Selection Sort:

Task 👍1:

Write an algorithm / steps for selection sort.

1. Start with an unsorted array

2. For each position i from 0 to n-1:

a. Find the minimum element in the unsorted part (from i to n-1)

b. Swap the minimum element with the element at position i

3. Repeat until the entire array is sorted

10.20 to 10.25

Task 2:

Write a pseudo code for the selection sort

procedure selectionSort(array, size)

for i = 0 to size-1

min\_idx = i

for j = i+1 to size-1

if array[j] < array[min\_idx]

min\_idx = j

swap array[i] and array[min\_idx]

end for

end procedure

10.26 to 10.30

Task 3:

Wap to make sure your list is sorted using selection sort.

def selection\_sort(arr):

n = len(arr)

for i in range(n):

min\_idx = i

for j in range(i+1, n):

if arr[j] < arr[min\_idx]:

min\_idx = j

arr[i], arr[min\_idx] = arr[min\_idx], arr[i]

return arr

# Test the function

arr = [64, 34, 25, 12, 22, 11, 90]

print("Original array:", arr)

print("Sorted array:", selection\_sort(arr))

10.31 to 10.38

Bubble Sort:

Task 4:

Write algorithm for the Bubble sort.

1. Start with an unsorted array

2. Compare adjacent elements

3. If left element is greater than right element, swap them

4. Continue this process for each pair of adjacent elements to the end

5. Repeat steps 2-4 until no swaps are needed

11.10 to 11.15

11.15 to 11.30

Task 5:

Write pseudo code for the bubble sort

procedure bubbleSort(array, size)

for i = 0 to size-1

swapped = false

for j = 0 to size-i-2

if array[j] > array[j+1]

swap array[j] and array[j+1]

swapped = true

if not swapped

break

end for

end procedure

11.30 to 11.35

Task 6:

Wap to make sure your list is sorted using Bubble sort.

def bubble\_sort(arr):

n = len(arr)

for i in range(n):

swapped = False

for j in range(0, n-i-1):

if arr[j] > arr[j+1]:

arr[j], arr[j+1] = arr[j+1], arr[j]

swapped = True

if not swapped:

break

return arr

# Test the function

arr = [64, 34, 25, 12, 22, 11, 90]

print("Original array:", arr)

print("Sorted array:", bubble\_sort(arr))

Task 6:

import java.util.\*;

public class BubbleSort {

public static void main(String args[]) {

int n = 5;

int[] arr = {10, 20 ,5 , 46, 80};

System.out.print("before Sort: ");

for(int i = 0; i<n; i++)

System.out.print(arr[i] + " ");

System.out.println();

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

int swaps = 0;

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

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

int temp;

temp = arr[j];

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

arr[j+1] = temp;

swaps = 1;

}

}

if(swaps == 0)

break;

}

System.out.print("After Sort: ");

for(int i = 0; i<n; i++)

System.out.print(arr[i] + " ");

System.out.println();

}

}

def bubble\_sort(arr):

n = len(arr)

for i in range(n):

swapped = False

for j in range(0, n-i-1):

if arr[j] > arr[j+1]:

arr[j], arr[j+1] = arr[j+1], arr[j]

swapped = True

if not swapped:

break

return arr

# Test the function

arr = [64, 34, 25, 12, 22, 11, 90]

print("Original array:", arr)

print("Sorted array:", bubble\_sort(arr))

Imagine bubbles rising to the surface - larger bubbles rise to the top.

Similarly, in bubble sort, larger elements "bubble up" to their correct positions.

import java.util.\*;

public class Bubble\_Sort {

public static void main(String args[]) {

int n = 5;

int[] arr = {10, 20 ,5 , 46, 80};

System.*out*.print("before Sort: ");

for(int i = 0; i<n; i++)

System.*out*.print(arr[i] + " ");

System.*out*.println();

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

int swaps = 0;

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

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

int temp;

temp = arr[j];

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

arr[j+1] = temp;

swaps = 1;

}

}

if(swaps == 0)

break;

}

System.*out*.print("After Sort: ");

for(int i = 0; i<n; i++)

System.*out*.print(arr[i] + " ");

System.*out*.println();

}

}

before Sort: 10 20 5 46 80

After Sort: 5 10 20 46 80

Task 7:

Insertion sort

1 If it is the first element, it is already sorted. return 1;

2 Pick next element

3 Compare with all elements in the sorted sub-list

4 Shift all the elements in the sorted sub-list that is greater than the value to be sorted

5 Insert the value

6 Repeat until list is sorted

The algorithm works like sorting playing cards in your hands:

1. Start with the first element (considered sorted)

2. Take the next element (key)

3. Compare with previous elements

4. Make space by shifting elements

5. Place the key in its correct position

6. Repeat for all elements

Task 8:

Insertion Sort(A)

for j = 2 to A.length

key = A[j]

i = j 1

while i > 0 and A[i] > key

A[i + 1] = A[i]

i = i -1

A[i + 1] = key

procedure InsertionSort(A)

for j = 1 to length[A]-1

key = A[j]

// Insert A[j] into sorted sequence A[0..j-1]

i = j - 1

while i >= 0 and A[i] > key

A[i + 1] = A[i]

i = i - 1

A[i + 1] = key

Task 9:

public class InsertionSort {

public static void main(String args[]) {

int n = 5;

int[] arr = {67, 44, 82, 17, 20};

System.out.print("before Sorting: ");

for(int i = 0; i<n; i++)

System.out.print(arr[i] + " ");

System.out.println();

for(int i = 1; i<n; i++) {

int key = arr[i];

int j = i;

while(j > 0 && arr[j-1]>key) {

arr[j] = arr[j-1];

j--;

}

arr[j] = key;

}

System.out.print("After Sorting: ");

for(int i = 0; i<n; i++)

System.out.print(arr[i] + " ");

System.out.println();

}

}

def insertion\_sort(arr):

n = len(arr)

for i in range(1, n):

key = arr[i]

j = i

# Move elements that are greater than key

# to one position ahead of their current position

while j > 0 and arr[j-1] > key:

arr[j] = arr[j-1]

j -= 1

arr[j] = key

return arr

# Test the function

if \_\_name\_\_ == "\_\_main\_\_":

arr = [67, 44, 82, 17, 20]

n = len(arr)

print("Before Sorting:", end=" ")

print(\*arr)

insertion\_sort(arr)

print("After Sorting:", end=" ")

print(\*arr)