

Q1.

Arraysort.java

```
/*
```

```
 * To change this license header, choose License Headers in Project Properties.
```

```
 * To change this template file, choose Tools | Templates
```

```
 * and open the template in the editor.
```

```
 */
```

```
/**
```

```
 *
```

```
 * @author manthan
```

```
 */
```

```
    abstract class ArraySort{
```

```
        protected int[] a ; //array to be sorted
```

```
        protected long numCompare ;
```

```
        protected long numSwap ;
```

```
        protected int numRecursion ;
```

```
        protected boolean display;
```

```
        protected static final IntUtil u = new IntUtil();
```

```
        private void sort1(int [] a, boolean ascend) {
```

```
            System.out.println("-----SORT1 Start-----");
```

```
            this.a = a ;
```

```

numCompare = 0 ;
numSwap = 0 ;
numRecursion = 0 ;
display = false ;
if (a.length > 0 && a.length < 20) {
    display = true ;
}

sort(ascend); //THIS CODE IS WRITTEN BY USER

if (ascend) {
    u.assertAscending(a) ;
}else {
    u.reverse(a) ;
    u.assertDescending(a) ;
}

if (display) {
    int n = a.length ;

    u.pLn(n) ;
    u.pLn(a,0,n);
}

u.printStatistics(a.length,numCompare,numSwap,0);

System.out.println("-----SORT1 End-----");
}

//I don't know how to write it

//Override by the concrete class

```

```
abstract protected void sort(boolean ascend) ;
```

```
private void testSort(int N, boolean ascend) {
```

```
    int [] a = u.generateRandomNumber(N,false);//Generates random pos and neg numbers
```

```
    sort1(a,ascend) ;
```

```
}
```

```
private void basicTests() {
```

```
    int b[][] = u.testArray();
```

```
    int l = b.length ;
```

```
    for (int i = 0; i < l; ++i) {
```

```
        int [] a = b[i] ;
```

```
        sort1(a,true);
```

```
    }
```

```
}
```

```
protected void testBench() {
```

```
    System.out.println("-----START-----");
```

```
    basicTests() ;
```

```
    for (int n = 10000; n < 50000; n = n + 10000) {
```

```
        testSort(n,true); //ascending order
```

```
    }
```

```
    for (int n = 10001; n < 50001; n = n + 10000) {
```

```
        testSort(n,false); //descending order
```

```
    }
```

```

int h = 1 ;

for (int i = 5000 + h; i < 25001 + h; i = i + 5000) {

    System.out.println("-----testing " + i + " SORTED ASCENDING numbers-----");

    int [] b = u.generateNumberInIncreasingOrder(i,1) ;

    sort1(a,true) ;

}

System.out.println("-----DONE!-----");

}

```

```

public static void main(String[] args) {

    System.out.println("ArraySort.java");

    //You cannot instantiate an object from abstract class

    //ArraySort a = new ArraySort() ;

    //a.testBench();

}

}

```

Bubblesort.java

```

/*

* To change this license header, choose License Headers in Project Properties.

* To change this template file, choose Tools | Templates

* and open the template in the editor.

*/

```

```

/**

```

```

*
* @author manthan
*/
public class BubbleSort extends ArraySort{

    protected void sort(boolean ascend) {

        if(display) ArraySort.u.pLn(a.length);

        boolean flag = true; // set flag to true to begin first pass

        int temp; // holding variable

        while (flag) {

            flag = false; // set flag to false awaiting a possible swap

            if(display) ArraySort.u.pLn(a);

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

                if (a[j] > a[j + 1]) // change to > for ascending sort

                {

                    temp = a[j]; // swap elements

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

                    a[j + 1] = temp;

                    flag = true; // shows a swap occurred

                    numSwap++;

                }

                numCompare++;

            }

        }
    }
}

```

```

    }

}

public static void main(String[] args) {

    System.out.println("ArraySort.java");

    BubbleSort a = new BubbleSort();

    a.testBench();

}

}

```

SS:

The screenshot shows the Eclipse IDE with the following components:

- Package Explorer:** Displays the project structure, including the 'src' folder containing 'BubbleSort.java'.
- Editor:** Shows the source code of 'BubbleSort.java', which extends 'ArraySort' and implements a bubble sort algorithm.
- Console:** Displays the output of the program, showing the execution of the 'testBench' method and the results of the bubble sort algorithm.

