Practical -1

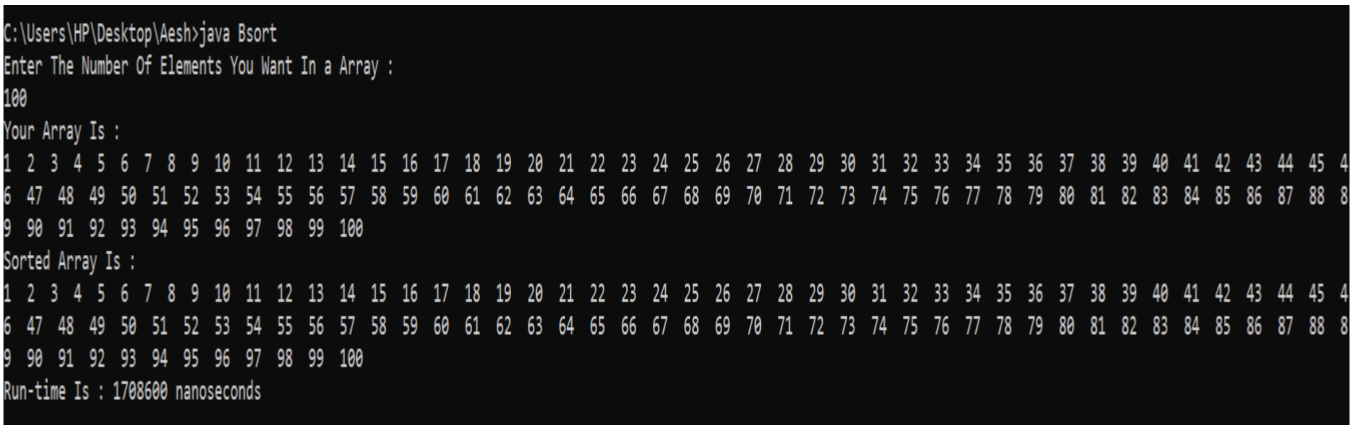
Aim: Write a program to sort given elements of an array in ascending order using bubble sort. Analyze the time complexity for best, average and worst case.

Code:

```
import java.util.Scanner;
//Program using bubble sort
class sort1
{
    void bsort(int[] a)
    {
        int n = a.length;
        for(int i=0;i<n;i++)
        {
            for(int j=i+1;j<n;j++)
            {
                if(a[i]>a[j])
                {
                    int temp = a[j];
                    a[j]=a[i];
                    a[i]=temp;
                }
            }
        }
    }
    void dis(int[] arr)
    {
        int n=arr.length;
        System.out.println("Sorted Array Is : ");
        for(int i=0;i<n;i++)
        {
            System.out.print(arr[i]+"  ");
        }
        System.out.println();
    }
}
class Bsort
{
    public static void main(String[] args)
    {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter The Number Of Elements You Want In a Array : ");
        int n=sc.nextInt();
        int[] a = new int [n];
        System.out.println("Enter The Values Of Elements In Array : ");
        for(int i=0;i<n;i++)
        {
            a[i]=sc.nextInt();
        }

        System.out.println("Your Array Is : ");
        for(int i=0;i<n;i++)
        {
            System.out.print(a[i]+"  ");
        }
        System.out.println();
        sort1 b = new sort1();
        long startTime = System.nanoTime();
        b.bsort(a);
        b.dis(a);
        long endTime = System.nanoTime();
        long totalTime = endTime - startTime;
        System.out.println("Run-time Is : "+totalTime+" nanoseconds");
    }
}
```

