Some Proof Techniques (Review)

Dianne R. Foreback, Ph.D.

Some information on slides originating from *Algorithm Design & Applications* by Michael T. Goodrich and Roberto Tamassia, © 2015 John Wiley & Sons, Inc.Goodrich and Tamassia, ISBN: 978-1118335918.

References

- **Some information on slides originating from**
  - ***Algorithm Design & Applications*** **by Michael T. Goodrich and Roberto Tamassia, © 2015 John Wiley & Sons, Inc.Goodrich and Tamassia,**
    ISBN: 978-1118335918.
  - ***Data Structures and Algorithm Analysis in C++, Fourth Edition*** **by Mark Allen Weiss, 2014 Addison-Wesley, ISBN-13: 978-0-13-286737-7, ISBN-10: 0-13-284737-7**

Topics

- Proof by Induction
- Proof by Counterexample
- Proof by Contradiction

Proof by Induction

Proof by induction can be used to prove recursive algorithm works

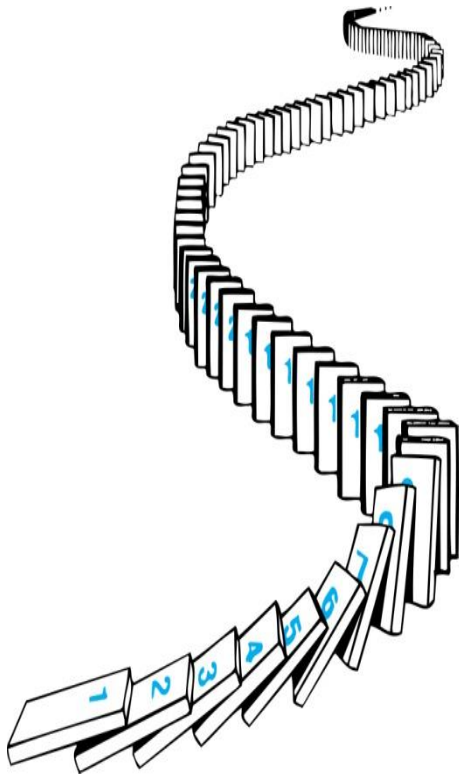
Three parts

1. Prove base case - establish theorem for some small (usually degenerate) value(s) – this step is almost always trivial
2. Assume inductive hypothesis – assume the theorem to be true for all cases up to some limit `k`
3. Inductive step - use inductive hypothesis assumption show theorem to be true for the next value `k+1`

Remembering How Mathematical Induction Works

Consider an infinite sequence of dominoes, labeled 1,2,3, …, where each domino is standing.

Let $P(n)$ be the proposition that the nth domino is knocked over.



Informal proof:

We know that the first domino is knocked down, i.e., $P(1)$ is true .

We also know that if whenever the kth domino is knocked over, it knocks over the $(k + 1)$st domino, i.e, $P(k) \rightarrow P(k + 1)$ is true for all positive integers k.

Hence, all dominos are knocked over.

$P(n)$ is true for all positive integers n.

Image from Data Structures and Algorithm Analysis in C++, Fourth Edition by Mark Allen Weiss, 2014 Addison-Wesley, ISBN-13: 978-0-13-286737-7, ISBN-10: 0-13-284737-7

Arithmetic Series *

$$\sum_{i=1}^{N} \mathrm{i} = \frac{N(N+1)}{2}$$

Greek uppercase letter sigma symbol to represent series.

The variable `i` is called the **index of summation**. It runs through the integers starting with its lower limit `i = 1` and ending with its upper limit **`N`**.

Proving a Summation Formula by Mathematical Induction

Example: Show that:

$$\sum_{i=1}^{n} = \frac{n(n+1)}{2}$$

Solution:

- BASIS STEP: *P*(1) is true since 1(1 + 1)/2 = 1.
- INDUCTIVE STEP: Assume true for *P*(*k*).

The inductive hypothesis is

$$\sum_{i=1}^{k} = \frac{k(k+1)}{2}$$

Under this assumption,

Note: Once we have this conjecture, mathematical induction can be used to prove it correct.

