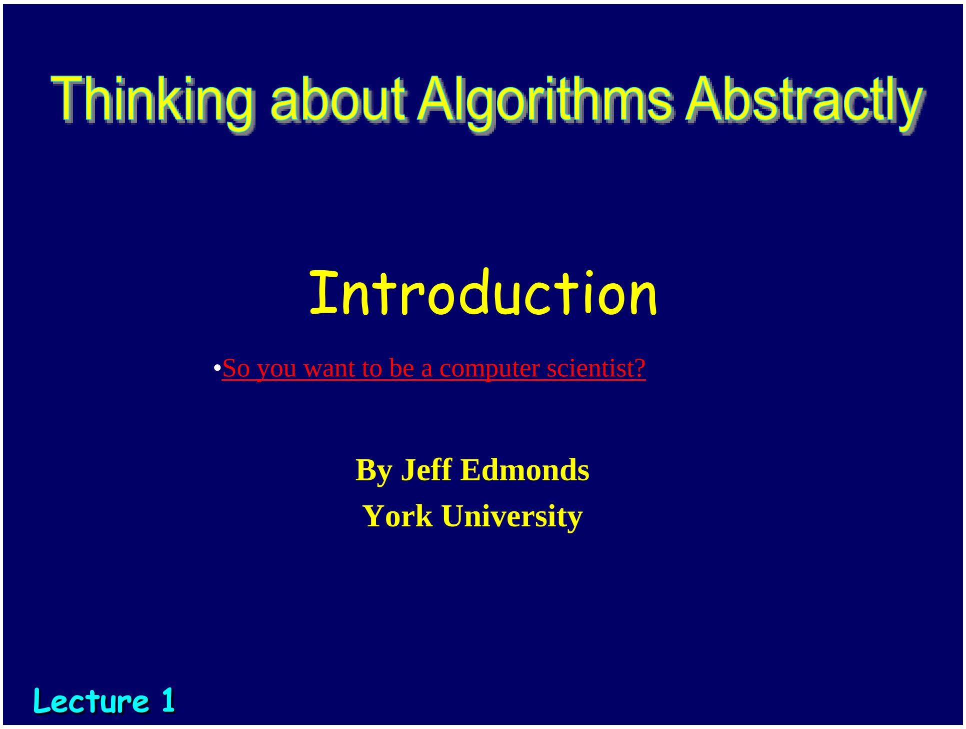


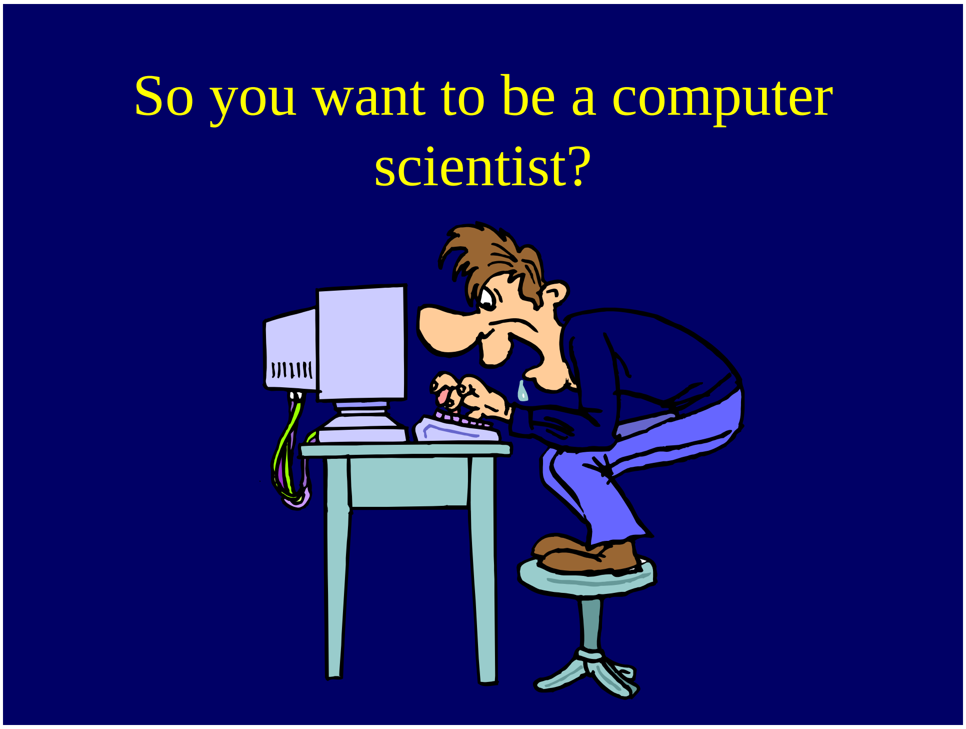
Thinking about Algorithms Abstractly

Introduction

- So you want to be a computer scientist?

