### CS 5114: Theory of Algorithms

Clifford A. Shaffer

Department of Computer Science Virginia Tech Blacksburg, Virginia

Spring 2014

Copyright © 2014 by Clifford A. Shaffer

CS 5114: Theory of Algorithms

Spring 2014 1 /

### **CS5114: Theory of Algorithms**

- Emphasis: Creation of Algorithms
- Less important:
  - Analysis of algorithms
  - Problem statement
  - Programming
- Central Paradigm: Mathematical Induction
  - Find a way to solve a problem by solving one or more smaller problems

CS 5114: Theory of Algorithms

Spring 2014 2 / 1

**Review of Mathematical Induction** 

- The paradigm of <u>Mathematical Induction</u> can be used to solve an enormous range of problems.
- Purpose: To prove a parameterized theorem of the form:

Theorem:  $\forall n \geq c, P(n)$ .

- ▶ Use only positive integers  $\geq c$  for n.
- Sample P(n):

 $n+1 \le n^2$ 

CS 5114: Theory of Algorithms

Spring 2014 3

# **Principle of Mathematical Induction**

- IF the following two statements are true:
  - $\bullet$  **P**(c) is true.
  - 2 For n > c, P(n-1) is true  $\rightarrow P(n)$  is true.

... **THEN** we may conclude:  $\forall n \geq c$ , **P**(n).

- The assumption "P(n − 1) is true" is the induction hypothesis.
- Typical induction proof form:
  - Base case
  - State induction Hypothesis
  - Prove the implication (induction step)
- What does this remind you of?

CS 5114: Theory of Algorithms

Spring 2014 4 / 1

CS 5114

CS 5114: Theory of Algorithms

Clintark, Studer

Clintark

Title page

Students should be familiar with inductive proofs, recursion, data structures, and programming at the CS3114 level.

CS 5114

CS5114: Theory of Algorithms

Creation of algorithms comes through exploration, discovery, techniques, intuition: largely by **lots** of examples and **lots** of practice (HW exercises).

We will use Analysis of Algorithms as a tool.

Problem statement (in the software eng. sense) is not important because our problems are easily described, if not easily solved. Smaller problems may or may not be the same as the original problem

Divide and conquer is a way of solving a problem by solving one more more smaller problems.

Claim on induction: The processes of constructing proofs and constructing algorithms are similar.

CS 5114

Review of Mathematical Induction

Induction

Review of Mathematical Induction

First we will refresh/expand our our familiarity with induction. Then we will try to apply an inductive approach to algorithm design.

P(n) is a statement containing n as a variable.

This sample P(n) is true for  $n \ge 2$ , but false for n = 1.

CS 5114

Principle of Mathematical Induction

\*\*Facilitating by distinction to the Company of th

Important: The goal is to prove the **implication**, not the theorem! That is, prove that  $\mathbf{P}(n-1) \to \mathbf{P}(n)$ . **NOT** to prove P(n). This is much easier, because we can assume that  $\mathbf{P}(n-1)$  is true.

Consider the truth table for implication to see this. Since  $A \to B$  is (vacuously) true when A is false, we can just assume that A is true since the implication is true anyway if A is false. That is, we only need to worry that the implication could be false if A is true.

The power of induction is that the induction hypothesis "comes for free." We often try to make the most of the extra information provided by the induction hypothesis.

This is like recursion! There you have a base case and a recursive call that must make progress toward the base case.

### Induction Example 1

Theorem: Let

$$S(n) = \sum_{i=1}^{n} i = 1 + 2 + \cdots + n.$$

Then,  $\forall n \ge 1$ ,  $S(n) = \frac{n(n+1)}{2}$ .

CS 5114: Theory of Algorithms

### Induction Example 2

**Theorem**:  $\forall n \geq 1, \forall \text{ real } x \text{ such that } 1 + x > 0,$  $(1+x)^n \ge 1 + nx.$ 

CS 5114: Theory of Algorithms

Spring 2014 6 / 1

# **Induction Example 3**

Theorem: 2¢ and 5¢ stamps can be used to form any denomination (for denominations  $\geq$  4).

CS 5114: Theory of Algorithms

# Colorings

4-color problem: For any set of polygons, 4 colors are sufficient to guarentee that no two adjacent polygons share the same color.

Restrict the problem to regions formed by placing (infinite) lines in the plane. How many colors do we need? Candidates:

- 4: Certainly
- 3: ?
- 2: ?
- 1: No!

Let's try it for 2... CS 5114: Theory of Algorithms

Spring 2014 8 / 1

2014-01-21 CS 2114 Induction Example 1

**Base Case**: P(n) is true since S(1) = 1 = 1(1+1)/2. Induction Hypothesis:  $S(i) = \frac{i(i+1)}{2}$  for i < n. Induction Step:

$$S(n) = S(n-1) + n = (n-1)n/2 + n$$
  
=  $\frac{n(n+1)}{2}$ 

Therefore,  $\mathbf{P}(n-1) \rightarrow \mathbf{P}(n)$ .

By the principle of Mathematical Induction,

$$\forall n \geq 1, S(n) = \frac{n(n+1)}{2}.$$

MI is often an ideal tool for verification of a hypothesis. Unfortunately it does not help us to construct a hypothesis.



What do we do induction on? Can't be a real number, so must

$$P(n): (1+x)^n \ge 1 + nx.$$

**Base Case**:  $(1+x)^1 = 1 + x \ge 1 + 1x$ **Induction Hypothesis**: Assume  $(1+x)^{n-1} \ge 1 + (n-1)x$ Induction Step:

$$(1+x)^n = (1+x)(1+x)^{n-1}$$

$$\geq (1+x)(1+(n-1)x)$$

$$= 1+nx-x+x+nx^2-x^2$$

$$= 1+nx+(n-1)x^2$$

$$> 1+nx.$$



**Base case**: 4 = 2 + 2.

**Induction Hypothesis**: Assume P(k) for  $4 \le k < n$ .

#### Induction Step:

Case 1: n-1 is made up of all 2¢ stamps. Then, replace 2 of these with a 5¢ stamp.