Show true for *P(k+1)*

$$\begin{aligned} 1 + 2 + \ldots + k + (k+1) &= \frac{k(k+1)}{2} + (k+1) \\ &= \frac{k(k+1) + 2(k+1)}{2} \\ &= \frac{(k+1)(k+2)}{2} \end{aligned}$$

◀

Proof by Counterexample

- Prove the following is false

$$\sum_{i=1}^{N} \mathrm{i} \leq 3N + 1$$

- To prove can give a counterexample showing that the inequality does not hold
  - *Let N = 6 then*

$$\sum_{i=1}^{6} \mathrm{i} \; = 1 + 2 + 3 + 4 + 5 + 6 = 21 > 3 * 6 + 1 = 19$$ □

Proof By Contradiction

Step 1: Assume the theorem is false

Step 2: Show this assumption implies that some known property is false

Step 3: Hence the original assumption (1) is erroneous (proving the assumption in step 1 is true)

Example

Theorem: There is an infinite number of primes

Proof:

Step 1: *Assume there are a finite number of primes, so there is some largest prime* P_k

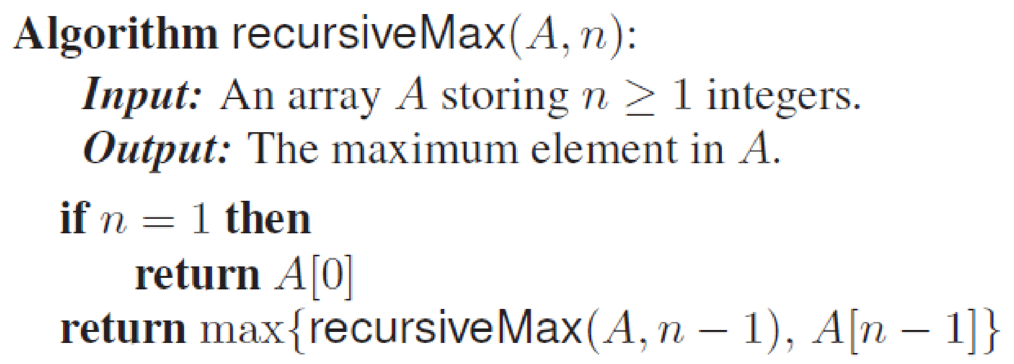
Let $P_1, P_2, \dots, P_k$ *be all the primes in order and consider*

$$N = P_1 * P_2 * \cdots * P_k + 1$$

Clearly, N is larger than P_k *so by our assumption (in Step 1) N cannot be prime.*

However, none of the factors $P_1, P_2, \dots, P_k$ *divides N exactly because there will always be a remainder of 1. This is a contradiction (of our assumption in Step 1), thus there is not a finite number of primes.*

Algo recursiveMax proof by induction



```
Algorithm recursiveMax($A, n$):
    Input: An array $A$ storing $n \geq 1$ integers.
    Output: The maximum element in $A$.
  if $n = 1$ then
      return $A[0]$
  return $\max\{\text{recursiveMax}(A, n-1),\ A[n-1]\}$
```

Algorithm 1.4: Algorithm recursiveMax

Algorithm originating from *Algorithm Design & Applications* by Michael T. Goodrich and Roberto Tamassia, © 2015 John Wiley & Sons, Inc.Goodrich and Tamassia, ISBN: 978-1118335918.

Claim: Algorithm recursiveMax(A, n) returns a maximum element from an array with n elements stored in array positions 0 to n-1.

Proof:
Let A be an array with n elements.

