Design Algorithm and Analysis

Lab – 2: Selection Sort

```
Aim :
To Perform Selection Sort on the Randomly generated arrays
```

```
Introduction :

It finds the smallest value among the unsorted elements of the array is passed
and every pass is inserted into the appropriate position into an array.
However, it is an in-place comparison sorting algorithm meaning it divide the
array into 2 parts first is sorted and the other is unsorted while sorted is
empty and unsorted is our array. the worst case and average case is O(n^2) .
used when the array is small or swapping cost doesn't matter as it checks all
elements compulsorily
```

```
Algorithm

SELECTION SORT(arr, n)

Step 1: Repeat Steps 2 and 3 for i = 0 to n-1
Step 2: CALL SMALLEST(arr, i, n, pos)
Step 3: SWAP arr[i] with arr[pos]
[END OF LOOP]
Step 4: EXIT

SMALLEST (arr, i, n, pos)
Step 1: [INITIALIZE] SET SMALL = arr[i]
Step 2: [INITIALIZE] SET pos = i
Step 3: Repeat for j = i+1 to n
if (SMALL > arr[j])
     SET SMALL = arr[j]
SET pos = j
[END OF if]
[END OF LOOP]
Step 4: RETURN pos
```

```
For Average Case :

import numpy
import time

arr = numpy.random.randint(100, size=(10))
start_time = time.time()

def selection(arr):
    count=0
    for i in range(len(arr)):
        min = i
        for j in range(i+1, len(arr)):
            count += 1
            if arr[min] > arr[j]:
                min = j
        arr[i], arr[min] = arr[min], arr[i]
    return count

def printArr(a):
    for i in range(len(a)):
        print (a[i], end = " ")

print("Before sorting array elements are - ")
printArr(arr)
count = selection(arr)
print("\nAfter sorting array elements are - ")
printArr(arr)
print("\nNumber of iterations: ", count)
print("\nTime taken: ", time.time() - start_time)

Before sorting array elements are –
25 73 28 41 41 57 89 73 68 73
After sorting array elements are –
25 28 41 41 57 68 73 73 73 89
Number of iterations: 45 Time taken: 0.0
```

For Best Case :

```
import numpy
import time

a = numpy.random.randint(100, size=(10))
arr = numpy.sort(a)
start_time = time.time()

def selection(arr):
    count=0
    for i in range(len(arr)):
        min = i
        for j in range(i+1, len(arr)):
            count += 1
            if arr[min] > arr[j]:
                min = j
        arr[i], arr[min] = arr[min], arr[i]
    return count

def printArr(a):
    for i in range(len(a)):
        print (a[i], end = " ")

print("Before sorting array elements are - ")
printArr(arr)
count = selection(arr)
print("\nAfter sorting array elements are - ")
printArr(arr)
print("\nNumber of iterations: ", count)
print("\nTime taken: ", time.time() - start_time)
```

```
Before sorting array elements are –
1 2 3 4 4 6 6 7 8 12 13 14 16 16 17 17 18 18 18 19 19 21 21 23 23 25 25 26 28
28 30 30 32 33 35 36 39 40 41 41 42 43 45 45 46 48 49 49 49 51 52 53 53 54 54
56 56 56 57 57 57 57 59 59 60 61 62 63 66 67 68 71 71 72 73 77 80 81 83 84 86
86 87 88 89 89 90 90 92 93 93 93 94 94 96 96 97 97 98 99
After sorting array elements are –
1 2 3 4 4 6 6 7 8 12 13 14 16 16 17 17 18 18 18 19 19 21 21 23 23 25 25 26 28
28 30 30 32 33 35 36 39 40 41 41 42 43 45 45 46 48 49 49 49 51 52 53 53 54 54
56 56 56 57 57 57 57 59 59 60 61 62 63 66 67 68 71 71 72 73 77 80 81 83 84 86
86 87 88 89 89 90 90 92 93 93 93 94 94 96 96 97 97 98 99
Number of iterations: 4950
Time taken: 0.0009987354278564453
```

