Algorithms Lab Assignment 1

Pranav Gade

February 7, 2022

Batch:	CS&AI
Roll no.:	LCI2020010

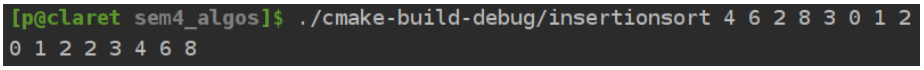
1 Insertion sort

Analysis of Insertion sort complexity.

1.1 Code

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

void sort(int* arr, int size);

int main(int argc, char** argv) {
    int size = argc - 1;
    int* arr = malloc(sizeof(int) * size);
    for (int i = 1; i <= size; ++i) {
        arr[i-1] = atoi(argv[i]);
    }

    sort(arr, size);

    for (int i = 0; i < size; ++i) {
        printf("%d ", arr[i]);
    }

    free(arr);
    return 0;
}

void sort(int* arr, int size) {
    for (int i = 0; i < size; ++i) {
        int curr = arr[i];
        int j = i-1;
        while (arr[j] > curr && j >= 0) {
            arr[j+1] = arr[j];
            j--;
        }
        arr[j+1] = curr;
    }
}
```

1.2 Output



```
[p@claret sem4_algos]$ ./cmake-build-debug/insertionsort 4 6 2 8 3 0 1 2
0 1 2 2 3 4 6 8
```

Figure 1: Insertion sort test output

1.3 Graph

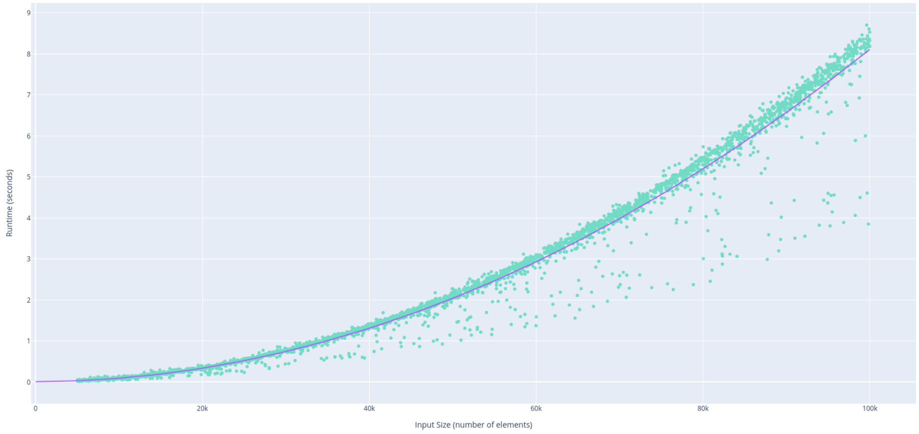


Figure 2: Insertion sort runtime v/s input size plot

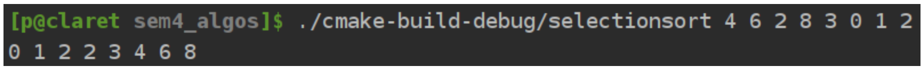
2 Selection sort

Analysis of Selection sort complexity.

2.1 Code

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

void sort(int* arr, int size);

int main(int argc, char** argv) {
    int size = argc - 1;
    int* arr = malloc(sizeof(int) * size);
    for (int i = 1; i <= size; ++i) {
        arr[i-1] = atoi(argv[i]);
    }

    sort(arr, size);

    for (int i = 0; i < size; ++i) {
        printf("%d ", arr[i]);
    }

    free(arr);
    return 0;
}

void sort(int* arr, int size) {
    for (int i = 0; i < size-1; ++i) {
        int idx = i;
        for (int j = i; j < size; ++j) {
            if (arr[idx] > arr[j]) idx = j;
        }
        int t = arr[idx];
        arr[idx] = arr[i];
        arr[i] = t;
    }
}
```