By Jeff Edmonds
York University

Lecture 1

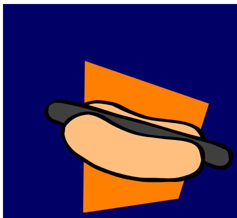


So you want to be a computer scientist?

Is your goal to be
a mundane programmer?

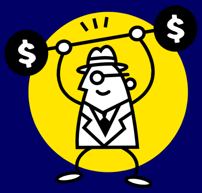
Or a great leader and thinker?

Original Thinking

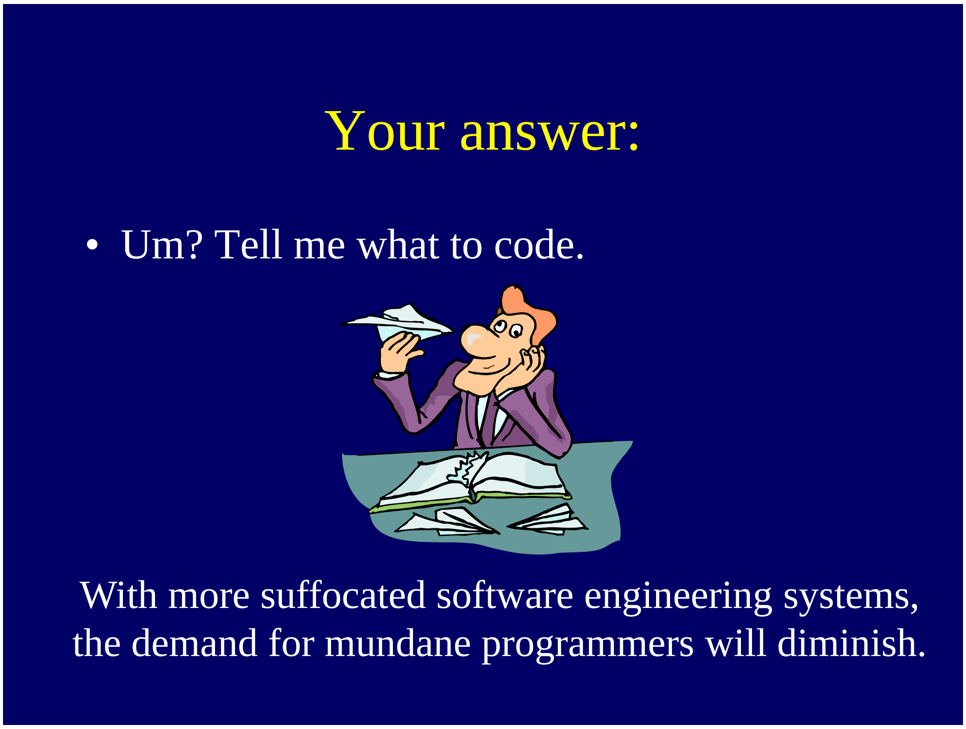


Boss assigns task:

- – Given today’s prices of pork, grain, sawdust, …
- – Given constraints on what constitutes a hotdog.
- – Make the cheapest hotdog.



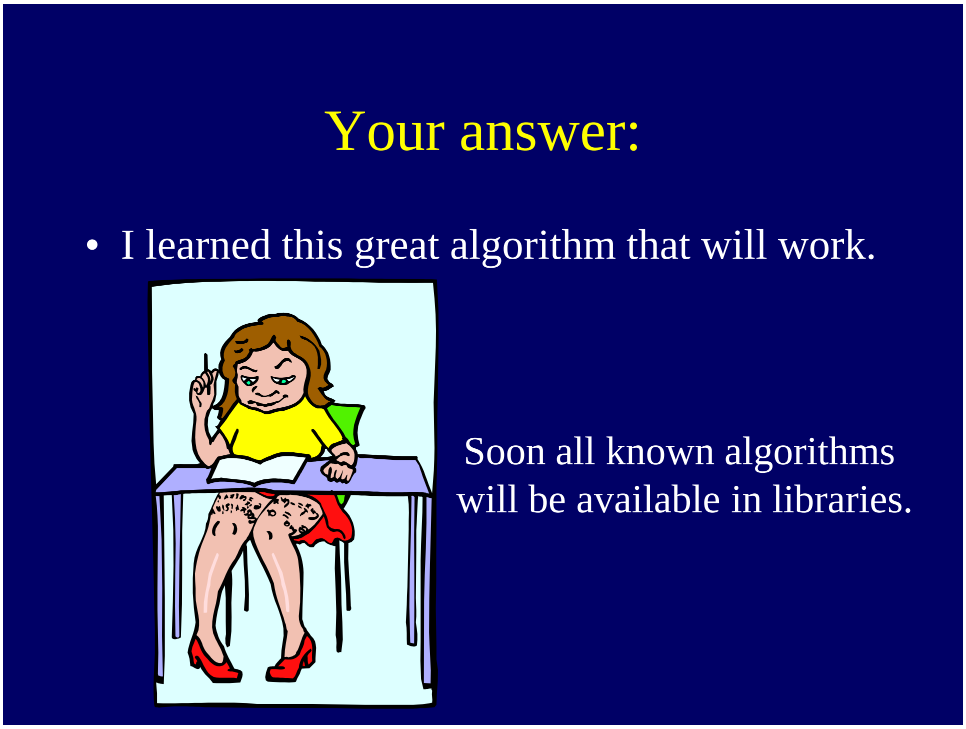
Everyday industry asks these questions.



