Bharatiya Vidya Bhavan's

SARDAR PATEL INSTITUTE OF TECHNOLOGY

(Autonomous Institute Affiliated to University of Mumbai)

.Munshi Nagar, Andheri (W), Mumbai – 400 058.

Name – Advika Kharat
UID – 2021300060
Subject – DAA
Class – SE Comps A

Experiment No. 1-b

Aim – Experiment on finding the running time of an algorithm.

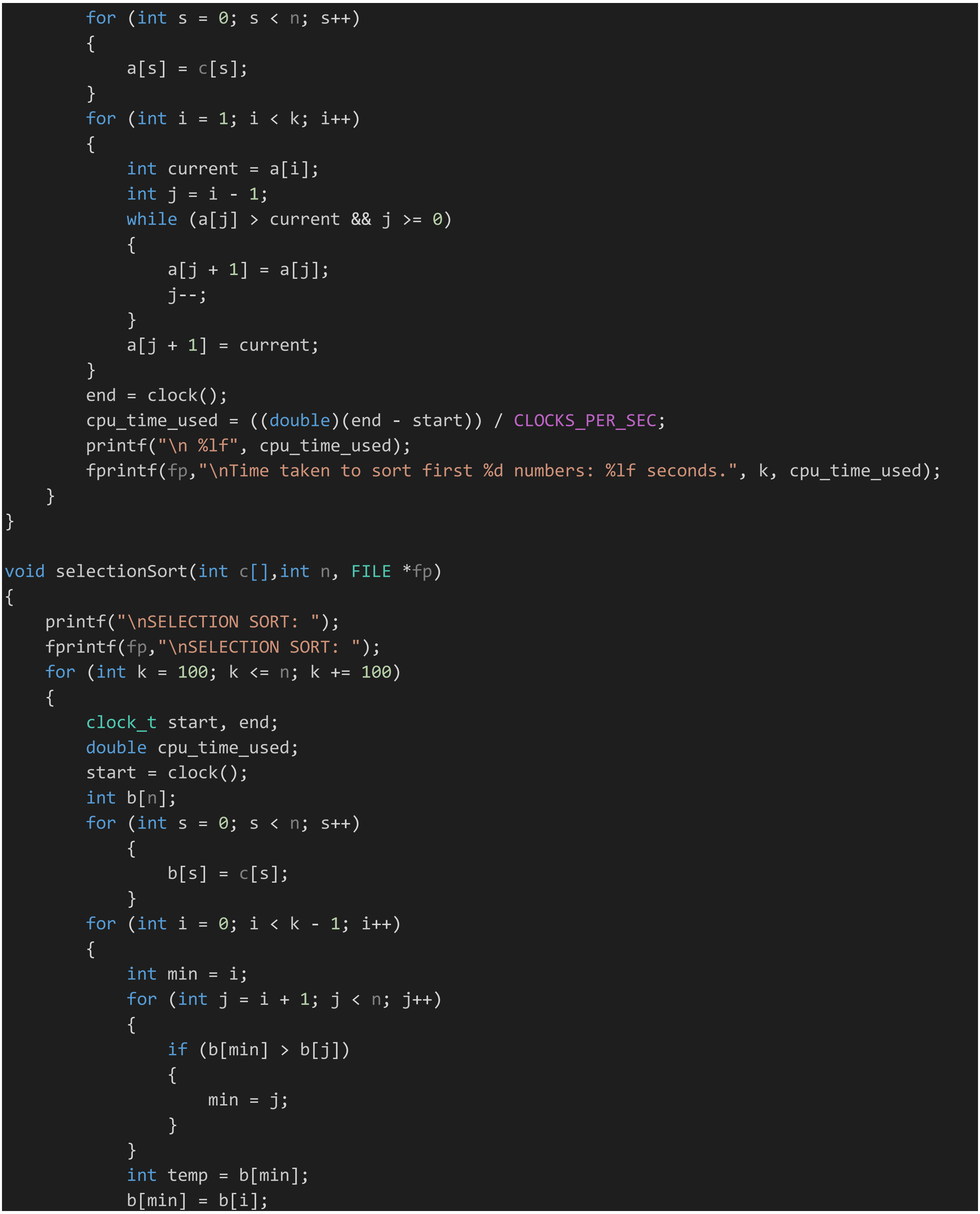
Details – The understanding of running time of algorithms is explored by implementing two basic sorting algorithms namely Insertion and Selection sorts. These algorithms work as follows.

Insertion sort– It works similar to the sorting of playing cards in hands. It is assumed that the first card is already sorted in the card game, and then we select an unsorted card. If the selected unsorted card is greater than the first card, it will be placed at the right side; otherwise, it will be placed at the left side. Similarly, all unsorted cards are taken and put in their exact place.