Console Output:

```

<terminated> BubbleSort [Java Application] C:\Program Files\Java\jdk1.7.0_79\bin\javaw.exe (Jul 14, 2016, 1:08:32 PM)
testing 15001 SORTED ASCENDING numbers
-----SORT1 Start-----
# n = 40001
# num compare (C) = 40000
# num swap (S) = 0
# C+S = 40000
T(n)=(C+S) / (n) = 0.9999750006249843 (n)
# nlogn = 611525.225607432
T(n)=(C+S) / (nlogn) = 0.06541022074807747 (n*logn)
# n*n = 1600080001

```

Output:

```

ArraySort.java
-----START-----
-----SORT1 Start-----

```

```

# n =          0
# num compare(C)      =  0
# num swap(S)         =  0
# C+S                =  0
Zero elements in array
-----SORT1 End-----
-----SORT1 Start-----
  0
 15
  0
 15
# n =          1
# num compare(C)      =  0
# num swap(S)         =  0
# C+S                =  0
T(n)=(C+S)/(n)= 0.0(n)
# nlogn              =  0.0
# n*n                =  1
T(n)=(C+S)/(n^2)= 0.0(n^2)
-----SORT1 End-----
-----SORT1 Start-----
  0  1
 15  5
  5 15
  0  1
  5 15
# n =          2
# num compare(C)      =  2
# num swap(S)         =  1
# C+S                =  3
T(n)=(C+S)/(n)= 1.5(n)
# nlogn              =  2.0
T(n)=(C+S)/(nlogn)= 1.5(n*logn)
# n*n                =  4
T(n)=(C+S)/(n^2)= 0.75(n^2)
-----SORT1 End-----
-----SORT1 Start-----
  0  1  2  3  4  5  6  7
 15  5 64  8 12 11  4 35
  5 15  8 12 11  4 35 64
  5  8 12 11  4 15 35 64
  5  8 11  4 12 15 35 64
  5  8  4 11 12 15 35 64
  5  4  8 11 12 15 35 64
  4  5  8 11 12 15 35 64
  0  1  2  3  4  5  6  7
  4  5  8 11 12 15 35 64
# n =          8
# num compare(C)      = 49
# num swap(S)         = 15
# C+S                = 64
T(n)=(C+S)/(n)= 8.0(n)
# nlogn              = 24.0
T(n)=(C+S)/(nlogn)= 2.6666666666666665(n*logn)
# n*n                = 64
T(n)=(C+S)/(n^2)= 1.0(n^2)
-----SORT1 End-----

```

```

-----SORT1 Start-----
 0  1  2  3  4  5
 6  5  4  3  2  1
 5  4  3  2  1  6
 4  3  2  1  5  6
 3  2  1  4  5  6
 2  1  3  4  5  6
 1  2  3  4  5  6
 0  1  2  3  4  5
 1  2  3  4  5  6
# n =          6
# num compare(C)      = 30
# num swap(S)         = 15
# C+S                 = 45
T(n)=(C+S)/(n)= 7.5(n)
# nlogn              = 15.509775004326936
T(n)=(C+S)/(nlogn)= 2.901396054259062(n*logn)
# n*n                = 36
T(n)=(C+S)/(n^2)= 1.25(n^2)
-----SORT1 End-----
-----SORT1 Start-----
 0  1  2  3  4  5
 1  2  3  4  5  6
 0  1  2  3  4  5
 1  2  3  4  5  6
# n =          6
# num compare(C)      = 5
# num swap(S)         = 0
# C+S                 = 5
T(n)=(C+S)/(n)= 0.8333333333333334(n)
# nlogn              = 15.509775004326936
T(n)=(C+S)/(nlogn)= 0.322377339362118(n*logn)
# n*n                = 36
T(n)=(C+S)/(n^2)= 0.1388888888888889(n^2)
-----SORT1 End-----
-----SORT1 Start-----
 0  1  2  3  4  5
 1  1  1  1  1  1
 0  1  2  3  4  5
 1  1  1  1  1  1
# n =          6
# num compare(C)      = 5
# num swap(S)         = 0
# C+S                 = 5
T(n)=(C+S)/(n)= 0.8333333333333334(n)
# nlogn              = 15.509775004326936
T(n)=(C+S)/(nlogn)= 0.322377339362118(n*logn)
# n*n                = 36
T(n)=(C+S)/(n^2)= 0.1388888888888889(n^2)
-----SORT1 End-----
-----SORT1 Start-----
 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15
 3  1  4  1  5  9  2  6  5  3  5  8  9  7  9  3
 1  3  1  4  5  2  6  5  3  5  8  9  7  9  3  9
 1  1  3  4  2  5  5  3  5  6  8  7  9  3  9  9
 1  1  3  2  4  5  3  5  5  6  7  8  3  9  9  9
 1  1  2  3  4  3  5  5  5  6  7  3  8  9  9  9

