Linear Search in Pseudocode

**Input:** Integer array $A$, integer $k$ being searched.

**Output:** The least index $i$ such that $A[i] = k$; otherwise $-1$.

**Algorithm** `linSearch`($A$, $k$)

1. for $i$ ← 0 to $A.length - 1$
2. if $A[i] = k$ then
3. return $i$
4. return $-1$

Assume each line takes constant time to execute once.
Let $c_i$ be the time for line $i$. Then

$$(c_1 + c_2)n + \min\{c_3, c_1 + c_4\} \leq T_{\text{linSearch}}(n) \leq (c_1 + c_2)n + \max\{c_3, c_1 + c_4\}.$$ 

---

**The Big-O Notation**

**Definition**

Let $f, g : \mathbb{N} \to \mathbb{R}$ be functions. We say that $f$ is $O(g)$ if there is some $n_0 \in \mathbb{N}$ and some $c > 0$ from $\mathbb{R}$ such that for all $n \geq n_0$ we have

$$0 \leq f(n) \leq cg(n).$$

In other words:

$$O(g) = \{f : \mathbb{N} \to \mathbb{R} \mid \text{there is an } n_0 \in \mathbb{N} \text{ and some } c > 0 \text{ in } \mathbb{R} \text{ such that } 0 \leq f(n) \leq cg(n), \text{ for all } n \geq n_0.\}$$

Then “$f$ is $O(g)$” means $f \in O(g)$.

Informally, we say “for sufficiently large $n$” instead of “there is some $n_0 \in \mathbb{N}$ such that for all $n \geq n_0$...”.

---

Intention

“$f$ is $O(g)$” tells us that the growth rate of $f$ is no worse than that of $g$. Could be better.

- $c$ allows us to adjust for constants: $n^2$ obviously has same growth rate as $3n^2$, $20n^2$, $100n^2$...
  - consider $n \to an$ then
    $$n^2 \to a^2 \cdot n^2$$
    $$3n^2 \to a^2 \cdot 3n^2$$
    $$20n^2 \to a^2 \cdot 20n^2$$
    ...
  - Positivity condition ($0 \leq f(n) \leq \cdots$) required to ensure some useful properties, always satisfied by runtimes!
- $n_0$ allows a settling in period of atypical behaviour.
- $O$ allows us to concentrate on the big picture rather than details (many being implementation dependent).
Notational Convention

Write
\[ f = O(g), \]
instead of
\[ f \in O(g). \]

- Makes it convenient to have chains reasoning with inequalities etc.
- Notation here is from left to right. \( f = O(g) \) does not mean that \( O(g) = f \! \)
- \( f = f_1 + O(g) = f_2 + O(g) \) does not imply that \( f_1 = f_2 \).
- Seems strange but easy to get used to it and very useful.

More Examples of \( O \)

3. \( \lg(n) = O(n) \)
   Intuitively: \( \lg(n) < n \) for all \( n \geq 1 \).
   Need a proof. Well
   \[ \lg(n) < n \iff n < 2^n, \quad \text{for all } n > 0. \]
   Use induction on \( n \) to prove rhs.
   - Base case \( n = 1 \) is clearly true.
   - For induction step assume claim holds for \( n \). Then
     \[ 2^{n+1} = 2 \cdot 2^n > 2n, \quad \text{by induction hypothesis.} \]
   To complete the proof just need to show that \( 2n \geq n + 1 \).
   Now
   \[ 2n \geq n + 1 \iff n \geq 1, \]
   and we have finished.
   So we take \( c = 1 \) and \( n_0 = 1 \).

Examples of \( O \)

1. \( 3n^3 = O(n^3) \).
   Need \( c \) and \( n_0 \) so that \( 3n^3 \leq cn^3 \) for all \( n \geq n_0 \).
   Take \( c = 3, n_0 = 0 \).

2. \( 3n^3 + 8 = O(n^3) \).
   For a constant \( c > 0 \) we have
   \[ 3n^3 + 8 \leq cn^3 \iff 3 + \frac{8}{n^3} \leq c \quad \text{provided } n > 0. \]
   As \( n \) increases \( 8/n^3 \) decreases. Thus
   \[ 3 + \frac{8}{n^3} \leq 11 \quad \text{for all } n > 0. \]
   So we take \( c = 11, n_0 = 1 \). We can also take \( c = 4, n_0 = 2 \) or \( c = 3 + 8/27, n_0 = 3 \) etc.