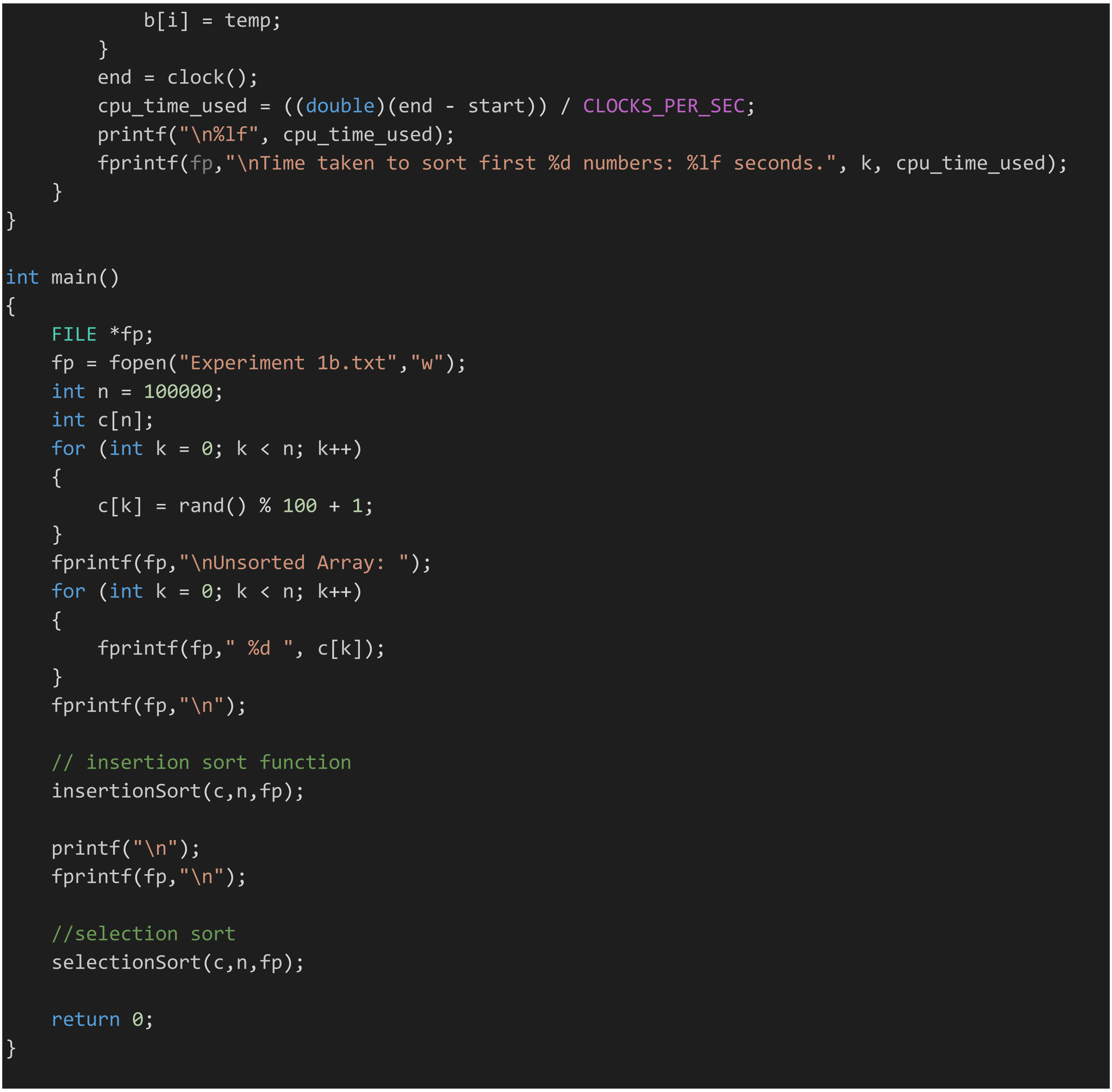
Selection sort– It first finds the smallest value among the unsorted elements of the array is selected in every pass and inserted to its appropriate position into the array. In this algorithm, the array is divided into two parts, first is sorted part, and another one is the unsorted part. Initially, the sorted part of the array is empty, and unsorted part is the given array. Sorted part is placed at the left, while the unsorted part is placed at the right. In selection sort, the first smallest element is selected from the unsorted array and placed at the first position. After that second smallest element is selected and placed in the second position. The process continues until the array is entirely sorted.

Code:

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

void insertionSort(int c[], int n, FILE *fp)
{
    printf("\nINSERTION SORT: ");
    fprintf(fp,"\nINSERTION SORT: ");
    for (int k = 100; k <= n; k += 100)
    {
        clock_t start, end;
        double cpu_time_used;
        start = clock();
        int a[n];
```



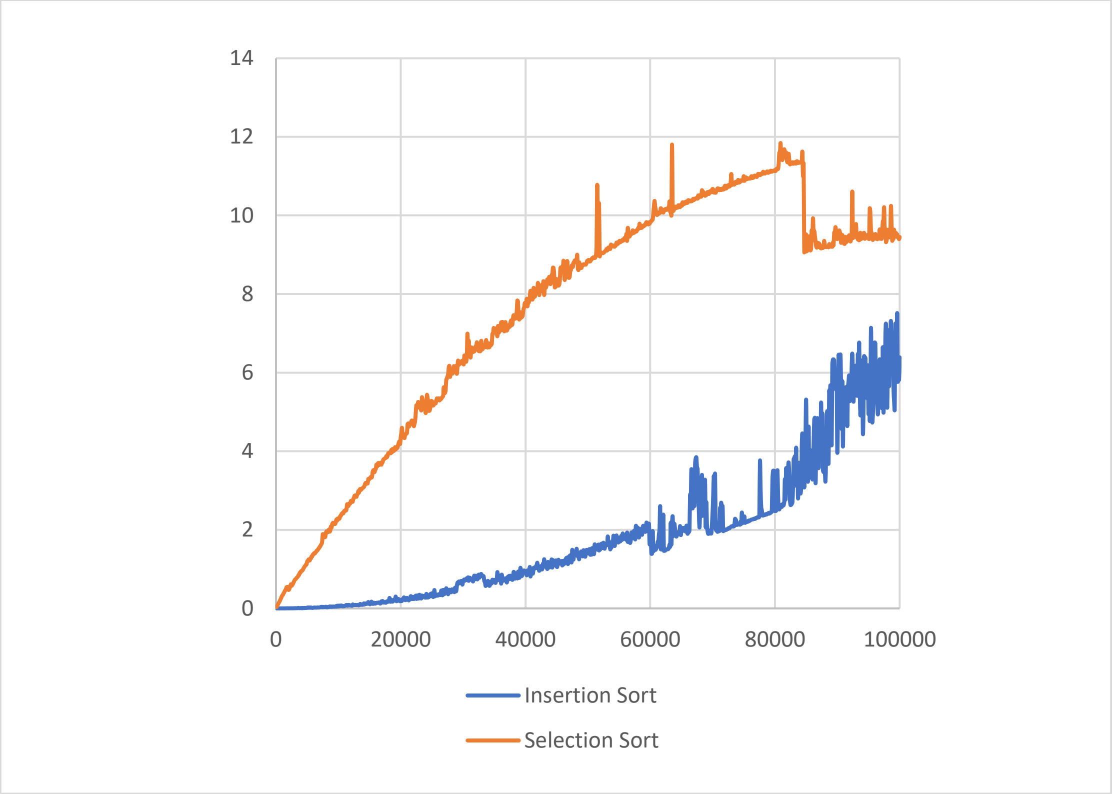
```
        for (int s = 0; s < n; s++)
        {
            a[s] = c[s];
        }
        for (int i = 1; i < k; i++)
        {
            int current = a[i];
            int j = i - 1;
            while (a[j] > current && j >= 0)
            {
                a[j + 1] = a[j];
                j--;
            }
            a[j + 1] = current;
        }
        end = clock();
        cpu_time_used = ((double)(end - start)) / CLOCKS_PER_SEC;
        printf("\n %lf", cpu_time_used);
        fprintf(fp,"\nTime taken to sort first %d numbers: %lf seconds.", k, cpu_time_used);
    }
}

void selectionSort(int c[],int n, FILE *fp)
{
    printf("\nSELECTION SORT: ");
    fprintf(fp,"\nSELECTION SORT: ");
    for (int k = 100; k <= n; k += 100)
    {
        clock_t start, end;
        double cpu_time_used;
        start = clock();
        int b[n];
        for (int s = 0; s < n; s++)
            {
                b[s] = c[s];
            }
        for (int i = 0; i < k - 1; i++)
        {
            int min = i;
            for (int j = i + 1; j < n; j++)
            {
                if (b[min] > b[j])
                {
                    min = j;
                }
            }
            int temp = b[min];
            b[min] = b[i];
```



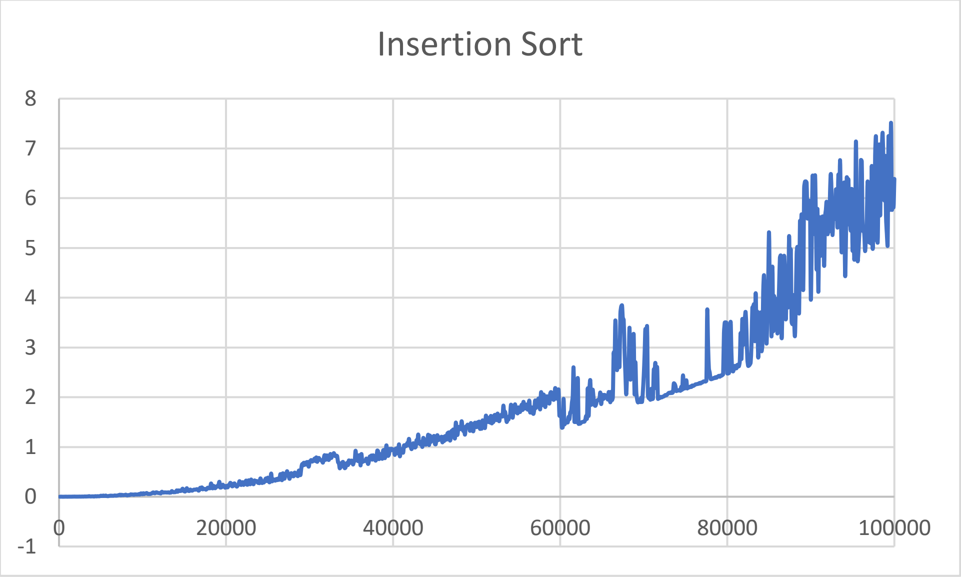
```
            b[i] = temp;
        }
        end = clock();
        cpu_time_used = ((double)(end - start)) / CLOCKS_PER_SEC;
        printf("\n%lf", cpu_time_used);
        fprintf(fp,"\nTime taken to sort first %d numbers: %lf seconds.", k, cpu_time_used);
    }
}

int main()
{
    FILE *fp;
    fp = fopen("Experiment 1b.txt","w");
    int n = 100000;
    int c[n];
    for (int k = 0; k < n; k++)
    {
        c[k] = rand() % 100 + 1;
    }
    fprintf(fp,"\nUnsorted Array: ");
    for (int k = 0; k < n; k++)
    {
        fprintf(fp," %d ", c[k]);
    }
    fprintf(fp,"\n");

