ACM India at a Glance

- **ACM**: world's largest educational and scientific computing society
  - **Mission**: advancing computing as science and profession
  - **Members**: ~100,000 worldwide, ~11000 in India
  - Comprising students, faculty, professionals
- **ACM India Chapters**: ~200 student chapters, ~20 professional chapters
- **ACM-W India**: empowering women in computing
- **Research Initiatives**
  - Student research: ARCS Symposium, best doctoral dissertation, partial travel grant
  - Research conferences: CODS-COMAD, ISEC, AIMS
- **ACM India Annual Event**
  - Discuss recent trends in technology and celebrate India's achievements in computing
- **Education Initiatives**
  - Summer and winter schools: ~2 week full-time course on technology area
  - Compute: Symposium focused on improving quality of computing education in India
  - CSpathshala: inculcate computational thinking in schools
- **Learning and Professional Development**
  - Eminent Speaker Program
  - Industry Webinars, Education Webinars
  - Blogs: theoreticians and practitioners sharing ideas, opinions
  - ACM global resources: Digital Library, ACM Learning Center
- **New prestigious awards instituted**
  - Acknowledge and celebrate outstanding contributions
- **ACM Membership in India**
  - Student? student member form
  - Professional? professional member form

Abhiram Ranade

- Immediate Past President, ACM India Council
- Professor of Computer Science and Engineering at IIT Bombay since 1995
- Assistant Professor of Electrical Engineering and Computer Science at the University of California, Berkeley during 1988-95
- Doctorate in Computer Science from Yale University
- B.Tech degree in Electrical Engineering from IIT Bombay
- Research interests - Algorithms and Combinatorial optimization.
- Has won Excellence in Teaching Awards of IIT Bombay in 2006-7 and 2010-11.
- Has published a book titled “An introduction to programming through C++” via McGraw Hill Education India. He has also offered this course as a MOOC through NPTEL SWAYAM.

Competency based learning: a personal perspective

Abhiram Ranade
IIT Bombay

ACM India Education Webinar
May 8, 2021

Learning as per ACM-IEEE Computing Curricula 2020

Competency = Task + Knowledge + Skill + disposition

- **Task:** The purpose served by the learning.
  The professionally/commercially desired outcome
- **Knowledge:** ideas, information that needs to be learned to produce the outcome
- **Skill:** Ability to apply the learned ideas/information.
- **Disposition:** Attitude/emotional aspects that helps in accomplishing the task
  - Proactiveness, Meticulousness, Passion, Perspective, Ethical behaviour

Should we care about this new definition of learning?

Competencies provide a better framework to understand where we are deficient.

- **Knowledge:** We (most Indian universities) are good. Good
- **Skill/Application:** Known deficiency. "Rote learning". ?
- **Task:** Many students wonder why they are learning something ?
  Feeling of being dragged.
  Not recognized as a (serious) problem in Outcome based education.
- **Disposition:** Our students do not have confidence, enthusiasm ?
  Worried about jobs, are not job ready..
  Most important innovation over Outcome based education.

Task and disposition are related to psychology/emotions.

Should teachers worry about students' emotions?

Many studies indicate happier students will work harder, learn better.

"I dont know why we are learning this" = "I dont see the bigger picture"
If the bigger picture is given, the student will understand better.
The bigger picture will motivate the student to work harder.

Can we really teach qualities listed under disposition:
Passion, Confidence, Intuition, Perspective, Ethical Behaviour

Thesis: Disposition = Love for the subject

Not official ACM-IEEE position

Disposition $\supseteq$ Passion, Confidence, Intuition, Perspective,...

These are all symptoms of love!

Confidence, intuition are acquired only after lot of hard work, calm thinking
To spend so much time and effort you need to like the subject.
Passion IS love!
If you are entering a profession, you should feel deep love for it.

You are really going to be married to your profession!

The same love will also help in the learning process!

Conjecture: Let us try to make the student fall in love with our subject.
Many required dispositions will follow.

Ethical disposition: Best taught by example.

The roadmap to learning

1. Student feels some attraction for the subject (or not).
2. Teachers introduce subject by highlighting its power and possibilities.

   Superficial attraction is created.
3. Teacher imparts knowledge.
4. Teacher gives practice problems.
5. Practice/repeated application develops skill, confidence.
6. Deeper love emerges if knowledge and practice problems are good.
7. Process repeats from step 2.

Key points:

- Synergy between knowledge, skill, disposition.
- Good teachers try hard to make their students like the subject.

  But now it is compulsory! Everyone must do it.
- Course designers must plan for it!

Outline

- Competency based course design
- Teaching strategies
- Example: Introductory Programming
- Example: Data structures
- Example: Design and analysis of algorithms
- The first lecture in DAA

Competency based course design

Course content = sequence of topics, topic = competency.

- Include application (task), theory (knowledge) and practice (skill) in each topic
  - Cannot say, "Theory in this course, applications in next"
  - The task in each topic should resemble job situation.
  - Encourages job readiness
  - Provides clear motivation to learn
  - Theory is understood better if application is known
- Course must contain competencies that have challenge
  - Student must feel proud/satisfied at the end
- Course must contain competencies that use interesting ideas
  - Interesting, elegant ideas make course likeable
- Course must reflect current practice, modern thinking.

Teaching strategies to improve disposition

First lecture: Use the first lecture to convey the challenges, some clever ideas.
Try to wow the class! Put your best foot forward!
Prepare and motivate student for what is to come.