```



```

1  1  2  3  3  4  5  5  5  6  3  7  8  9  9  9
1  1  2  3  3  4  5  5  5  3  6  7  8  9  9  9
1  1  2  3  3  4  5  5  3  5  6  7  8  9  9  9
1  1  2  3  3  4  5  3  5  5  6  7  8  9  9  9
1  1  2  3  3  4  3  5  5  5  6  7  8  9  9  9
1  1  2  3  3  3  4  5  5  5  6  7  8  9  9  9
0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15
1  1  2  3  3  3  4  5  5  5  6  7  8  9  9  9
# n = 16
# num compare(C) = 165
# num swap(S) = 31
# C+S = 196
T(n)=(C+S)/(n)= 12.25(n)
# nlogn = 64.0
T(n)=(C+S)/(nlogn)= 3.0625(n*logn)
# n*n = 256
T(n)=(C+S)/(n^2)= 0.765625(n^2)
-----SORT1 End-----
-----SORT1 Start-----
# n = 10000
# num compare(C) = 97590240
# num swap(S) = 25235068
# C+S = 122825308
T(n)=(C+S)/(n)= 12282.5308(n)
# nlogn = 132877.1237954945
T(n)=(C+S)/(nlogn)= 924.3525483666789(n*logn)
# n*n = 100000000
T(n)=(C+S)/(n^2)= 1.22825308(n^2)
-----SORT1 End-----
-----SORT1 Start-----
# n = 20000
# num compare(C) = 398280085
# num swap(S) = 101113653
# C+S = 499393738
T(n)=(C+S)/(n)= 24969.6869(n)
# nlogn = 285754.247590989
T(n)=(C+S)/(nlogn)= 1747.6336474787993(n*logn)
# n*n = 400000000
T(n)=(C+S)/(n^2)= 1.248484345(n^2)
-----SORT1 End-----
-----SORT1 Start-----
# n = 30000
# num compare(C) = 895320155
# num swap(S) = 225193077
# C+S = 1120513232
T(n)=(C+S)/(n)= 37350.441066666666(n)
# nlogn = 446180.2464081182
T(n)=(C+S)/(nlogn)= 2511.3465713026517(n*logn)
# n*n = 900000000
T(n)=(C+S)/(n^2)= 1.2450147022222222(n^2)
-----SORT1 End-----
-----SORT1 Start-----
# n = 40000
# num compare(C) = 1569560760
# num swap(S) = 400292933
# C+S = 1969853693
T(n)=(C+S)/(n)= 49246.342325(n)