Your answer:

- Um? Tell me what to code.

With more suffocated software engineering systems, the demand for mundane programmers will diminish.



Your answer:

- I learned this great algorithm that will work.

Soon all known algorithms will be available in libraries.

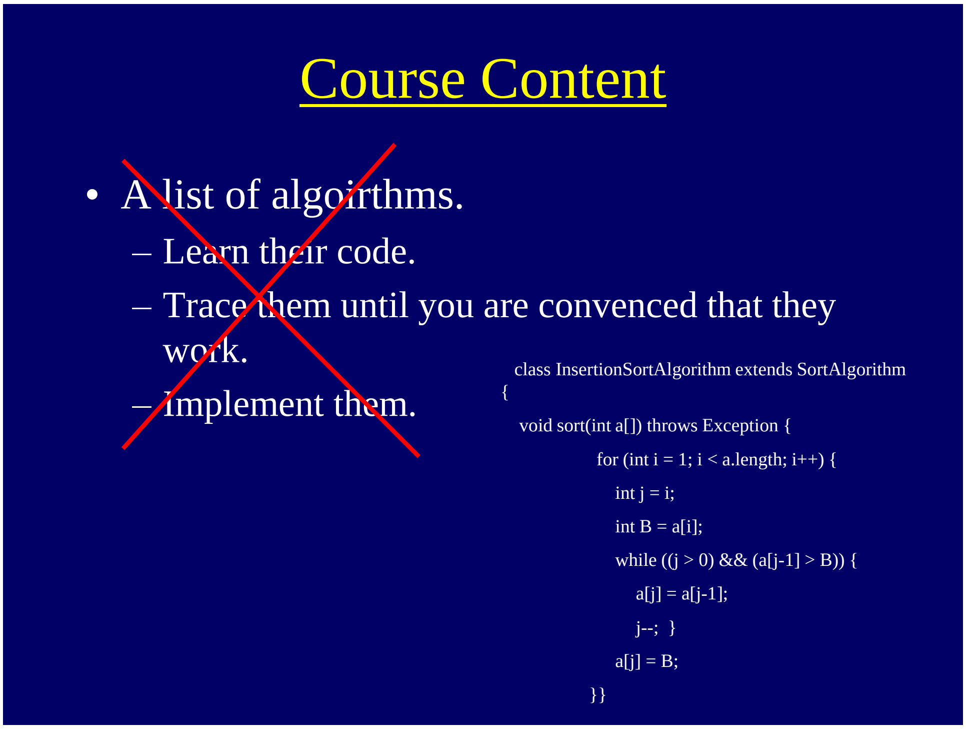
Your answer:

- I can develop a new algorithm for you.

Great thinkers
will always be needed.

The future belongs to the computer scientist who has

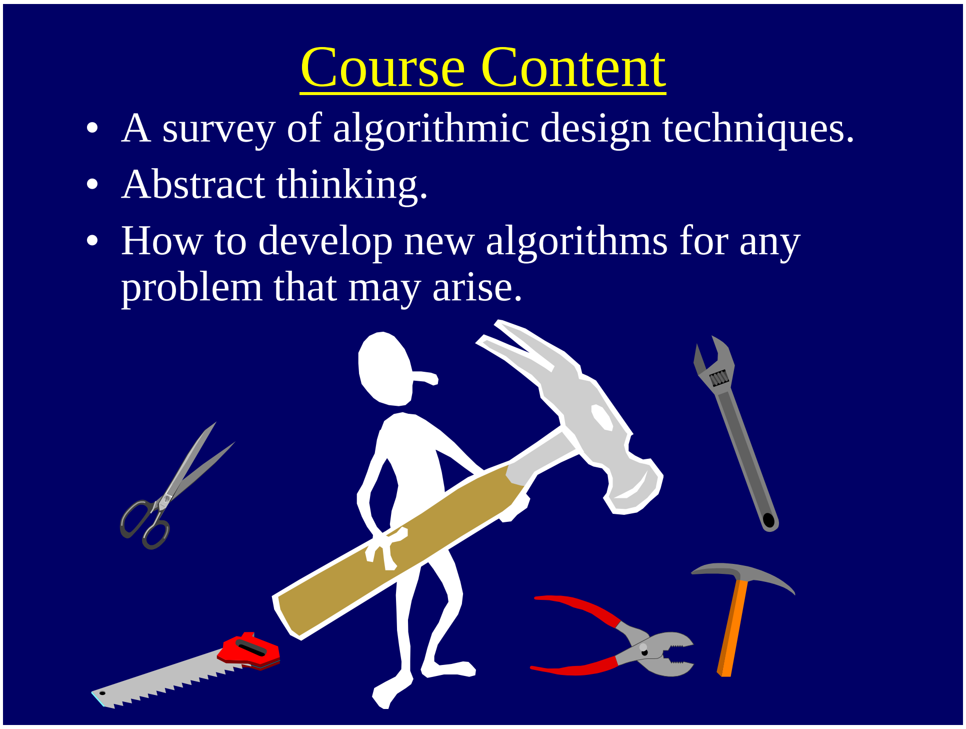
- Content: An up to date grasp of fundamental problems and solutions
- Method: Principles and techniques to solve the vast array of unfamiliar problems that arise in a rapidly changing field