Give demos: of what the student will be able to acheive at the end of the course.

Scaffolding: Predeveloped libraries which students can use
Example: graphics library. Enables interesting programs and visualization

Repeatedly explain the Task: Very important to keep students focussed.

Assessment: Use realistic problems. Students feel more motivated and connected.
If you give a realistic problem, the student is also learning as she is being assessed.
No learning in synthetic problems.

Teaching to develop skills

- Skill = applying knowlege in an unseen situation

  Indian education system is weak on this.
- Let us take baby steps: apply knowledge to situations not totally new but similar to those seen earlier?
- Gradually increase the amount of unknown/unfamiliar element.
- Skill of solving unseen problems will give confidence for job situations.
- There should be solution path for every problem:
  - "You should have thought of x because we did that in solving y"
  - Dont expect students to innovate, be happy if they do.

Presenting useful and elegant ideas is necessary.
But students will feel frustrated if there is no good process for mastering their application.

Introductory programming

Task: Writing a program — Not "Learn programming language X."

The programming language is a means, not the end.

Key skill: Given an English language specification, produce a program

Assuming they know how to solve the problem by hand

Task $\neq$ designing new algorithms

Don't be too ambitious, nor too unambitious

Knowledge:

Programming language syntax, semantics

Standard programming patterns: Repeat {Read command, modify state, print answer}

Skill: Introspect over how you solve the problem and identify patterns in it.

Express the patterns using loops, conditionals, recursion, functions.

Need lot of drill problems. Problems should be realistic.

Bank accounts, water tanks that overflow, games, kinematics, picture drawing

Scaffolding: Graphics. Greatly useful for exercises, illustrating ideas.

Data Structures

Standard Task: Writing programs requiring data structures

Traditional skill taught

- Typically everything is expected to be coded from scratch
- Some combinations of data structures or design of new data structures.
- A lot of time is spent on linked lists which are basically obsolete

Today's requirement: use built-in data structures, vector, map

- Select correct data structure given "word problem".
- Combination may be needed e.g. Distance["Mumbai"]["Pune"] is map of maps
- Configuring a data structure: "Order by x coordinate"

Algorithms

Task: Algorithm design from scratch:
Generally considered difficult
Often not even attempted.
Suggestion: Ask student to design algorithms for minor variations of problems done in class.

Task: Use of packages such as LP/ILP solvers
Many IEOR professionals design algorithms every day using these.
Requires different kind of skill/training
Basic ideas about using LP/ILP solvers will help students in jobs.
Could be easier than designing algorithms from scratch

First lecture in algorithms

Should give a flavour of the course

- Clever ideas can produce fast algorithms
- Some clever ideas are general and can be used again and again
- Everyone can acquire the skill of adapting an idea to slightly different situation.

Discuss a problem which is important, or related to an important problem.

Optimization and Counting

Many algorithm design problems have the following form.

- Some **constraints** are given on the structure of an object.
- We want to know whether and how many such objects can be constructed.

  Object satisfying constraints: "feasible solution"

  "How many feasible solutions are there?"
- If many constructions are possible, which is the best as per a certain "optimization" function?

  "Optimal solution"

This lecture: Counting "Poetic Meters"

- Simple problem studied in ancient India.
- The solution ideas will be useful for "more important" problems.

Poetic meter: singing style of a poem

Example: Poetic meter Shardulvikridit

2	2	2	1	1	2	1	2	1	1	1	2
Ya	kun	den	du	tu	sha	r	Haa	r	dha	wa	la

2	2	1	2	2	1	2
ya	shubh	ra	vas	tra	vru	ta

Poetic meter: Sequence of 1s and 2s. Shardulvikridit: 2221121211122212212

- Length of the sequence = number of syllables in each line of poem.
- ith number in sequence = Singing duration of ith syllable
- Sum of the sequence = (Total) singing duration of line

Shardulvikridit: 19 syllables(11 long, 8 short). Total duration 30.

Question: How many poetic meters have duration D?

Virahanka, Indian prosodist, 6th-8th century AD

Our goal: Write a program to find the answer.

Remarks

- Counting poetic meters problem $\in$ "Packing problems"

  We are packing syllables into duration D.
- Techniques we learn are useful for more complex packing problems
  - In how many ways you can give change for D paise using standard coins.
  - What is the minimum number of coins needed to give change for D paise.
  - How to fill a container with items of maximum value..
- Related to a problem you have known by an Italian name.
- Math in Ancient India! Math in Poetry!

Comments

Counting poetic meters is not computationally important, but is closely related to important problems.

The obvious algorithm runs in time exponential in the duration.
The less obvious algorithm runs in time linear in the duration.

Show the programs in lecture 1, run them and observe time.

A very unobvious algorithm runs in time polylog in the duration.

Poetic meters problem originated in India, and was solved in India.
Can be expressed as "How many towers of height H can you build given red bricks of height 1 and blue bricks of height 2?"

Summary

Competency based learning says emotional response of students is important.

Might be exactly what we need to hear.

Learning needs synergy between task, knowledge, skill.
Courses must be designed so that teachers can produce this synergy.

Skill development requires lots and lots of exercises, of varied difficulty and similarity.
Take baby steps at least in getting students to solve unseen problems.

Explain "why are we studying this" many, many, many times.

First lecture of the course is very important.

I am happy to discuss further. Send me email.