Successful approaches for teaching Computer Science

Ajay Bansal

School of Computing, Informatics, and Decision Systems Engineering (CIDSE)
Arizona State University

Seminar @ East Carolina University, Nov 22, 2019

IRA A. FULTON SCHOOLS OF engineering
ARIZONA STATE UNIVERSITY

Research Interests

Software Engineering

- Software Analysis & Design
- Software Verification & Validation (Co-logic Programming)
- Software Development (Programming Paradigms)

Knowledge Representation & AI

- Model Checking, Planning algorithms with time constraints
- Data Mining, Natural Language processing, Machine Learning

Programming Languages (Declarative Programming)

Automated Reasoning

- Non-monotonic Reasoning (Goal-directed Answer set Programming)
- Rule-based reasoning engines for Game development

Semantic Computing

- Semantic description language for Web Services (USDL)
- Web service discovery & composition (using Constraint Logic Programming)

Been There… Done That…

- 12+ years of CS teaching experience
- Taught 20+ unique CS courses in 80+ course offerings
- Taught courses in the areas of:

  Algorithms, Data Structures, Programming Paradigms, Discrete Math, Software Engineering, Database Management, Artificial Intelligence, Theory of Computation, Programming Languages, Internet Computing, Web UI development, Introduction to Programming, Object-oriented programming, Compiler design…

- Teaching Awards:
  - **2019 - Fulton Teaching Excellence Award (**given to one faculty member across all engineering programs at ASU (20,000+ students and 500+ faculty))
  - **2018 - Best Teacher Award –Top 5%** (Fulton Schools of Engineering, ASU)
  - **2015 - Best Teacher Award –Top 5%** (Fulton Schools of Engineering, ASU)
  - **2013 - Best Teacher Award (**Department of Engineering, CTI @ASU)

Once upon a time …

Course Evaluation:

- …
- …
- The course was too slow…
- …
- It was too fast…
- …
- …
- The pace was just right…
- …

It was the ‘Introduction to Programming’ course

Challenges in CS Teaching

- Students come from very diverse backgrounds in CS courses
  - Undergrad – AP credits from high school, know multiple languages vs never programmed before…
  - Graduate – from various undergrad disciplines, non-majors…
- A wide spectrum of the levels of preparatory training that students come in with
- Cannot assume that the students have the pre-requisite knowledge to be successful in the course
- Big class sizes as more students get attracted to CS programs