For Worst case :

```
import numpy
import time

a = numpy.random.randint(100, size=(10))
b = numpy.sort(a)
arr = numpy.flip(b)
start_time = time.time()

def selection(arr):
    count=0
    for i in range(len(arr)):
        min = i
        for j in range(i+1, len(arr)):
            count += 1
            if arr[min] > arr[j]:
                min = j
        arr[i], arr[min] = arr[min], arr[i]
    return count

def printArr(a):
    for i in range(len(a)):
        print (a[i], end = " ")

print("Before sorting array elements are - ")
printArr(arr)
count = selection(arr)
print("\nAfter sorting array elements are - ")
printArr(arr)
print("\nNumber of iterations: ", count)
print("\nTime taken: ", time.time() - start_time)
```

```
Before sorting array elements are - 99 99 99 99 99 99 99 98 98 98 98 98 98 98 98 98 98 97 97 97 97 97
97 96 96 96 96 96 96 96 95 95 95 95 95 95 95 95 95 95 94 94 94 94 94 94 94 94 94 94 94 94 94 93 93 93
93 93 93 93 93 93 92 92 92 92 92 92 92 92 92 92 92 92 91 91 91 91 91 91 91 90 90 90 90 90 90 90 90
89 89 89 89 89 89 89 88 88 88 88 88 88 88 88 88 88 88 88 87 87 87 87 87 87 87 86 86 86 86 86 86
86 86 85 85 85 85 85 85 85 84 84 84 84 84 84 84 84 83 83 83 83 83 83 83 83 83 83 83 83 83 83 82 82
82 82 82 82 82 82 81 81 81 81 81 81 80 80 80 80 80 80 80 80 80 80 80 80 79 79 79 79 79 79 79 79 79 78
78 78 78 78 78 78 78 77 77 77 77 77 77 77 77 76 76 76 76 76 76 76 76 76 76 76 76 76 76 76 75 75 75 75
75 75 75 75 75 74 74 74 74 74 74 74 73 73 73 73 73 73 73 73 73 73 72 72 72 72 72 72 72 72 72 72 72 71
71 71 70 70 70 70 70 70 70 70 70 70 69 69 69 69 69 69 69 69 68 68 68 68 68 67 67 67 67 67 67 67 67 67
67 67 67 67 67 66 66 66 66 66 66 66 66 65 65 65 65 65 65 65 65 65 65 65 64 64 64 64 64 64 64 64 64 63 63
63 63 63 63 63 63 63 62 62 62 62 62 62 62 62 62 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 60
60 60 60 60 60 60 60 60 60 60 60 59 59 59 59 59 59 59 59 59 59 59 59 58 58 58 58 58 58 58 58 58 58
58 57 57 57 57 57 57 57 56 56 56 56 56 56 56 55 55 55 55 55 55 55 55 55 55 55 55 55 55 54 54 54 54 54
54 54 54 54 54 54 54 53 53 53 53 53 53 53 53 53 52 52 52 52 52 52 52 52 52 52 52 52 52 51 51 51 51 51
51 51 51 51 51 50 50 50 50 50 50 50 50 50 49 49 49 48 48 48 48 48 48 48 48 48 48 48 48 47 47 47 47 47
```

```
47 47 46 46 46 46 46 46 46 46 46 45 45 45 45 45 45 45 45 45 45 45 44 44 44 44 44 44 44 44 44 44 44 44
44 44 43 43 43 43 43 43 43 43 43 43 43 43 42 42 42 42 42 42 42 42 42 42 42 41 41 41 41 41 41 41 41 41
41 40 40 40 40 40 40 40 40 40 39 39 39 39 39 39 39 39 39 39 39 39 39 38 38 38 38 38 38 38 38 38 38 38
38 38 38 38 38 37 37 37 37 37 37 37 37 37 36 36 36 36 36 36 36 36 36 36 36 35 35 35 35 35 35 35 35 34
34 34 34 34 34 34 34 34 34 34 34 34 33 33 33 33 33 33 33 33 33 33 32 32 32 32 32 32 32 32 32 32 32 31 31
31 31 31 31 31 31 31 31 30 30 30 30 30 30 29 29 29 29 29 29 29 29 29 29 29 29 29 29 29 29 28 28 28 28 28
28 28 28 28 28 28 28 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 26 26 26 26 26 26 26 26 26 26 25 25
25 25 25 25 25 25 25 25 25 25 25 25 25 24 24 24 24 24 24 24 24 24 24 24 24 24 23 23 23 23 23 23 23 22
22 22 22 22 22 22 22 22 22 21 21 21 21 21 21 21 21 21 21 20 20 20 20 20 20 20 20 19 19 19 19 19 19
19 19 19 19 18 18 18 18 18 18 18 18 18 18 17 17 17 17 17 17 16 16 16 16 16 16 16 16 16 16 16 16 16 16
15 15 15 15 15 15 15 14 14 14 14 14 14 14 14 14 14 14 14 13 13 13 13 13 13 13 12 12 12 12 12 12 12 12
12 12 12 12 12 12 12 12 12 11 11 11 11 11 11 11 11 11 10 10 10 10 10 10 10 10 9 9 9 9 9 8 8 8 8
8 8 8 8 8 8 8 8 8 7 7 7 7 7 7 7 7 6 6 6 6 6 6 6 6 6 6 5 5 5 5 5 5 5 5 5 5 5 4 4 4 4 4 4 4 4 3 3 3 3
