# SORTING ALGORITHMS

## Program for Bubble sort, Selection sort, Insertion sort of following type of arrays:

**1)Random array**

**2)Sorted array**

**3)Reverse sorted array**

**Source code:**

```
import random
import time
def insertionsort(mylist):
    n=len(mylist)
    for i in range(1,n):
        key=mylist[i]
        j=i-1
        while mylist[j]>key and j>-1:
            mylist[j+1]=mylist[j]
            j=j-1
        mylist[j+1]=key


def bubblesort(mylist):
    n=len(mylist)
    for i in range(n):
        for j in range(n-i-1):
            if mylist[j]>mylist[j+1]:
                temp=mylist[j]
                mylist[j]=mylist[j+1]
                mylist[j+1]=temp
```

```
def selectionsort(mylist):
   n=len(mylist)
   for i in range(n):
      for j in range(i,n):
         if mylist[i]>mylist[j]:
            temp=mylist[i]
            mylist[i]=mylist[j]
            mylist[j]=temp
def calc1(mylist):
   start=time.time()
   bubblesort(mylist)
   end=time.time()
   exec_time=end-start
   list1.append(exec_time)
   start=time.time()
   insertionsort(mylist)
   end=time.time()
   exec_time=end-start
   list2.append(exec_time)
   start=time.time()
   selectionsort(mylist)
   end=time.time()
   exec_time=end-start
   list3.append(exec_time)
def calc2(mylist):
   start=time.time()
   bubblesort(mylist)
   end=time.time()
```

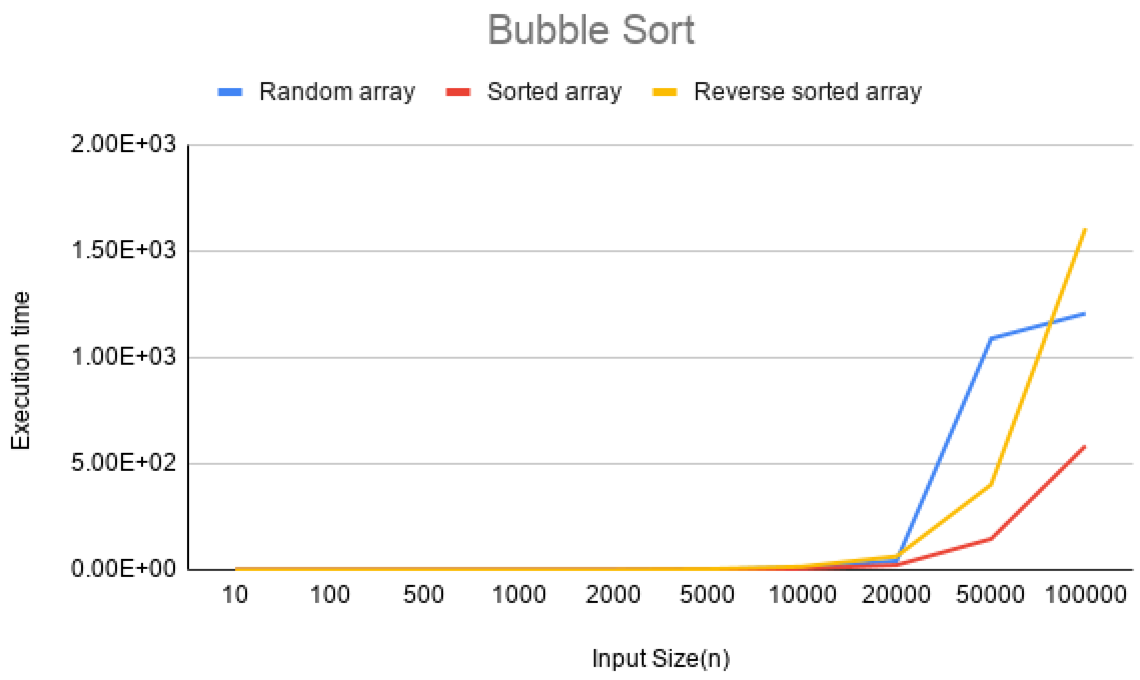
```
    exec_time=end-start
    list4.append(exec_time)
    start=time.time()
    insertionsort(mylist)
    end=time.time()
    exec_time=end-start
    list5.append(exec_time)
    start=time.time()
    selectionsort(mylist)
    end=time.time()
    exec_time=end-start
    list6.append(exec_time)

def calc3(mylist):
    start=time.time()
    bubblesort(mylist)
    end=time.time()
    exec_time=end-start
    list7.append(exec_time)
    start=time.time()
    insertionsort(mylist)
    end=time.time()
    exec_time=end-start
    list8.append(exec_time)
    start=time.time()
    selectionsort(mylist)
    end=time.time()
    exec_time=end-start
    list9.append(exec_time)
```

```
tc=int(input())
list1=[]
list2=[]
list3=[]
list4=[]
list5=[]
list6=[]
list7=[]
list8=[]
list9=[]

for i in range(tc):
    mylist=[]
    size=int(input())
    for i in range(size):
        b=random.randrange(1,100)
        mylist.append(b)
    calc1(mylist)
    mylist.sort()
    calc2(mylist)
    mylist.reverse()
    calc3(mylist)
print(list1)
print(list2)
print(list3)
print(list4)
print(list5)
print(list6)
print(list7)

print(list8)

print(list9)
```

