Analyzing Algorithms Part II

In this lesson, we'll dry run insertion sort and count the instructions executed on an array of length 5.

We'll now dry run our algorithm on the input array and trace the number of instructions executed in each iteration. We'll sum up the number of instructions across all the iterations to get a final count of instructions for the entire run. In the previous section, we worked out the instruction counts of the inner and outer loop to be those below:

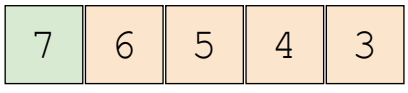
Number of instructions executed per iteration of outer for loop = 4
Number of instructions executed per iteration of inner while loop = 7

The code from the previous section is reproduced below:

```
1.          for (int i = 0; i < input.length; i++) {
2.              int key = input[i];
3.              j = i - 1;
4.              while (j >= 0 && input[j] > key) {
5.                  if (input[j] > key) {
6.                      int tmp = input[j];
7.                      input[j] = key;
8.                      input[j + 1] = tmp;
9.                      j--;
10.                  }
11.             }
12.         }
```

First Iteration of Outer Loop | when i = 0, key = 7

j = -1 inner loop doesn't execute



Total cost for 1st iteration = Outer Loop + Inner Loop
= 4 + [(iterations * 7) + 2]

= 4 + [(iterations * 7) + 2]

= 4 + [(0 * 7) + 2]
= 6 instructions

Second Iteration of Outer Loop | when i = 1, key = 6

j = 0

Total cost for 2nd iteration = Outer Loop + Inner Loop
= 4 + [(iterations * 7) + 2]
= 4 + [(1 * 7) + 2]
= 13 instructions

Third Iteration of Outer Loop | when i = 2, key = 5

j = 1

j = 0

Total cost for 3rd iteration = Outer Loop + Inner Loop
= 4 + [(iterations * 7) + 2]
= 4 + [(2 * 7) + 2]
= 20 instructions

Fourth Iteration of Outer Loop | when i = 3, key = 4

j = 2

j = 1

j = 0

Total cost for 1st iteration = Outer Loop + Inner Loop
= 4 + [(iterations * 7) + 2]
= 4 + [(3 * 7) + 2]
= 27 instructions

Fifth Iteration of Outer Loop | when i = 4, key = 3

j=3

j=2

j=1

j=0

Total cost for 1st iteration = Outer Loop + Inner Loop
= 4 + [(iterations * 7) + 2]
= 4 + [(4 * 7) + 2]

= 4 + [(4 * 7) + 2]
= 34 instructions

However, remember that to exit the loop the increment and the inequality statements will execute for one last time, so we need to add that cost too.
= 34 + 2
= 36 instructions

Iteration	Instructions Executed
1	6
2	13
3	20
4	27
5	36
Total Cost	6 + 13 + 20 + 27 + 36 = 102 instructions

One can observe that as the size of the array increases, the inner loop's iterations will correspondingly increase, and the running time of the program will increase.

Using Formulas

From the previous section, we can plug the formulas we produced to get the same result. There were 5 iterations of the outer for loop, so the instructions executed for the outer *for* loop are:

$$[2 * (n + 1) + 2n], \ where\ n\ is\ number\ of\ iterations$$

$$= [2 * (5 + 1) + 2 * 5]$$

$$= [12 + 10]$$

$$= 22\ instructions$$

The number of iterations for the inner loop varies for each iteration of the outer loop given by **(k x 7) + 2**

- No iteration on first run of outer loop.
  = (0 x 7) + 2 = 2
- 1 iteration in second run of outer loop:
  = (1 x 7) + 2 = 9
- 2 iterations on third run of outer loop:
  = (2 x 7) + 2 = 16
- 3 iterations on fourth run of outer loop:
  = (3 x 7) + 2= 23
- 4 iterations on fifth run of outer loop:
  = (4 x 7) + 2= 30
- Total instructions executed for the inner while loop are then: 2 + 9 + 16 + 23 + 30 = 80 instructions.

The number of instructions for the entire run comes out to be ***80*** + ***22*** = ***102 instructions***. Note how our formulas match the exact count we found from the dry run!