    // insertion sort function
    insertionSort(c,n,fp);

    printf("\n");
    fprintf(fp,"\n");

    //selection sort
    selectionSort(c,n,fp);

    return 0;
}
```

Output:

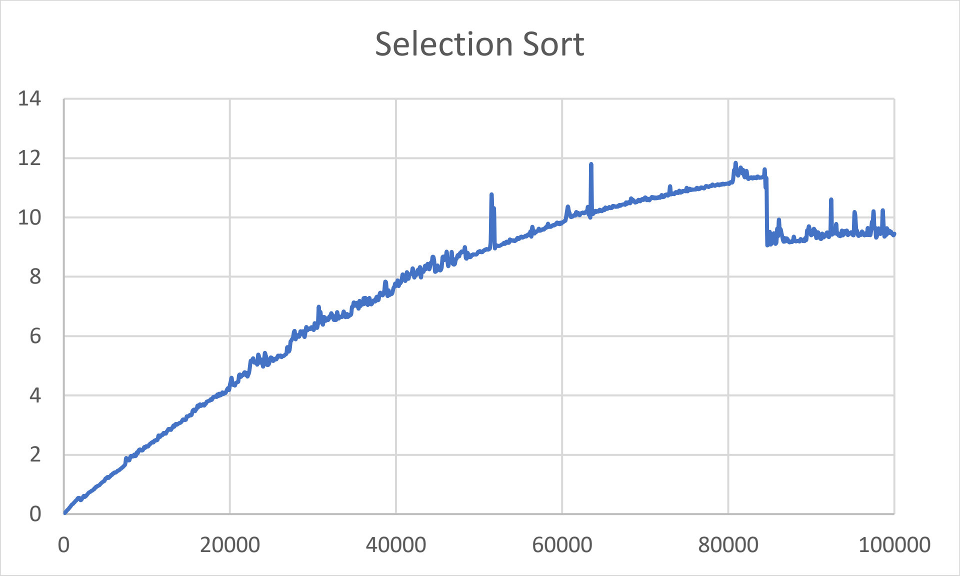