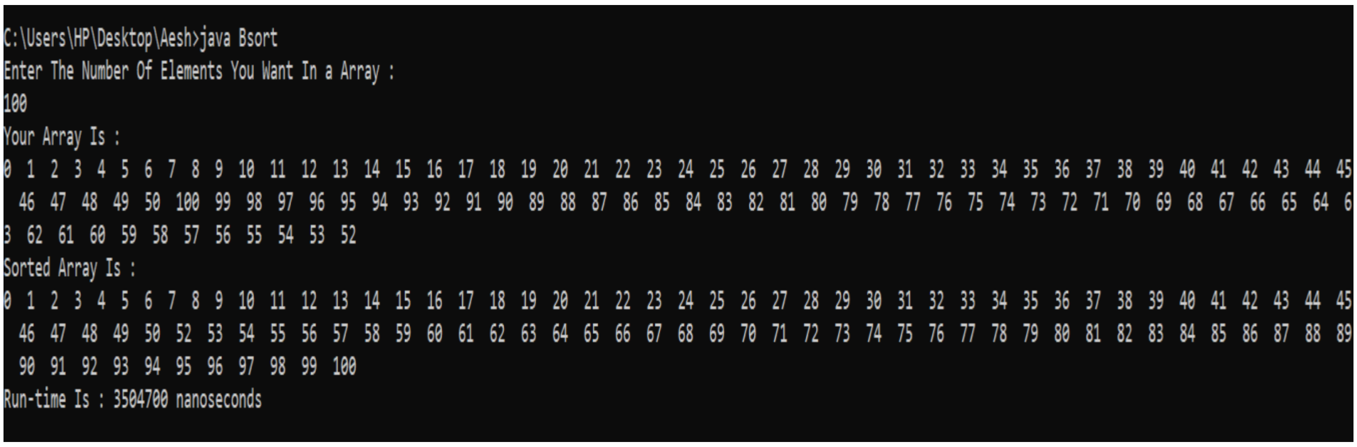
Output:

For Best Case:



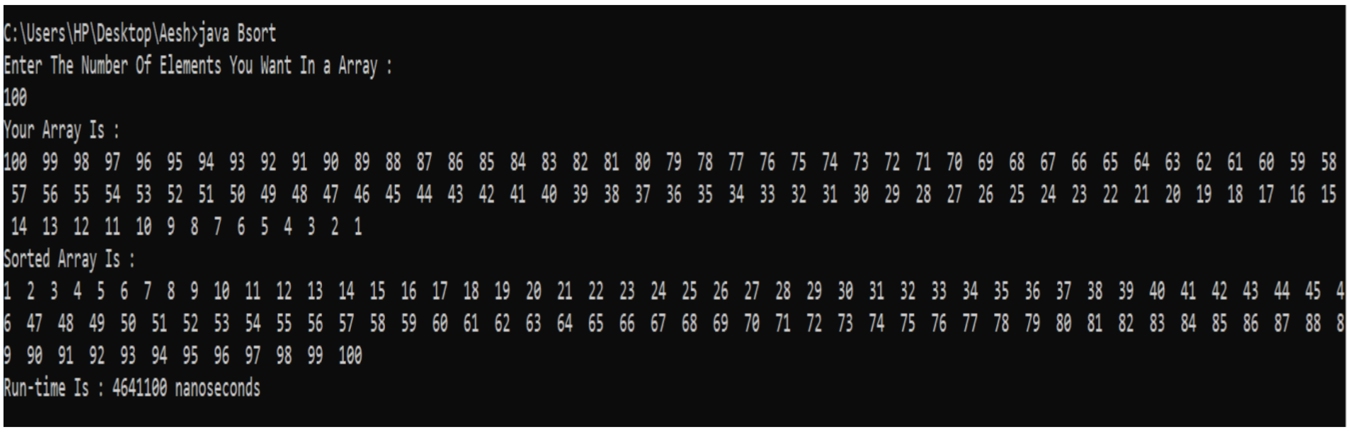
```
C:\Users\HP\Desktop\Aesh>java Bsort
Enter The Number Of Elements You Want In a Array :
100
Your Array Is :
1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  4
6  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  8
9  90  91  92  93  94  95  96  97  98  99  100
Sorted Array Is :
1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  4
6  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  8
9  90  91  92  93  94  95  96  97  98  99  100
Run-time Is : 1708600 nanoseconds
```

For Average Case:



```
C:\Users\HP\Desktop\Aesh>java Bsort
Enter The Number Of Elements You Want In a Array :
100
Your Array Is :
0  1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45
 46  47  48  49  50  100  99  98  97  96  95  94  93  92  91  90  89  88  87  86  85  84  83  82  81  80  79  78  77  76  75  74  73  72  71  70  69  68  67  66  65  64  6
3  62  61  60  59  58  57  56  55  54  53  52
Sorted Array Is :
0  1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45
 46  47  48  49  50  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89
 90  91  92  93  94  95  96  97  98  99  100
Run-time Is : 3504700 nanoseconds
```

