Algorithm and Code

1 Sort the student id by quick sort

**Algorithm:**

1 Start the program

2 Intiliaze an integer array to store the student id(array name= studentId)

3 create a function called as quick sort.

quick sort(studentId, start, end)

4 Select any element as pivot, here we select the leftmost value of the array, or the starting point as the pivot.

5 Since the pivot element is on the left, we start comparing the rightmost value to the pivot and will keep moving towards left.

6 Comparing each element to the pivot we swap the value of the pivot with the current element which is being compared if,

studentId[pivot]> studentId[current location]

7 If step 6 happens, our pivot moves to the location which the pointer is pointing, so, we start comparing values from the left and keep moving towards the pivot. Repeat the steps 5 to 7 until the value of pivot, the value of left and right pointers are pointing to the same element.

8 Once the three values, that is, studentId[pivot], studentId[left], studentId[right] are the same value(or pointing to the same element) our array is sorted.

9 display the sorted array

10 end the program.

**Code:**

public class Quick

{

int partition (int studentId[], int start, int end)

{

int pivot = studentId[end];

int i = (start - 1);

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

{

if (studentId[j] < pivot)

{

i++;

int t = studentId[i];

studentId[i] = studentId[j];

studentId[j] = t;

}

}

int t = studentId[i+1];

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

studentId[end] = t;

return (i + 1);

}

void quicksort(int studentId[], int start, int end) {

if (start < end)

{

int p = partition(studentId, start, end);

quicksort(studentId, start, p - 1);

quicksort(studentId, p + 1, end);

}

}

void printArr(int studentId[], int n)

{

int i;

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

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

}

public static void main(String[] args) {

int studentId[] = { 13, 18, 27, 2, 19, 25 };

int n = studentId.length;

System.out.println("\nBefore sorting array elements are - ");

Quick q1 = new Quick();

q1.printArr(studentId, n);

q1.quicksort..6(studentId, 0, n - 1);

System.out.println("\nAfter sorting array elements are - ");

q1.printArr(studentId, n);

System.out.println();

}

}

2 Sort the student id by using insertion sort

**Algorithm:**

1 start the program

2 create a function sort which takes the array studentId as an argument.

3 repeat the steps 3 to 4 until i ranges from 1 to studentId.length()-1.

4 check the key element to its predecessor element, if predecessor elements is greater than the key element swap them.

5 once all the elements are sorted, print the sorted array

6 end the program.

**Code:**

class Insert {

void sort(int studentId[])

{

int n = studentId.length;

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

int key = studentId[i];

int j = i - 1;

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

studentId[j + 1] = studentId[j];

j = j - 1;

}

studentId[j + 1] = key;

}

}

static void printArray(int studentId[])

{

int n = studentId.length;

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

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

System.out.println();

}

public static void main(String args[])

{

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

Insert ob = new Insert();

ob.sort(studentId);

System.out.println("After sorting:");

printArray(studentId);

}

}

3 Sort the student Id by using stack

**Algorithm:**

1 start the program

2 create a new stack

3 store the array elements into the stack using push and pop

4 create a temporary stack and push all the elements of the stack into the temporary stack

5 sort the elements in the stack using push and pop

7 return the stack elements to the array, the array is sorted

8 print the sorted array

9 end the program

**Code:**

import java.util.\*;

class SortUsingStack

{

static Stack<Integer> sortStack(Stack<Integer> input)

{

Stack<Integer> tmpStack =

new Stack<Integer>();

while (!input.empty())

{

int tmp = input.peek();

input.pop();

while (!tmpStack.empty() &&

tmpStack.peek() < tmp)

{

input.push(tmpStack.peek());

tmpStack.pop();

}

tmpStack.push(tmp);

}

return tmpStack;

}

static void sortArrayUsingStacks(int []studentId,int n)

{

Stack<Integer> input =

new Stack<Integer>();

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

input.push(studentId[i]);

Stack<Integer> tmpStack = sortStack(input);

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

{

studentId[i] = tmpStack.peek();

tmpStack.pop();

}

}

public static void main(String args[])

{

int []studentId = {10, 5, 15, 45};

int n = studentId.length;

sortArrayUsingStacks(studentId, n);

System.out.println("After sorting");

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

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

}

}