Graph : Insertion sort and Selection sort : time vs blocks



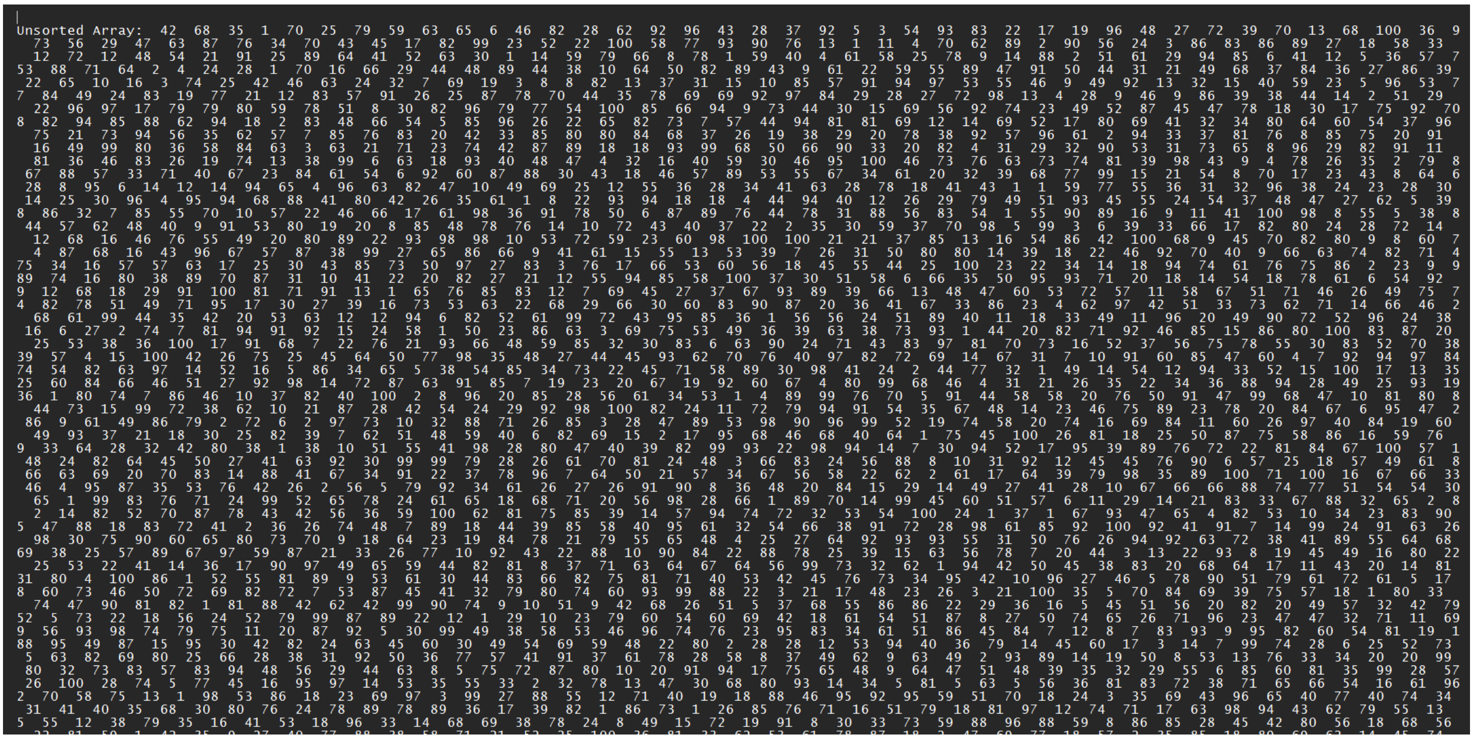
Graph : Insertion sort: time vs blocks



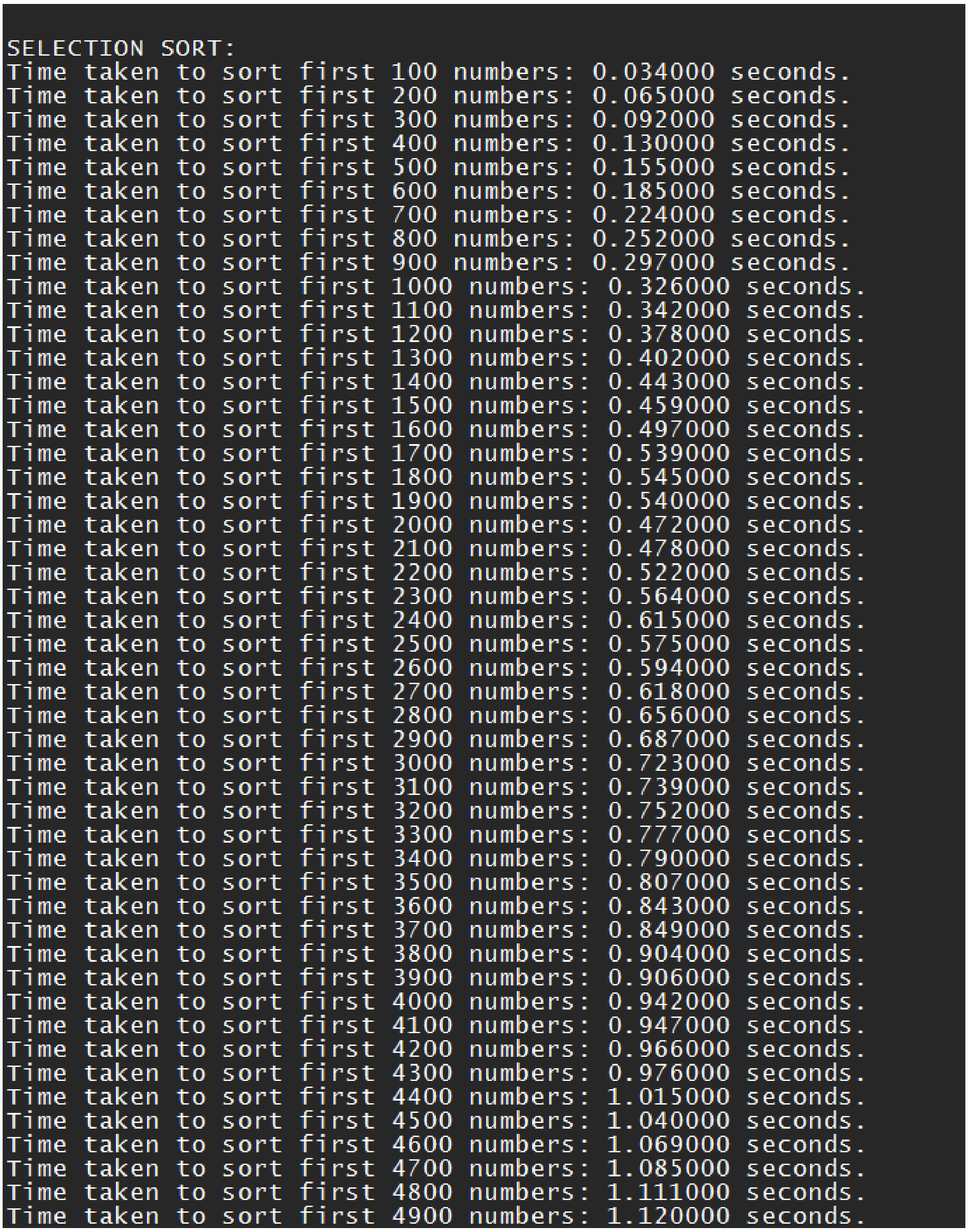
Graph : Selection sort : time vs blocks



Output in text file:



```
Unsorted Array:  42  68  35  1  70  25  79  59  63  65  6  46  82  28  62  92  96  43  28  37  92  5  3  54  93  83  22  17  19  96  48  27  72  39  70  13  68  100  36  9
  73  56  29  47  63  87  76  34  70  43  45  17  82  99  23  52  22  100  58  77  93  90  76  13  1  11  4  70  62  89  2  90  56  24  3  86  83  86  89  27  18  58  33
  12  72  12  48  54  21  91  25  89  64  41  52  63  30  1  14  59  79  66  8  78  1  59  40  4  61  58  25  78  9  14  88  2  51  61  29  94  85  6  41  12  5  36  57  7
53  88  71  64  2  4  24  28  1  70  16  66  29  44  48  89  44  38  10  64  50  82  89  43  9  61  22  59  55  89  47  91  50  44  31  21  49  68  37  84  36  27  86  39
  22  65  10  16  3  74  25  42  46  63  24  32  7  69  19  3  8  8  82  13  37  31  15  10  85  57  91  94  97  53  55  46  9  49  92  13  32  15  40  59  23  5  96  53  7
7  84  49  24  83  19  77  21  12  83  57  91  26  25  87  78  70  44  35  78  69  69  92  97  84  29  28  27  72  98  13  4  28  9  46  9  86  39  38  44  14  2  51  29
  22  96  97  17  79  79  80  59  78  51  8  30  82  96  79  77  54  100  85  66  94  9  73  44  30  15  69  56  92  74  23  49  52  87  45  47  78  18  30  17  75  92  70
8  82  94  85  88  62  94  18  2  83  48  66  54  5  85  96  26  22  65  82  73  7  57  44  94  81  81  69  12  14  69  52  17  80  69  41  32  34  80  64  60  54  37  96
  75  21  73  94  56  35  62  57  7  85  76  83  20  42  33  85  80  80  84  68  37  26  19  38  29  20  78  38  92  57  96  61  2  94  33  37  81  76  8  85  75  20  91
  16  49  99  80  36  58  84  63  3  63  21  71  23  74  42  87  89  18  18  93  99  68  50  66  90  33  20  82  4  31  29  32  90  53  31  73  65  8  96  29  82  91  11
  81  36  46  83  26  19  74  13  38  99  6  63  18  93  40  48  47  4  32  16  40  59  30  46  95  100  46  73  76  63  73  74  81  39  98  43  9  4  78  26  35  2  79  8
  67  88  57  33  71  40  67  23  84  61  54  6  92  60  87  88  30  43  18  46  57  89  53  55  67  34  61  20  32  39  68  77  99  15  21  54  8  70  17  23  43  8  64  6
  28  8  95  6  14  12  14  94  65  4  96  63  82  47  10  49  69  25  12  55  36  28  34  41  63  28  78  18  41  43  1  1  59  77  55  36  31  32  96  38  24  23  28  30
  14  25  30  96  4  95  94  68  88  41  80  42  26  35  61  1  8  22  93  94  18  18  4  44  94  40  12  26  29  79  49  51  93  45  55  24  54  37  48  47  27  62  5  39
8  86  32  7  85  55  70  10  57  22  46  66  17  61  98  36  91  78  50  6  87  89  76  44  78  31  88  56  83  54  1  55  90  89  16  9  11  41  100  98  8  55  5  38  8
  44  57  62  48  40  9  91  53  80  19  20  8  85  48  78  76  14  10  72  43  40  37  22  2  35  30  59  37  70  98  5  99  3  6  39  33  66  17  82  80  24  28  72  14
  12  68  16  46  76  55  49  20  80  89  22  93  98  98  10  53  72  59  23  60  98  100  100  21  21  37  85  13  16  54  86  42  100  68  9  45  70  82  80  9  8  60  7
  4  87  68  16  43  96  67  57  87  38  99  27  65  86  66  9  41  61  15  55  13  53  39  7  26  31  50  80  80  14  39  18  22  46  92  70  40  9  66  63  74  82  71  4
75  34  16  57  57  63  17  25  30  43  85  73  50  97  27  83  3  76  17  66  53  60  56  18  45  55  44  25  100  23  22  34  14  18  94  74  61  76  75  86  2  23  9  9
89  74  16  80  38  89  70  87  31  10  41  22  20  82  27  21  12  55  94  85  58  100  37  30  51  58  6  66  35  50  95  93  71  20  18  14  54  18  78  61  6  54  92
9  12  68  18  29  91  100  81  71  91  13  1  65  76  85  83  12  7  69  45  27  37  67  93  89  39  66  13  48  47  60  53  72  57  11  58  67  51  71  46  26  49  75  7
4  82  78  51  49  71  95  17  30  27  39  16  73  53  63  22  68  29  66  30  60  83  90  87  20  36  41  67  33  86  23  4  62  97  42  51  33  73  62  71  14  66  46  2
  68  61  99  44  35  42  20  53  63  12  12  94  6  82  52  61  99  72  43  95  85  36  1  56  56  24  51  89  40  11  18  33  49  11  96  20  49  90  72  52  96  24  38
  16  6  27  2  74  7  81  94  91  92  15  24  58  1  50  23  86  63  3  69  75  53  49  36  39  63  38  73  93  1  44  20  82  71  92  46  85  15  86  80  100  83  87  20
  25  53  38  36  100  17  91  68  7  22  76  21  93  66  48  59  85  32  30  83  6  63  90  24  71  43  83  97  81  70  73  16  52  37  56  75  78  55  30  83  52  70  38
39  57  4  15  100  42  26  75  25  45  64  50  77  98  35  48  27  44  45  93  62  70  76  40  97  82  72  69  14  67  31  7  10  91  60  85  47  60  4  7  92  94  97  84
74  54  82  63  97  14  52  16  5  86  34  65  5  38  54  85  34  73  22  45  71  58  89  30  98  41  24  2  44  77  32  1  49  14  54  12  94  33  52  15  100  17  13  35
