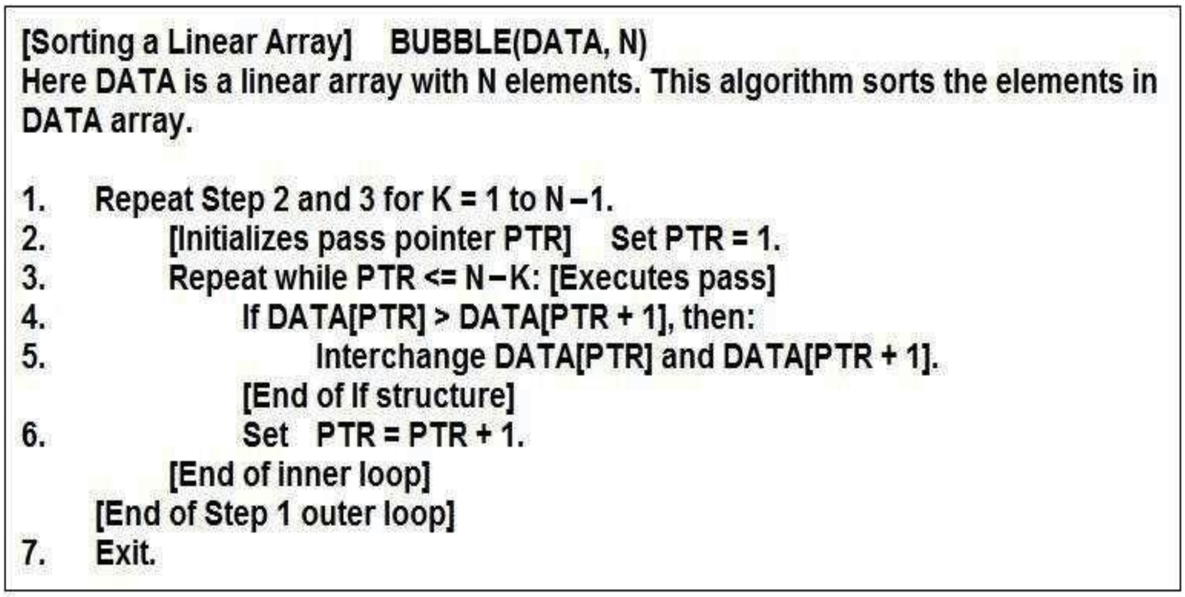
Bubble Sort

Bubble sort is the simplest sorting algorithm that works by repeatedly swapping the adjacent elements if they are in the wrong order. This algorithm is not suitable for large data sets as its average and worst-case time complexity is quite high.

Bubble Sort Algorithm

In Bubble Sort algorithm,

- traverse from left and compare adjacent elements and the higher one is placed at right side.
- In this way, the largest element is moved to the rightmost end at first.
- This process is then continued to find the second largest and place it and so on until the data is sorted.



```
[Sorting a Linear Array]    BUBBLE(DATA, N)
Here DATA is a linear array with N elements. This algorithm sorts the elements in
DATA array.

1.    Repeat Step 2 and 3 for K = 1 to N – 1.
2.          [Initializes pass pointer PTR]    Set PTR = 1.
3.          Repeat while PTR <= N – K: [Executes pass]
4.                If DATA[PTR] > DATA[PTR + 1], then:
5.                      Interchange DATA[PTR] and DATA[PTR + 1].
                  [End of If structure]
6.                Set   PTR = PTR + 1.
            [End of inner loop]
      [End of Step 1 outer loop]
7.    Exit.
```

Bubble Sort ‘C’ Program:

```
#include <stdio.h>

int main()

{

   int  n, j, i, swap;

   printf("Enter number of elements\n");

   scanf("%d", &n);

   int array[n];

   printf("Enter %d integers\n", n);
```

```
    for (i= 0; i < n; i++)
    {
       scanf("%d", &array[i]);
    }
    for (i = 0 ; i < n - 1; i++)
    {
       for (j = 0 ; j < n - i- 1; j++)
       {
          if (array[j] > array[j+1])
          {
             swap       = array[j];
             array[j]   = array[j+1];
             array[j+1] = swap;
          }
       }
    }

    printf("Sorted list in ascending order:\n");
     for (i = 0; i < n; i++)
       printf("%d\n", array[i]);
    return 0;
 }
```

Selection Sort

Selection sort is a simple and efficient sorting algorithm that works by repeatedly selecting the smallest (or largest) element from the unsorted portion of the list and moving it to the sorted portion of the list.

The algorithm repeatedly selects the smallest (or largest) element from the unsorted portion of the list and swaps it with the first element of the unsorted part. This process is repeated for the remaining unsorted portion until the entire list is sorted.

Selection Sort Algorithm

EBOOK: 9:10

Selection Sort 'C' Code:

```
#include <stdio.h>
// Function to perform selection sort
void selectionSort(int arr[], int n) {
    int i, j, min_idx;
    // One by one move boundary of unsorted subarray
    for (i = 0; i < n-1; i++) {
        // Find the minimum element in unsorted array
        min_idx = i;
        for (j = i+1; j < n; j++)
            if (arr[j] < arr[min_idx])
                min_idx = j;

        // Swap the found minimum element with the first element
        int temp = arr[min_idx];
        arr[min_idx] = arr[i];
        arr[i] = temp;
    }
}

// Function to print an array
void printArray(int arr[], int size) {
    int i;
    for (i=0; i < size; i++)
        printf("%d ", arr[i]);
    printf("\n");
}

int main() {
    int arr[] = {64, 25, 12, 22, 11};
    int n = sizeof(arr)/sizeof(arr[0]);
    printf("Original array: ");
    printArray(arr, n);
    selectionSort(arr, n);
    printf("Sorted array: ");
    printArray(arr, n);
    return 0;
}
```