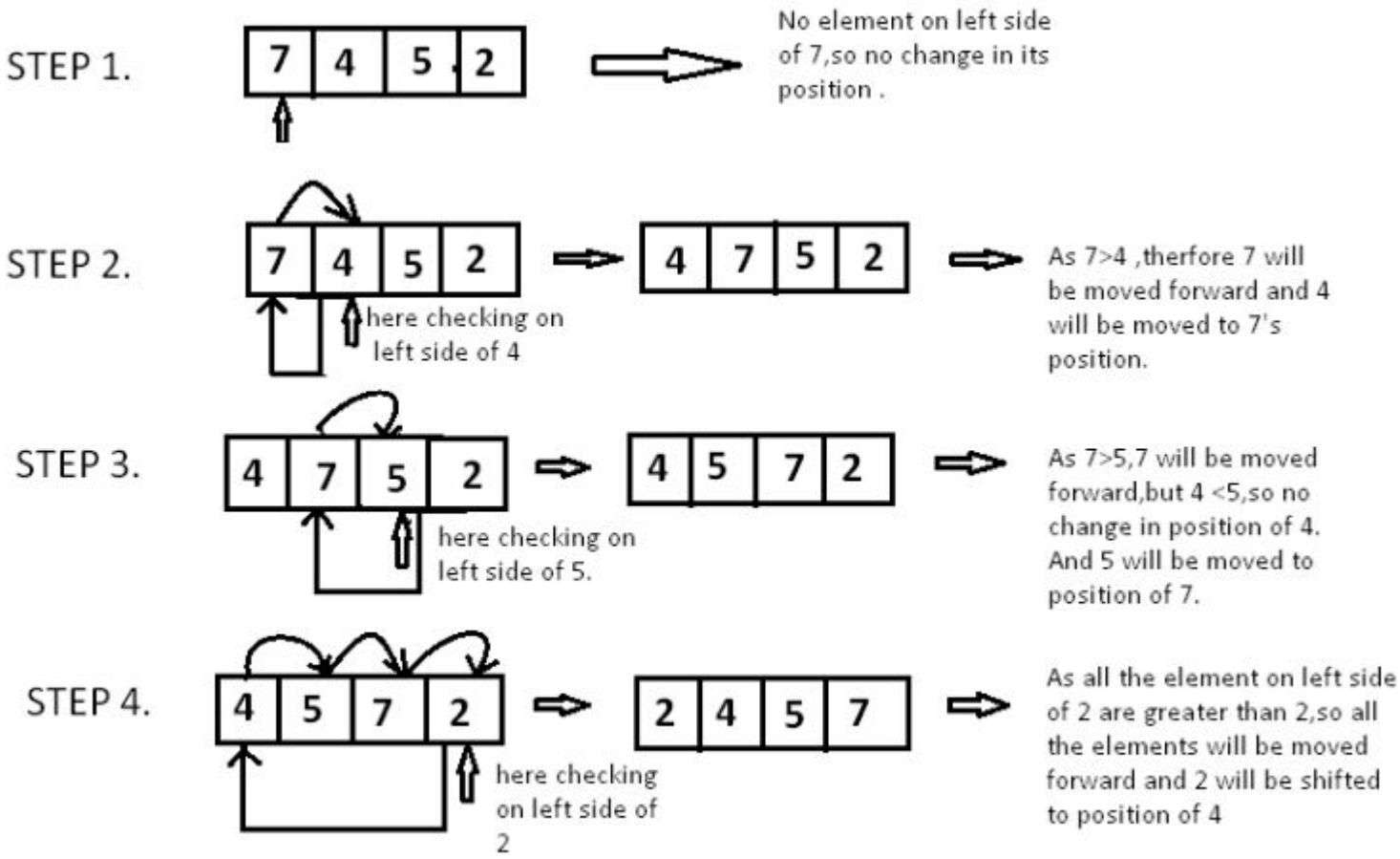
4. [Place value in proper position]