## Observation table of Bubble sort:

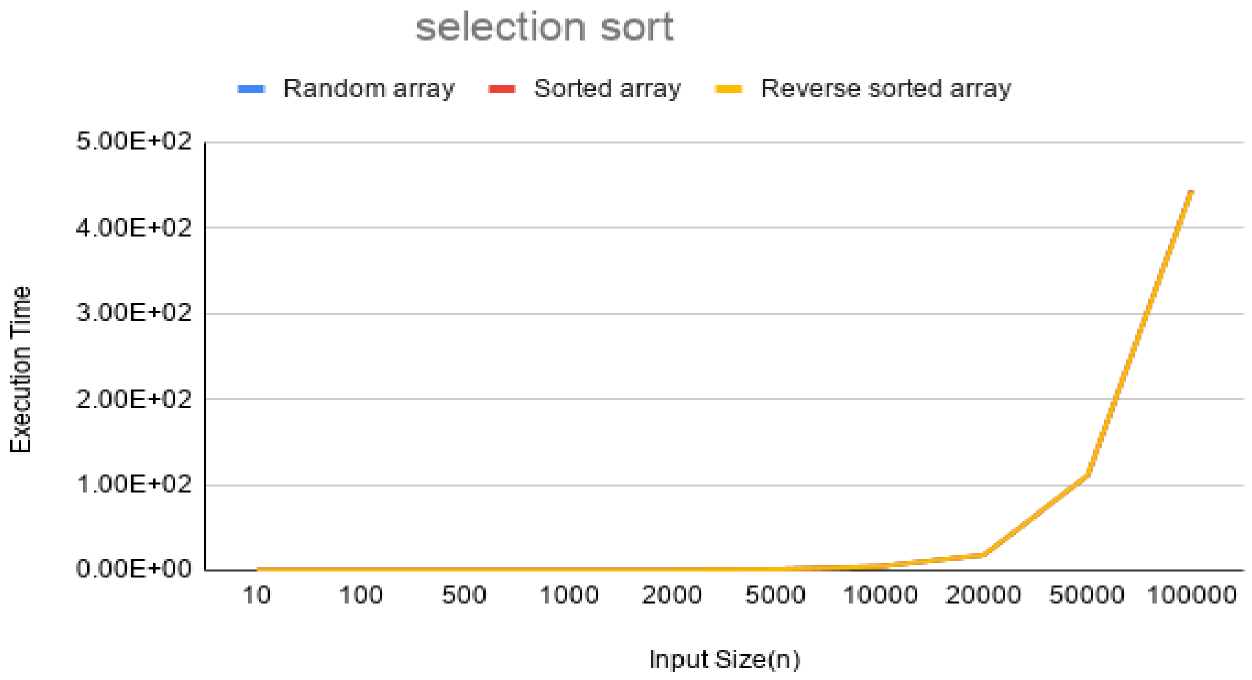
| Input Size(n) | Random array | Sorted array | Reverse sorted array |
|---|---|---|---|
| 10 | 1.5735626220703125e-05 | 8.106231689453125e-06 | 1.52587890625e-05 |
| 100 | 0.0016443729400634766 | 0.0005116462707519531 | 0.0013167858123779297 |
| 500 | 0.023614406585693336 | 0.012209415435791016 | 0.03322649002075195 |
| 1000 | 0.09896254539489746 | 0.05207419395446777 | 0.14172744750976562 |
| 2000 | 0.40331411361694336 | 0.21912312507629395 | 0.6100995540618896 |
| 5000 | 2.7132110595703125 | 1.4203836917877197 | 3.9436004161834717 |
| 10000 | 10.554478168487549 | 5.600391626358032 | 15.82536506652832 |
| 20000 | 42.7580840587616 | 22.957401514053345 | 63.900449991226196 |
| 50000 | 1087.579746246338 | 145.68381309509277 | 401.46946001052856 |

| | | | |
|---|---|---|---|
| 100000 | 1205.3558273315430 | 584.2181475162506 | 1606.9161143302917 |