3 3 3 3 3 3 3 3 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0
After sorting array elements are - 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 3 3
3 3 3 3 3 3 3 3 3 4 4 4 4 4 4 4 4 5 5 5 5 5 5 5 5 5 5 5 6 6 6 6 6 6 6 6 6 7 7 7 7 7 7 7 7 8 8 8
8 8 8 8 8 8 8 8 8 9 9 9 9 9 10 10 10 10 10 10 10 10 10 11 11 11 11 11 11 11 11 11 12 12 12 12 12
12 12 12 12 12 12 12 12 12 12 12 13 13 13 13 13 13 13 14 14 14 14 14 14 14 14 14 14 15 15 15
15 15 15 15 16 16 16 16 16 16 16 16 16 16 16 16 16 17 17 17 17 17 17 18 18 18 18 18 18 18 18 18
19 19 19 19 19 19 19 19 19 20 20 20 20 20 20 20 20 21 21 21 21 21 21 21 21 21 21 22 22 22 22 22
22 22 22 22 22 23 23 23 23 23 23 23 24 24 24 24 24 24 24 24 24 24 24 24 25 25 25 25 25 25 25 25 25
25 25 25 25 25 25 26 26 26 26 26 26 26 26 26 26 27 27 27 27 27 27 27 27 27 27 27 27 27 27 28 28 28
28 28 28 28 28 28 28 28 29 29 29 29 29 29 29 29 29 29 29 29 29 29 30 30 30 30 30 30 31 31 31 31
31 31 31 31 31 31 32 32 32 32 32 32 32 32 32 32 32 33 33 33 33 33 33 33 33 33 34 34 34 34 34 34 34 34
34 34 34 34 34 35 35 35 35 35 35 35 35 36 36 36 36 36 36 36 36 36 36 36 37 37 37 37 37 37 37 37 37 38
38 38 38 38 38 38 38 38 38 38 38 38 38 39 39 39 39 39 39 39 39 39 39 39 39 39 40 40 40 40 40 40
40 40 40 41 41 41 41 41 41 41 41 41 42 42 42 42 42 42 42 42 42 42 42 43 43 43 43 43 43 43 43 43 43
43 43 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 45 45 45 45 45 45 45 45 45 45 46 46 46 46 46 46 46
46 46 47 47 47 47 47 47 47 48 48 48 48 48 48 48 48 48 48 48 48 49 49 49 50 50 50 50 50 50 50 50 50 51
51 51 51 51 51 51 51 51 51 52 52 52 52 52 52 52 52 52 52 52 52 52 53 53 53 53 53 53 53 53 54 54 54
54 54 54 54 54 54 54 54 55 55 55 55 55 55 55 55 55 55 55 55 55 55 56 56 56 56 56 56 56 57 57 57 57
57 57 57 58 58 58 58 58 58 58 58 58 58 58 58 59 59 59 59 59 59 59 59 59 59 59 60 60 60 60 60 60 60
60 60 60 60 60 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 62 62 62 62 62 62 62 62 62 63 63 63 63
63 63 63 63 63 64 64 64 64 64 64 64 64 65 65 65 65 65 65 65 65 65 65 66 66 66 66 66 66 66 66 67
67 67 67 67 67 67 67 67 67 67 67 67 68 68 68 68 68 69 69 69 69 69 69 69 69 70 70 70 70 70 70 70
70 70 71 71 71 72 72 72 72 72 72 72 72 72 72 72 73 73 73 73 73 73 73 73 73 73 74 74 74 74 74 74 74 75
75 75 75 75 75 75 75 75 76 76 76 76 76 76 76 76 76 76 76 76 76 76 76 76 77 77 77 77 77 77 77 78 78 78
78 78 78 78 78 79 79 79 79 79 79 79 79 79 80 80 80 80 80 80 80 80 80 80 80 80 81 81 81 81 81 81 82 82
82 82 82 82 82 83 83 83 83 83 83 83 83 83 83 83 83 83 84 84 84 84 84 84 84 85 85 85 85 85 85
85 85 86 86 86 86 86 86 86 87 87 87 87 87 87 87 88 88 88 88 88 88 88 88 88 88 88 88 89 89 89
89 89 89 89 90 90 90 90 90 90 90 90 91 91 91 91 91 91 91 92 92 92 92 92 92 92 92 92 92 92 93 93
93 93 93 93 93 93 94 94 94 94 94 94 94 94 94 94 94 94 94 94 95 95 95 95 95 95 95 95 95 95 96 96 96 96
96 96 96 97 97 97 97 97 97 98 98 98 98 98 98 98 98 98 99 99 99 99 99 99 99
Number of iterations: 499500
Time taken: 0.07754039764404297
```

Result :

N	Comparison	Worst Case (in sec)	Comparison	Best Case (in sec)	Comparison	Average Case (in sec)
10	45	0	45	0	45	0
100	4950	0.00199913	4950	0.00099945	4950	0.00101161
1000	499500	0.073531866	499500	0.075998067	499500	0.0833742
10000	49995000	7.123580932	49995000	7.27520465	49995000	6.950613498
100000	499950000	724.56	499950000	752.44	499950000	718.8

Applications :

Facial Recogintion
Google Search
Scheduling - transport network

References :

Selection Sort: A Comparative Study
by H. L. Bodla and S. K. Sharma