What is Computer Science?

What is it that distinguishes it from the separate subjects with which it is related? What is the linking thread which gathers these disparate branches into a single discipline? My answer to these questions is simple — *it is the art of programming a computer*. It is the art of designing efficient and elegant methods of getting a computer to solve problems, theoretical or practical, small or large, simple or complex.

C.A.R (Tony) Hoare
What is Computer Science?

- **If it's programming, then**
  - Is there life in a cubicle?
  - Are there opportunities outside of Google?
  - Are all the jobs in Bangalore?
  - Why don't biology majors take computer science course?
  - Can we find palindromes recursively? Who cares?

- **Managing and understanding information and the Internet**
  - Is the Internet a phenomenon? A revolution (industrial)?
  - What is information?
  - What is innovation and is it teachable or understandable?
  - Questions, questions, ... how about answers?
What can be programmed?

• What class of problems can be solved?
  ➢ G5, TRS-80, Pascal, C++, Scheme, Cray, Pencil/Paper
  ➢ 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
- **Curriculum 2000 changes**
  - Count all codes, why?
- **Add a GUI too**
  - Web interface
  - Database ...

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
Graph coloring (see colorable.cpp)

- Can vertices of a graph be colored so that no two adjacent vertices share the same color?
  - What is minimum # colors
  - Can graph be k-colored?

- Two problems, second is called a decision problem, first is an optimization problem

- Can a graph be 2-colored?
  - Depth first search, mark vertex with a color and ...

- Can a graph be k-colored?
  - Backtrack search
Graph coloring continued

- **Two-color problem solving using depth-first search, see code in colorable.cpp that uses stack**
  - Every reachable vertex put on stack,
  - Every edge processed once
  - Complexity is O(.....)

- **K-colorable problem tries each of k-colors**
  - For each color, use it on a vertex and then visit all adjacent vertices that aren’t colored yet
  - Backtrack to undo colorings if they don’t work out before trying next color
  - Recurrence is *at best*: \( T(n) = k \cdot T(n-1) + O(1) \)
  - What is solution to Towers of Hanoi problem?
Towers of Hanoi

- Move disks from "from" peg to "to" peg
- What is the recurrence: \(T(n) = 2T(n-1) + O(1)\) Solution?

```c
void Move(int from, int to, int aux, int numDisks)
    // pre: numDisks on peg from,
    // post: numDisks moved to peg to
{
    if (numDisks == 1) {
        cout << from << " to " << to << endl;
    }
    else {
        Move(from, aux, to, numDisks-1);
        Move(from, to, aux, 1);
        Move(aux, to, from, numDisks-1);
    }
}
```
Tim Berners-Lee

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?
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`

```cpp
bool DoesHalt(const string& progname,
               const string& s)
// post: returns true if progname halts given s
//       as input, false otherwise

int main()
{
    string f = PromptString("enter filename ");
    string s = PromptString("input for "+filename);
    if (DoesHalt(f,s)) cout << "does halt" << endl;
    else               cout << "does not halt" << endl;
}
```

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

```cpp
#include "halt.h"
int main()
{
    string f = PrompString("enter filename ");
    if (DoesHalt(f,f))
    {
        while (true)
        {
            // do nothing forever
        }
    }
    return 0;
}
```

- **We want to show writing `DoesHalt` is impossible**
  - Proof by contradiction:
    - Assume possible, show impossible situation results
Can we write confuse.cpp?

- Legal if `doesHalt` exists
  - What have we assumed?
- What are consequences of running confuse on itself?
  - Trouble?

```
P
DoesHalt

S

confuse

if DoesHalt(..,..) loop
else exit
```
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?
  - Consider spanning tree

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 ...
Complexity Classifications

- This route hits all cities for less than $2,000 — verify properties of route efficiently.
- Hard to *find* optimal solution

Pack trucks with barrels, use minimal # trucks

Ideas?

Problems are the “same hardness”: solve one efficiently, solve them all
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 can be solved efficiently---one problem is *reducible* to another
  - 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)
  - An NP complete problem is in NP (guessable/verify) and is the same “difficulty” as every other problem in NP
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

- Top secret
- Public-key cryptography
- Randomized primality testing
Shafi Goldwasser

- RCS professor of computer science at MIT
  - Co-inventor of zero-knowledge proof protocols

How do you convince someone that you know something without revealing “something”

- Consider card readers for dorms
  - Access without tracking

Work on what you like, what feels right, I now of no other way to end up doing creative work
Why is programming fun?

What delights may its practitioner expect as a reward?

First is the sheer joy of making things

Second is the pleasure of making things that are useful

Third is the fascination of fashioning complex puzzle-like objects of interlocking moving parts

Fourth is the joy of always learning

Finally, there is the delight of working in such a tractable medium. The programmer, like the poet, works only slightly removed from pure thought-stuff.

Fred Brooks
What is computer science?

- **What is a computation?**
  - Can formulate this precisely using mathematics
  - Can say “anything a computer can compute”
  - Study both theoretical and empirical formulations, build machines as well as theoretical models

- **How do we build machines and the software that runs them?**
  - Hardware: gates, circuits, chips, cache, memory, disk, ...
  - Software: operating systems, applications, programs

- **Art, Science, Engineering**
  - How do we get better at programming and dealing with abstractions
  - What is hard about programming?
Fred Brooks

- ... on computing pioneer Howard Aiken "the problem was not to keep people from stealing your ideas, but to make them steal them."

- Duke valedictorian 1953, started UNC Computer Science Dept in 1964, won Turing Award in 1999

- Mythical-Man Month, "Adding man-power to a late project makes it later", ... "There is no silver-bullet for Software Engineering... [because of essential complexity]"