**TITLE: SORTING**

Sorting is the process of placing data in ascending or descending order based on a linear connection between the data pieces. Different kinds of sorting are employed. Among them are:

**Bubble Sorting: Bubble Sort** is the simplest [sorting algorithm](https://www.geeksforgeeks.org/sorting-algorithms/) 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.

Algorithm for Bubble sort:

Input: n, list[n]

Func bubbleSort(n,list)

for all elements of list:

if list[i]>list[i+1] then:

swap (list[i],list[i+1])

return list

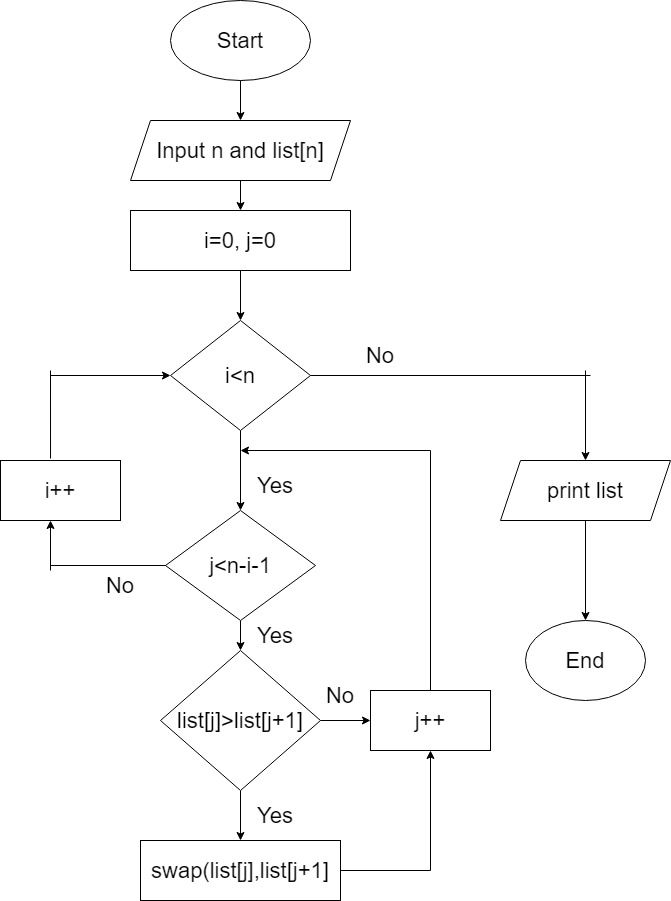
****

Figure: Bubble Sort

**WAP to implement bubble sort.**

#include <stdio.h>

void bubbleSort(int \*arr,int size){

int temp;

for (int i=0;i<size-1;i++){

for (int j=0;j<size -1 -i;j++){

if (arr[j] > arr[j+1]){

temp = arr[j+1];

arr[j+1] = arr[j];

arr[j] = temp;

}

}

}

}

void printarr(int \*arr,int size){

for (int i=0;i<size;i++){

printf("%d ",arr[i]);

}

}

int main(){

int size = 5;

int arr[size] = {1,5,4,7,2};

printf("Before Sorting\n");

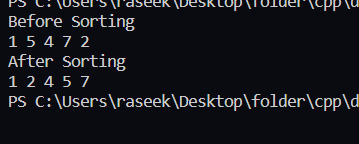
printarr(&arr[0], size);

bubbleSort(&arr[0],size);

printf("\nAfter Sorting\n");

printarr(&arr[0],size);

}



**Selection Sort:** The selection sort method sorts an array by repeatedly selecting the least element from the unsorted portion and inserting it at the beginning (while taking into account ascending order). In a given array, the method maintains two subarrays.

Algorithm for selection sort:

1. Set MIN location to 0
2. Search the minimum value in the list
3. Swap the minimum value with location MIN
4. Increment MIN to point to next element
5. Repeat until the list is sorted

Pseudocode:

Input: n,list[n]

Func selectionsort(n,list[n])

for i=0 to n-1 do:

min=i

for j=i+1 to n-1:

if list[j]<list[min] then

min=j

end if

end for

if indexMin!=I then

swap (list[min],list[i])