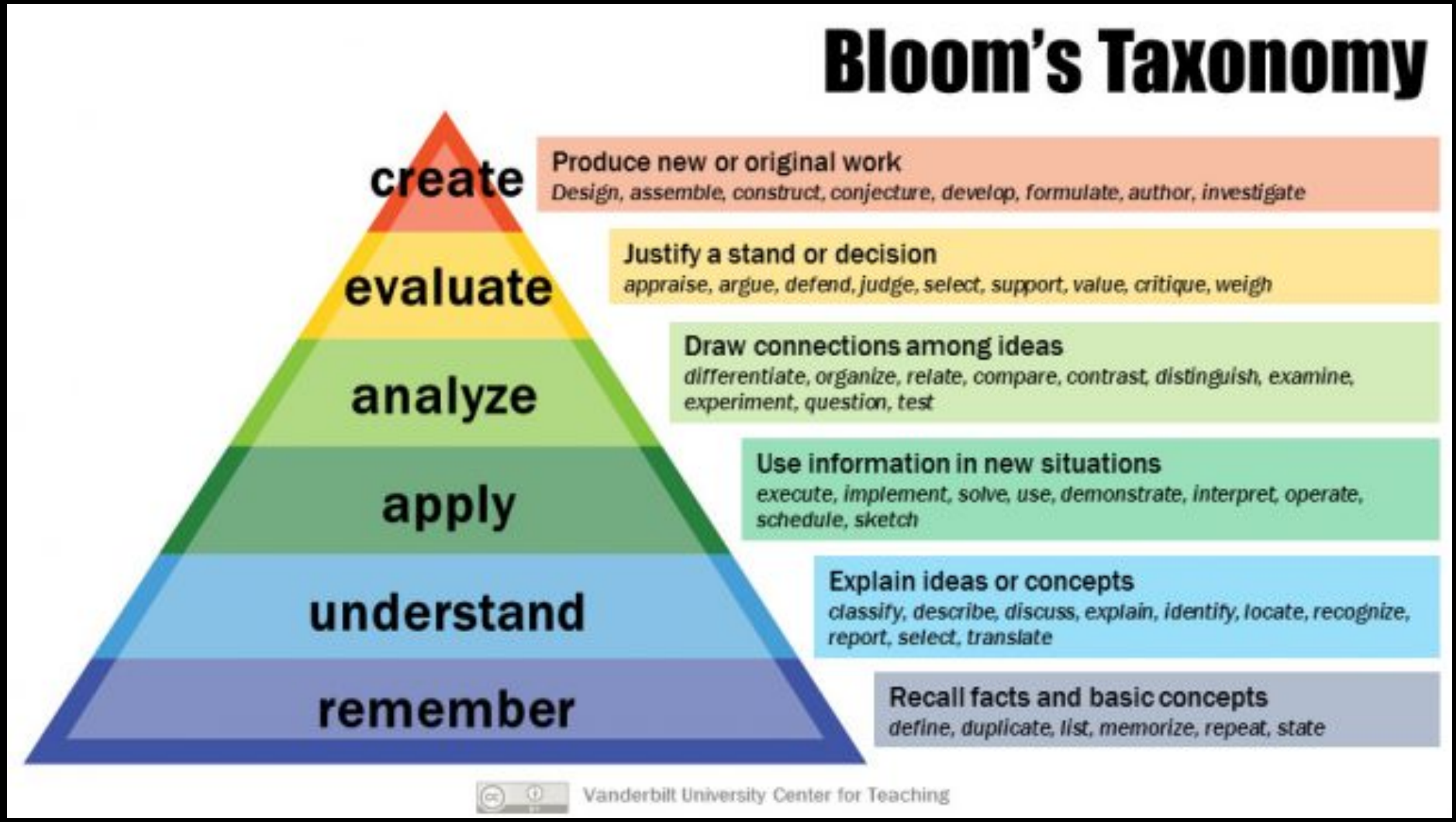
Challenges in CS Teaching

- Problem Solving
  - A fundamental skill that spans across all subfields of CS
  - Very important, particularly in the software industry, as a software is an encoding of our solution to a problem
  - No widely accepted formal way to teach Problem Solving
- Algorithmic Thinking
  - An important skill for Computational Problem Solving
  - Usually confined to Algorithms design course

Requirements for my courses

- Teach assuming basic preparatory training or pre-requisite knowledge
- Motivate and Engage students at all levels
- Incorporate Problem Solving with every topic taught in the course
- Inspire Algorithmic Thinking to solve these problems
- Use scalable approaches

Outcomes Based Education



Let's re-invent the wheel…

Discovery Learning based Lectures:

- Concepts to be taught are presented as problems or situations
- (Existing) Solutions to these problems are (re)invented in class
- Topics are covered as they may have been invented in the first place

Examples: Loops in languages, Sum of Cubes, Divide & Conquer …

"Art of teaching is the art of assisting discovery"

– Mark Van Doren

Let's break it up…

Algorithmic Thinking:

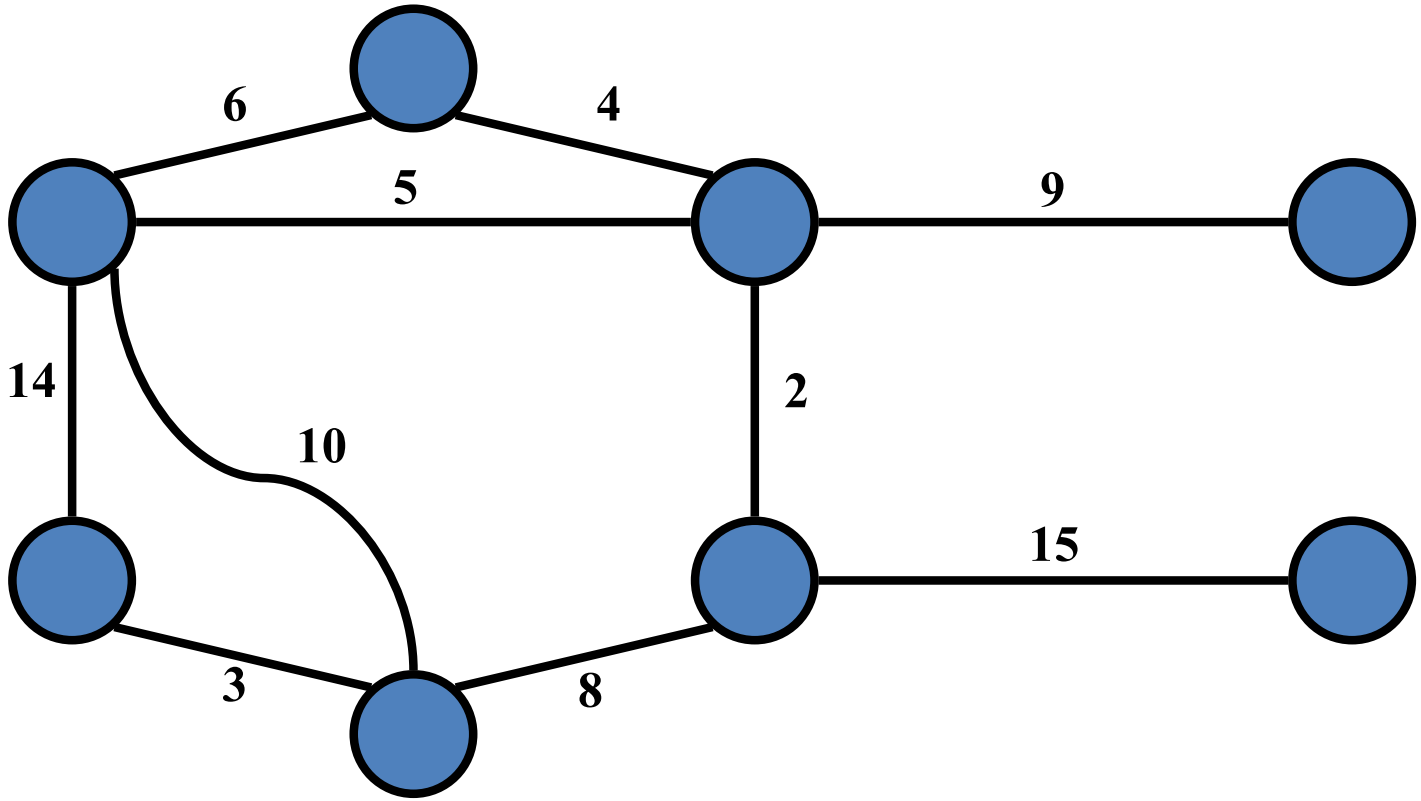
Program = Algorithm + Data Structures

- Niklaus Wirth

Example: Minimum spanning tree

Minimum Spanning Tree

Problem: given a connected, undirected, weighted graph, find a *spanning tree* using edges that minimize the total weight



Algorithm…

```
MST()
{
    T = ∅;
    for each v ∈ V
        MakeSet(v);
    sort E by increasing edge weight w
    for each (u,v) ∈ E (in sorted order)
        if FindSet(u) ≠ FindSet(v)
            T = T U {{u,v}};
            Union(FindSet(u), FindSet(v));
}
```

Let's break it up…

Algorithmic Thinking:

Algorithm = Logic + Control

- Robert Kowalski

Examples: Recursion, Change making problem…

Tell Them Why…

Use inspired:

- **Real world examples**
- **Example: Sanskrit Poetry**

Reasoning inspired:

- **Comparative Examples**
- **Example: Quick Sort Vs Bubble/Selection Sort**
- **Example: Brute Force Vs Dynamic Programming**

Awe-some Lectures

Some ‘awe’ in every lecture:

examples, programs, story, anecdotes, …

Examples: Representation in Algorithms, Factorial, Cryptarithmetic puzzles, …

“Teaching should be such that what is offered is perceived as a valuable gift” - Einstein

SEND + MORE = MONEY

- Each letter represents a unique digit from 0 to 9.
- Two letters cannot represent the same digit.
- What digit each letter represents to satisfy the simple equation below?

