

CSE 21—Mathematics for Algorithm and System Analysis

Spring, 2006

June 6, 2006

Lecture 19

Asymptotic Notation

Instructor: Neil Rhodes

Breadth/Depth-First Search

Breadth-First Search (BFS)

- Add root to queue
while queue is not empty
 retrieve vertex from head of queue
 print vertex
 add children (in order) to tail of queue

Depth-First Search (DFS)

- Visit(root)
- procedure Visit(vertex) *preorder traversal*
 print vertex
 foreach child of vertex
 visit(child)

Spanning Tree

Definition

- A *spanning tree* of a simple graph $G=(V,E)$ is a subgraph $T=(V,E')$ which is a tree

Minimal Spanning Tree

- Given a connected weighted graph $G=(V,E,W)$ (where W is a function with domain E and codomain $\mathbb{R}$)
- A minimal spanning tree $T=(V,E',W')$ of G is a spanning tree whose sum of weights is no more than that of any other spanning tree of G

Minimal spanning Tree

Generating a minimal spanning tree for a simple graph $G=(V,E,W)$ (Prim's)

- Start with $E'=\{\}$
- Start with $V'=\{v\}$ for some v in V .
- While $|V'| < |V|$
 - Find the edge e from E with exactly one edge in V' of minimum weight
 - Add e to E'
 - Add the other vertex of e to V'

Alternative (Kruskal's)

- Start with $E'=\{\}$
- While $T=(V,E')$ is not connected
 - Find the cheapest edge from E that doesn't create a cycle in (V,E')
 - Add e to E'

Computational Tractability

Worst case running time. Obtain bound on largest possible running time of algorithm on input of a given size N , and see how this scales with N .

- Generally captures efficiency in practice.
- Draconian view, but hard to find effective alternative.

Desirable scaling property. When the input size increases by a factor of 2, the algorithm should only slow down by some constant factor C .

There exists constants $c > 0$ and $d > 0$ such that on every input of size N , its running time is bounded by $c N^d$ steps.

Def. An algorithm is **efficient** if it has **polynomial** running time.

Justification. It really works in practice!

Asymptotic Order of Growth

Big-Oh of $f(n)$

Upper bounds. $T(n)$ is $O(f(n))$ if there exist constants $c > 0$ and $n_0 \geq 0$ such that for all $n \geq n_0$ we have $0 \leq T(n) \leq c \cdot f(n)$.

Big-Omega of $f(n)$

Lower bounds. $T(n)$ is $\Omega(f(n))$ if there exist constants $c > 0$ and $n_0 \geq 0$ such that for all $n \geq n_0$ we have $T(n) \geq c \cdot f(n) \geq 0$

Tight bounds. $T(n)$ is $\Theta(f(n))$ if $T(n)$ is both $O(f(n))$ and $\Omega(f(n))$.

Ex: $T(n) = 32n^2 + 17n + 32$.

- $T(n)$ is $O(n^2)$, $O(n^3)$, $\Omega(n^2)$, $\Omega(n)$, and $\Theta(n^2)$.
- $T(n)$ is not $O(n)$, $\Omega(n^3)$, $\Theta(n)$, or $\Theta(n^3)$, .

Slight abuse of notation. $T(n) = O(f(n))$.

Vacuous statement. Any comparison-based sorting algorithm requires at least $O(n \log n)$ comparisons.

Properties

Transitivity. If $f = O(g)$ and $g = O(h)$ then $f = O(h)$.

Additivity. $O(f(n)) + O(g(n)) = O(\max\{f(n), g(n)\})$

Multiplication by a constant. $O(k \cdot g(n)) = O(g(n)) \quad k > 0$

Example

Prove that $f(n) = 3n^3 - 10n^2 + n - 10 = O(n^3)$

Example

Prove that $f(n) = 3n^3 - 10n^2 + n - 10 \neq O(n^2)$

Example

Prove that $f(n) = 3n^3 - 10n^2 + n - 10 = \Omega(n^3)$