25  60  84  66  46  51  27  92  98  14  72  87  63  91  85  7  19  23  20  67  19  92  60  67  4  80  99  68  46  4  31  21  26  35  22  34  36  88  94  28  49  25  93  19
36  1  80  74  7  86  46  10  37  82  40  100  2  8  96  20  85  28  56  61  34  53  1  4  89  99  76  70  5  91  44  58  58  20  76  50  91  47  99  68  47  10  81  80  8
  44  73  15  99  72  38  62  10  21  87  28  42  54  24  29  92  98  100  82  24  11  72  79  94  91  54  35  67  48  14  23  46  75  89  23  78  20  84  67  6  95  47  2
  86  9  61  49  86  79  2  72  6  2  97  73  10  32  88  71  26  85  3  28  47  89  53  98  90  96  99  52  19  74  58  20  74  16  69  84  11  60  26  97  40  84  19  60
  49  93  37  21  18  30  25  82  39  7  62  51  48  59  40  6  82  69  15  2  17  95  68  46  68  40  64  1  75  45  100  26  81  18  25  50  87  75  58  86  16  59  76
9  33  64  28  32  42  80  38  1  38  10  51  55  41  98  28  80  47  40  39  82  99  93  22  98  94  14  7  30  94  52  17  95  39  89  76  72  22  81  84  67  100  57  1
  48  24  82  64  45  50  27  41  63  92  30  99  99  79  28  26  61  70  81  24  48  3  66  83  24  56  88  8  10  31  92  12  45  45  76  90  6  57  25  18  57  49  61  8
  66  63  69  20  70  83  14  88  41  67  34  91  22  37  78  96  7  64  50  21  57  34  67  56  58  22  62  2  61  17  64  39  79  98  35  89  100  71  100  16  67  66  33
  46  4  95  87  35  53  76  42  26  2  56  5  79  92  34  61  26  27  26  91  90  8  36  48  20  84  15  29  14  49  27  41  28  10  67  66  66  88  74  77  51  54  54  30
  65  1  99  83  76  71  24  99  52  65  78  24  61  65  18  68  71  20  56  98  28  66  1  89  70  14  99  45  60  51  57  6  11  29  14  21  83  33  67  88  32  65  2  8
  2  14  82  52  70  87  78  43  42  56  36  59  100  62  81  75  85  39  14  57  94  74  72  32  53  54  100  24  1  37  1  67  93  47  65  4  82  53  10  34  23  83  90
5  47  88  18  83  72  41  2  36  26  74  48  7  89  18  44  39  85  58  40  95  61  32  54  66  38  91  72  28  98  61  85  92  100  92  41  91  7  14  99  24  91  63  26
  98  30  75  90  60  65  80  73  70  9  18  64  23  19  84  78  21  79  55  65  48  4  25  27  64  92  93  93  55  31  50  76  26  94  92  63  72  38  41  89  55  64  68
69  38  25  57  89  67  97  59  87  21  33  26  77  10  92  43  22  88  10  90  84  22  88  78  25  39  15  63  56  78  7  20  44  3  13  22  93  8  19  45  49  16  80  22
  25  53  22  41  14  36  17  90  97  49  65  59  44  82  81  8  37  71  63  64  67  64  56  99  73  32  62  1  94  42  50  45  38  83  20  68  64  17  11  43  20  14  81
31  80  4  100  86  1  52  55  81  89  9  53  61  30  44  83  66  82  75  81  71  40  53  42  45  76  73  34  95  42  10  96  27  46  5  78  90  51  79  61  72  61  5  17
8  60  73  46  50  72  69  82  72  7  53  87  45  41  32  79  80  74  60  93  99  88  22  3  21  17  48  23  26  3  21  100  35  5  70  84  69  39  75  57  18  1  80  33
  74  47  90  81  82  1  81  88  42  62  42  99  90  74  9  10  51  9  42  68  26  51  5  37  68  55  86  86  22  29  36  16  5  45  51  56  20  82  20  49  57  32  42  79
52  5  73  22  18  56  24  52  79  99  87  89  22  12  1  29  10  23  79  60  54  60  69  42  18  61  54  51  87  8  27  50  74  65  26  71  96  23  47  47  32  71  11  69
9  56  93  98  74  79  75  11  20  87  92  5  30  99  49  38  58  53  46  96  74  76  23  95  83  34  61  51  86  45  84  7  12  8  7  83  93  9  95  82  60  54  81  19  1
88  95  49  87  15  95  30  42  82  24  63  45  60  30  49  54  69  59  48  22  80  2  28  28  12  53  94  40  36  79  14  45  60  17  3  14  7  99  74  28  6  25  52  73
  5  63  82  69  80  25  66  28  38  31  92  50  36  77  57  41  91  37  61  78  28  58  8  37  49  62  9  63  49  2  93  89  14  19  50  8  53  13  76  33  34  20  20  99
  80  32  73  83  57  83  94  48  56  29  44  63  8  5  75  72  87  80  10  20  91  94  17  75  65  48  9  64  47  51  48  39  35  32  29  55  6  85  60  81  35  99  28  57
  26  100  28  74  5  77  45  16  95  97  14  53  35  55  33  2  32  78  13  47  30  68  80  93  14  34  5  81  5  63  5  56  36  81  83  72  38  71  65  66  54  16  61  96
2  70  58  75  13  1  98  53  86  18  23  69  97  3  99  27  88  55  12  71  40  19  18  88  46  95  92  95  59  51  70  18  24  3  35  69  43  96  65  40  77  40  74  34
  31  41  40  35  68  30  80  76  24  78  89  78  89  36  17  39  82  1  86  73  1  26  85  76  71  16  51  79  18  81  97  12  74  71  17  63  98  94  43  62  79  55  13
5  55  12  38  79  35  16  41  53  18  96  33  14  68  69  38  78  24  8  49  15  72  19  91  8  30  33  73  59  88  96  88  59  8  86  85  28  45  42  80  56  18  68  56
```