Course Content

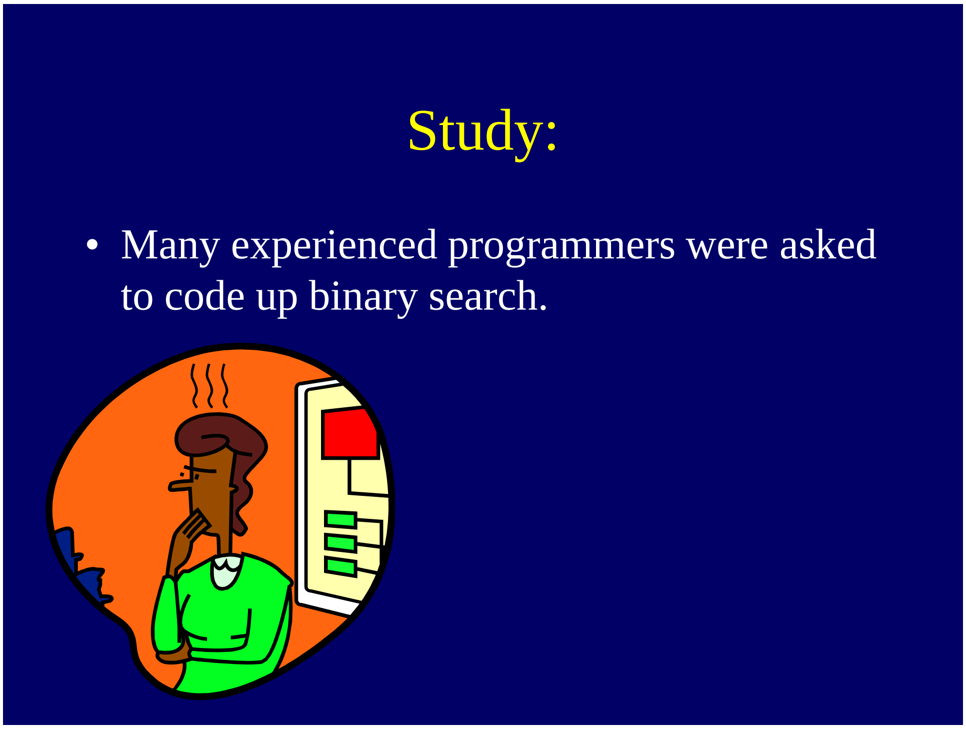
- ~~A list of algoirthms.~~
  - ~~Learn their code.~~
  - ~~Trace them until you are convenced that they work.~~
  - ~~Implement them.~~

```
  class InsertionSortAlgorithm extends SortAlgorithm
{
   void sort(int a[]) throws Exception {
                for (int i = 1; i < a.length; i++) {
                    int j = i;
                    int B = a[i];
                    while ((j > 0) && (a[j-1] > B)) {
                        a[j] = a[j-1];
                        j--;  }
                    a[j] = B;
               }}
```



