I want you to realize that, if you can imagine a computer doing something, you can program a computer to do that. Unbounded opportunity... limited only by your imagination. And a couple of laws of physics.

- TCP/IP, HTTP
  - How, Why, What, When?
Graphs: Structures and Algorithms

- How do packets of bits/information get routed on the internet?
  - Message divided into packets on client (your) machine
  - Packets sent out using routing tables toward destination
    - Packets may take different routes to destination
    - What happens if packets lost or arrive out-of-order?
  - Routing tables store local information, not global (why?)

- What about The Oracle of Bacon, Erdos Numbers, and Word Ladders?
  - All can be modeled using graphs
  - What kind of connectivity does each concept model?

- Graphs are everywhere in the world of algorithms (world?)

Vocabulary

- Graphs are collections of vertices and edges (vertex also called node)
  - Edge connects two vertices
    - Direction can be important, directed edge, directed graph
    - Edge may have associated weight/cost
  - A vertex sequence $v_0, v_1, \ldots, v_n$ is a path where $v_k$ and $v_{k+1}$ are connected by an edge.
    - If some vertex is repeated, the path is a cycle
    - A graph is connected if there is a path between any pair of vertices

Graph questions/algorithms

- What vertices are reachable from a given vertex?
  - Two standard traversals: depth-first, breadth-first
  - Find connected components, groups of connected vertices

- Shortest path between any two vertices (weighted graphs?)
  - Breadth first search is storage expensive
  - Dijkstra’s algorithm is efficient, uses a priority queue too!

- Longest path in a graph
  - No known efficient algorithm

- Visit all vertices without repeating? Visit all edges?
  - With minimal cost? Hard!

Depth, Breadth, other traversals

- We want to visit every vertex that can be reached from a specific starting vertex (we might try all starting vertices)
  - Make sure we don’t visit a vertex more than once
  - Why isn’t this an issue in trees?
  - Mark vertex as visited, use set/array/map for this
    - Can keep useful information to help with visited status
  - Order in which vertices visited can be important
  - Storage and runtime efficiency of traversals important

- What other data structures do we have: stack, queue, ...
  - What happens when we traverse using priority queue?
Breadth first search

- In an unweighted graph this finds the shortest path between a start vertex and every vertex
  - Visit every node one away from start
  - Visit every node two away from start
    - This is every node one away from a node one away
  - Visit every node three away from start, ...

- Put vertex on queue to start (initially just one)
  - Repeat: take vertex off queue, put all adjacent vertices on
  - Don’t put a vertex on that’s already been visited (why?)
  - When are 1-away vertices enqueued? 2-away? 3-away?
  - How many vertices on queue?

Code for breadth first

```java
public void breadth(String vertex){
    Set visited = new TreeSet();
    LinkedList q = new LinkedList();
    q.addLast(vertex);
    visited.add(vertex);
    while (q.size() > 0){
        String current = (String) q.removeFirst();
        // process current
        for (each v adjacent to current){
            if (!visited.contains(v)){// not visited
                visited.add(v);
                q.addLast(v);
            }
        }
    }
}
```

Pseudo-code for depth-first search

```java
void depthfirst(String vertex){
    if (!alreadySeen(vertex)) {
        markAsSeen(vertex);
        System.out.println(vertex);
        for (each v adjacent to vertex) {
            depthfirst(v);
        }
    }
}
```

- Clones are stacked up, problem? Can we make use of stack explicit?

Edsger Dijkstra

- Turing Award, 1972
- Operating systems and concurrency
- Algol-60 programming language
- Goto considered harmful
- Shortest path algorithm
- Structured programming
  - “Program testing can show the presence of bugs, but never their absence”
- A Discipline of programming
  - “For the absence of a bibliography I offer neither explanation nor apology”
**What does this position entail?**

- Do you want to build quantitative models millions of people will use, based on data from the world's largest online laboratory? Are you passionate about formulating relevant questions and producing solutions to initially ill-defined problems? Do the challenges and opportunities of terabytes of data excite you? Can you think abstractly and apply your ideas to the real world? Can you contribute to the big picture and are not afraid to handle the details?
- We are looking for people with the right blend of vision, intellectual curiosity, and hands-on skills, who want to be part of a highly visible, entrepreneurial team.

[http://www.ph.tn.tudelft.nl/PRInfo/jobs/msg00185.html](http://www.ph.tn.tudelft.nl/PRInfo/jobs/msg00185.html)

---

**What is this about?**

- Ideal candidates will have a track record of creating innovative solutions, and typically a Ph.D. in computer science, physics, statistics, or electrical engineering. Significant research experience is desired in fields including active learning, probabilistic graphical models and Bayesian networks, data mining and visualization, Web search and information retrieval, judgment and decision making, consumer modeling, and behavioral economics.
- What is data mining? What is machine learning?
Finally, ...