For Worst Case:



```
C:\Users\HP\Desktop\Aesh>java Bsort
Enter The Number Of Elements You Want In a Array :
100
Your Array Is :
100  99  98  97  96  95  94  93  92  91  90  89  88  87  86  85  84  83  82  81  80  79  78  77  76  75  74  73  72  71  70  69  68  67  66  65  64  63  62  61  60  59  58
 57  56  55  54  53  52  51  50  49  48  47  46  45  44  43  42  41  40  39  38  37  36  35  34  33  32  31  30  29  28  27  26  25  24  23  22  21  20  19  18  17  16  15
 14  13  12  11  10  9  8  7  6  5  4  3  2  1
Sorted Array Is :
1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  4
6  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  8
9  90  91  92  93  94  95  96  97  98  99  100
Run-time Is : 4641100 nanoseconds
```

Practical -2

Aim: Write a program to sort given elements of an array in ascending order using selection sort. Analyse the time complexity for best, average, and worst case.

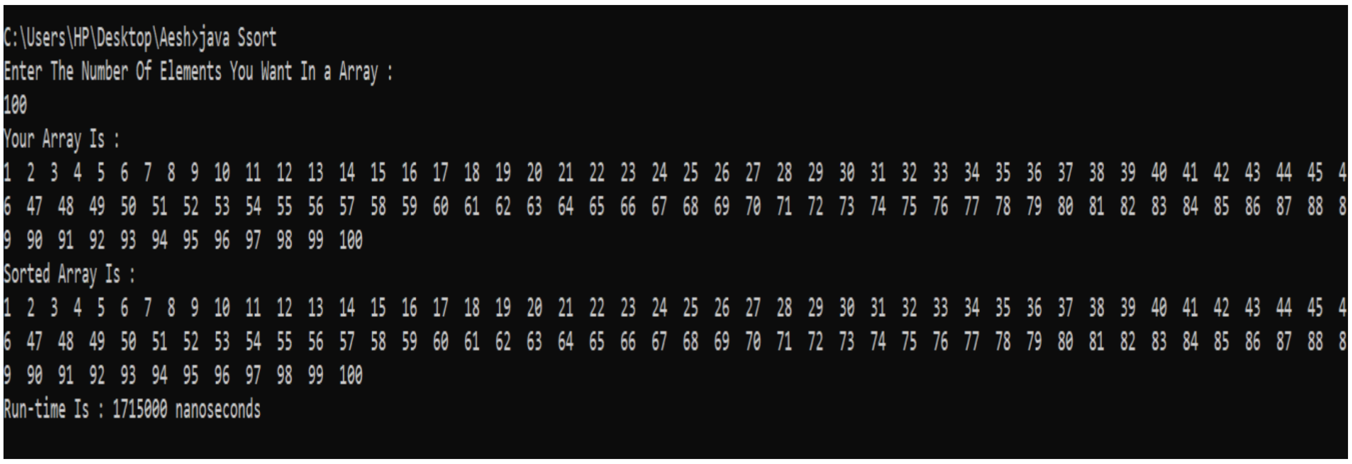
Code:

```
import java.util.Scanner;
//Program using selection sort
class sort2
{
   void ssort(int[] a)
   {
      int n = a.length;
      for(int i=0;i<n;i++)
      {
         int min = i;
         for(int j=i+1;j<n;j++)
         {
            if (a[min] > a[j])
            {
               min = j;
            }
         }
         int temp = a[min];
         a[min]=a[i];
```

```
            a[i]=temp;
        }
    }
    void dis(int[] arr)
    {
        int n=arr.length;
        System.out.println("Sorted Array Is : ");
        for(int i=0;i<n;i++)
        {
            System.out.print(arr[i]+"  ");
        }
        System.out.println();
    }
}
class Ssort
{
    public static void main(String[] args)
    {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter The Number Of Elements You Want In a Array : ");
        int n=sc.nextInt();
        int[] a = new int [n];
        System.out.println("Enter The Values Of Elements In Array : ");
        for(int i=0;i<n;i++)
        {
            a[i]=sc.nextInt();
        }
        System.out.println("Your Array Is : ");
        for(int i=0;i<n;i++)
        {
            System.out.print(a[i]+"  ");
        }
        System.out.println();
        sort2 b = new sort2();
        long startTime = System.nanoTime();
        b.ssort(a);
        b.dis(a);
        long endTime = System.nanoTime();
        long totalTime = endTime - startTime;
        System.out.println("Run-time Is : "+totalTime+" nanoseconds");
    }
}
```

