Algorithms
A Look At Efficiency
1B

Big O Notation

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.

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

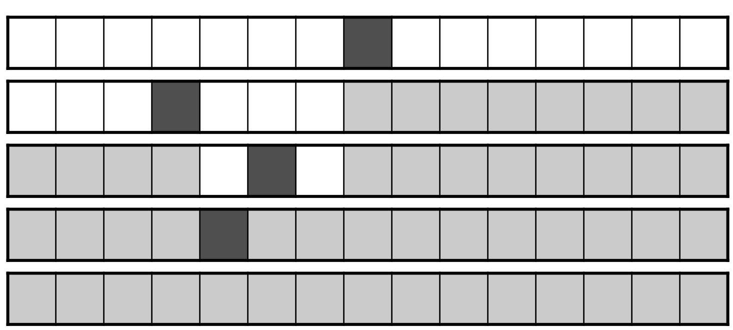
- Unique II Algorithm $C(n) = n(n-1)/2 = O(n^2)$

Example 5

```
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



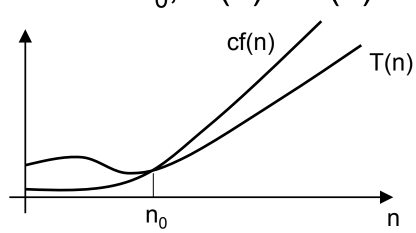
Binary Search: Worst Case

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

Array length	Iterations
15	4
31	5
63	6
n	

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 $n > n_0$, $cf(n) \geq T(n)$.



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: $c = 5 + 3/n_0 + 9/n_0^2$.
  - If $n_0 = 3$, then $c = 7$.
- Thus, $7n^2 > 5n^2 + 3n + 9$ for all $n > 3$.
  - So $5n^2 + 3n + 9 = O(n^2)$

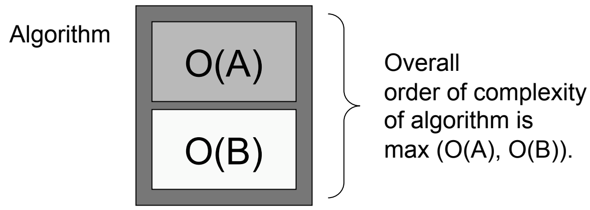
More about Big O

- Big O gives us an upper-bound approximation on the complexity of a computation.
- We can say that the following computation is $O(n^3)$:

```
for (int i = 0; i < n; i++)
   for (int j = 0; j < n; j++)
     System.out.println(i + " " + j);
```

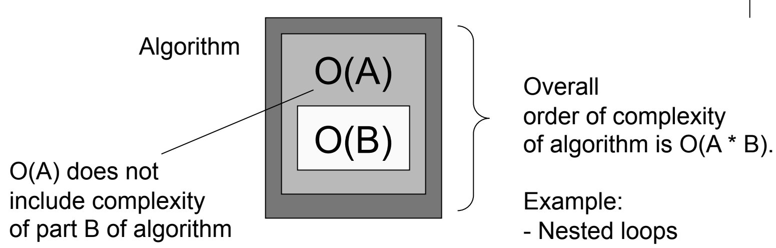
- A tighter bound would be $O(n^2)$, but both are technically correct.

Order of Complexity



- 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

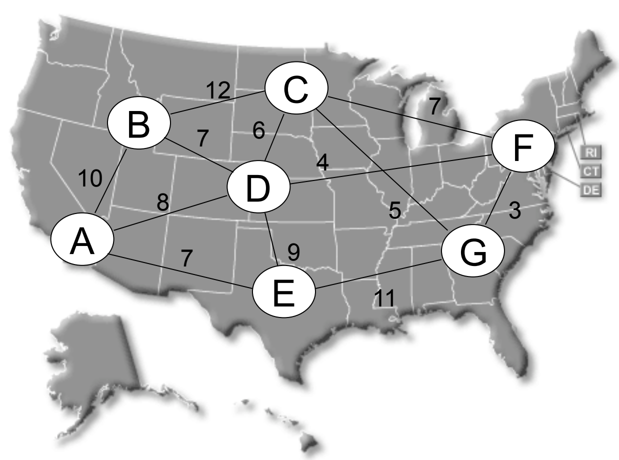


- Examples:
  - O(log N) * O(N) = O(N log N)
  - O(N log N) * O(N) = O(N^2 log N)
  - O(N) * O(1) = O(N)

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

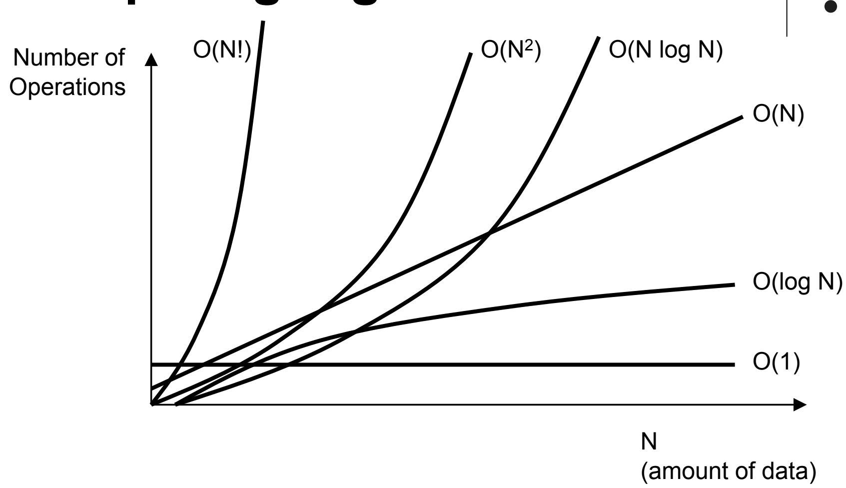


Traveling Salesperson

- 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)
  - ...

  how to build a route
- Maximum number of routes: ______ = O(______)

Comparing Big O Functions



Algorithmic Time

	O(n)	O(n^2)	O(n!)
n = 10	**1 msec**	**1 msec**	**1 msec**
n = 100	10 msec	100 msec	$\frac{100!}{10!}$ msec
n = 1,000	100 msec	10 sec	
n = 10,000	1 sec	16 min 40 sec	
n = 100,000	10 sec	27.7 hr	
n = 1,000,000	1 min 40 sec	115.74 days	