K[J+1] <- Key

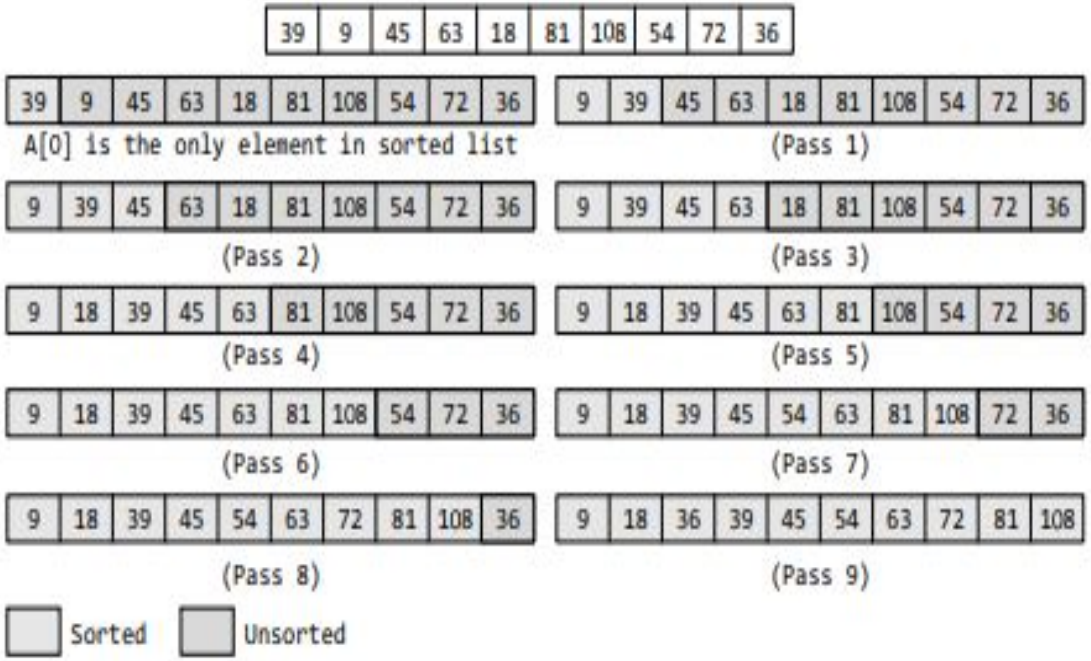
5. [Finish]

Exit

Insertion Sort



Insertion Sort



Insertion Sort

- Easy to implement and efficient to use on small sets of data.
- It can be efficiently implements on data sets that are already almost sorted.
- It performs better than selection sort.
- It requires less memory space.