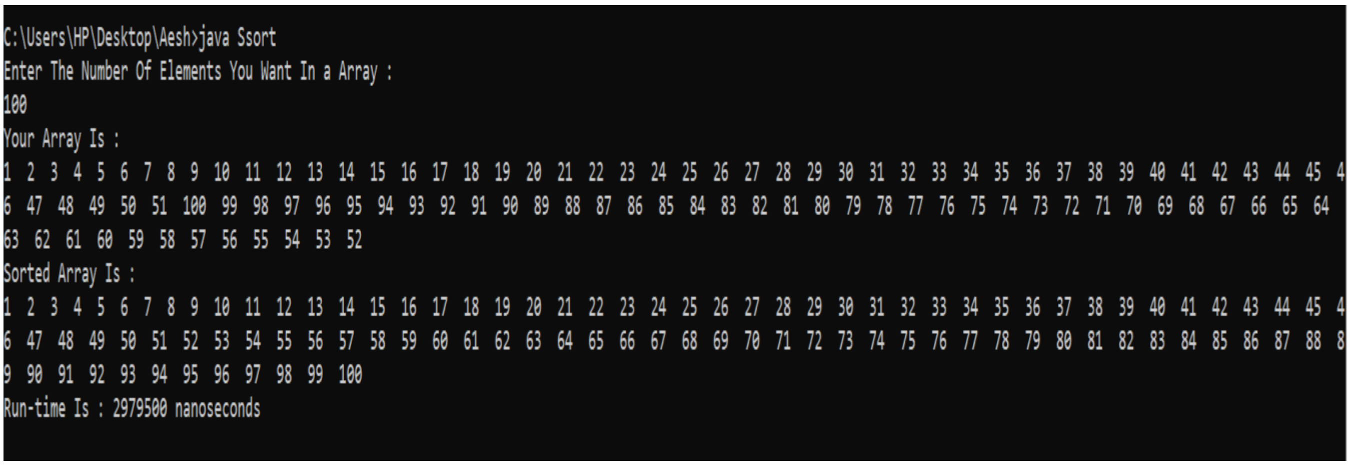
Output:

For Best Case:



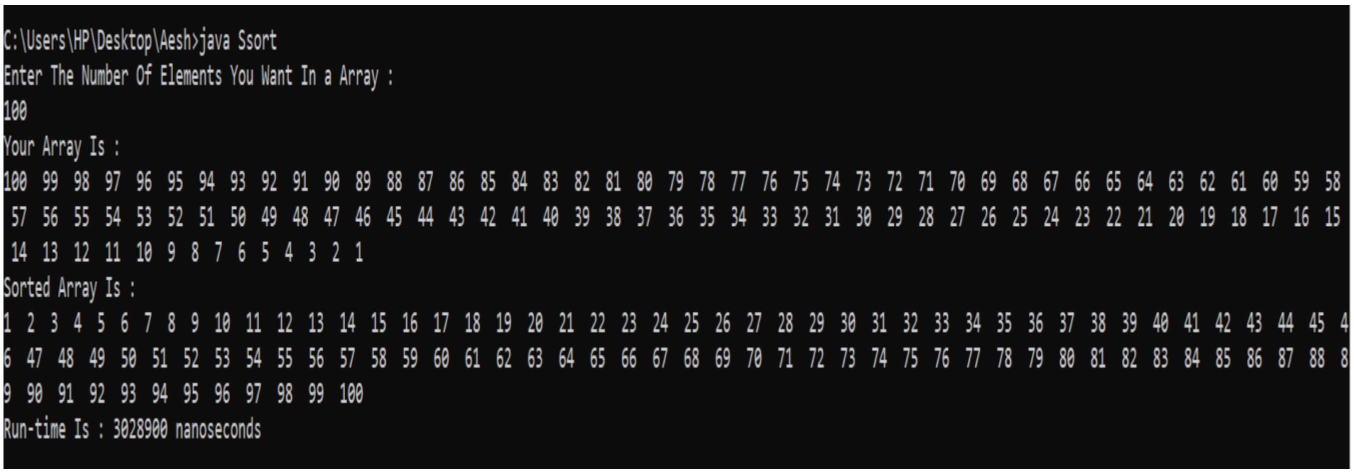
```
C:\Users\HP\Desktop\Aesh>java Ssort
Enter The Number Of Elements You Want In a Array :
100
Your Array Is :
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 4
6 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 8
9 90 91 92 93 94 95 96 97 98 99 100
Sorted Array Is :
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 4
6 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 8
9 90 91 92 93 94 95 96 97 98 99 100
Run-time Is : 1715000 nanoseconds
```