```
SELECTION SORT:
Time taken to sort first 100 numbers: 0.034000 seconds.
Time taken to sort first 200 numbers: 0.065000 seconds.
Time taken to sort first 300 numbers: 0.092000 seconds.
Time taken to sort first 400 numbers: 0.130000 seconds.
Time taken to sort first 500 numbers: 0.155000 seconds.
Time taken to sort first 600 numbers: 0.185000 seconds.
Time taken to sort first 700 numbers: 0.224000 seconds.
Time taken to sort first 800 numbers: 0.252000 seconds.
Time taken to sort first 900 numbers: 0.297000 seconds.
Time taken to sort first 1000 numbers: 0.326000 seconds.
Time taken to sort first 1100 numbers: 0.342000 seconds.
Time taken to sort first 1200 numbers: 0.378000 seconds.
Time taken to sort first 1300 numbers: 0.402000 seconds.
Time taken to sort first 1400 numbers: 0.443000 seconds.
Time taken to sort first 1500 numbers: 0.459000 seconds.
Time taken to sort first 1600 numbers: 0.497000 seconds.
Time taken to sort first 1700 numbers: 0.539000 seconds.
Time taken to sort first 1800 numbers: 0.545000 seconds.
Time taken to sort first 1900 numbers: 0.540000 seconds.
Time taken to sort first 2000 numbers: 0.472000 seconds.
Time taken to sort first 2100 numbers: 0.478000 seconds.
Time taken to sort first 2200 numbers: 0.522000 seconds.
Time taken to sort first 2300 numbers: 0.564000 seconds.
Time taken to sort first 2400 numbers: 0.615000 seconds.
Time taken to sort first 2500 numbers: 0.575000 seconds.
Time taken to sort first 2600 numbers: 0.594000 seconds.
Time taken to sort first 2700 numbers: 0.618000 seconds.
Time taken to sort first 2800 numbers: 0.656000 seconds.
Time taken to sort first 2900 numbers: 0.687000 seconds.
Time taken to sort first 3000 numbers: 0.723000 seconds.
Time taken to sort first 3100 numbers: 0.739000 seconds.
Time taken to sort first 3200 numbers: 0.752000 seconds.
Time taken to sort first 3300 numbers: 0.777000 seconds.
Time taken to sort first 3400 numbers: 0.790000 seconds.
Time taken to sort first 3500 numbers: 0.807000 seconds.
Time taken to sort first 3600 numbers: 0.843000 seconds.
Time taken to sort first 3700 numbers: 0.849000 seconds.
Time taken to sort first 3800 numbers: 0.904000 seconds.
Time taken to sort first 3900 numbers: 0.906000 seconds.
Time taken to sort first 4000 numbers: 0.942000 seconds.
Time taken to sort first 4100 numbers: 0.947000 seconds.
Time taken to sort first 4200 numbers: 0.966000 seconds.
Time taken to sort first 4300 numbers: 0.976000 seconds.
Time taken to sort first 4400 numbers: 1.015000 seconds.
Time taken to sort first 4500 numbers: 1.040000 seconds.
Time taken to sort first 4600 numbers: 1.069000 seconds.
Time taken to sort first 4700 numbers: 1.085000 seconds.
Time taken to sort first 4800 numbers: 1.111000 seconds.
Time taken to sort first 4900 numbers: 1.120000 seconds.
```

Observation : From the above output and graph, we can infer that Insertion sort is much faster than Selection sort for an array of 100,000 elements. While Insertion Sort takes around 6 seconds to sort the entire array, Selection sort takes 10 seconds to do the same. This implies that while it won't matter for smaller arrays, insertion sort is much more efficient than selection sort for a large array of elements.

Conclusion: Hereby, I have successfully run the code for insertion sort and selection sort while getting the desired output. I have thoroughly understood the concept of sorting of elements and using the clock() function to compute the time required to execute a function in C through this experiment.