## Algorithms for searching an array

The algorithms and analysis here are adapted from the reading, *Algorithms Unlocked* by Cormen.

**Algorithm 1** An algorithm for searching an array

1: **procedure** LINEARSEARCH($A, n, x$) ▷ Find $x$ in array $A$ of length $n$
2: Set *answer* to NOT-FOUND.
3: Set $i$ to 1.
4: **while** $i \leq n$ **do**
5: If $A[i] = x$, then set *answer* to the value of $i$.
6: Increment $i$.
7: **end while**
8: **return** the value of *answer* as the output.
9: **end procedure**

Runtime analysis:

- Work is performed on lines 2-6 and line 8. The other lines contain only descriptive text.
- Let $t_i$ denote the amount of time for each execution of line $i$.
- On line 5, two things are happening: $A[i]$ is being checked and (sometimes) the value of *answer* is being updated. Let's use $t'_5$ and $t''_5$ to denote the time of these two activities.

Mininum running time (occurs when $x$ is never found):

$$t_2 + t_3 + (n+1) \cdot t_4 + n \cdot t'_5 + 0 \cdot t''_5 + n \cdot t_6 + t_8 = (t_4 + t'_5 + t_6) \cdot n + (t_2 + t_3 + t_4 + t_8)$$

Maximum running time (occurs when $x$ is found in every array cell):

$$t_2 + t_3 + (n+1) \cdot t_4 + n \cdot t'_5 + n \cdot t''_5 + n \cdot t_6 + t_8 = (t_4 + t'_5 + t''_5 + t_6) \cdot n + (t_2 + t_3 + t_4 + t_8)$$

Either way, we can see that the runtime grows *linearly* with $n$.

## Big-Oh Notation and Asymptotic runtime

When we analyze algorithms, we want to avoid getting bogged down into the details such as the cost of individual operations (e.g., specifying $t_2$, $t_3$, etc.). Asymptotic runtime analysis ignores these constant factors.

We say that $f(n) = O(g(n))$ if there exist positive constants $c$ and $n_0$ such that $0 \leq f(n) \leq c \cdot g(n)$ for all $n \geq n_0$.

LINEARSEARCH has a $O(n)$ runtime in the best and worst case. SENTINELLINEARSEARCH has a $O(1)$ runtime in the best case and $O(n)$ runtime in the worst case.

**Algorithm 2** A more efficient algorithm for searching an array

**procedure** SENTINELLINEARSEARCH($A, n, x$) ▷ Find $x$ in array $A$ of length $n$
    Save $A[n]$ into *last* and then put $x$ into $A[n]$.
    Set $i$ to 1.
    **while** $A[i] \neq x$ **do**
        Increment $i$.
    **end while**
    Restore $A[n]$ from *last*.
    If $i < n$ or $A[n] = x$, then return the vlaue of $i$ as the output.
    Otherwise, return NOT-FOUND as the output.
**end procedure**

When we talk about the asymptotic runtime of an algorithm, we typically consider the *worst-case* input and then we find a $g(n)$ such that the runtime of the algorithm is $O(g(n))$. The function $g(n)$ is typically a simple function of $n$, such as $n$, $n^2$, $2^n$, etc.

Finally, we typically want to find the *smallest* $g(n)$ such that the runtime is $O(g(n))$. For example, it is true that LINEARSEARCH is $O(n^2)$ as well as $O(n^3)$, etc. but we prefer to say $O(n)$ because this is the smallest function of $n$ for which the big-Oh relationship holds.