Functions, Graphs, Trees and proofs

CS154
Chris Pollett
Jan 31, 2007.

Outline

- O-Notation
- Equivalence Relations
- Graphs and Trees
- Proofs and Proof Strategies
- Strings

Growth Rates of Functions

- **Def$^{\underline{n}}$** Let **N** be the nonnegative integers. Let f and g be functions from **N** to **N**.
  - 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"
  - We write $f(n) = \Omega\ (g(n))$ if $g(n) = O(f(n))$.
  - We write $f(n) = \Theta(g(n))$ if $\mathrm{f(n)} = \Omega(g(n))$ and $f(n) = O(g(n))$.
- For example, $n^2+1 = O(n^2)$. To see this notice, for all $n \geq 1$, $n^2+1 \leq n^2+ n^2 \leq 2 \cdot n^2$. So m=1, c=2 in the above definition.
- You might want to convince yourself that:

  $n^3 = \Omega(n^2+n+1)$ and $n^3 + n^2 = \Theta(n^3)$.

Equivalence Relations

- One particularly useful kind of relation is an **equivalence relation**. Such a relation acts like ‘=’.
- Like the binary relation equals we will write equivalences in infix notation. i.e., we’ll write xRy rather than $(x,y) \in R$ or R(x,y).
- A binary relation R is an equivalence relation if for each x,y,z:
  - R is **reflexive**, that is, xRx. (xRx is just R written in infix and we write xRx to mean xRx = TRUE).
  - R is **symmetric**, that is, xRy implies yRx
  - R is **transitive**, that is, xRy and yRz implies xRz.
- The **equivalence class of x**, denoted [x], is the set:

$$\{y \mid xRy\}$$

- We often write $\equiv$ or ~ rather than R for equivalence relations.

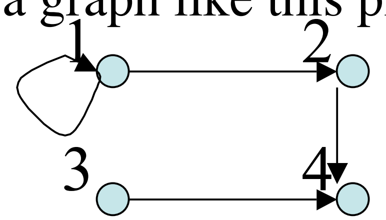
Example Equivalence Relations

- Last day, we defined the natural numbers in terms of sets.
- Let '-' be coded as 0, and '+' be coded as 1.
- **Z** - the integers are $\{[(sgn, n)] \mid sgn \in \{-, +\} \wedge n \in \mathbf{N}\}$ under the equivalence relation:

  $(sgn, n) \sim (sgn', n')$ if $n=n'$ and $sgn = sgn'$ or if $n=n'=0$
- To keep things simple we abbreviate (+, n) as n and (-, n) as -n. The $n=n'=0$ case is so that $-0 \sim 0$.
- You might want to think how addition, subtraction, and less than can be defined within this definition of the integers.
- Once we do this, we get the usual view of the integers as ..-2,-1,0,1,2..
- **Q** - the rational numbers can be defined as the set of equivalence classes of pairs of integers (p,q) (which we write as p/q) such that $q \geq 1$ and where $p/q \sim p'/q'$ if and only if $p \cdot q' = p' \cdot q$.
- For example, $1/2 \sim 2/4$ as $1 \cdot 4 = 2 \cdot 2$.

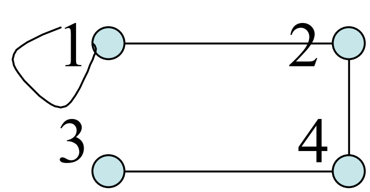
Graphs

- A **graph (**sometimes called a **directed graph**) is a pair G=(V,E) where V is a set of vertices (aka points or nodes) and $E \subseteq VxV$ is a set of edges between points. For example, ({1,2,3,4}, {(1,2),(2,4), (1,1),(3,4)})
- We can draw a graph like this pictorially:



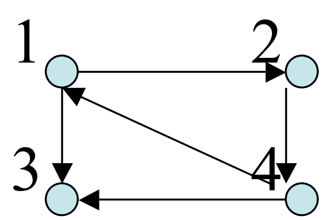
- An edge of the form (v,v) is called a **loop.** For example, (1,1) above.
- An **undirected graph** (or just a **graph**) is graph in which we can ignore the direction on the edges. One way to do this is to require that if (v,w) is in E then (w,v) is also in E.
- For example, the undirected version of the above graph would be :

  ({1,2,3,4}, {(1,2),(2,1), (2,4),(4,2), (1,1),(3,4), (4,3)})



