

Algorithms Lab Assignment 2

Pranav GADE

February 10, 2022

Batch: CS&AI
Roll no.: LCI2020010

1 Bucket sort

Analysis of Bucket sort complexity.

1.1 Code

```
1 #include <stdlib.h>
2 #include <stdio.h>
3 #include <limits.h>
4
5 void sort(int* arr, int size);
6
7 int main(int argc, char** argv) {
8     int size = argc - 1;
9     int* arr = malloc(sizeof(int) * size);
10    for (int i = 1; i <= size; ++i) {
11        arr[i-1] = atoi(argv[i]);
12    }
13
14    sort(arr, size);
15
16    for (int i = 0; i < size; ++i) {
17        printf("%d ", arr[i]);
18    }
19
20    free(arr);
21    return 0;
22 }
23
24 void insertionsort(int* arr, int size) {
25     for (int i = 0; i < size; ++i) {
26         int curr = arr[i];
27         int j = i-1;
28         while (arr[j] > curr && j >= 0) {
29             arr[j+1] = arr[j];
30             j--;
31         }
32         arr[j+1] = curr;
```

```

33     }
34 }
35
36 void sort(int* arr, int size) {
37     int buckets[16][size];
38     int sizes[16] = {0};
39
40     int max = INT_MIN;
41     for (int i = 0; i < size; ++i) {
42         if (arr[i] > max) max = arr[i];
43     }
44     int bit = 0;
45     while (max > 0) {
46         max /= 2;
47         bit++;
48     }
49     bit -= 4;
50     if (bit < 0) bit = 0;
51     for (int i = 0; i < size; ++i) {
52         int pos = (arr[i] & (0xf << bit)) >> bit;
53         buckets[pos][sizes[pos]++] = arr[i];
54     }
55
56     for (int i = 0; i < 16; ++i) insertionsort(buckets[i], sizes[i]);
57
58     int pos = 0;
59     int ptr = 0;
60     for (int i = 0; i < size; ++i) {
61         while (ptr >= sizes[pos]) {
62             pos++;
63             ptr = 0;
64         }
65         arr[i] = buckets[pos][ptr++];
66     }
67 }

```

1.2 Output

```

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

```

Figure 1: Bucket sort test output

1.3 Graph

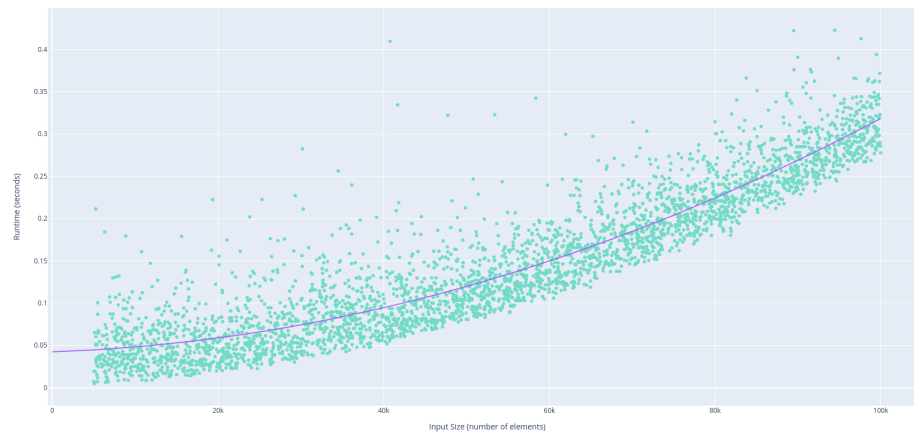


Figure 2: Bucket sort runtime v/s input size plot

2 Counting sort

Analysis of Counting sort complexity.

2.1 Code

```
1 #include <stdlib.h>
2 #include <stdio.h>
3 #include <limits.h>
4
5 void sort(int* arr, int size);
6
7 int main(int argc, char** argv) {
8     int size = argc - 1;
9     int* arr = malloc(sizeof(int) * size);
10    for (int i = 1; i <= size; ++i) {
11        arr[i-1] = atoi(argv[i]);
12    }
13
14    sort(arr, size);
15
16    for (int i = 0; i < size; ++i) {
17        printf("%d ", arr[i]);
18    }
19
20    free(arr);
21    return 0;
22 }
23
24 void sort(int* arr, int size) {
25     int max = INT_MIN;
```

```

26     int min = INT_MAX;
27     for (int i = 0; i < size; ++i) {
28         if (arr[i] > max) max = arr[i];
29         if (arr[i] < min) min = arr[i];
30     }
31
32     int len = max - min + 2;
33     int counts[len];
34     for (int i = 0; i < len; ++i) counts[i] = 0;
35     for (int i = 0; i < size; ++i) counts[arr[i] - min]++;
36
37     int ptr = 0;
38     for (int i = 0; i < size; ++i) {
39         while (counts[ptr] <= 0) ptr++;
40         arr[i] = ptr;
41         counts[ptr]--;
42     }
43 }