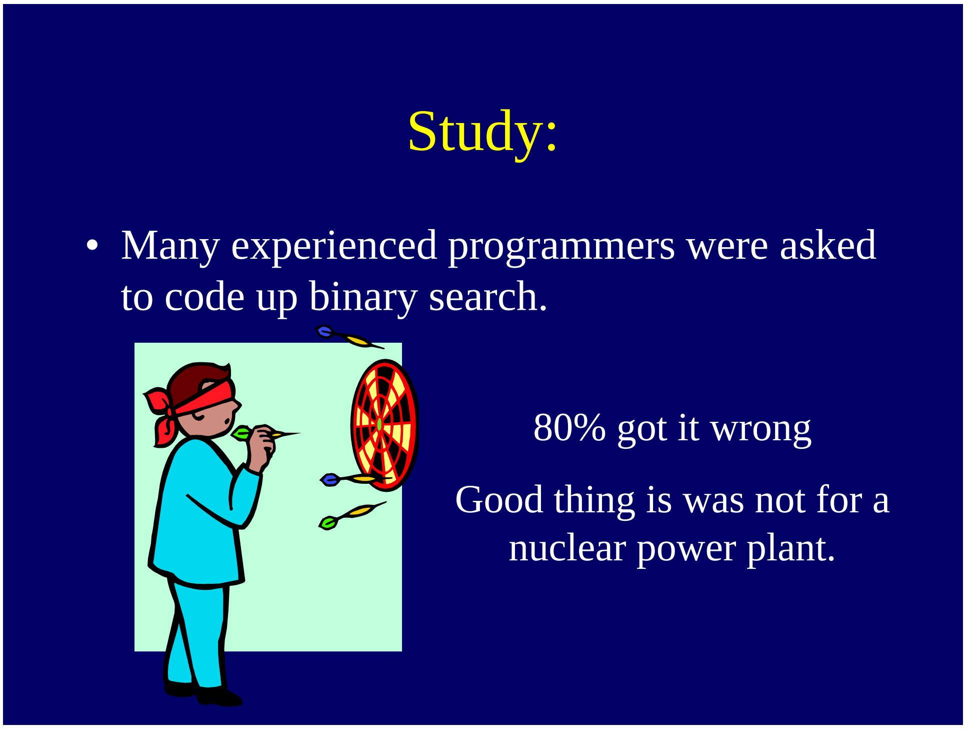
Course Content

- A survey of algorithmic design techniques.
- Abstract thinking.
- How to develop new algorithms for any problem that may arise.



Study:

- Many experienced programmers were asked to code up binary search.

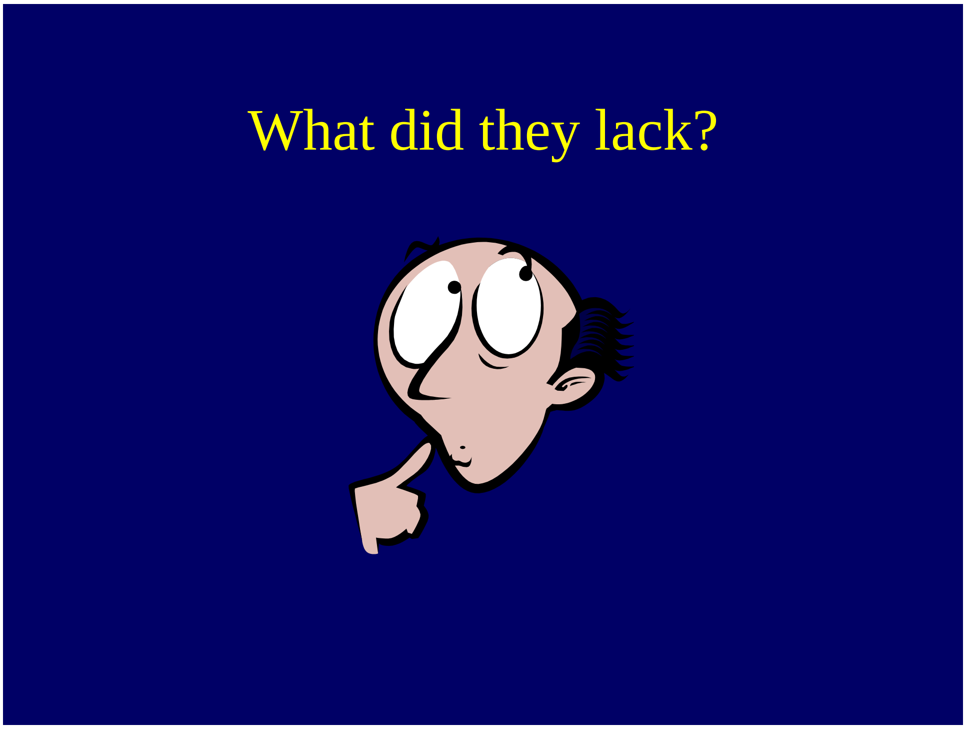


Study:

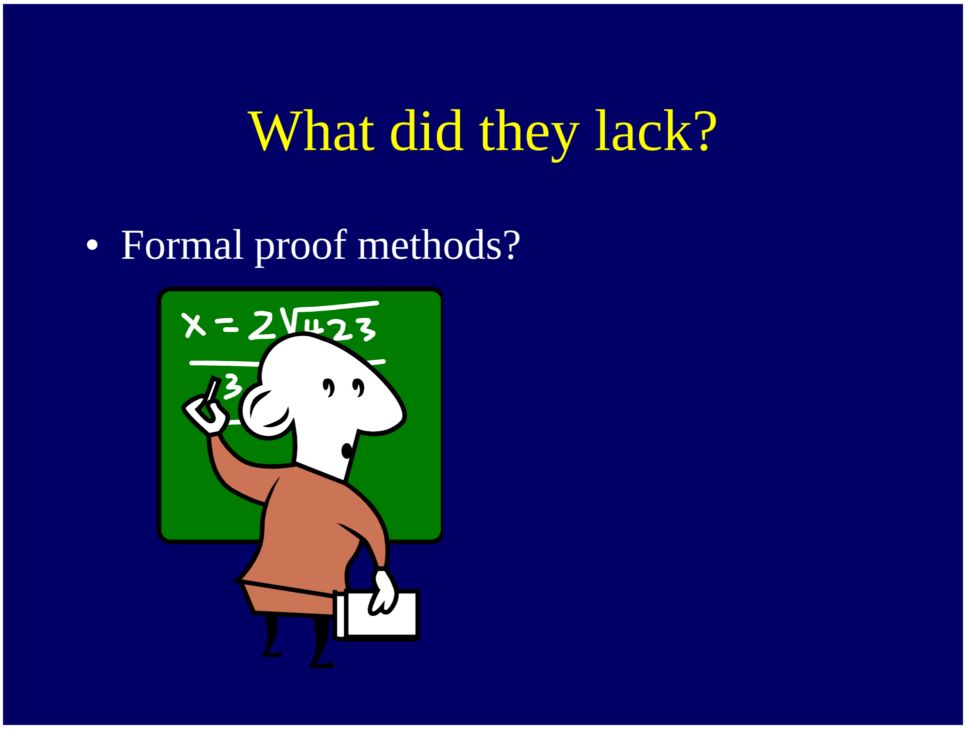
- Many experienced programmers were asked to code up binary search.