5. \( 2^{100} = O(1) \).
   Take \( n_0 = 0 \) and \( c = 2^{100} \).
"Laws" of Big-O

**Theorem:** Let \( f_1, f_2, g_1, g_2 : \mathbb{N} \to \mathbb{R} \) be functions. Then:
1. For any constant \( a > 0 \) in \( \mathbb{R} \): \( f_1 = O(g_1) \Rightarrow af_1 \in O(g_1) \).
2. \( f_1 = O(g_1) \) and \( f_2 = O(g_2) \) \( \Rightarrow f_1 + f_2 = O(g_1 + g_2) \).
3. \( f_1 = O(g_1) \) and \( f_2 = O(g_2) \) \( \Rightarrow f_1 f_2 = O(g_1 g_2) \).
4. \( f_1 = O(g_1) \) and \( g_1 = O(g_2) \) \( \Rightarrow f_1 = O(g_2) \).
5. For any \( d \in \mathbb{N} \) if \( f_1 \) is polynomial of degree \( d \) with positive leading coefficient then \( f_1 = O(n^d) \).
6. For any constants \( a > 0 \) and \( b > 1 \) in \( \mathbb{R} \): \( n^a = O(b^a) \).
7. For any constant \( a > 0 \) in \( \mathbb{R} \): \( \lg(n^a) = O(\lg(n)) \).
8. For any constants \( a > 0 \) and \( b > 0 \) in \( \mathbb{R} \): \( \lg^a(n) = O(n^b) \).

Example (using Laws for \( O \))

\[
871 n^3 + 13 n^2 \lg^5(n) + 18 n + 566 = O(n^3).
\]

\[
871 n^3 + 13 n^2 \lg^5(n) + 18 n + 566
\]

\[
= 871 n^3 + 13 n^2 O(n) + 18 n + 566 \quad \text{by (8)}
\]

\[
= 871 n^3 + O(n^2) + 18 n + 566 \quad \text{by (3)}
\]

\[
= 871 n^3 + 18 n + 566 + O(n^3)
\]

\[
= O(n^3) + O(n^3) \quad \text{by (5)}
\]

\[
= O(n^3) \quad \text{by (2) \& (1)}
\]

Big-\( \Omega \) and Big-\( \Theta \)

**Definition**

Let \( f, g : \mathbb{N} \to \mathbb{R} \) be functions.

1. We say that \( f \) is \( \Omega(g) \) if there is an \( n_0 \in \mathbb{N} \) and \( c > 0 \) in \( \mathbb{R} \) such that for all \( n \geq n_0 \) we have
   \[
   f(n) \geq cg(n) > 0.
   \]

2. We say that \( f \) is \( \Theta(g) \), or \( f \) has the same asymptotic growth rate as \( g \), if \( f \) is \( O(g) \) and \( \Omega(g) \).

- Have corresponding 'laws of \( \Omega \)' (see notes).
- \( f = \Omega(g) \iff g = O(f) \).  [Prove this.]

Examples of \( f = \Omega(g) \)

1. Let \( f(n) = 3n^3 \) and \( g(n) = n^3 \).
   (combining this with Ex 1. for \( O \) gives \( 3n^3 = \Theta(n^3) \))

   Let \( n_0 = 0 \) and \( c = 1 \). Then for all \( n \geq n_0 \),
   \[
   f(n) = 3 n^3 \geq cg(n) = g(n).
   \]

2. Let \( f(n) = \lg(n) \) and \( g(n) = \lg(n^2) \).

   Well
   \[
   \lg(n^2) = 2 \lg(n).
   \]

   So take \( c = 1/2 \) and \( n_0 = 1 \). Then for every \( n \geq n_0 \) we have,
   \[
   f(n) = \lg(n) = \frac{1}{2} \lg(n) = \frac{1}{2} \lg(n^2) = \frac{1}{2} g(n).
   \]
Quick quiz: True or False?

$\sqrt{n^3} = O(n^2)$?

True. $\sqrt{n^3} = n^{3/2} \leq n^2$.

$2^{\lfloor \log n \rfloor} = O(n)$?

True. $2^{\lfloor \log n \rfloor} \leq 2^\log n = n$.

$2^{\lfloor \log n \rfloor} = \Omega(n)$?

True: Let $n \geq 2$. Then $\lfloor \log n \rfloor \geq (\log n) - 1 \geq 0$. Hence $2^{\lfloor \log n \rfloor} \geq 2^{\log n - 1} = n/2$. So take $n_0 = 2$, $c = 1/2$.

$n \log n = \Theta(n^2)$?

False: We do have $n \log n = O(n^2)$ but $n \log n$ is not $\Omega(n^2)$.

Further Reading

- Lecture notes 2 (handed out).
- If you have Goodrich & Tamassia [GT]:
  All of the chapter on “Analysis Tools” (especially the “Seven functions” and “Analysis of Algorithms” sections).
  NB: the title of the book is as given in slides of lecture 1, not as in note 1.
- If you have [CLRS]:
  Read chapter 3 on “Growth of Functions”.
- Wikipedia has a page about asymptotic notation:
  en.wikipedia.org/wiki/Asymptotic_notation