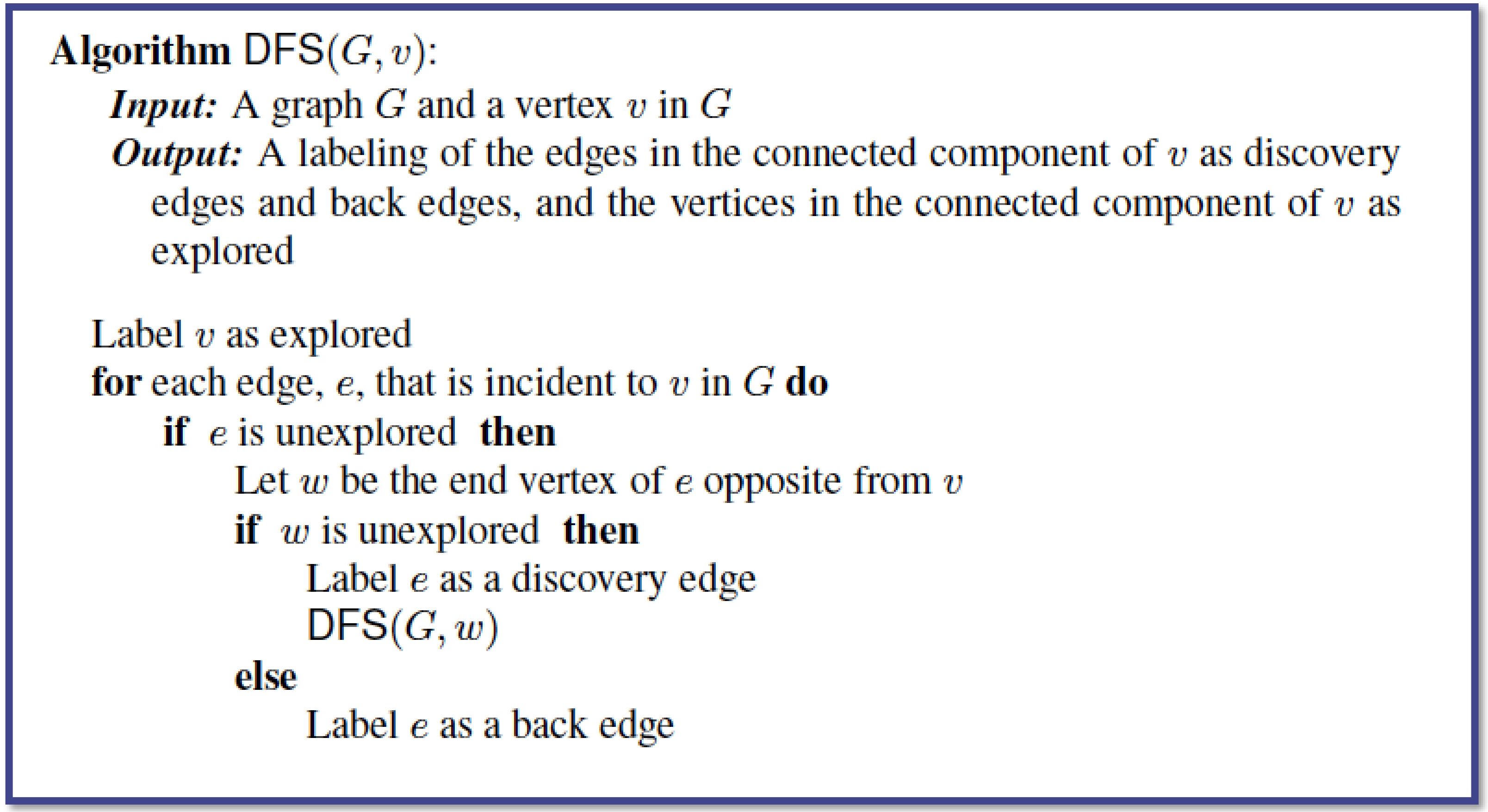
Base Case: When n = 1, only one element is in the array, thus it is the maximum element and it is returned via "return A[0]".

Inductive Hypothesis: Assume that when there are n=k >1 elements, arrayMax(A, k) returns the maximum element in the array of size k.

Inductive Step: (Now we must explain that when n = k+1 >1 elements, that the largest of these k+1 elements is returned.)

