Big O

- Instead of using the exact number of operations to express the complexity of a computation, we use a more general notation called "Big O".
- Big O expresses the type of complexity function:
  - Linear $O(n)$
  - Quadratic $O(n^2)$
  - Logarithmic $O(\log n)$
  - Log-Linear $O(n \log n)$
  - Exponential $O(2^n)$
  - Constant $O(1)$
Big O

- Let $C$ represent a function for the number of comparisons needed for an algorithm as a function of the size of the input array(s).

  - Search $C(n) = n = O(n)$
  - Unique I $C(n) = n^2 = O(n^2)$
  - Diff $C(m,n) = mn + n = O(mn)$

    - If arrays are the same size: $O(n^2)$

More about Big O

- Consider a computation that performs $5n^2 + 3n + 9$ operations on $n$ data elements.

  - The graph comparing the number of data elements to the number of computations will be quadratic.

    $5n^2 + 3n + 9 = O(n^2)$

- Unique II Algorithm $C(n) = n(n-1)/2 = O(n^2)$
Example 5

```java
public static int binarySearch(int[] list, int target) {
    int min = 0, max = list.length-1, mid = 0;
    boolean found = false;
    while (!found && min <= max) {
        mid = (min + max) / 2;      // (integer div!)
        if (list[mid] == target)
            found = true;
        else if (target < list[mid])
            max = mid-1;
        else
            min = mid+1;
    }
    if (found) return mid;
    else return -1;
}
```

Worst Case

- Example: `list.length = n = 15`

```plaintext
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
```

```plaintext
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
```
Binary Search: Worst Case

- How many iterations are needed before we end up with no elements left to examine?

<table>
<thead>
<tr>
<th>Array length</th>
<th>Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>4</td>
</tr>
<tr>
<td>31</td>
<td>5</td>
</tr>
<tr>
<td>63</td>
<td>6</td>
</tr>
<tr>
<td>n</td>
<td></td>
</tr>
</tbody>
</table>

Big O: Formal Definition

- Let $T(n)$ = the number of operations performed in an algorithm as a function of $n$.
- $T(n) = O(f(n))$ if and only if there exists two constants, $n_0 > 0$ and $c > 0$, and a function $f(n)$ such that for all $n > n_0$, $c f(n) \geq T(n)$. 
Example Again

- Let $T(n) = 5n^2 + 3n + 9$. Show that $T(n) = O(n^2)$.
  - Find $c$ and $n_0$ such that for all $n > n_0$,
    $cn^2 > 5n^2 + 3n + 9$.
- Find intersection point such that $cn^2 = 5n^2 + 3n + 9$.
- Let $n = n_0$ and solve for $c$:
  - If $n_0 = 3$, then $c = 7$.
- Thus, $7n^2 > 5n^2 + 3n + 9$ for all $n > 3$.
- So $5n^2 + 3n + 9 = O(n^2)$

More about Big O

- Big O gives us an upper-bound approximation on the complexity of a computation.
- We can say that the following computation is $O(n^3)$:
  ```java
  for (int i = 0; i < n; i++)
    for (int j = 0; j < n; j++)
      System.out.println(i + " " + j);
  ```
- A tighter bound would be $O(n^2)$, but both are technically correct.
Order of Complexity

Algorithm

\[ O(A) \]
\[ O(B) \]

Overall order of complexity of algorithm is \( \max (O(A), O(B)) \).

- Examples:
  - \( O(\log N) + O(N) = O(N) \)
  - \( O(N \log N) + O(N) = O(N \log N) \)
  - \( O(N \log N) + O(N^2) = O(N^2) \)
  - \( O(2^N) + O(N^2) = O(2^N) \)

Order of Complexity

Algorithm

\[ O(A) \]
\[ O(B) \]

Overall order of complexity of algorithm is \( O(A \times B) \).

Example: - Nested loops

- Examples:
  - \( O(\log N) \times O(N) = O(N \log N) \)
  - \( O(N \log N) \times O(N) = O(N^2 \log N) \)
  - \( O(N) \times O(1) = O(N) \)
Traveling Salesperson

- Given: a network of nodes representing cities and edges representing flight paths (weights represent cost)
- Is there a route that takes the salesperson through every city and back to the starting city with cost no more than K?
  - The salesperson can visit a city only once (except for the start and end of the trip).
Traveling Salesperson

- If there are $N$ cities, what is the maximum number of routes that we might need to compute?
- Worst-case: There is a flight available between every pair of cities.
- Compute cost of every possible route.
  - Pick a starting city
  - Pick the next city (N-1 choices remaining)
  - Pick the next city (N-2 choices remaining)
  - ...

- Maximum number of routes: ______ = $O(______)$

Comparing Big O Functions

<table>
<thead>
<tr>
<th>Number of Operations</th>
<th>$O(N)$</th>
<th>$O(N \log N)$</th>
<th>$O(N^2)$</th>
<th>$O(N!)$</th>
</tr>
</thead>
</table>

$N$ (amount of data)
## Algorithmic Time

<table>
<thead>
<tr>
<th>n</th>
<th>O(n)</th>
<th>O(n²)</th>
<th>O(n!)</th>
</tr>
</thead>
<tbody>
<tr>
<td>n = 10</td>
<td>1 msec</td>
<td>1 msec</td>
<td>1 msec</td>
</tr>
<tr>
<td>n = 100</td>
<td>10 msec</td>
<td>100 msec</td>
<td>100! msec</td>
</tr>
<tr>
<td>n = 1,000</td>
<td>100 msec</td>
<td>10 sec</td>
<td>10!</td>
</tr>
<tr>
<td>n = 10,000</td>
<td>1 sec</td>
<td>16 min 40 sec</td>
<td></td>
</tr>
<tr>
<td>n = 100,000</td>
<td>10 sec</td>
<td>27.7 hr</td>
<td></td>
</tr>
<tr>
<td>n = 1,000,000</td>
<td>1 min 40 sec</td>
<td>115.74 days</td>
<td></td>
</tr>
</tbody>
</table>