Introducing Complexity Theory via Reachability

CS254

Chris Pollett

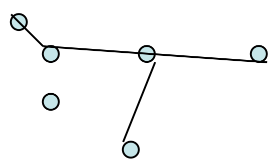
Aug. 23, 2006.

Outline

- Graph Reachability
- Estimating algorithm runtimes

Graph Reachability

- A **graph** G is an ordered pair (V,E) of nodes V and edges E.



- Given a graph G and two nodes 1, n $\in$ V, the REACHABILITY problem is to try to determine if there is a path from 1 to n.
- An *instance* of the problem, is the problem for a particular graph and a particular pair of nodes.
- The problem is called a *decision problem* because the answer is yes or no.

Specifying algorithms

- Next week we will discuss Turing Machine a formal setting to specify algorithms to solve decision problems.
- Here, though, is one way to solve this problem:
  0) Let S be a set of nodes. Initially, set $S=\{1\}$.
  1) A node is called *marked* if it has ever appeared in S.
  2) Repeat until either n is marked or S is empty.
     a) Choose a node i from S and remove it from S. For each edge (i,j) out of i in E, if node j is *unmarked,* mark it, and add it to S.
  3) If n is marked then answer yes; otherwise, answer no.
- Although this works it leaves out important details…

More on Specifying Algorithms

- We left out:
  - How graphs are represented as an input to the algorithm? (Turns out won't matter much).
  - How do we choose i? Choice might affect whether the algorithm is breadth-first or depth-first, or some other kind of search. It turns out the algorithm works regardless of the precise choice.
  - How efficient is the algorithm? It turns out to be efficient taking roughly n^2 steps and roughly linear space.

Asymptotics

- The "roughly" of the last slide can be made more precise using various definitions for asymptotic behaviors:

Def$^{\underline{n}}$ Let **N** be the nonnegative integers. Let f and g be functions from **N** to **N**.

1) We write $f(n) = O(g(n))$ if there are positive integer c, m such that $f(n) \leq c \bullet g(\mathrm{n})$ for all $n \geq m$. "f grows as g or slower"
2) We write $f(n) = \Omega\ (g(n))$ if $g(n) = O(f(n))$.
3) We write $f(n) = \Theta(g(n))$ if $\mathrm{f(n)} = \Omega(g(n))$ and $f(n) = O(g(n))$.

- For example, if $p(n)$ is a polynomial of degree d, one can show $p(n)= \Theta(n^d)$.

Tractable versus Intractable

- For the purposes of this class , we will view the algorithms with polynomial, worst-case, asymptotic runtime as *tractable* (i.e., doable) and those with nonpolynomial asymptotic runtime as *intractable*.
- For example, a $\Omega(2^n)$ algorithm would be considered intractable.