Algorithms and Theory of Computation

Lecture 2: Big-O Notation
Graph Algorithms

Xiaohui Bei

MAS 714

August 14, 2018
O, \Omega, \text{ and } \Theta

Let \( T, f \) are two monotone increasing functions that maps from \( \mathbb{N} \) to \( \mathbb{R}^+ \).

**Asymptotic Upper Bounds**

We say \( T(n) = O(f(n)) \) if there exists constants \( c > 0 \) and \( n_0 \geq 0 \), such that for all \( n \geq n_0 \), we have \( T(n) \leq c \cdot f(n) \).

**Examples**

- \( T(n) = 1000n + 100 \implies T(n) = O(n) \)
  - set \( c = 1001 \) and \( n_0 = 100 \)

- \( T(n) = pn^2 + qn + r \) for constants \( p, q, r > 0 \implies T(n) = O(n^2) \)
  - set \( c = p + q + r \) and \( n_0 = 1 \)
  - also correct to say \( T(n) = O(n^3) \)
Some Remarks

- **Equals sign.** $O(f(n))$ is a set of functions, but computer scientists often write $T(n) = O(f(n))$ instead of $T(n) \in O(f(n))$.
  - Consider $f(n) = 5n^3$ and $g(n) = 3n^2$, we write $f(n) = O(n^3) = g(n)$, but it does not mean $f(n) = g(n)$.

- **Domain.** The domain of $f(n)$ is typically the natural numbers $\{0, 1, 2, \ldots\}$.
  - Sometimes we restrict to a subset of the natural numbers.
  - Other times we extend to the reals.

- **Nonnegative functions.** When using big-O notation, we assume that the functions involved are (asymptotically) nonnegative.
\( O, \Omega, \text{ and } \Theta \)

Let \( T, f \) are two monotone increasing functions that maps from \( \mathbb{N} \) to \( \mathbb{R}^+ \).

**Asymptotic Lower Bounds**

We say \( T(n) = \Omega(f(n)) \) if there exists constants \( \epsilon > 0 \) and \( n_0 \geq 0 \), such that for all \( n \geq n_0 \), we have \( T(n) \geq \epsilon \cdot f(n) \).

**Examples**

- \( T(n) = pn^2 + qn + r \) for constants \( p, q, r > 0 \) \( \Rightarrow \) \( T(n) = \Omega(n^2) \)
  - set \( \epsilon = p \) and \( n_0 = 1 \)
  - also correct to say \( T(n) = \Omega(n) \)

**Meaningful Statement.** Any compare-based sorting algorithm requires \( \Omega(n \log n) \) compares in the worst case.

**Meaningless Statement.** Any compare-based sorting algorithm requires \( O(n \log n) \) compares in the worst case.
Let $T, f$ are two monotone increasing functions that maps from $\mathbb{N}$ to $\mathbb{R}^+$. 

**Asymptotic Tight Bounds**

We say $T(n) = \Theta(f(n))$ if there exists constants $c_1, c_2 > 0$ and $n_0 \geq 0$, such that for all $n \geq n_0$, we have $c_1 \cdot f(n) \leq T(n) \leq c_2 \cdot f(n)$.

**Alternative definition**

$T(n) = \Theta(f(n))$ if $T(n)$ is both $O(f(n))$ and also $\Omega(f(n))$.

**Examples**

- $T(n) = pn^2 + qn + r$ for constants $p, q, r > 0 \implies T(n) = \Theta(n^2)$
  - set $c_1 = p, c_2 = p + q + r$ and $n_0 = 1$
  - $T(n)$ is neither $\Theta(n)$ nor $\Theta(n^3)$
Properties of Asymptotic Growth Rates

1. If \( f = O(g) \) and \( g = O(h) \), then \( f = O(h) \).

2. If \( f = \Omega(g) \) and \( g = \Omega(h) \), then \( f = \Omega(h) \).

3. If \( f = \Theta(g) \) and \( g = \Theta(h) \), then \( f = \Theta(h) \).

4. If \( f = O(h) \) and \( g = O(h) \), then \( f + g = O(h) \).

5. If \( g = O(f) \), then \( f + g = \Theta(f) \).

6. If \( \lim_{n \to \infty} \frac{f(n)}{g(n)} = c > 0 \), then \( f = \Theta(g) \).
Some Common Functions

1. **Polynomials.** Let $f$ be a polynomial of degree $d$, in which the coefficient $a_d$ is positive. Then $f = O(n^d)$.
   - Asymptotic rate of growth is determined by their “high-order term”.

---

**Polynomial-Time Algorithm**

A polynomial-time algorithm is one with running time $O(n^d)$ for some constant $d$.

---

2. **Logarithms.** For every $a, b > 1$ and every $x > 0$, we have
   \[ \log_a n = \Theta(\log_b n) = O(n^x). \]
   - No need to specify base (assuming it’s a constant).
   - Logarithms are always better than polynomials.

