**Algorithm of Radix Sort**

Step 1:Find the largest number in ARR as LARGE

Step 2: [INITIALIZE] SET NOP = Number of digits

in LARGE

Step 3: SET PASS =0

Step 4: Repeat Step 5 while PASS <= NOP-1

Step 5: SET I = 0 and INITIALIZE buckets

Step 6:Repeat Steps 7 to 9 while I<n-1< li=""></n-1<>

Step 7: SET DIGIT = digit at PASSth place in A[I]

Step 8: Add A[I] to the bucket numbered DIGIT

Step 9: INCREMENT bucket count for bucket

numbered DIGIT

[END OF LOOP]

Step 10: Collect the numbers in the bucket

[END OF LOOP]

Step 11: END

**Algorithm of Shell Sort**

Shell\_Sort(Arr, n)

Step 1: SET FLAG = 1, GAP\_SIZE = N

Step 2: Repeat Steps 3 to 6 while FLAG = 1 OR GAP\_SIZE > 1

Step 3:SET FLAG = 0

Step 4:SET GAP\_SIZE = (GAP\_SIZE + 1) / 2

Step 5:Repeat Step 6 for I = 0 to I < (N -GAP\_SIZE)

Step 6:IF Arr[I + GAP\_SIZE] > Arr[I]

SWAP Arr[I + GAP\_SIZE], Arr[I]

SET FLAG = 0

Step 7: END

**General Algorithm**

**The general algorithm for Quicksort is given below.**

|  |
| --- |
| quicksort(A, low, high)  begin  Declare array A[N] to be sorted      low = 1st element; high = last element; pivot  if(low < high)      begin      pivot = partition (A,low,high);      quicksort(A,low,pivot-1)      quicksort(A,pivot+1,high)      End  end |

.

|  |
| --- |
| procedure partition (arr[], low, high)  begin      // pivot (Element to be placed at right position)      pivot = arr[high];       i = (low - 1)  // Index of smaller element      for j = low to high      {          if (arr[j] <= pivot)          {              i++;    // increment index of smaller element              swap arr[i] and arr[j]          }      }      swap arr[i + 1] and arr[high])      return (i + 1)  end procedure |
|  |

Algorithm for linked list