Solution:

```
  S E N D
+ M O R E
---------
M O N E Y
```

```
  9 5 6 7
+ 1 0 8 5
---------
1 0 6 5 2
```

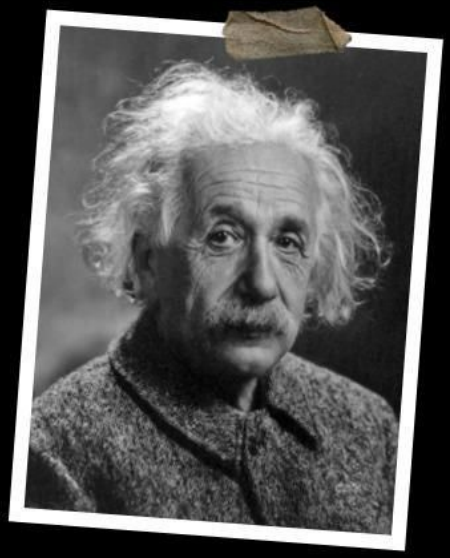
SEND + MORE = MONEY

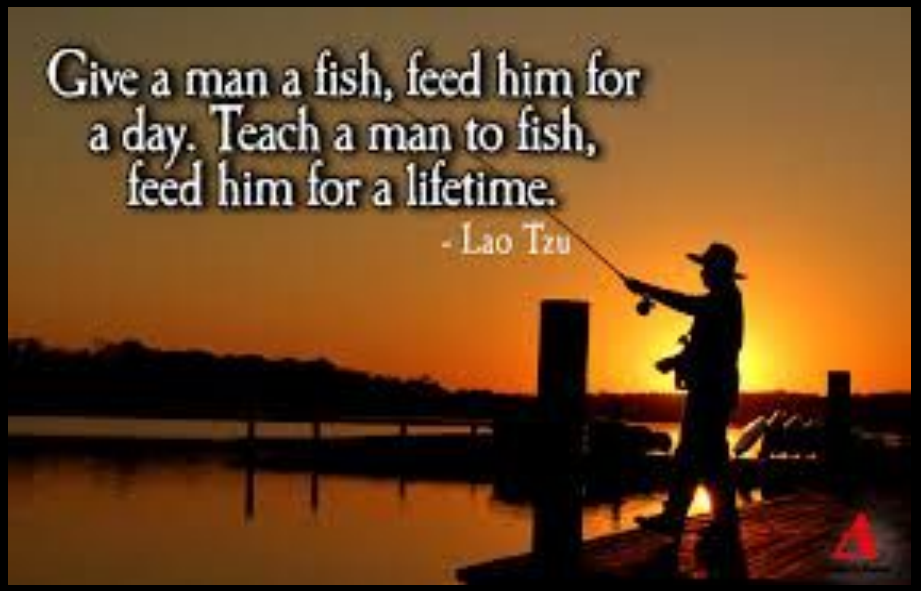
Declarative Programming:

```
solve(Digits) :-
    Digits = [S,E,N,D,M,O,R,Y],
    Digits :: [0..9],
    alldifferent(Digits),
                1000*S + 100*E  + 10*N + D
              + 1000*M + 100*O + 10*R + E
      #= 10000*M + 1000*O + 100*N  + 10*E + Y,
    labeling(Digits).
```

```
? – solve(X)
    X = [9,5,6,7,1,0,8,2]
```

"Education is not the learning of facts, but the training of the mind to think."
-Albert Einstein





Thank You!!

ajay.bansal@asu.edu