80% got it wrong

Good thing is was not for a nuclear power plant.

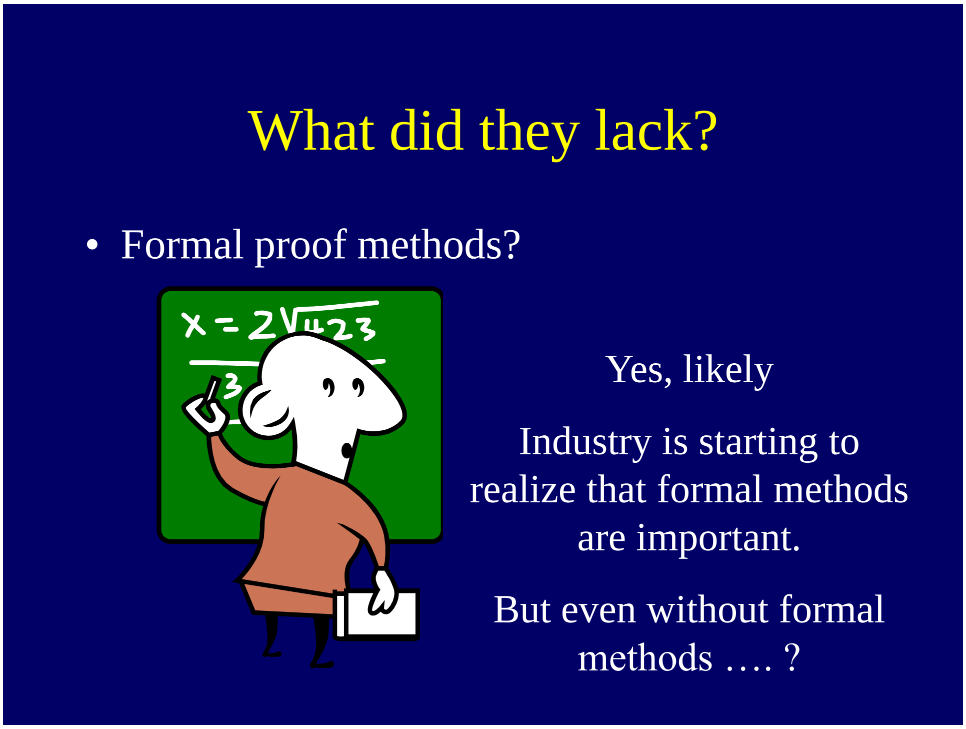


What did they lack?



What did they lack?

- Formal proof methods?



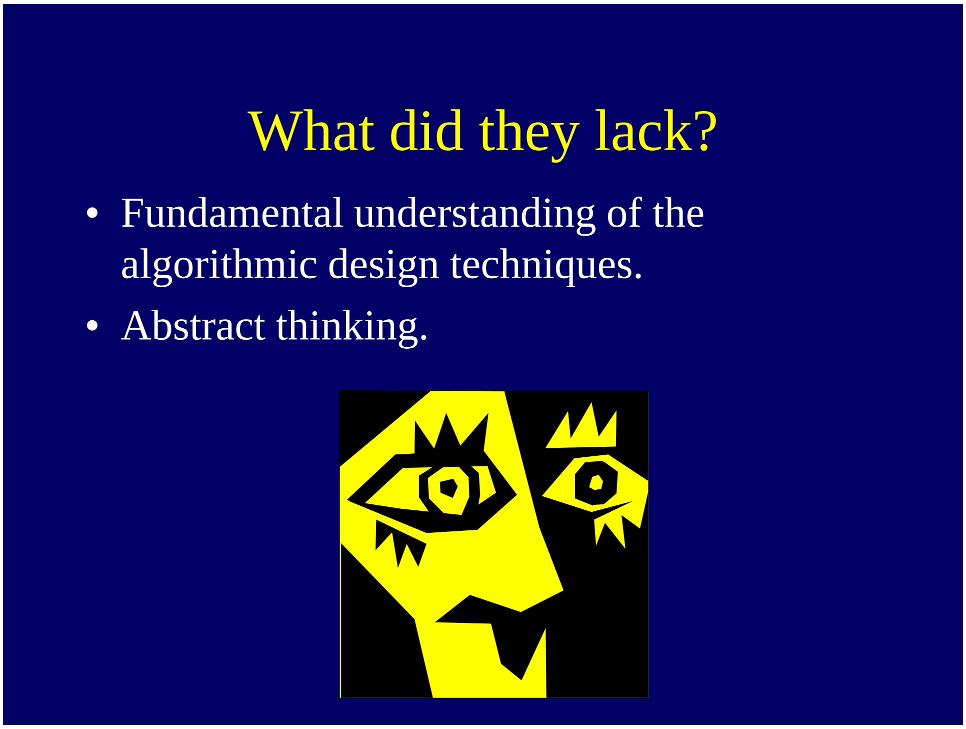
What did they lack?

