CSE 421
Algorithms
Richard Anderson
Lecture 4

Announcements
• Homework 2, Due October 11, 1:30 pm.
• Reading
  – Chapter 2.1, 2.2
  – Chapter 3 (Mostly review)
  – Start on Chapter 4

Today
• Finish discussion of asymptotics
  – O, Ω, Θ
• Graph theory terminology
• Basic graph algorithms

Formalizing growth rates
• $T(n) = O(f(n)) \quad [T : \mathbb{Z}^+ \rightarrow \mathbb{R}^+]$
  – If sufficiently large $n$, $T(n)$ is bounded by a constant multiple of $f(n)$
  – Exist $c, n_0$, such that for $n > n_0$, $T(n) < c f(n)$
• $T(n) = O(f(n))$ will be written as:
  $T(n) = O(f(n))$
  – Be careful with this notation

Order the following functions in increasing order by their growth rate
a) $n \log^4 n$
b) $2n^2 + 10n$
c) $2^{n^{100}}$
d) $100n + \log^8 n$
e) $n^{100}$
f) $3^n$
g) $1000 \log^{10} n$
h) $n^{1/2}$

Ordering growth rates
• For $b > 0$ and $x > 0$
  – $\log^n n$ is $O(n^x)$
• For $r > 1$ and $d > 0$
  – $n^d$ is $O(r^n)$
Lower bounds

- \( T(n) = \Omega(f(n)) \)
  - \( T(n) \) is at least a constant multiple of \( f(n) \)
  - There exists an \( n_0 \) and \( \varepsilon > 0 \) such that \( T(n) > \varepsilon f(n) \) for all \( n > n_0 \)
- Warning: definitions of \( \Omega \) vary

- \( T(n) = \Theta(f(n)) \) if \( T(n) = O(f(n)) \) and \( T(n) = \Omega(f(n)) \)

True or False

- \( n \log n \) is \( O(n^2) \)
- \( n^3 \) is \( O(4n^3 + 2n + n) \)
- \( n^{-1} \) is \( O(n^2) \)
- \( n^{-1} \) is \( \Omega(n^{-2}) \)
- \( f(n) = n^2 \) if \( n \) is even, \( 0 \) if \( n \) is odd
  - \( f(n) \) is \( \Omega(n^2) \)

Useful Theorems

- If \( \lim (f(n) / g(n)) = c \) for \( c > 0 \) then \( f(n) = \Theta(g(n)) \)

- If \( f(n) \) is \( O(g(n)) \) and \( g(n) \) is \( O(h(n)) \) then \( f(n) \) is \( O(h(n)) \)

- If \( f(n) \) is \( O(h(n)) \) and \( g(n) \) is \( O(h(n)) \) then \( f(n) + g(n) \) is \( O(h(n)) \)

Graph Theory

- \( G = (V, E) \)
  - \( V \) – vertices
  - \( E \) – edges
- Undirected graphs
  - Edges sets of two vertices \( \{u, v\} \)
- Directed graphs
  - Edges ordered pairs \( (u, v) \)
- Many other flavors
  - Edge / vertices weights
  - Parallel edges
  - Self loops

Definitions

- Path: \( v_1, v_2, \ldots, v_k \) with \( (v_i, v_{i+1}) \) in \( E \)
  - Simple Path
  - Cycle
  - Simple Cycle
- Distance
- Connectivity
  - Undirected
  - Directed (strong connectivity)
- Trees
  - Rooted
  - Unrooted

Graph search

- Find a path from \( s \) to \( t \)

\[ S = \{s\} \]

While there exists \( u, v \) in \( E \) with \( u \) in \( S \) and \( v \) not in \( S \)

\[ \text{Pred}(v) = u \]

Add \( v \) to \( S \)

If \( v = t \) then path found
Breadth first search

• Explore vertices in layers
  – s in layer 1
  – Neighbors of s in layer 2
  – Neighbors of layer 2 in layer 3 . . .

Key observation

• All edges go between vertices on the same layer or adjacent layers

Bipartite

• A graph V is bipartite if V can be partitioned into V₁, V₂ such that all edges go between V₁ and V₂
• A graph is bipartite if it can be two colored

Testing Bipartiteness

• If a graph contains an odd cycle, it is not bipartite

Algorithm

• Run BFS
• Color odd layers red, even layers blue
• If no edges between the same layer, the graph is bipartite
• If edge between two vertices of the same layer, then there is an odd cycle, and the graph is not bipartite