1. Bubble Sort

import java.util.Arrays;

public class Linked {

static void bubbleSortAlgorithm(int array[]) {

int size = array.length;

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

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

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

int temp = array[j];

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

array[j + 1] = temp;

}

}

public static void main(String args[]) {

int[] data = { 6, 5, 8, 1, 2 };

bubbleSortAlgorithm(data);

System.out.println("The array performing the Bubble Sort Algorithm is:");

System.out.println(Arrays.toString(data));

}

}

The array performing the Bubble Sort Algorithm is:

[1, 2, 5, 6, 8]

2. Quick Sort

class Linked {

static void swap(int[] arr, int i, int j)

{

int temp = arr[i];

arr[i] = arr[j];

arr[j] = temp;

}

static int partition(int[] arr, int low, int high)

{

int pivot = arr[high];

int i = (low - 1);

for (int j = low; j <= high - 1; j++) {

if (arr[j] < pivot) {

i++;

swap(arr, i, j);

}

}

swap(arr, i + 1, high);

return (i + 1);

}

static void quickSort(int[] arr, int low, int high)

{

if (low < high) {

int pi = partition(arr, low, high);

quickSort(arr, low, pi - 1);

quickSort(arr, pi + 1, high);

}

}

public static void printArr(int[] arr)

{

for (int i = 0; i < arr.length; i++) {

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

}

}

public static void main(String[] args)

{

int[] arr = { 10, 7, 8, 9, 1, 5 };

int N = arr.length;

quickSort(arr, 0, N - 1);

System.out.println("Sorted array:");

printArr(arr);

}

}

Sorted array:

1 5 7 8 9 10

3. Selection Sort

public class Linked

{

void sort(int arr[])

{

int n = arr.length;

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

{

int min\_idx = i;

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

if (arr[j] < arr[min\_idx])

min\_idx = j;

int temp = arr[min\_idx];

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

arr[i] = temp;

}

}

void printArray(int arr[])

{

int n = arr.length;

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

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

System.out.println();

}

public static void main(String args[])

{

Linked ob = new Linked();

int arr[] = {64,25,12,22,11};

ob.sort(arr);

System.out.println("Sorted array");

ob.printArray(arr);

}

}

C:\Users\mujah\Downloads\Algorithm and Data Structure>java Linked

Sorted array

11 12 22 25 64

4. Insertion Sort

public class Linked {

void sort(int arr[])

{

int n = arr.length;

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

int key = arr[i];

int j = i - 1;

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

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

j = j - 1;

}

arr[j + 1] = key;

}

}

static void printArray(int arr[])

{

int n = arr.length;

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

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

System.out.println();

}

public static void main(String args[])

{

int arr[] = { 12, 11, 13, 5, 6 };

Linked ob = new Linked();

ob.sort(arr);

printArray(arr);

}

}

C:\Users\mujah\Downloads\Algorithm and Data Structure>java Linked

5 6 11 12 13

5. Merge Sort

class Linked {

void merge(int arr[], int l, int m, int r)

{

int n1 = m - l + 1;

int n2 = r - m;

int L[] = new int[n1];

int R[] = new int[n2];

for (int i = 0; i < n1; ++i)

L[i] = arr[l + i];

for (int j = 0; j < n2; ++j)

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

int i = 0, j = 0;

int k = l;

while (i < n1 && j < n2) {

if (L[i] <= R[j]) {

arr[k] = L[i];

i++;

}

else {

arr[k] = R[j];

j++;

}

k++;

}

while (i < n1) {

arr[k] = L[i];

i++;

k++;

}

while (j < n2) {

arr[k] = R[j];

j++;

k++;

}

}

void sort(int arr[], int l, int r)

{

if (l < r) {

int m = l + (r - l) / 2;

sort(arr, l, m);

sort(arr, m + 1, r);

merge(arr, l, m, r);

}

}

static void printArray(int arr[])

{

int n = arr.length;

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

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

System.out.println();

}

public static void main(String args[])

{

int arr[] = { 12, 11, 13, 5, 6, 7 };

System.out.println("Given array is");

printArray(arr);

Linked ob = new Linked();

ob.sort(arr, 0, arr.length - 1);

System.out.println("\nSorted array is");

printArray(arr);

}

}

C:\Users\mujah\Downloads\Algorithm and Data Structure>java Linked

Given array is

12 11 13 5 6 7

Sorted array is

5 6 7 11 12 13

6. Quick Sort

class Linked{

static void swap(int[] arr, int i, int j)

{

int temp = arr[i];

arr[i] = arr[j];

arr[j] = temp;

}

static int partition(int[] arr, int low, int high)

{

int pivot = arr[high];

int i = (low - 1);

for (int j = low; j <= high - 1; j++) {

if (arr[j] < pivot) {

i++;

swap(arr, i, j);

}

}

swap(arr, i + 1, high);

return (i + 1);

}

static void quickSort(int[] arr, int low, int high)