- Formal proof methods?

Yes, likely

Industry is starting to realize that formal methods are important.

But even without formal methods …. ?



What did they lack?

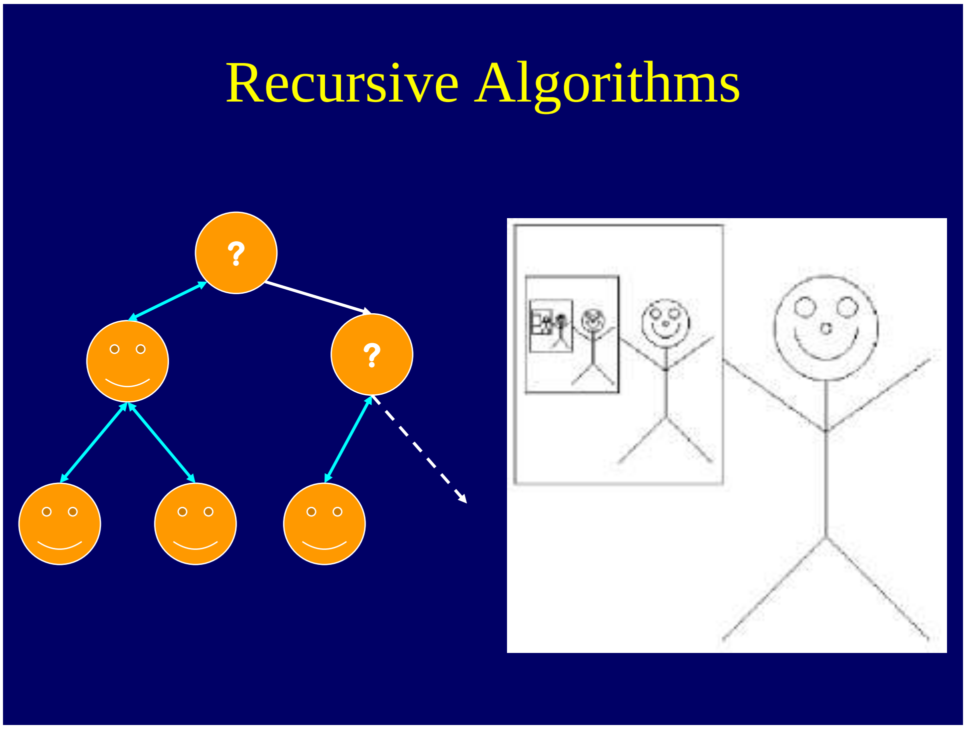
- Fundamental understanding of the algorithmic design techniques.
- Abstract thinking.