## Observation table of Selection sort:

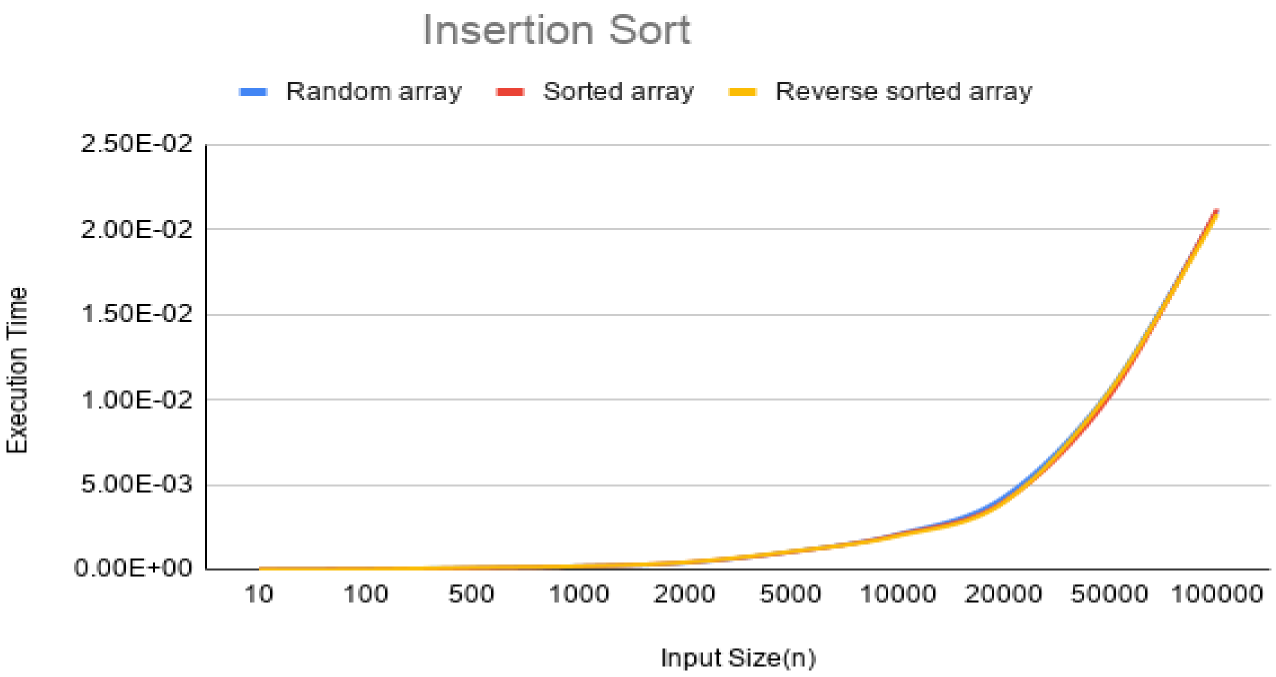
| Input Size(n) | Random array | Sorted array | Reverse sorted array |
|---|---|---|---|
| 10 | 9.298324584960938e-06 | 8.106231689453125e-06 | 7.62939453125e-06 |
| 100 | 0.002482175827026367 | 0.00048160552978515625 | 0.0004029273986816406 |
| 500 | 0.010422468185424805 | 0.01037907600402832 | 0.010891914367675781 |
| 1000 | 0.0422823429107666 | 0.04233264923095703 | 0.04248666763305664 |
| 2000 | 0.1808764934539795 | 0.17335247993469238 | 0.17928314208984375 |
| 5000 | 1.088284969329834 | 1.090080738067627 | 1.1104421615600586 |
| 10000 | 4.353042364120483 | 4.424070596694946 | 4.342396020889282 |
| 20000 | 17.565318822860718 | 17.603698253631592 | 17.775641918182373 |
| 50000 | 110.56216216087341 | 110.79182982444763 | 110.83180069923401 |
| 100000 | 443.48854899406433 | 444.07488083839417 | 442.5390658378601 |



## Observation table of Insertion sort:

| Input Size(n) | Random array | Sorted array | Reverse sorted array |
|---|---|---|---|
| 10 | 4.0531158447265625e-06 | 2.384185791015625e-06 | 2.384185791015625e-06 |
| 100 | 3.790855407714844e-05 | 1.9073486328125e-05 | 1.8596649169921875e-05 |
| 500 | 9.465217590332031e-05 | 9.179115295410156e-05 | 0.00011420249938964844 |
| 1000 | 0.00019502639770507812 | 0.0001902580261230468 | 0.00019264221191406250 |
| 2000 | 0.0003905296325683594 | 0.0004119873046875 | 0.00042033195495605470 |
| 5000 | 0.001042842864990234 | 0.00102996826171875 | 0.00105762481689453120 |
| 10000 | 0.002075910568237304 | 0.00203704833984375 | 0.00197625160217285160 |
| 20000 | 0.004269838333129883 | 0.004004240036010742 | 0.003949880599975586 |
| 50000 | 0.010607004165649414 | 0.010282039642333984 | 0.010552167892456055 |

| | | | |
|---|---|---|---|
| 100000 | 0.021106004714965820 | 0.0212059020996093750 | 0.0208961963653564450 |



## Conclusion:

The Time complexity of Bubble sort for random array O(n^2)

The Time complexity of Bubble sort for sorted array O(n)

The Time complexity of Bubble sort for reverse sorted array O(n^2)

The Time complexity of Selection sort for random array O(n^2)

The Time complexity of Selection sort for sorted array O(n^2)

The Time complexity of Selection sort for reverse sorted array O(n^2)

The Time complexity of Insertion sort for random array O(n^2)

The Time complexity of Insertion sort for sorted array O(n)

The Time complexity of Insertion sort for reverse sorted array O(n^2)

Time complexity of insertion sort is better than selection and insertion sort so it is more efficient than the other two sortings