**Algorithms and Codes**

1. sort using selection sort based on its employeeid

**Algorithm:**

1. Declare the entire array initially as unsorted.
2. Identify the smallest element from the unsorted array and swap it with the first element, turning this into sorted array.
3. The sorted array will hold the elements of the array in sorted manner.
4. Then this process is repeated and smallest element is searched from unsorted array and moved to sorted array.

**Code:**

public class Program4

{

public static void main(String args[])

{

Employee[] arr;

arr = new Employee[5];

arr[0] = new Employee(1701289270, "Satyabrata", "Demo Address1");

arr[1] = new Employee(1701289219, "Omm Prasad", "Demo Address2");

arr[2] = new Employee(1701289218, "Jai Bhatt", "Demo Address3");

arr[3] = new Employee(1701289221, "Farhan Qureshi", "Demo Address4");

arr[4] = new Employee(1701289213, "Raju Rastogi", "Demo Address5");

for(int i=0;i<4;i++)

{

int min=i;

for(int j=i+1;j<5;j++)

{

if(arr[j].employeeID < arr[min].employeeID)

{

min=j;

}

}

int temp=arr[min].employeeID;

arr[min].employeeID=arr[i].employeeID;

arr[i].employeeID=temp;

}

System.out.println("After Selection Sort: ");

for(int i=0;i<5;i++)

{

System.out.println("Employee ID: "+arr[i].employeeID+" Name: "+arr[i].employeeName+" Address: "+arr[i].employeeAddress);

}

}

}

class Employee

{

public int employeeID;

public String employeeName;

public String employeeAddress;

Employee(int employeeID,String employeeName,String employeeAddress)

{

this.employeeID=employeeID;

this.employeeName=employeeName;

this.employeeAddress=employeeAddress;

}

}

1. sort using quick sort based on its employeeid

**Algorithm:**

1. This algorithm works in a recursive manner, declaring one element from the array as pivot element, and declaring the first and last elements as i and j.
2. Usually, any element can be declared as pivot, but here the first element is to be declared as pivot.
3. The goal is to keep all elements smaller than pivot at left side and all elements larger than pivot at the right side of the pivot.
4. The pivot is compared with the right element. If the right element is bigger than pivot, j is decremented.
5. If not, then element at j is swapped with pivot.
6. Similarly, now comparison is done from pivot to left side with i.
7. If element at i is smaller than pivot, i is incremented.
8. Else elements at i and pivot are swapped.
9. This is repeated until pivot, i and j, all three point to the same array element. This indicates that the pivot element is at the desired position.
10. Now 2 sub arrays created to the left and right of pivot will hold elements smaller than pivot in left sub array and elements larger than pivot in right sub array.
11. This entire process will be done again on the newly created sub arrays, until all elements are sorted.

**Code:**

public class Demo

{

int partition (Employee array[], int start, int end)

{

Employee pivot = array[end];

int i = (start - 1);

for (int j = start; j <= end - 1; j++)

{

if (array[j].EmployeeID < pivot.EmployeeID)

{

i++;

Employee tmp = array[i];

array[i] = array[j];

array[j] = tmp;

}

}

Employee tmp = array[i+1];

array[i+1] = array[end];

array[end] = tmp;

return (i + 1);

}

void quick(Employee array[], int start, int end)

{

if (start < end)

{

int part = partition(a, start, end);

quick(a, start, part - 1);

quick(a, part + 1, end);

}

}

public static void main(String args[])

{

Employee[] arr;

arr = new Employee[5];

arr[0]=new Employee(1,"Ritu","Rajkot");

arr[1]=new Employee(2,"Sakshi","Jamnagar");

arr[2]=new Employee(13,"Tanvi","Ahemdabad");

arr[3]=new Employee(4,"Jinkal","Morbi");

arr[4]=new Employee(15,"Gunjan","Pune");

for(int i=0;i<4;i++)

{

int min=i;

for(int j=i+1;j<5;j++)

{

if(arr[j].EmployeeID < arr[min].EmployeeID)

{

min=j;

}

}

int temp=arr[min].EmployeeID;

arr[min].EmployeeID=arr[i].EmployeeID;

arr[i].EmployeeID=temp;

}

System.out.println("After Quick Sort: ");

for(int i=0;i<5;i++)

{

System.out.println("Employee Id: "+arr[i].EmployeeID+" Name: "+arr[i].employeeName+" Address: "+arr[i].employeeAddress);

}

}

}

class Employee

{

public int EmployeeID;

public String employeeName;

public String employeeAddress;

Employee(int EmployeeID,String employeeName,String employeeAddress)

{

this.EmployeeID=EmployeeID;

this.employeeName=employeeName;

this.employeeAddress=employeeAddress;

}

}

1. sort using stack based on its employeeid

**Algorithm:**

1. Array elements are scanned to find the largest element from the array.
2. Stack of size same as array is created.
3. The largest element is then pushed into the stack.
4. This process is repeated, largest element from the array is found and pushed into the stack.
5. Once the stack is full, the elements will be arranged in such a way that the top element of the stack will be the smallest element of the array.
6. Then the stack items will be popped one by one and added to array in such a way that first pop of stack will be stored at array index 0, next pop at array index 1, and so on.
7. This way, once all the elements are popped from the stack, they will be stored in array in sorted manner.

**Code:**

import java.util.\*;

class Employee {

public int employeeID;

public String employeeName;

public String employeeAddress;

Employee(int employeeID, String employeeName, String employeeAddress)

{

this.employeeID = employeeID;

this.employeeName = employeeName;

this.employeeAddress = employeeAddress;

}

}

public class Demo {

public static Stack<Employee> sortstack(Stack<Employee> stk)

{

Stack<Employee> tmpStack = new Stack<Employee>();

while(!stk.isEmpty())

{

Employee tmp = stk.pop();

while(!tmpStack.isEmpty() && tmpStack.peek().employeeID > tmp.employeeID)

{

stk.push(tmpStack.pop());

}

tmpStack.push(tmp);

}

return tmpStack;

}

public static void main(String args[])

{

Employee[] arr;

arr = new Employee[5];

arr[0] = new Employee(1701289270, "Satyabrata", "Demo Address1");

arr[1] = new Employee(1701289219, "Omm Prasad", "Demo Address2");

arr[2] = new Employee(1701289218, "Jai Bhatt", "Demo Address3");

arr[3] = new Employee(1701289221, "Farhan Qureshi", "Demo Address4");

arr[4] = new Employee(1701289213, "Raju Rastogi", "Demo Address5");

for(int i=0;i<5;i++)

{

System.out.println("Employee ID: "+arr[i].employeeID+", Name: "+arr[i].employeeName+", Address: "+ arr[i].employeeAddress);

}

Stack<Employee> stk = new Stack<>();

for(int x=0;x<5;x++)

{

stk.push(arr[x]);

}

Stack<Employee> tmpStack = sortstack(stk);

arr[0] = new Employee(0, "", "");

arr[1] = new Employee(0, "", "");

arr[2] = new Employee(0, "", "");

arr[3] = new Employee(0, "", "");

arr[4] = new Employee(0, "", "");

int k=4;

while (!tmpStack.empty())

{

arr[k]=tmpStack.pop();

k--;

}

System.out.println("After Stack Sort:");

for(int i=0;i<5;i++)

{

System.out.println("Employee ID: "+arr[i].employeeID+", Name: "+arr[i].employeeName+", Address: "+ arr[i].employeeAddress);

}

}

}