Case 2: n-1 includes a 5¢ stamp. Then, replace this with 3 2¢ stamps.



Induction is useful for much more than checking equations!

If we accept the statement about the general 4-color problem, then of course 4 colors is enough for our restricted version.

If 2 is enough, then of course we can do it with 3 or more.

### **Two-coloring Problem**

Given: Regions formed by a collection of (infinite) lines in the plane.

Rule: Two regions that share an edge cannot be the same color.

**Theorem**: It is possible to two-color the regions formed by n lines

CS 5114: Theory of Algorithms

Spring 2014 9

### **Strong Induction**

IF the following two statements are true:

**P**(*c*)

**2**  $P(i), i = 1, 2, \cdots, n-1 \rightarrow P(n),$ 

... **THEN** we may conclude:  $\forall n \geq c$ , **P**(n).

Advantage: We can use statements other than P(n-1) in proving P(n).

CS 5114: Theory of Algorithms

Spring 2014 10 / 1

# **Graph Problem**

An **Independent Set** of vertices is one for which no two vertices are adjacent.

**Theorem**: Let G = (V, E) be a <u>directed</u> graph. Then, G contains some independent set S(G) such that every vertex can be reached from a vertex in S(G) by a path of length at most 2.

Example: a graph with 3 vertices in a cycle. Pick any one vertex as S(G).

CS 5114: Theory of Algorithms

Spring 2014 11 / 1

# **Graph Problem (cont)**

**Theorem**: Let G = (V, E) be a <u>directed</u> graph. Then, G contains some independent set S(G) such that every vertex can be reached from a vertex in S(G) by a path of length at most 2.

**Base Case**: Easy if  $n \le 3$  because there can be no path of length > 2.

**Induction Hypothesis**: The theorem is true if |V| < n. **Induction Step** (n > 3):

Pick any  $v \in V$ .

Define:  $N(v) = \{v\} \cup \{w \in V | (v, w) \in E\}.$ 

H = G - N(v).

Since the number of vertices in H is less than n, there is an independent set S(H) that satisfies the theorem for H.

 CS 5114

Two-coloring Problem

One Regions breakly a calledian of princip loss as the Research to the State Coloring Problem

Two-coloring Problem

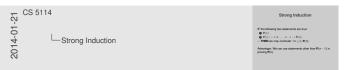
Picking what to do induction on can be a problem. Lines? Regions? How can we "add a region?" We can't, so try induction on lines.

**Base Case**: n=1. Any line divides the plane into two regions. **Induction Hypothesis**: It is possible to two-color the regions formed by n-1 lines.

**Induction Step**: Introduce the *n*'th line.
This line cuts some colored regions in two.

Reverse the region colors on one side of the *n*'th line. A valid two-coloring results.

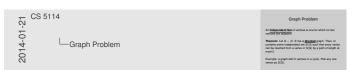
- · Any boundary surviving the addition still has opposite colors.
- Any new boundary also has opposite colors after the switch.



The previous examples were all very straightforward – simply add in the *n*'th item and justify that the IH is maintained. Now we will see examples where we must do more sophisticated (creative!) maneuvers such as

- go backwards from n.
- prove a stronger IH.

to make the most of the IH.



It should be obvious that the theorem is true for an undirected graph.

Naive approach: Assume the theorem is true for any graph of n-1 vertices. Now add the nth vertex and its edges. But this won't work for the graph  $1 \leftarrow 2$ . Initially, vertex 1 is the independent set. We can't add 2 to the graph. Nor can we reach it from 1.

Going forward is good for proving existance. Going backward (from an arbitrary instance into the IH) is usually necessary to prove that a property holds in all instances. This is because going forward requires proving that you reach all of the possible instances.

Graph Problem (cont)

Graph Problem (cont)

Baseman (et al. (1/87)) or gasping eyes, him, 0

or control amon and control amon

N(v) is all vertices reachable (directly) from v. That is, the Neighbors of v.

*H* is the graph induced by V - N(v).

OK, so why remove both v and N(v) from the graph? If we only remove v, we have the same problem as before. If G is  $1 \to 2 \to 3$ , and we remove 1, then the independent set for H must be vertex 2. We can't just add back 1. But if we remove both 1 and 2, then we'll be able to do something...

## **Graph Proof (cont)**

There are two cases:

**③**  $S(H) \cup \{v\}$  is not independent. Let  $w \in S(H)$  such that  $(w, v) \in E$ . Every vertex in N(v) can be reached by w with path of length ≤ 2. So, set S(G) = S(H).

By Strong Induction, the theorem holds for all G.

CS 5114: Theory of Algorithms

Spring 2014 13 / 1

CS 5114

Graph Proof (cont)

There are the classes:

O 2011 of the are the classes:

Control of the classes:

Control

" $S(H) \cup \{v\}$  is not independent" means that there is an edge from something in S(H) to v.

IMPORTANT: There cannot be an edge from v to S(H) because whatever we can reach from v is in N(v) and would have been removed in H.

We need strong induction for this proof because we don't know how many vertices are in  $N(\nu)$ .

We must remove N(v) instead of just v because of this case: We remove just v to yield H. S(H) turns out to have something that can be reached from v. So, when we add v back to reform G, v cannot become part of S(G) (because that would violate the definition of independent set). But if v is 3 steps away from anything in S(H), we must add it to satisfy the theorem. So are stuck.