CSL202: Discrete Mathematical Structures

Ragesh Jaiswal, CSE, IIT Delhi

Induction and Recursion

- We will only consider simple graphs for this discussion which are graphs that do not have self loops or multi-edges (i.e., multiple edges between a pair of vertices).

Definition (Strongly connected graph)

An undirected graph is called strongly connected iff for every pair of vertices in the graph there is a path between these vertices.

Definition (Tree)

An undirected graph is called a tree iff the graph is strongly connected and does not have any cycles.

Definition (Cycle)

A sequence of vertices $v_1, v_2, ..., v_k$ in an undirected graph is called a cycle iff $k > 3$, $v_1 = v_k$, $v_1, v_2, ..., v_{k-1}$ are distinct, and for every $1 \leq i \leq k-1$, there is an edge between v_i and v_{i+1}.

- Show that: Every tree with n vertices has exactly $(n-1)$ edges.

Definition (Strong induction)

To prove that $P(n)$ is true for all positive integers n, where $P(n)$ is a propositional function, we complete two steps:

- Basis step: We verify that $P(1)$ is true.
- Inductive step: We show that the conditional statement $[P(1) \wedge P(2) \wedge ... \wedge P(k)] \rightarrow P(k+1)$ is true for all positive integers k.

- Strong induction is sometimes called *second principle of mathematical induction* or *complete induction*.

Definition (Strong induction)

To prove that $P(n)$ is true for all positive integers n, where $P(n)$ is a propositional function, we complete two steps:

- Basis step: We verify that $P(1)$ is true.
- Inductive step: We show that the conditional statement $[P(1) \wedge P(2) \wedge ... \wedge P(k)] \rightarrow P(k+1)$ is true for all positive integers k.

- If the inductive step is valid only for integers greater than a particular integer.
  - Basis step: We verify that the propositions $P(b), P(b+1), ..., P(b+j)$ are true.
  - Inductive step: We show that $[P(b) \wedge P(b+1) \wedge ... \wedge P(k)] \rightarrow P(k+1)$ is true for every integer $k \geq b+j$.

- Show that if n is an integer greater than 1, then n can be written as the product of primes.

- Show that if n is an integer greater than 1, then n can be written as the product of primes.
- Prove that every amount of postage of 12 cents or more can be formed using just 4-cent and 5-cent stamps.

- The process of defining objects in terms of itself is called *recursion*.
- Examples
  - Recursively defined functions.
  - Recursively defined sets and structures

- Recursively defined functions.
  - Functions with non-negative integers as its domain.
  - Basis step: Specify the value of the function at 0.
  - Recursive step: Give a rule for finding its value at an integer from its values at smaller integers.
  - Such functions may alternatively be interpreted as a sequence.
- Example: Fibonacci sequence
  - Basis step: $f_0 = 0, f_1 = 1$.
  - Recursive step: $f_n = f_{n-1} + f_{n-2}$ for $n = 2, 3, 4, ...$
  - Show that whenever $n \geq 3, f_n > \alpha^{n-2}$, where $\alpha = (1 + \sqrt{5})/2$.

- Recursively defined sets and structures
  - Basis step: An initial collection of elements is specified.
  - Recursive step: Rules for forming new elements in the set from those already known to be in the set are provided.
  - *Exclusion Rule*: Recursively defined set contains nothing other than those elements specified in the basis step or generated by applications of the recursive step.
- Examples:
  - Multiples of 3:
    - Basis step: $3 \in S$
    - Recursive step: If $x \in S$ and $y \in S$, then $x + y \in S$.
  - The set Σ^* of strings over the alphabet Σ:
    - Basis step: ?
    - Recursive step: ?

- Examples:
  - Multiples of 3:
    - Basis step: $3 \in S$
    - Recursive step: If $x \in S$ and $y \in S$, then $x + y \in S$.
  - The set Σ^* of strings over the alphabet Σ:
    - Basis step: $\lambda \in Sigma^*$ (where λ is the empty string containing no symbols).
    - Recursive step: If $w \in \Sigma^*$ and $x \in \Sigma$, then $wx \in \Sigma^*$.
  - Well-formed formula in propositional logic:
    - Basis step: T, F, and s, where s is a propositional variable, are well-formed formulae.
    - Recursive step: If E and F are well-formed formulae, then $(\neg E)$, $(E \wedge F)$, $(E \vee F)$, $(E \rightarrow F)$, and $(E \leftrightarrow F)$ are well-formed formulae.

- More examples:
  - Rooted trees:
    - Basis step: A single vertex r is a rooted tree.
    - Recursive step: Suppose that $T_1, T_2, ..., T_n$ are disjoint rooted trees with roots $r_1, r_2, ..., r_n$, respectively. Then the graph formed by starting with a root r, which is not in any of the rooted trees $T_1, T_2, ..., T_n$, and adding an edge from r to each of the vertices $r_1, r_2, ..., r_n$, is also a rooted tree.
  - Full binary trees:
    - Basis step: There is a full binary tree consisting only of a single vertex r.
    - Recursive step: If T_1 and T_2 are disjoint full binary trees, there is a full binary tree, denoted by $T_1 \cdot T_2$, consisting of a root r together with edges connecting the root to each of the roots of the left subtree T_1 and the right subtree T_2.

- A set S defined recursively:
  - Basis step: $3 \in S$
  - Recursive step: If $x \in S$ and $y \in S$, then $x + y \in S$.
- Let A be the set of all positive integers divisible by 3.
- Show that $S \subseteq A$.
- *Structural Induction*
  - Basis step: Show that the result holds for all elements specified in the basis step of the recursive definition to be in the set.
  - Recursive step: Show that if the statement is true for each of the elements used to construct new elements in the recursive step of the definition, the result holds for these new elements.
- Argue the validity of structural induction.

- Well-formed formula in propositional logic:
  - Basis step: T, F, and s, where s is a propositional variable, are well-formed formulae.
  - Recursive step: If E and F are well-formed formulae, then $(\neg E)$, $(E \wedge F)$, $(E \vee F)$, $(E \rightarrow F)$, and $(E \leftrightarrow F)$ are well-formed formulae.
- Show that every well-formed formula for compound propositions, as defined above, contains an equal number of left and right parentheses.

Definition

An algorithm is called *recursive* if it solves a problem by reducing it to an instance of the same problem with smaller input.

- Example:

Extended-Euclid-GCD(a, b)
  If$(b = 0)$, then return$(a, 1, 0)$
  else
    Compute integers q, r such that $a = qb + r$ and $0 \leq r < b$.
    Let $(d, x, y) =$ Extended-Euclid-GCD(b, r)
    return$(d, y, x - yq)$

- Prove that the above algorithm returns the correct answer.

End