**Sorting Algorithms**

**1.Boubble Sort:**

import java.util.Scanner;

class BoubleSort

{

public static void intfun1(int a[],int n ) //for numbers sorting

{

for(int i=0;i<n;i++) //performance no of rounds i.e n-1

{

int flag=0;

for(int j=0;j<n-1-i;j++) //performance swapping of adjacent { elements except last element of priv round

if(a[j]>a[j+1])

{

int temp=a[j];

a[j]=a[j+1];

a[j+1]=temp;

flag=1;

}

}

if(flag==0) //means already sorted

{

break;

}

}

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

{

System.out.println(a[i]+" ");

}

}

public static void Stringfun(String a[] ) //for text sorting i.e deepak tutorials

{

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

{

int flag=0;

for(int j=0;j<a.length-1-i;j++)

{

if(a[j].compareTo(a[j+1])>0)

{

String temp=a[j];

a[j]=a[j+1];

a[j+1]=temp;

flag=1;

}

}

if(flag==0)

{

break;

}

}

for(int k=0;k<a.length;k++)

{

System.out.print(a[k]+" ");

}

}

public static void main(String args[])

{

Scanner ip=new Scanner(System.in);

int n=ip.nextInt();

int a[]=new int[n];

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

{

a[i]=ip.nextInt();

}

BoubleSort.intfun1(a,n); //BoubleSort.intfun2(a);

String s[]=new String[n];

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

{

s[i]=ip.nextLine();

}

BoubleSort.Stringfun(s);

}

}

\* "it performance n-1 rounds" to sort list

\* BS is a simple sorting algorithm, that works by repeatedly stepping through the list to be sorted.

\* Comparing each pair of adjacent items and swapping them if they are in the wrong order.

it is repeated until no swaps are needed.

\* it is a comparison sort. Although the algorithm is simple,

most of the other sorting algorithms are more efficient for large lists.

======> worst-case and average complexity both О(n2).

\* insertion sort, tend to have better performance than bubble sort.

\* bubble sort is not a practical sorting algorithm when n is large.

public static void intfun2(int[] a)

{

boolean sorted = false;

int temp;

while(!sorted)

{

sorted = true;

for (int i = 0; i < a.length - 1; i++)

{

if (a[i] > a[i+1])

{

temp = a[i];

a[i] = a[i+1];

a[i+1] = temp;

sorted = false;

}

}

}

)

-------------------------------------end boubble sort -----------------------

**2.Quick Sort:**

/\*Write a java program to implement Quicksort

expected output

Enter number of integer elements

5

Enter 5 integer elements

30

20

10

40

70

Sorted array

10 20 30 40 70

\*/

import java.util.\*;

class Quicksort

{

void swap (int a[], int left, int right)

{

int left ,right,flr;

left=low;

right=hight;

flr=a[low];

while

//add code here... to swap elements

}//end swap

void quicksort( int a[], int low, int high )

{

if (a[low]<pivot)

{

low++;

}

else (a[high]>pivot)

{

high--;

}

}

// add code-here... to sort (use recursion)

//call partition()

// ..add code-here... to sort left-part of pivot item (use recursion)

// ...add code-here... to sort right-part of pivot item (use recursion)

} //end quicksort

int partition( int a[], int low, int high )

{

int left ,right,flr;

left=low;

right=hight;

flr=a[low];

while //add code here....

}

void printArray(int a[], int n)

{

int i;

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

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

}

}

--------------------------------end quick sort --------------------------

**3.Selection Sort:**

import java.util.Scanner;

class SelectionSort

{

public static void sortfun1(int a[],int n)

{

int min,temp=0;

for(int i=0;i<n;i++) // swaps or places that smallest value into its proper { location.

min=i;

for(int j=i+1;j<n;j++) // finds Smallest element index in list

{

if(a[j]<a[min])

{

min=j;

}

}

temp=a[i];

a[i]=a[min];

a[min]=temp;

}

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

{

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

}

}

public static void sortfun2(String a[],int n)

{

int min;

String temp="";

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

{

min=i;

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

{

if(a[j].compareTo(a[min])<0)

{

min=j;

}

}

temp=a[i];

a[i]=a[min];

a[min]=temp;

}

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

{

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

}

}

public static void main(String args[])

{

Scanner ip=new Scanner(System.in);

int n=ip.nextInt();

int a[]=new int[n];

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

{

a[i]=ip.nextInt();

}

SelectionSort.sortfun1(a,n);

int k=ip.nextInt();

String s[]=new String[k];

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

{

s[i]=ip.nextLine();

}

SelectionSort.sortfun2(s,k);

}

}

/\*

\* selection sort is a combination of searching and sorting.

\* Selecting the lowest element(initial) requires scanning all n elements (this takes n − 1 comparisons) and

then swapping it into the first position.

\* in the selection sort, the inner loop finds the next smallest (or largest) value and

the outer loop places that value into its proper location.

\* Finding the next lowest element requires scanning the remaining n − 1 elements and so on,

for (n − 1) + (n − 2) + ... + 2 + 1 = n(n − 1) / 2 ∈ Θ(n2) comparisons.