```

2.2 Output

```

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

```

Figure 3: Counting sort test output

2.3 Graph

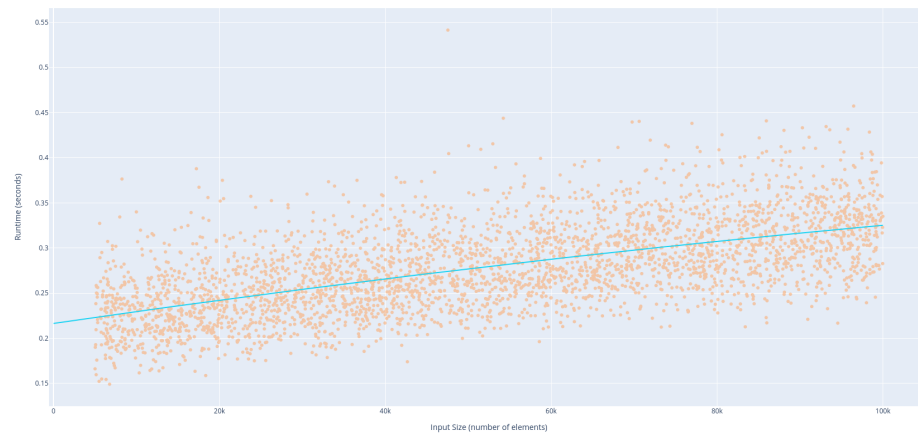


Figure 4: Counting sort runtime v/s input size plot

3 Radix sort

Analysis of Radix sort complexity.

3.1 Code

```
1 #include <stdlib.h>
2 #include <stdio.h>
3 #include <limits.h>
4
5 void sort(int* arr, int size);
6
7 int main(int argc, char** argv) {
8     int size = argc - 1;
9     int* arr = malloc(sizeof(int) * size);
10    for (int i = 1; i <= size; ++i) {
11        arr[i-1] = atoi(argv[i]);
12    }
13
14    sort(arr, size);
15
16    for (int i = 0; i < size; ++i) {
17        printf("%d ", arr[i]);
18    }
19
20    free(arr);
21    return 0;
22 }
23
24 void sort_helper(int* arr, int size, int bit) {
25     int buckets[16][size];
26     int sizes[16] = {0};
27     for (int i = 0; i < size; ++i) {
28         int pos = (arr[i] & (0xf << bit)) >> bit;
29         buckets[pos][sizes[pos]++] = arr[i];
30     }
31     int pos = 0;
32     int ptr = 0;
33     for (int i = 0; i < size; ++i) {
34         while (ptr >= sizes[pos]) {
35             pos++;
36             ptr = 0;
37         }
38         arr[i] = buckets[pos][ptr++];
39     }
40 }
41
42 void sort(int* arr, int size) {
43     int max = INT_MIN;
44     for (int i = 0; i < size; ++i) {
45         if (arr[i] > max) max = arr[i];
46     }
47     int bit = 0;
48     while (max > 0) {
49         max /= 2;
50         bit++;
51     }
52     bit -= 4;
53     if (bit < 0) bit = 0;
54
55     int ptr = 0;
```

```

56     while (bit >= ptr) {
57         sort_helper(arr, size, ptr);
58         ptr += 4;
59     }
60 }

```

3.2 Output

```

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

```

Figure 5: Radix sort test output

3.3 Graph

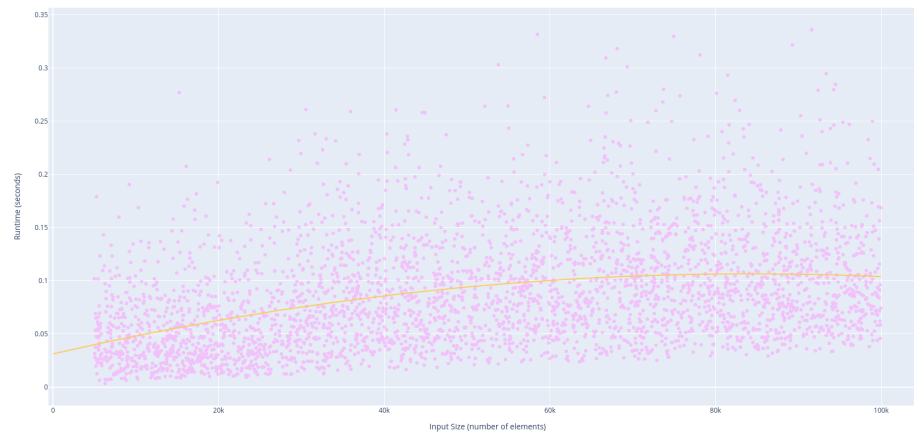


Figure 6: Radix sort runtime v/s input size plot

4 Footnotes

Code to generate graphs and this file is on [github](#)