For Average Case:



```
C:\Users\HP\Desktop\Aesh>java Ssort
Enter The Number Of Elements You Want In a Array :
100
Your Array Is :
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 4
6 47 48 49 50 51 100 99 98 97 96 95 94 93 92 91 90 89 88 87 86 85 84 83 82 81 80 79 78 77 76 75 74 73 72 71 70 69 68 67 66 65 64
63 62 61 60 59 58 57 56 55 54 53 52
Sorted Array Is :
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 4
6 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 8
9 90 91 92 93 94 95 96 97 98 99 100
Run-time Is : 2979500 nanoseconds
```

For Worst Case:



```
C:\Users\HP\Desktop\Aesh>java Ssort
Enter The Number Of Elements You Want In a Array :
100
Your Array Is :
100 99 98 97 96 95 94 93 92 91 90 89 88 87 86 85 84 83 82 81 80 79 78 77 76 75 74 73 72 71 70 69 68 67 66 65 64 63 62 61 60 59 58
 57 56 55 54 53 52 51 50 49 48 47 46 45 44 43 42 41 40 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15
 14 13 12 11 10 9 8 7 6 5 4 3 2 1
Sorted Array Is :
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 4
6 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 8
9 90 91 92 93 94 95 96 97 98 99 100
Run-time Is : 3028900 nanoseconds
```

Practical-3

Aim: Write a program to implement Heap Sort.

Code:

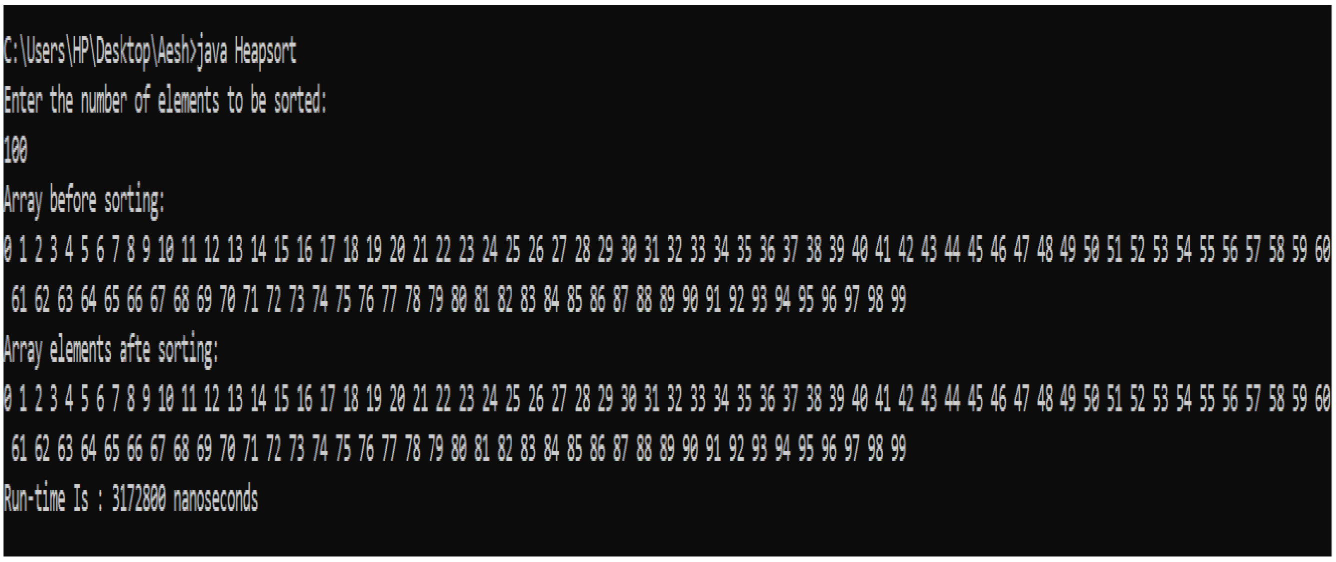
```
import java.util.Scanner;
class Array
{
    void heapify(int a[],int n,int i)
    {
        int largest=i;
        int left=2*i+1;
        int right=2*i+2;

        if(left<n && a[left]>a[largest])
        {
            largest=left;
        }
        if(right<n && a[right]>a[largest])
        {
            largest=right;

        }
        if(largest !=i)
        {
            int temp=a[i];
            a[i]=a[largest];
            a[largest]=temp;

            heapify(a,n,largest);
        }
    }


    void heapsort(int a[])
    {
        int n=a.length;
        for(int i=n/2-1;i>=0;i--)
        {
            heapify(a,n,i);
        }
        for(int i=n-1;i>=0;i--)
        {
            int temp=a[0];
            a[0]=a[i];
            a[i]=temp;

            heapify(a,i,0);
        }
    }

    void display(int a[])
    {
        int n=a.length;
        for(int i=0;i<n;i++)
        {
            System.out.print(a[i]+" ");

        }
        System.out.println();
    }
}
```

```
class Heapsort
{
   public static void main (String args[])
   {
      Array arr = new Array();
      Scanner sc = new Scanner(System.in);
      System.out.println("Enter the number of elements to be sorted: ");
      int n=sc.nextInt();
      int[] a=new int [n];
      int x=n;
      for (int i=0;i<n;i++)
      {
         if(i<50)
         {
            a[i]=i;
         }
         else
         {
            a[i]=x;
            x--;
         }

      }
      System.out.println("Array before sorting: ");
      arr.display(a);
      long startTime = System.nanoTime();
      arr.heapsort(a);
      System.out.println("Array elements after sorting: ");
      arr.display(a);
      long endTime = System.nanoTime();
```

```
        long totalTime = endTime - startTime;

        System.out.println("Run-time Is : "+totalTime+" nanoseconds");

    }

}
```

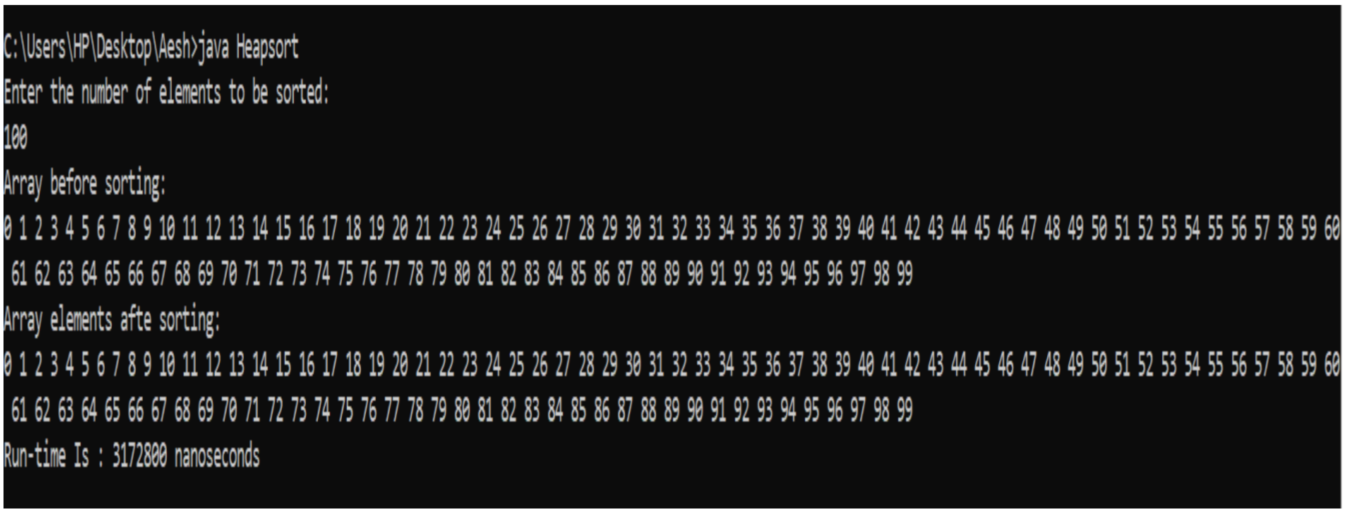
Output:

