Comparing Algorithms

- Need to define **computational cost** (running time)
  - number of **basic operations** required by the algorithm to process an input of a certain **size**

\[ T(n) \]

- basic operations are always **relevant** to the problem
  - ex: find max in an array: # of comparisons
  - ex: sum elements in an array: # of additions
  - size is always the size of the input
  - ex: sum elements in an array: # of elements

What is the running time?

```java
1 for (int i = 0 ; i < n ; i ++) {
2    // 1 basic operation
3 }
4 5 for (int i = 0 ; i < n ; i ++) {
6    for (int j = 0 ; j < n ; j ++) {
7        // 1 basic operation
8    }
9 }
10 11 for (int i = 0 ; i < n ; i ++) {
12    for (int j = 0 ; j < n*n ; j ++) {
13        // 1 basic operation
14    }
15 }
```
What is the running time?

```
for (int i = 0; i < n; i++) {
    for (int j = 0; j < i; j++) {
        // 1 basic operation
    }
}
for (int i = 0; i < n; i++) {
    for (int j = 0; j < n; j++) {
        for (int k = 0; k < n; k++) {
            // 1 basic operation
        }
    }
}
for (int i = 0; i < n; i++) {
    for (int j = 0; j < i*i; j++) {
        for (int k = 0; k < j; k++) {
            // 1 basic operation
        }
    }
}
```

Different types of analysis

- **Worst-case:** maximum time of algorithm on any input
- **Average-case:** expected time of algorithm over all inputs
- **Best-case:** minimum time of algorithm on some (optimal) input

Worst-case, Average-case, Best-case

- Examples:
  - factorial of a number (iterative algorithm)
  - sequential search (return first occurrence)
  - sequential search (return last occurrence)