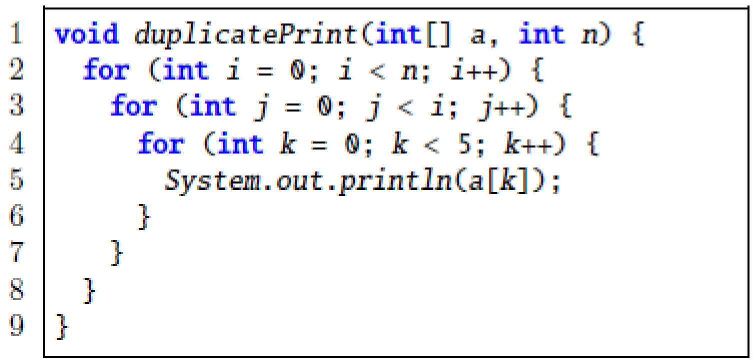
EECS2030 Fall 2019

Lab 8

Analyzing the Time Complexity of a Program

Siddarth Kanna Kannapiran

216469850



```java
void duplicatePrint(int[] a, int n) {
  for (int i = 0; i < n; i++) {
    for (int j = 0; j < i; j++) {
      for (int k = 0; k < 5; k++) {
        System.out.println(a[k]);
      }
    }
  }
}
```

Examples:

If n = 5:

- The outer loop runs n times, which is 5 times.
- The middle loop runs i times, which is also 5 times.
- The inner loop runs 5 times regardless of the value of n.
  Total of 125 times

If n = 6:

- The outer loop runs n times, which is 6 times.
- The middle loop runs i times, which is also 6 times.
- The inner loop runs 5 times regardless of the value of n.
  Total of 180 times

If n = 7:

- The outer loop runs n times, which is 7 times.
- The middle loop runs i times, which is also 7 times.
- The inner loop runs 5 times regardless of the value of n.
  Total of 245 times

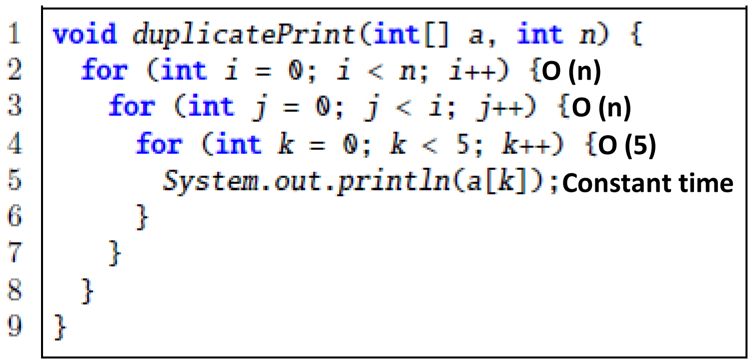
If n = 8:

- The outer loop runs n times, which is 8 times.
- The middle loop runs i times, which is also 8 times.
- The inner loop runs 5 times regardless of the value of n.
  Total of 320 times

All the above examples run for $f(n) = 5 \times n^2$ time.

The inner most loop runs 5 times regardless of the value of n, so it runs a constant time.

The number of times the other two loops runs will be dependent on the value of n.



```
void duplicatePrint(int[] a, int n) {
  for (int i = 0; i < n; i++) {O (n)
    for (int j = 0; j < i; j++) {O (n)
      for (int k = 0; k < 5; k++) {O (5)
        System.out.println(a[k]);Constant time
      }
    }
  }
}
```

We have two loops that run n times, and one that runs 5 times. If we multiply all of them together, we get the expression **O ($5n^2$)** which is the same as **O (n^2)**. Therefore, this program has an asymptotic upper bound of **O (n^2).**