Consider that there are k+1 elements in the array.
When n=k+1 > 1, the recursive call recursiveMax(A, k) is made. From the inductive hypothesis, we know that from the subarray of elements 0 to k-1, the maximum element is returned. Now, when n=k+1, the maximum element is either element A[k] or the maximum from this subarray that is returned earlier. Examining the "return" statement, the element that is returned is the maximum of these two. □

Algo DFS(G, s)

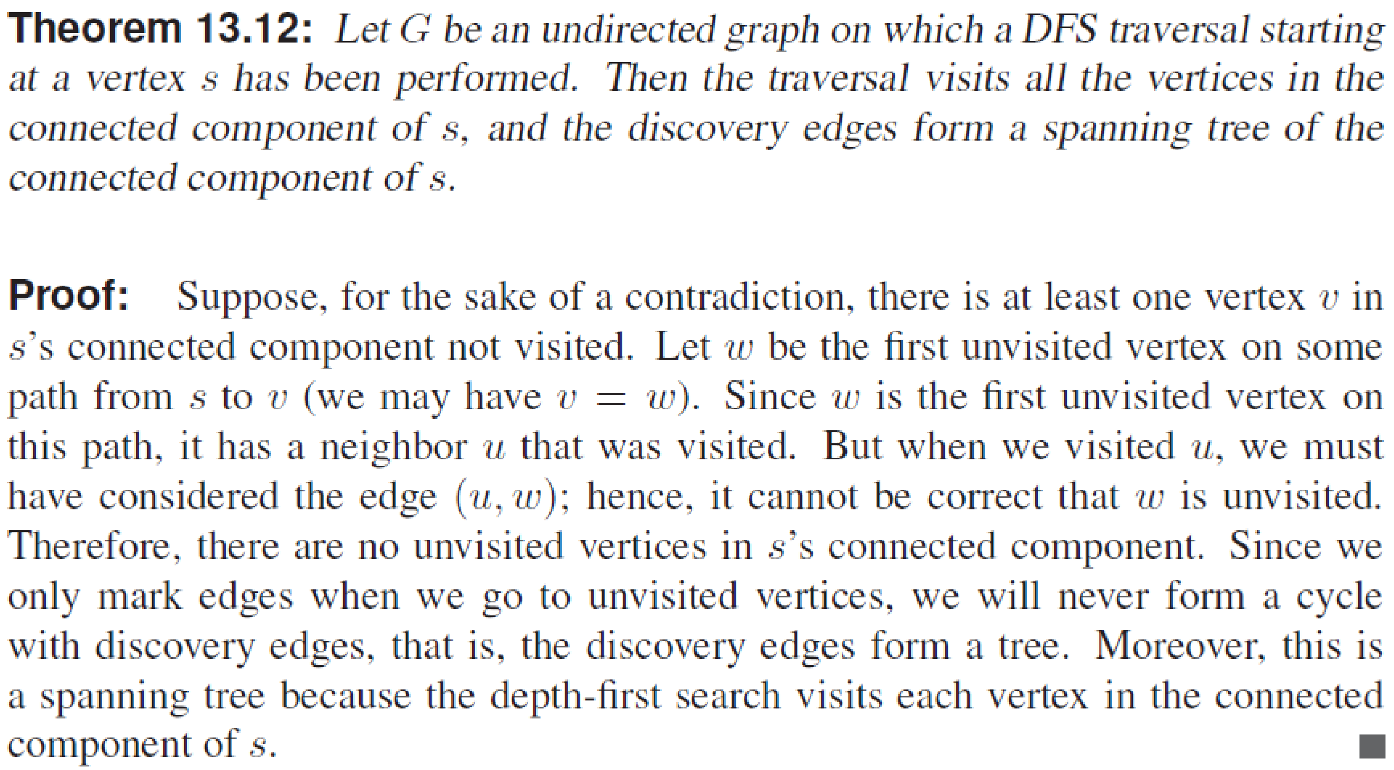


```
Algorithm DFS($G, v$):
    Input: A graph $G$ and a vertex $v$ in $G$
    Output: A labeling of the edges in the connected component of $v$ as discovery
        edges and back edges, and the vertices in the connected component of $v$ as
        explored

  Label $v$ as explored
  for each edge, $e$, that is incident to $v$ in $G$ do
      if $e$ is unexplored then
          Let $w$ be the end vertex of $e$ opposite from $v$
          if $w$ is unexplored then
              Label $e$ as a discovery edge
              DFS($G, w$)
          else
              Label $e$ as a back edge
```