More on Graphs

- Last day, we defined the cartesian power of a set A^n=Ax..n time..xA.
- A **sequence** of elements from a set A is a tuple in A^n for some n.
- A sequence of edges of the form ((v_1, v_2), (v_2, v_3), …,(v_{n-1},v_n)) in a graph is called a **walk.** For example, w=((1,2), (2,4), (4,1), (1,2)) below is a walk.
- The length of a **walk** is the number of edges in it. length(w) = 4
- A **path** is a walk in which no edge is repeated. For example, p=((1,2), (2,4), (4,1), (1,3)) below is a path, w is not.
- A **simple path** is a path that does not go out of any vertex more than once. For example, p´=((1,2), (2,4), (4,3)) is simple, p is not a simple path.
- A **cycle** is a path which begins and ends at the same node. A cycle is **simple** if it does not repeat nodes except the end point twice. For example, ((1,2), (2,4), (4,1), (1,2), (2,4), (4,1)) is a cycle but is not simple; whereas, ((1,2), (2,4), (4,1)) is a simple cycle.

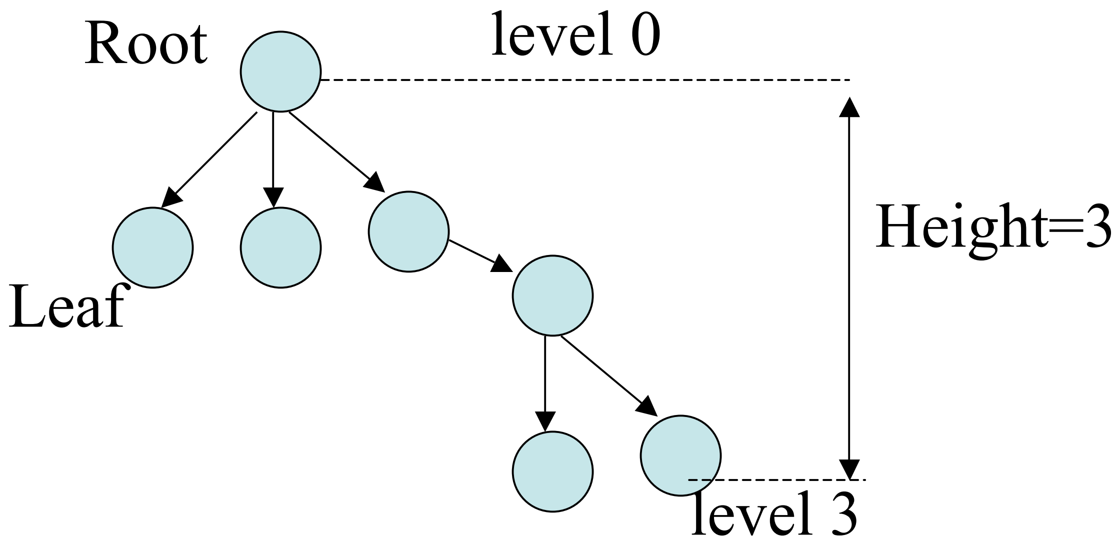


Finding Simple Paths

- In this course, we will find it useful to have an algorithm which on inputs a graph $G=(V,E)$ and two vertices s, t, computes a simple path from s to t, if there is a simple path between these points.
- To do this we maintain a set A of active nodes and a set S of seen nodes.
- Initialize $A=\{s\}$, $S=\emptyset$.
- Repeat until either $t \in A$ or $A=\emptyset$
  - Pick an $x \in A$, set $S := S \cup \{x\}$.
  - Let $\mathrm{UnseenChild}(x) := \{y \mid (x,y) \in E \wedge y \notin S\}$.
  - Set $A := A \cup \mathrm{UnseenChild}(x) - \{x\}$
- If $A=\emptyset$ then output “there is no path s to t”
- Otherwise, we can find a path in reverse order by looking in S for some x such that $(x,t) \in E$, then looking in S for some y such that $(y,x) \in E$, and so on until we get back to s. This will be a simple path.

Trees

- A **tree** is a graph without cycles, and that has one distinct vertex, called the **root,** such that there is exactly one path from the root to every other vertex.
- The root has no incoming edges.
- Any node without outgoing edges is called a **leaf**.
- In (v,w) is an edge in a tree, then v is called the **parent** of w and w is called the **child** of v.
- The **level** of a vertex is the number of edges in the path from the root to that vertex.
- The **height** of a tree is the largest level number of any vertex.



Definitions, Theorems, Proofs

- **Definitions** describes the objects and notions that we use. We want our definitions to be as precise as possible.
- Once we have made some definitions we make **mathematical statements** involving them.
- A **proof** is a convincing logical argument that a statement is true.
- A **theorem** is a mathematical statement which has been proved true.
- A **lemma** is a simple mathematical statement which has been proved true and which will be used in the proof of a theorem.
- A **proposition** is a mathematical statement with an easy proof. One can view it like a warm-up result, which does not immediately lead to the proof of a theorem
- A **corollary** is a mathematical statement which can be proved easily once some theorem is known.