end if

end for

end selectionsort

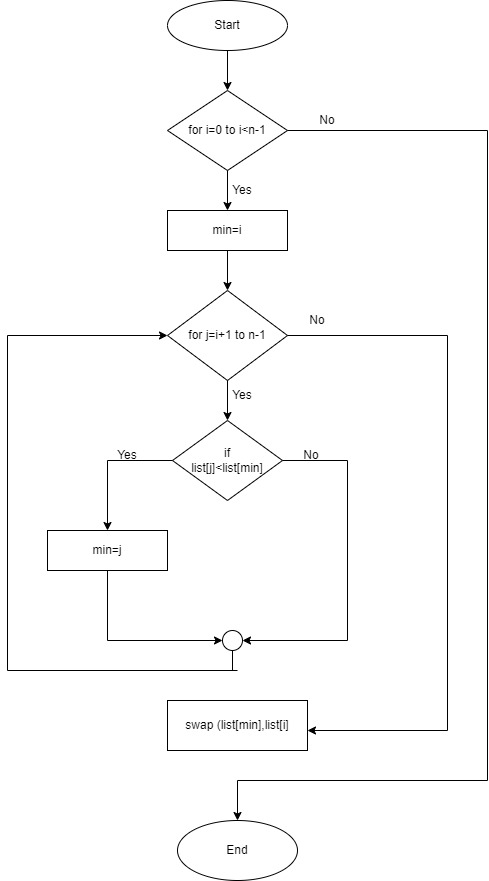


Figure: Selection sort

**WAP to implement selection sort.**

#include<stdio.h>

void printarr(int \*arr,int size){

for (int i=0;i<size;i++){

printf("%d ",arr[i]);

}

}

int main(){

int size=10,min=0,temp;

int arr[size] = {23,10,20,3,45,76,67,24,2,11};

printf("Bofore sorting:\n");

printarr(&arr[0],size);

for (int i=0;i<size-1;i++){

min = i;

for (int j=i+1;j<size;j++){

if (arr[j] < arr[min]){

min = j;

}

}

if (min!=i){

temp = arr[min];

arr[min] = arr[i];

arr[i] = temp;

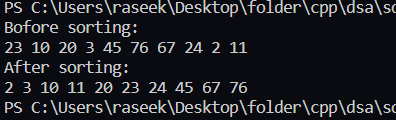
}

}

printf("\nAfter sorting:\n");

printarr(&arr[0],size);

}



**Insertion sort:** Insertion sort is an in-place sorting algorithm.  It sorts without using any additional data structures. Insertion sort is a sorting algorithm that, after each iteration, inserts an unsorted element in the proper location. Similar to how we sort cards in our hand in a card game, insertion sort works similarly. Since we already know that the first card is sorted, we choose an unsorted card.

**Algorithm for insertion sort:**

1. If the element is the first element, assume that it is already sorted. Return 1.
2. Pick the next element, and store it separately in a **key.**
3. Now, compare the **key** with all elements in the sorted array.
4. If the element in the sorted array is smaller than the current element, then move to the next element. Else, shift greater elements in the array towards the right.
5. Insert the value.
6. Repeat until the array is sorted.

Pseudocode:

for i=1 to n

key=list[i]

j=i-1

while j>0 and list[j]>key do

list[j+1]=list[j]

j=j-1

end while

list[j+1]=key

end for

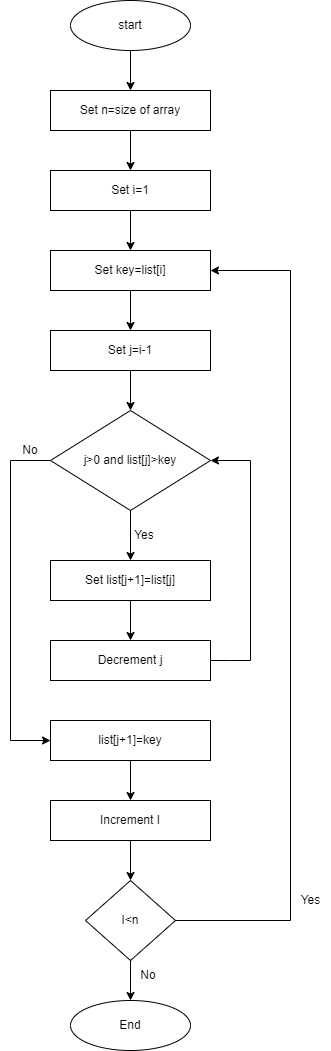


Figure: Insertion sort

**WAP to implement insertion sort.**

#include <stdio.h>

void printArray(int array[], int size) {

for (int i = 0; i < size; i++) {

printf("%d ", array[i]);

}

printf("\n");

}

void insertionSort(int array[], int size) {

for (int step = 1; step < size; step++) {

int key = array[step];

int j = step - 1;

while (key < array[j] && j >= 0) {

array[j + 1] = array[j];

--j;

}

array[j + 1] = key;

}

}

int main() {

int data[] = {23,42,32,1,35};

int size = sizeof(data) / sizeof(data[0]);

printf("Before sorting\n");

for(int i=0;i<size;i++){

printf("%d ",data[i]);

}

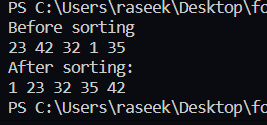
insertionSort(data, size);

printf("\nAfter sorting:\n");

printArray(data, size);

return 0;

}

****