Best Case:



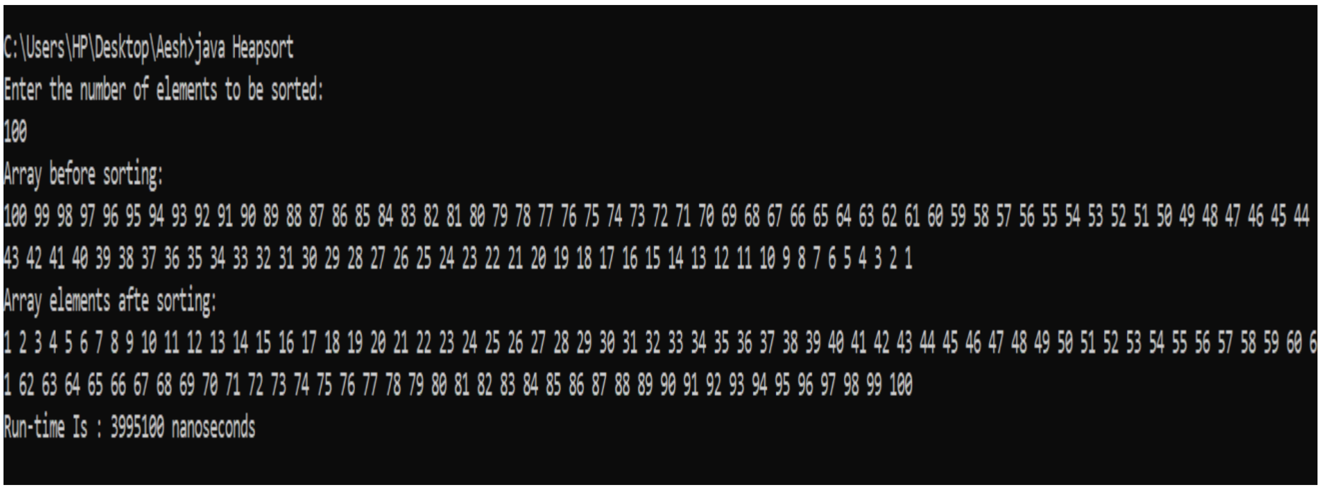
```
C:\Users\HP\Desktop\Aesh>java Heapsort
Enter the number of elements to be sorted:
100
Array before sorting:
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60
 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99
Array elements afte sorting:
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60
 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99
Run-time Is : 3172800 nanoseconds
```

Average Case:



```
C:\Users\HP\Desktop\Aesh>java Heapsort
Enter the number of elements to be sorted:
100
Array before sorting:
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60
 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99
Array elements afte sorting:
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60
 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99
Run-time Is : 3172800 nanoseconds
```

Worst Case:



```
C:\Users\HP\Desktop\Aesh>java Heapsort
Enter the number of elements to be sorted:
100
Array before sorting:
100 99 98 97 96 95 94 93 92 91 90 89 88 87 86 85 84 83 82 81 80 79 78 77 76 75 74 73 72 71 70 69 68 67 66 65 64 63 62 61 60 59 58 57 56 55 54 53 52 51 50 49 48 47 46 45 44
43 42 41 40 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1
Array elements afte sorting:
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 6
1 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100
Run-time Is : 3995100 nanoseconds
```