{

if (low < high) {

int pi = partition(arr, low, high);

quickSort(arr, low, pi - 1);

quickSort(arr, pi + 1, high);

}

}

public static void printArr(int[] arr)

{

for (int i = 0; i < arr.length; i++) {

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

}

}

public static void main(String[] args)

{

int[] arr = { 10, 7, 8, 9, 1, 5 };

int N = arr.length;

quickSort(arr, 0, N - 1);

System.out.println("Sorted array:");

printArr(arr);

}

}

C:\Users\mujah\Downloads\Algorithm and Data Structure>java Linked

Sorted array:

1 5 7 8 9 10

7. Sorting Strings using Bubble Sort

class Linked {

static int MAX = 100;

public static void sortStrings(String[] arr, int n)

{

String temp;

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

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

if (arr[j].compareTo(arr[i]) > 0) {

temp = arr[j];

arr[j] = arr[i];

arr[i] = temp;

}

}

}

}

public static void main(String[] args)

{

String[] arr = { "one", "two",

"three", "four", "five " };

int n = arr.length;

sortStrings(arr, n);

System.out.println(

"Strings in sorted order are : ");

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

System.out.println("String " + (i + 1) + " is "

+ arr[i]);

}

}

Strings in sorted order are :

String 1 is five

String 2 is four

String 3 is one

String 4 is three

String 5 is two

9. Bubble Sort for Linked List by Swapping nodes

class Node {

int data;

Node next;

public Node(int data) {

this.data = data;

this.next = null;

}

}

class LinkedList {

Node head;

public void swapNodes(int x, int y) {

if (x == y) {

return;

}

Node prevX = null, currX = head;

while (currX != null && currX.data != x) {

prevX = currX;

currX = currX.next;

}

Node prevY = null, currY = head;

while (currY != null && currY.data != y) {

prevY = currY;

currY = currY.next;

}

if (currX == null || currY == null) {

return;

}

if (prevX != null) {

prevX.next = currY;

} else {

head = currY;

}

if (prevY != null) {

prevY.next = currX;

} else {

head = currX;

}

Node temp = currX.next;

currX.next = currY.next;

currY.next = temp;

}

public void bubbleSort() {

if (head == null) {

return;

}

int count = 0;

Node start = head;

while (start != null) {

count++;

start = start.next;

}

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

Node curr = head;

while (curr != null && curr.next != null) {

if (curr.data > curr.next.data) {

swapNodes(curr.data, curr.next.data);

}

curr = curr.next;

}

}

}

public void printList() {

Node tmp = head;

while (tmp != null) {

System.out.print(tmp.data + " -> ");

tmp = tmp.next;

}

System.out.println("None");

}

}

public class Linked {

public static void main(String[] args) {

int[] arr = {78, 20, 10, 32, 1, 5};

LinkedList llist = new LinkedList();

llist.head = new Node(arr[0]);

Node start = llist.head;

for (int i = 1; i < arr.length; i++) {

start.next = new Node(arr[i]);

start = start.next;

}

System.out.println("Linked list before sorting");

llist.printList();

llist.bubbleSort();

System.out.println("Linked list after sorting");

llist.printList();

}

}

C:\Users\mujah\Downloads\Algorithm and Data Structure>java Linked

Linked list before sorting

78 -> 20 -> 10 -> 32 -> 1 -> 5 -> None

Linked list after sorting

1 -> 5 -> 10 -> 20 -> 32 -> 78 -> None

10.Bubble Sort On Doubly Linked List

class Linked

{

static class Node

{

int data;

Node prev;

Node next;

};

static Node insertAtTheBegin( Node start\_ref, int data)

{

Node ptr1 = new Node();

ptr1.data = data;

ptr1.next = start\_ref;

if (start\_ref != null)

(start\_ref).prev = ptr1;

start\_ref = ptr1;

return start\_ref;

}

static void printList( Node start)

{

Node temp = start;

System.out.println();

while (temp != null)

{

System.out.print( temp.data + " ");

temp = temp.next;

}

}

static Node bubbleSort( Node start)

{

int swapped, i;

Node ptr1;

Node lptr = null;

if (start == null)

return null;

do

{

swapped = 0;

ptr1 = start;

while (ptr1.next != lptr)

{

if (ptr1.data > ptr1.next.data)

{

int t = ptr1.data;

ptr1.data = ptr1.next.data;

ptr1.next.data = t;

swapped = 1;

}

ptr1 = ptr1.next;

}

lptr = ptr1;

}

while (swapped != 0);

return start;

}

public static void main(String args[])

{

int arr[] = {12, 56, 2, 11, 1, 90};

int list\_size, i;

Node start = null;

for (i = 0; i < 6; i++)

start=insertAtTheBegin(start, arr[i]);

System.out.printf("\n Doubly Linked list before sorting ");

printList(start);

start = bubbleSort(start);

System.out.printf("\n Doubly Linked list after sorting ");

printList(start);

}

}

Doubly Linked list before sorting

90 1 11 2 56 12

Doubly Linked list after sorting

1 2 11 12 56 90