Time Comlexity

- BestCase :O(n)
- AverageCase: $O(n^2)$
- WortCase: $O(n^2)$

Radix Sort

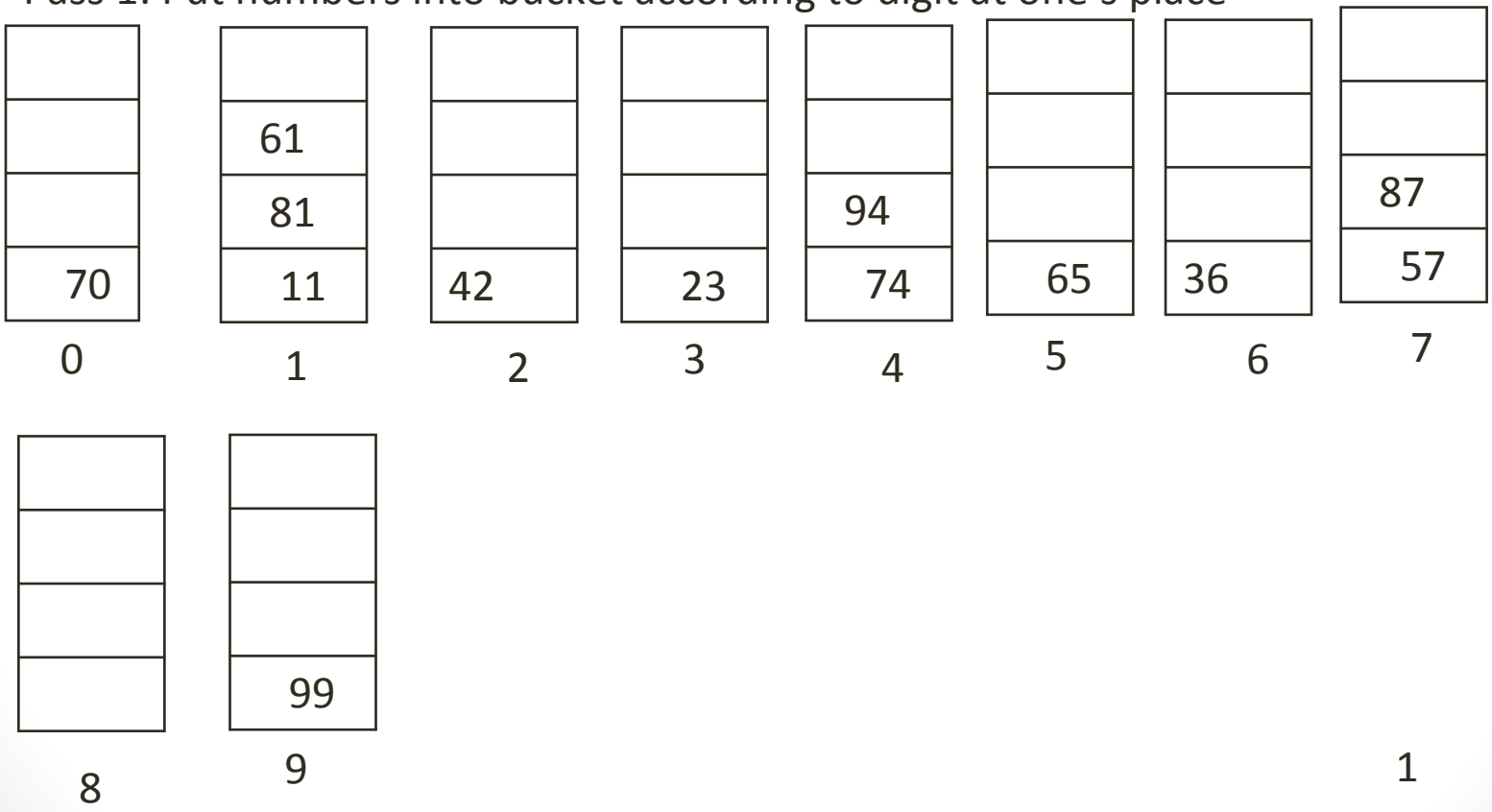
- Radix sort is a linear sorting algorithm for integers and uses the concept of sorting names in alphabetical order.
- When we have a list of sorted names, the radix is 26 (or 26 buckets) because there are 26 letters in the English alphabet. So radix sort is also known as bucket sort.
- When radix sort is used on integers, sorting is done on each of the digits in the number.
- The sorting procedure proceeds by sorting the least significant to the most significant digit.
- While sorting the numbers, we have ten buckets, each for one digit (0, 1, 2, …, 9) and the number of passes will depend on the length of the number having maximum number of digts.

Algorithm

1. Repeat thrugh step 6 for each digit in the key.
2. Initialize the pockes
3. Repeat thru step 5 until the end of the linked list.
4. Obtain the next digit of the key
5. Insert the record in the appropriate pocket
6. Combine the pockets to form a new linked list.