```

```

# nlogn      = 611508.495181978
T(n)=(C+S)/(nlogn)= 3221.3022525775277 (n*logn)
# n*n       = 1600000000
T(n)=(C+S)/(n^2)= 1.231158558125 (n^2)
-----SORT1 End-----
-----SORT1 Start-----
# n =      10001
# num compare(C)      = 99420000
# num swap(S)         = 25008949
# C+S                 = 124428949
T(n)=(C+S)/(n)= 12441.650734926507 (n)
# nlogn              = 132891.85427504728
T(n)=(C+S)/(nlogn)= 936.3173512687125 (n*logn)
# n*n               = 100020001
T(n)=(C+S)/(n^2)= 1.2440406694257082 (n^2)
-----SORT1 End-----
-----SORT1 Start-----
# n =      20001
# num compare(C)      = 394400000
# num swap(S)         = 100312700
# C+S                 = 494712700
T(n)=(C+S)/(n)= 24734.398280085996 (n)
# nlogn              = 285769.9780344762
T(n)=(C+S)/(nlogn)= 1731.157007473739 (n*logn)
# n*n               = 400040001
T(n)=(C+S)/(n^2)= 1.2366580811002448 (n^2)
-----SORT1 End-----
-----SORT1 Start-----
# n =      30001
# num compare(C)      = 885570000
# num swap(S)         = 225636260
# C+S                 = 1111206260
T(n)=(C+S)/(n)= 37038.97403419886 (n)
# nlogn              = 446196.561802084
T(n)=(C+S)/(nlogn)= 2490.3962852427567 (n*logn)
# n*n               = 900060001
T(n)=(C+S)/(n^2)= 1.2345913147628031 (n^2)
-----SORT1 End-----
-----SORT1 Start-----
# n =      40001
# num compare(C)      = 1595000000
# num swap(S)         = 399640052
# C+S                 = 1994640052
T(n)=(C+S)/(n)= 49864.754681132974 (n)
# nlogn              = 611525.225607432
T(n)=(C+S)/(nlogn)= 3261.746152856918 (n*logn)
# n*n               = 1600080001
T(n)=(C+S)/(n^2)= 1.2465877023357659 (n^2)
-----SORT1 End-----
-----testing 5001 SORTED ASCENDING numbers-----
-----SORT1 Start-----
# n =      40001
# num compare(C)      = 1600040000
# num swap(S)         = 800019916
# C+S                 = 2400059916
T(n)=(C+S)/(n)= 59999.9979000525 (n)
# nlogn              = 611525.225607432

```

```

T(n)=(C+S)/(nlogn)= 3924.7112228543065 (n*logn)
# n*n      = 1600080001
T(n)=(C+S)/(n^2)= 1.4999624484401015 (n^2)
-----SORT1 End-----
-----testing 10001 SORTED ASCENDING numbers-----
-----SORT1 Start-----
# n =      40001
# num compare(C)      = 40000
# num swap(S)         = 0
# C+S                 = 40000
T(n)=(C+S)/(n)= 0.9999750006249843 (n)
# nlogn              = 611525.225607432
T(n)=(C+S)/(nlogn)= 0.06541022074807747 (n*logn)
# n*n                = 1600080001
T(n)=(C+S)/(n^2)= 2.4998750046873436E-5 (n^2)
-----SORT1 End-----
-----testing 15001 SORTED ASCENDING numbers-----
-----SORT1 Start-----
# n =      40001
# num compare(C)      = 40000
# num swap(S)         = 0
# C+S                 = 40000
T(n)=(C+S)/(n)= 0.9999750006249843 (n)
# nlogn              = 611525.225607432
T(n)=(C+S)/(nlogn)= 0.06541022074807747 (n*logn)
# n*n                = 1600080001
T(n)=(C+S)/(n^2)= 2.4998750046873436E-5 (n^2)
-----SORT1 End-----
-----testing 20001 SORTED ASCENDING numbers-----
-----SORT1 Start-----
# n =      40001
# num compare(C)      = 40000
# num swap(S)         = 0
# C+S                 = 40000
T(n)=(C+S)/(n)= 0.9999750006249843 (n)
# nlogn              = 611525.225607432
T(n)=(C+S)/(nlogn)= 0.06541022074807747 (n*logn)
# n*n                = 1600080001
T(n)=(C+S)/(n^2)= 2.4998750046873436E-5 (n^2)
-----SORT1 End-----
-----testing 25001 SORTED ASCENDING numbers-----
-----SORT1 Start-----
# n =      40001
# num compare(C)      = 40000
# num swap(S)         = 0
# C+S                 = 40000
T(n)=(C+S)/(n)= 0.9999750006249843 (n)
# nlogn              = 611525.225607432
T(n)=(C+S)/(nlogn)= 0.06541022074807747 (n*logn)
# n*n                = 1600080001
T(n)=(C+S)/(n^2)= 2.4998750046873436E-5 (n^2)
-----SORT1 End-----
-----DONE!-----

```