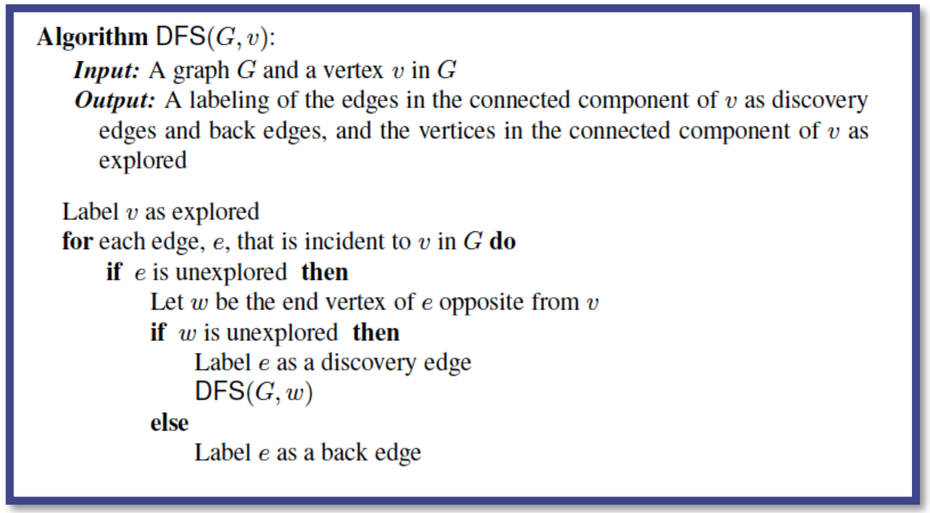
Algorithm originating from *Algorithm Design & Applications* by Michael T. Goodrich and Roberto Tamassia, © 2015 John Wiley & Sons, Inc.Goodrich and Tamassia, ISBN: 978-1118335918.

Algo DFS(G, s) proof from the book



Theorem 13.12: *Let G be an undirected graph on which a DFS traversal starting at a vertex s has been performed. Then the traversal visits all the vertices in the connected component of s, and the discovery edges form a spanning tree of the connected component of s.*

Proof: Suppose, for the sake of a contradiction, there is at least one vertex v in s's connected component not visited. Let w be the first unvisited vertex on some path from s to v (we may have $v = w$). Since w is the first unvisited vertex on this path, it has a neighbor u that was visited. But when we visited u, we must have considered the edge (u, w); hence, it cannot be correct that w is unvisited. Therefore, there are no unvisited vertices in s's connected component. Since we only mark edges when we go to unvisited vertices, we will never form a cycle with discovery edges, that is, the discovery edges form a tree. Moreover, this is a spanning tree because the depth-first search visits each vertex in the connected component of s. ■



```
Algorithm DFS($G, v$):
  Input: A graph $G$ and a vertex $v$ in $G$
  Output: A labeling of the edges in the connected component of $v$ as discovery
    edges and back edges, and the vertices in the connected component of $v$ as
    explored

  Label $v$ as explored
  for each edge, $e$, that is incident to $v$ in $G$ do
      if $e$ is unexplored then
          Let $w$ be the end vertex of $e$ opposite from $v$
          if $w$ is unexplored then
              Label $e$ as a discovery edge
              DFS($G, w$)
          else
              Label $e$ as a back edge
```

From *Algorithm Design & Applications* by Michael T. Goodrich and Roberto Tamassia, © 2015 John Wiley & Sons, Inc.Goodrich and Tamassia, ISBN: 978-1118335918.