Measuring algorithm performance

Big O

Example: Sequential search of an unordered array E

```c
int seqSearch (int[] E, int n, int k)
1    int ans, index;
2    ans = -1;
3    for (i = 0; i < n; i++) {
4        if (k == E[i])
5            ans = i;
6            break;
7    }
8    return ans;
```

Worst-case: \( T_{\text{worst}}(n) = 3 + 2n \)
Best-case: \( T(n) = 4 \)
Average-case: ???

Average case:
If \( I \) is an instance of size \( n \) of the problem and \( D \) is the set of all distinct possible instances of size \( n \), then the average case or the expected time is

\[
T_{\text{avg}}(n) = \sum_{I \in D} \Pr(I)T(I)
\]

\( S_i \) is the event that \( E[i] = k \)

\[
T_{\text{avg}}(n) = \sum_{i=0}^{n-1} \Pr(S_i)T(S_i)
= \sum_{i=0}^{n-1} \left( \frac{1}{n} \times (2i + 4) \right)
= 3 + n
\]

\[
T_{\text{avg}}(n) = \Pr(\text{succ}) T_{\text{succ}}(n) + \Pr(\text{fail}) T_{\text{fail}}(n)
= 50\% \times (3+n) + 50\% \times (3+2n)
= 3 + \frac{3}{2}n
\]

Lessons learned:
- Worst-case: the most commonly used
  - worst-case gives the upper bound, i.e., guarantees the algorithm will never take any longer
  - worst-case actually happens quite often
  - Worst-case is not much worse than the average (the previous example is the case)
- Average: harder to analyze
- Best-case: not that useful

Big Oh, etc.
- **Upper bound**: \( O(g(n)) = \{ f(n) : \) there exist positive constant \( c \) and \( n_0 \) such that \( 0 \leq f(n) \leq c g(n) \) for all \( n \geq n_0 \)\}
  - Meaning: \( g(n) \) grows faster up to a constant factor than any functions in \( O(g(n)) \).
- **Lower bound**: \( \Omega(g(n)) = \{ f(n) : \) there exist positive constant \( c \) and \( n_0 \) such that \( 0 \leq c g(n) \leq f(n) \) for all \( n \geq n_0 \} \)
- **Tight bound**: \( \Theta(g(n)) = \{ f(n) : \) there exist positive constant \( c_1, c_2 \) and \( n_0 \) such that \( 0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n) \) for all \( n \geq n_0 \} \)
Examples:

1. \(3n^2 + 5n + 24 = \Theta(n^2)\)

   \[\text{Proof:} \text{ By definition, we need to find } c_1, c_2 \text{ and } n_0 \text{ such that for all } n \geq n_0, c_1 n^2 \leq 3n^2 + 5n + 24 \leq c_2 n^2 \quad (1)\]

   Divide \((1)\) by \(n^2, c_1 \leq 3 + \frac{5}{n} + \frac{24}{n^2} \leq c_2 \]

   The Inequalities in \((2)\) hold for all \(n > 5, c_1 = 3, \text{ and } c_2 = 5.\]

   QED

Properties of Asymptotic Notation and Comparison of Functions

- Transitivity
  \(f \in O(g), g \in O(h) \Rightarrow f \in O(h); f \in \Theta(g), g \in \Theta(h) \Rightarrow f \in \Theta(h); f \in \Omega(g), g \in \Omega(h) \Rightarrow f \in \Omega(h).\)

- Reflexivity
  \(f \in O(f(n)); f \in \Theta(f(n)); f \in \Omega(f(n)).\)

- Symmetry
  \(f \in O(g) \iff g \in \Omega(f); f \in \Theta(g) \iff g \in \Theta(f); O(f+g) = O(\max(f,g)).\)

“little-oh of \(g\) of \(n\)”

\(o(g(n)) = \{f(n): \text{ for any positive constant } c, \text{ there exists a constant } n_0 \text{ such that for all } n \geq n_0, 0 \leq f(n) \leq c g(n)\}\)

Meaning:

- \(o(g(n))\) contains functions in \(O(g(n))\) excluding those in \(\Theta(g(n)).\)
- \(f(n)\) becomes insignificant relative to \(g(n)\) as \(n\) approaches infinity:
  \[\lim_{n \to \infty} f(n)/g(n) = 0.\]

  e.g., \(2n = o(n^2),\) but \(2n^2 \neq o(n^2).\)

  e.g., \(\lg(n) = o(\sqrt{n})\) (Proof by using L’Hospital rule)

Typical asymptotic complexities are

- \(\Omega(n)\), logarithmic (sub-linear)
- \(\Theta(n)\), linear
- \(\Omega(n \lg n), n \lg n\)
- \(\Omega(n^2)\), quadratic
- \(\Omega(n^3)\), cubic
- \(\Theta(2^n)\), exponential

Note:

- Logarithms:
  \(\lg n = \log_2 n; \lg e = \log_{10} n; \lg b = \log_b 0 / \log_b e; \)
  Under the log \(\Theta,\) the base of a logarithm does not matter.

- Factorial \(n!\) grows faster than exponential:
  \(\omega(n!); \omega(n^{\omega(n)}); \omega(n^{\Omega(n)}); \omega(n) = \Theta(n!);\)

  See Section 3.2 in CLRS for more details