Radix Sort

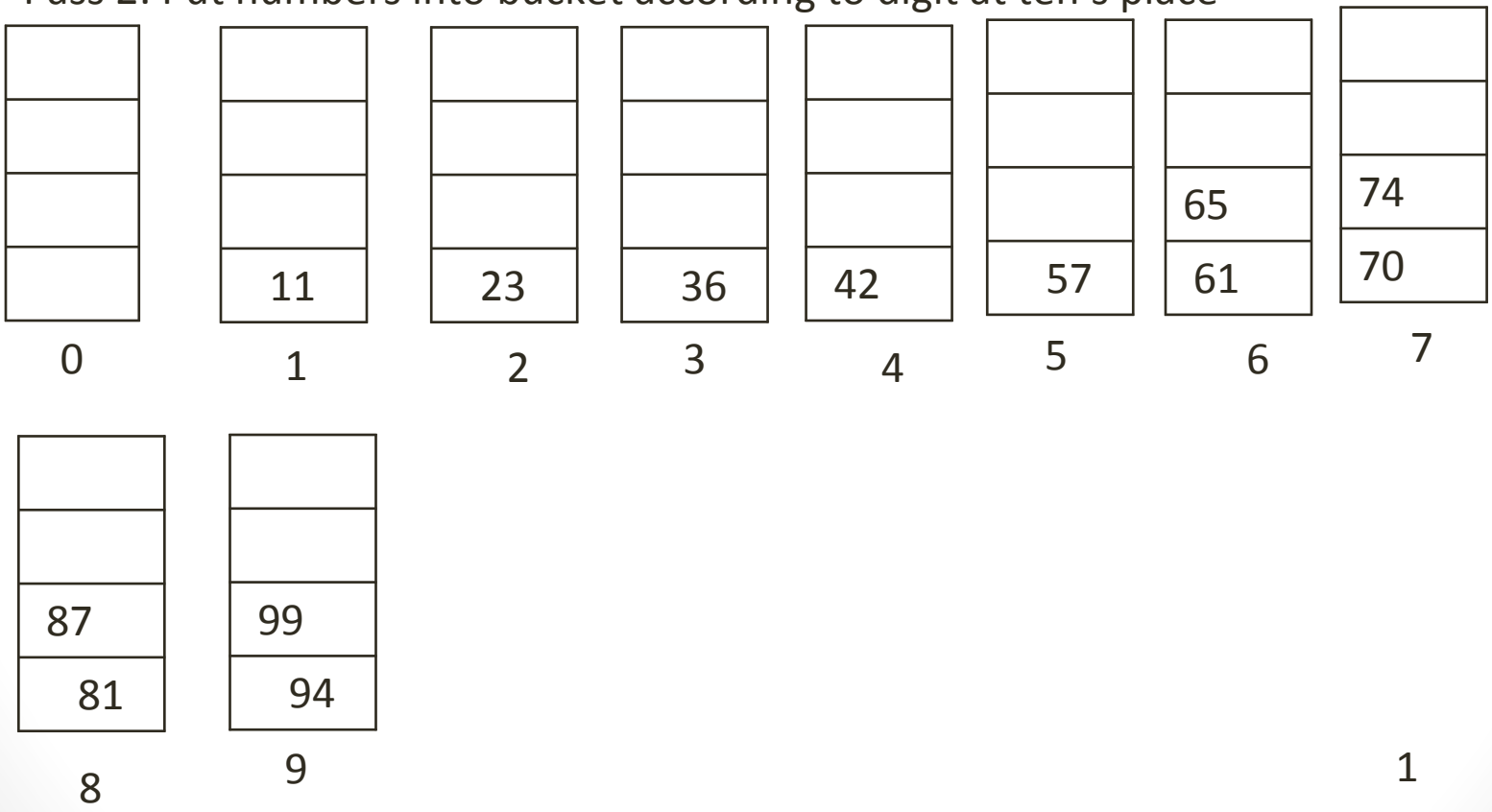
- 42,23,74,11,65,57,94,36,99,87,70,81,61
- Pass 1: Put numbers into bucket according to digit at one's place



Now combine the contents of bucket in fifo order .
70,11,81,61,42,23,74,94,65,36,57,87,99

Radix Sort

- 70,11,81,61,42,23,74,94,65,36,57,87,99
- Pass 2: Put numbers into bucket according to digit at ten's place



Now combine the contents of bucket in FIFO order.
11,23,36,42,57,61,65,70,74,81,87,94,99 .