3. **Exponentials.** For every $r > 1$ and every $d > 0$, we have $n^d = O(r^n)$.
   - Polynomials are always better than exponentials.
More Notations

Asymptotic Smaller

We say $T(n) = o(f(n))$ if $\lim_{n \to \infty} \frac{T(n)}{f(n)} = 0$.

Asymptotic Larger

We say $T(n) = \omega(f(n))$ if $\lim_{n \to \infty} \frac{f(n)}{T(n)} = 0$. 
Asymptotic Upper Bounds

We say $T(m, n) = O(f(m, n))$ if there exists constants $c > 0$, $m_0 \geq 0$ and $n_0 \geq 0$, such that for all $m \geq m_0$ and $n \geq n_0$, we have $T(m, n) \leq c \cdot f(m, n)$.

Similar definitions for $\Omega$ and $\Theta$.

Examples

$T(m, n) = 32mn^2 + 17mn + 32n^3$

- $T(m, n)$ is both $O(mn^2 + n^3)$ and $O(mn^3)$.
- $T(m, n)$ is neither $O(n^3)$ nor $O(mn^2)$. 
We discussed algorithms in pseudocode, but how the data will be represented in an actual implementation of the algorithm?

- **Data Structures**: a particular way of organizing data such that it can be used efficiently in an algorithm.
- Appropriately designed data structures can help to give more efficient algorithms.
- Data structure in “Number Addition”: arrays.

**Program = Algorithm + Data Structure**
Common Data Structures

- Elementary Data Structures
  - Arrays
  - Linked Lists
  - Stacks and Queues
- Priority Queues
- Hash Tables
- Binary Search Trees
- many others
Graph Algorithms
Graphs

One of the most important combinatoric object in Computer Science, Optimization, Combinatorics.

Figure: Seven Bridges of Königsberg
A graph $G$ consists of a set $V$ of vertices and a set $E$ of edges

- **Directed Graph:** Each edge $e \in E$ is an ordered pair $(u, v)$ for some $u, v \in V$.
- **Undirected Graph:** Each edge $e \in E$ is an unordered pair $\{u, v\}$ for some $u, v \in V$.

**Example**

In the above undirected graph $G = (V, E)$:

- $V = \{1, 2, 3, 4, 5, 6\}$
- $E = \{\{1, 2\}, \{1, 3\}, \{2, 3\}, \{2, 4\}, \{3, 4\}, \{3, 5\}, \{4, 5\}, \{4, 6\}\}$
Applications

**Transportation Networks**
The map of routes served by an airline carrier/railway company.
- vertices are airports/train stations;
- an edges from $u$ to $v$ if there is a flight/railway track between them.

**Information Networks**
The World Wide Web can be viewed as a directed graph.
- vertices are Web pages;
- an edge from $u$ to $v$ if $u$ has a hyperlink to $v$.

**Social Networks**
A collection of people who interact with each other forms an undirected graph.
- vertices are people
- an edge joining $u$ and $v$ if they are friends.

**Dependency Networks**
A directed graph that captures the interdependencies among a collection of objects, e.g., in an university:
- vertices are courses offered
- an edge from $u$ to $v$ if $u$ is a prerequisite for $v$. 
Notation and Convention

- usually use \( n = |V| \) and \( m = |E| \)
- \( u \) and \( v \) are the end points of an edge \( e = \{u, v\} \)
- multi-graphs may have
  - loops which are edges \((u, u)\)
  - multi-edges which are different edges between same pair of vertices
- in most of this class we only consider simple graphs
Graph Representation

Adjacency Matrix

Represent a graph $G = (V, E)$ by an $n \times n$ adjacency matrix $A$, where


- The matrix is symmetric for undirected graphs.

Advantages:

- can check if an edge $(u, v) \in E$ in $O(1)$ time
- can do linear algebra on the matrix

Disadvantages:

- require $\Omega(n^2)$ space even when $m = o(n^2)$
- cannot examine all edges incident to a given node efficiently
Graph Representation

Adjacency List

Represent a graph $G = (V, E)$ by **adjacency lists**, which is an array $\text{Adj}$ of length $n$, where

- each entry $\text{Adj}[v]$ is a list containing all vertices adjacent to vertex $v$

Advantages:

- take $O(n + m)$ space, which is close to optimal
- can browse all edges incident to a given vertex efficiently

Disadvantages:

- hard to find a specific edge

Standard representation for graphs.
A Concrete Representation

Linked List

Linked lists are a data structure to represent sequence whose length is arbitrary and changeable.

- each entry in a linked list consists of a cell for data and a pointer that points to the next entry
- there is a pointer to the first element
- the last entry points to NIL

- Advantages: easy to add an element into a linked list, and to sequentially read the list.
- Disadvantages: no random access.
Example of a Linked List

Example of an Adjacency List