2.2 Output



```
[p@claret sem4_algos]$ ./cmake-build-debug/selectionsort 4 6 2 8 3 0 1 2
0 1 2 2 3 4 6 8
```

Figure 3: Selection sort test output

2.3 Graph

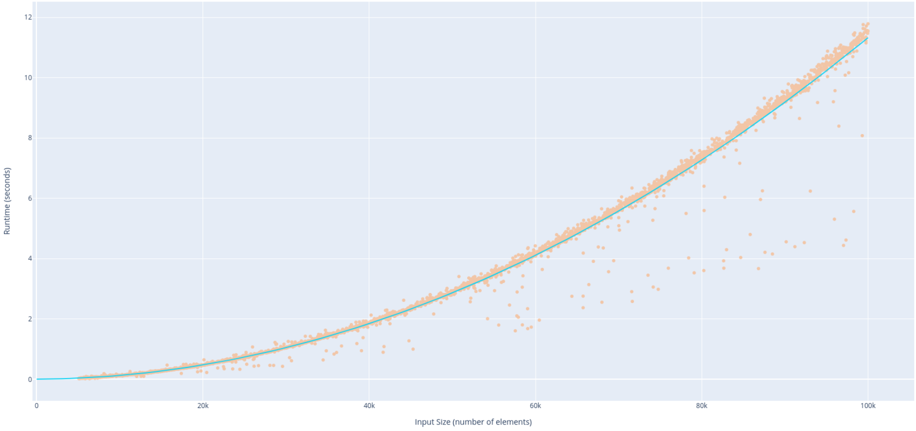


Figure 4: Selection sort runtime v/s input size plot

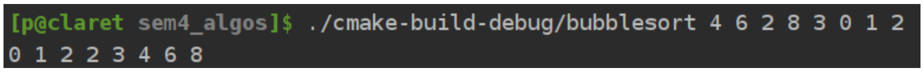
3 Bubble sort

Analysis of Bubble sort complexity.

3.1 Code

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

void sort(int* arr, int size);

int main(int argc, char** argv) {
    int size = argc - 1;
    int* arr = malloc(sizeof(int) * size);
    for (int i = 1; i <= size; ++i) {
        arr[i-1] = atoi(argv[i]);
    }

    sort(arr, size);

    for (int i = 0; i < size; ++i) {
        printf("%d ", arr[i]);
    }

    free(arr);
    return 0;
}

void sort(int* arr, int size) {
    for (int i = 0; i < size-1; ++i) {
        for (int j = 0; j < size-i-1; ++j) {
            if (arr[j] > arr[j+1]) {
                int t = arr[j+1];
                arr[j+1] = arr[j];
                arr[j] = t;
            }
        }
    }
}
```

3.2 Output



```
[p@claret sem4_algos]$ ./cmake-build-debug/bubblesort 4 6 2 8 3 0 1 2
0 1 2 2 3 4 6 8
```

Figure 5: Bubble sort test output

3.3 Graph

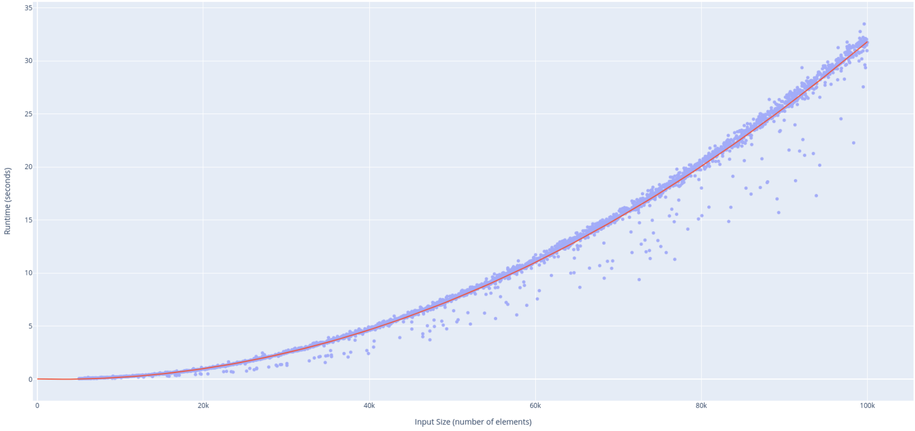


Figure 6: Bubble sort runtime v/s input size plot

4 Footnotes

Graphs are screenshots from https://jsfiddle.net/z51x9asg/show
Code to generate graphs and this file is on github