Chapter 5: Analyzing Algorithm

Running Time

1. How long does a program take to run?

2. How many “steps” does a program take to run?

3. How many “steps” does a program take, in terms of its input size?

4. Approximately how many steps...

5. Approximately how many steps... as the input size gets “large”?

Comparing functions formally

def f(n):
    for i in range(n):
        print(i)
Defn Let \( f, g : \mathbb{N} \to \mathbb{R}_{\geq 0} \).

We say \( g \) is absolutely dominated by \( f \) when
\[
\forall n \in \mathbb{N}, \ g(n) \leq f(n).
\]

We say \( g \) is eventually dominated by \( f \)
\[
\exists n_0 \in \mathbb{R}^+, \forall n \in \mathbb{N}, \ n \geq n_0 \Rightarrow g(n) \leq f(n).
\]

Example.
\[
n + 1000000 \text{ is eventually dominated by } n^2.
\]

We say \( g \) is eventually dominated up to a constant factor by \( f \) when
\[ \exists n_0, c \in \mathbb{R}^+, \forall n \in \mathbb{N}, n \geq n_0 \implies g(n) \leq cf(n) \]

Ex. \( f(n) = n \)
\[ g(n) = 3n + 1 \]

\( g \) is eventually dominated up to a constant factor by \( f \).

A shorter way of saying "\( g \) is e.d. up to a c by \( f \)" is "\( g \) is Big-Oh of \( f \)" denoted \( g \in O(f) \).

E.g., \( 3n + 1 \in O(n) \).
\[ a, b \in \mathbb{R}^+ \quad \text{and} \quad an + b \in O(n) \]

good exercise to prove!

\[ f, g : \mathbb{N} \rightarrow \mathbb{R}^+ \]

\[ g \in O(f) : \exists c, n_0 \in \mathbb{R}^+ \quad \forall n \in \mathbb{N}, \quad n \geq n_0 \Rightarrow g(n) \leq cf(n) \]

\[ O(f) = \{ g : \mathbb{N} \rightarrow \mathbb{R}^+ \quad \text{and} \quad \exists c, n_0 \in \mathbb{R}^+, \forall n \in \mathbb{N}, \quad n \geq n_0 \Rightarrow g(n) \leq cf(n) \} \]

"The step count of my algorithm is \( 3n + 2 = O(n) \)"
Properties of Big-Oh

Theorem (Big-Oh Hierarchy Theorem)

Let \( a, b \in \mathbb{R}^+ \). The following are true:

1. If \( a > 1 \) and \( b > 1 \), then \( \log_a n \in O(\log_b n) \).

2. If \( a \leq b \), then \( n^a \in O(n^b) \).

3. If \( 1 \leq a \leq b \), then \( a^n \in O(b^n) \).

4. If \( a > 1 \), then \( \log_a n \in O(n^b) \).

5. If \( b > 1 \), then \( n^a \in O(b^n) \).

\[
\log_2 n \in O(n^{0.0000000001})
\]

\[
n^{1000000} \in O(1.0000000001^n)
\]
45. If \( a > 1 \), then \( n^a \notin O(\log n) \).

\[
\begin{align*}
\log n & \\
\sqrt{n} & \\
n^{0.999} & \\
\text{linear } & n, 3n+1, 3Qn & \\
n^{1.0001} & \\
n^2 & \\
n^3 & \\
2^n & \\
\end{align*}
\]

**Combinations of functions**

**Ex. (warmup)**

**Worksheet:** For all \( f, g : N \rightarrow R^\ge \), \( g \in O(f) \Rightarrow f + g \in O(f) \).
Theorem (general sum)

\[ \forall f, g, h : \mathbb{N} \rightarrow \mathbb{R}_{\geq 0}, f \in O(h) \land g \in O(h) \implies f + g \in O(h) \]

\[ f(n) = n^{1,0000} , g(n) = 2^n \]

\[ h(n) = (2.0000)^n \]

\[ n^{1,0000} + 2^n \in O(2.0000^n) \]

Theorem (constant multiples)

\[ \forall f, g : \mathbb{N} \rightarrow \mathbb{R}_{\geq 0}, \forall k \in \mathbb{R}^+, \]

\[ f(n) = gn \]

\[ f(n) = n^{1,0000} \]

\[ n^{1,0000} + 2^n \in O(2.0000^n) \]
\( g \in O(f) \Rightarrow k \cdot g \in O(f) \).

\( n \in O(n^2) \)

\( n \in O(n) \)

\( n \in O(n) \)