What is the Internet?

- The Internet was originally designed as an "overlay" network running on top of existing phone and other networks. It is based on a small set of software protocols that direct routers inside the network to forward data from source to destination, while applications run on the Internet to rapidly scale into a critical global service. However, this success now makes it difficult to create and test new ways of protecting it from abuses, or from implementing innovative applications and services.

http://www.intel.com/labs/features/idf09041.htm

What can be programmed?

- What class of problems can be solved?
  - G5, 1000Mhz Pentium III, Cray, pencil?
  - Alan Turing proved some things, hypothesized others
    - Halting problem, Church-Turing thesis

- What class of problems can be solved efficiently?
  - Problems with no practical solution
    - What does practical mean?
  - Problems for which we can’t find a practical solution
    - Solving one solves them all
    - Would you rather be rich or famous?
Schedule students, minimize conflicts

- Given student requests, available teachers
  - write a program that schedules classes
  - Minimize conflicts
- Add a GUI too
  - Web interface
  - ...
  - ...

I can’t write this program because I’m too dumb

One better scenario

I can’t write this program because it’s provably impossible

Another possible scenario

I can’t write this program but neither can all these famous people

Entscheidungsproblem

- What can we program?
- What can’t we program?

Can we write a program that will determine if any program P will halt when run on input S?
  - Input to halt: P and S
  - Output: yes/no halts
The halting problem: writing `doesHalt`

```java
public class ProgramUtils {
    /**
     * Returns true if program halts on input,
     * otherwise returns false (program loops)
     */
    public static boolean doesHalt(String progname, String input) {
    }
}
```

- A compiler is a program that reads other programs as input
  - Can a word counting program count its own words?
- The `doesHalt` method might simulate, analyze, ...
  - One program/function that works for any program/input

How to tell if Foo stops on 123 456

```java
public static void main(String[] args) {
    String prog = "Foo.java";
    String input = "123 456"
    if (ProgramUtils.doesHalt(prog, input)) {
        System.out.println(prog + " stops");
    } else {
        System.out.println(prog + " 4ever");
    }
}
```

- Can user enter name of program? Input?
  - What's the problem with this program?

Consider the class `Confuse.java`

```java
public static void main(String[] args) {
    String prog = "Foo.java";
    if (ProgramUtils.doesHalt(prog, prog)) {
        while (true) {
            // do nothing forever
        }
    }
}
```

- We want to show writing `doesHalt` is impossible
  - Proof by contradiction:
  - Assume possible, show impossible situation results
- Can a program read a program? Itself?

Can we write `Confuse.java`?

- Legal if `doesHalt` exists
  - What have we assumed?
- What are consequences of running `confuse` on itself?
  - Trouble?
What's a meta catalog? Top 10 sites?

- Consider a website of interesting sites
  - Does the website list itself? Is this a problem?

- Consider a website that lists every useless website
  - Would this be a useful resource?
  - Does the website list itself?

- What about a site of all the sites that list themselves?
  - What about sites that don’t list themselves? nolist.com

Not impossible, but impractical

- Towers of Hanoi
  - How long to move n disks?

- What combination of switches turns the light on?
  - Try all combinations, how many are there?
  - Is there a better way?

Travelling Salesperson

- Visit every city exactly once
- Minimize cost of travel or distance
- Is there a tour for under $2,000? less than 6,000 miles?
- Is close good enough?
  - Within 10% of optimal
  - Within 50% of optimal
  - ...

  Try all paths, from every starting point -- how long does this take?

  a, b, c, d, e, f, g
  b, a, c, d, e, f, g ...

Are hard problems easy?

- P = easy problems, NP = “hard” problems
  - P means solvable in polynomial time
    - Difference between N, N^2, N^10?
  - NP means non-deterministic, polynomial time
    - guess a solution and verify it efficiently

- Question: P = NP?
  - if yes, a whole class of difficult problems, the NP-complete problems, can be solved efficiently
  - if no, none of the hard problems can be solved efficiently
  - showing the first problem was NP complete was an exercise in intellectual bootstrapping, satisfiability/Cook/(1971)
Theory and Practice

- Number theory: pure mathematics
  - How many prime numbers are there?
  - How do we factor?
  - How do we determine primeness?

- Computer Science
  - Primality is “easy”
  - Factoring is “hard”
  - Encryption is possible

Computer Science in a Nutshell

- Public-key cryptography
- Randomized primality testing
- Encryption is possible