Course Content

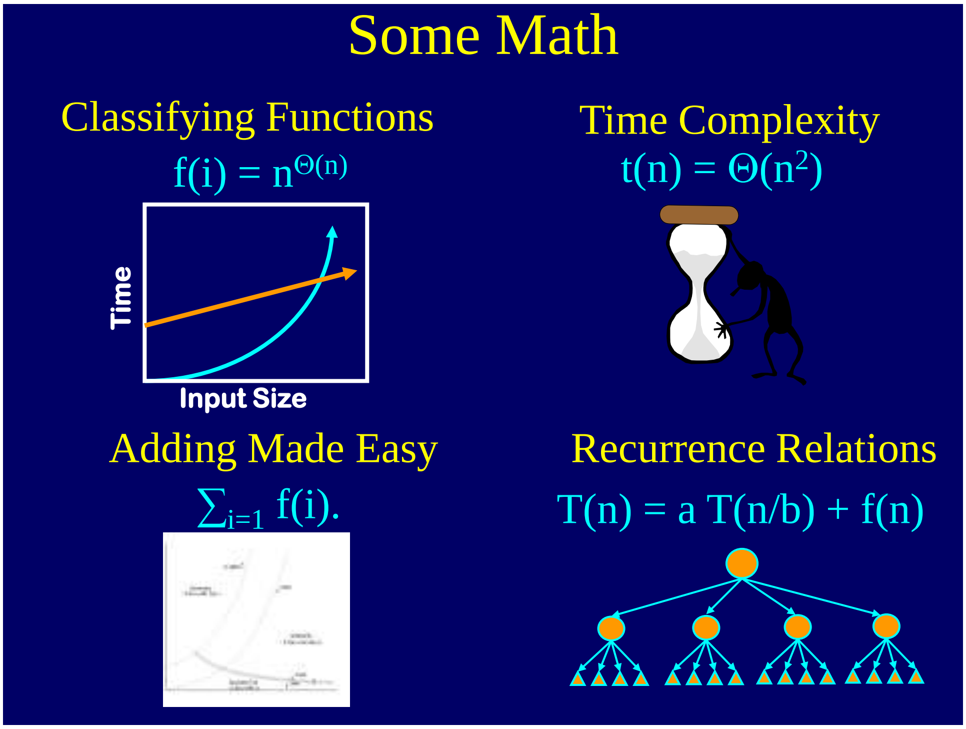
Notations, analogies, and abstractions
for developing,
thinking about,
and describing algorithms
so correctness is transparent

A survey of fundamental ideas and algorithmic design techniques

For example . . .



Recursive Algorithms



Some Math

Classifying Functions

$f(i) = n^{\Theta(n)}$

Time
Input Size

Time Complexity

$t(n) = \Theta(n^2)$

Adding Made Easy

$\sum_{i=1} f(i).$

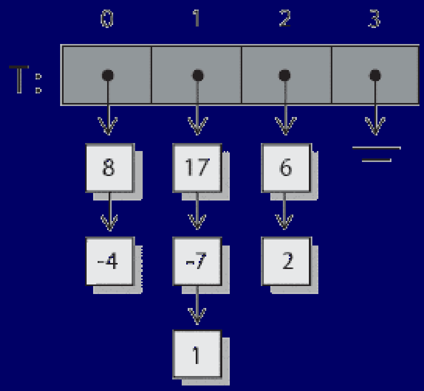
Recurrence Relations

$T(n) = a\,T(n/b) + f(n)$

Sort, Search Algorithms

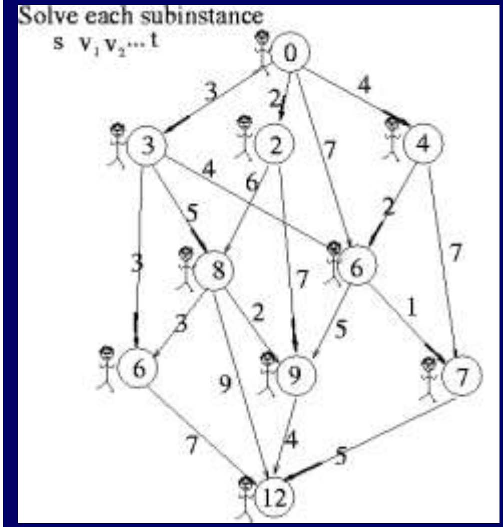
Graph Related Algorithms

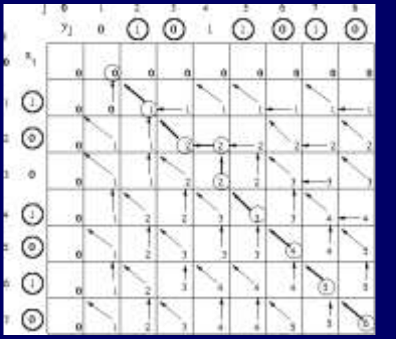
Hashing



Greedy Algorithms

Dynamic Programing





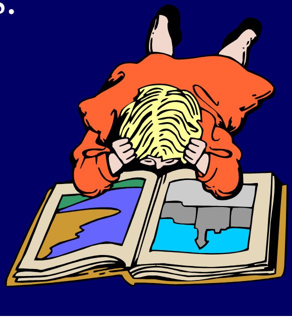
Useful Learning Techniques

Read Ahead

You are expected to read the lecture notes **before** the lecture.

This will facilitate more productive discussion during class.

Like in an English class



Also please proof read assignments & tests.

Explaining

- We are going to test you on your ability to explain the material.
- Hence, the best way of studying is to explain the material over and over again out loud to yourself, to each other, and to your stuffed bear.

While going along with your day

Day Dream

Mathematics is not all linear thinking.

Allow the essence of the material to seep into your subconscious

Pursue ideas that percolate up and flashes of inspiration that appear.

Be Creative

- Ask questions.
- Why is it done this way and not that way?

Guesses and Counter Examples

- Guess at potential algorithms for solving a problem.
- Look for input instances for which your algorithm gives the wrong answer.
- Treat it as a game between these two players.

Refinement:

The best solution comes from a process of repeatedly refining and inventing alternative solutions

Rudich www.discretemath.com