Limits and Future of Computing
Where do we go from here?

Back to Algorithms...
Recall that algorithms provide computational solutions for problems
- Problems can be solved by multiple algorithms
- We can "rank" problems by the fastest algorithm that solves them

Some problems are efficiently solvable:
- Algorithms solve them with time: constant, logarithmic, linear, quadratic
- In general, "polynomial time" - time bounded by $n^c$ for some constant $c$

What about problems for which we don't know efficient solutions?
- Are there limits to what we can compute efficiently?

But there are some hard problems...
Example: The Traveling Salesperson Problem (TSP)
Given a map, what is the shortest route that visits all cities and returns home?

A small example:

Question: What order to visit the cities (start from and return to "A")?
But there are some hard problems...

Example: The Traveling Salesperson Problem (TSP)

Given a map, what is the shortest route that visits all cities and returns home?

A small example:

Can try all possibilities:
A-B-C-D-A = 97
A-B-D-C-A = 108
A-C-B-D-A = 141
A-C-D-B-A = 108
A-D-B-C-A = 141
A-D-C-B-A = 97

Question: What order to visit the cities (start from and return to "A")?

What happens when the number of cities grows?

What about 7 cities?

For a complete map of 7 cities, there are 6 choices for first city to visit, then 5 remaining cities for the second city, then 4, then 3, ... So there are

4*5*4*3*2*1 = 720 routes

<table>
<thead>
<tr>
<th>Cities</th>
<th>Number of Routes</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>362,880</td>
</tr>
<tr>
<td>15</td>
<td>172,972,800</td>
</tr>
<tr>
<td>20</td>
<td>2,432,902,008</td>
</tr>
</tbody>
</table>

Testing 1 billion routes/sec would take 121,645,100 seconds... or over 3.85 years

NP-complete Problems

Some problems that share a common computational structure

Is there an algorithm that efficiently solves the TSP?

We don't know!!

TSP (in yes/no form) is an NP-complete problem

- Many important problems (thousands!) are NP-complete
- They share some common properties
  - Can verify solutions efficiently
  - If can solve any NP-complete problem efficiently, can solve them all efficiently

This is known as the P vs NP problem, and is the biggest unsolved problem in computer science

Clay Institute: $1 million "bounty" for a solution to this problem!
Some Awareness in Popular Media

Numb3rs TV show: Often deals with difficult problems, including NP Completeness (e.g., episode 2)

Some problems are known to be solvable, but not efficiently (known!)
- "Generalized checkers": Computing optimal checkers strategy for an n x n checkers board

Some problems do not have algorithmic solutions at all!

The "Halting Problem"
- Programs are just bits stored in files, just like any other file
- Therefore, programs can be inputs to other programs
- The Halting Problem: Given a program to run with a specific input, will it eventually halt and give an answer?

Obviously would be great if we could solve (no more programs that hang!)

Unfortunately, the Halting Problem is undecidable (uncomputable): no algorithm, no matter how clever or complex, can solve the Halting Problem for all inputs (i.e., for all programs)

Coping with NP-hard Problems

Lots of very important, practical problems are NP-hard

Is it just hopeless*?

Let's look at some strange cutting-edge research directions...

* If you want exact answers, that is. Approximation algorithms are sometimes "good enough"!
Quantum Computing
The physics of the matter...

In the strange world of quantum physics, particles/matter can be in multiple states simultaneously - in quantum superposition.

Classic physics thought experiment: Schroedinger's cat

Simultaneously alive and dead, until observed (so the cat isn’t an observer? but that’s not really the point...)

Quantum Computing
So what does this mean for computing?

In standard computing, bits are 0 or 1
In quantum computing, qubits can simultaneously be both
• Computations work on a superposition of values until observed

So what?
• If working with data in many states simultaneously, can potentially do many calculations simultaneously!

Quantum Computing
What can you do with a quantum computer?

Grover’s Algorithm for database searching
Invented in 1996

Problem: Searching an unsorted list (like "contains" in BYOB)
Classical: Requires linear (n) time
Quantum: Grover’s algorithm works in $n^{1/2}$ (square root of n) time.
Searching for a 64-bit crypto key:
Classical: 2^n steps (584 years @1GHz)
Quantum: 2^(1/2) steps (4 seconds @1GHz)

Shor’s Algorithm for factoring
Invented in 1994

Problem: Factor a large number ("large" can mean hundreds of digits or more)
Importance: If you can factor, you can break RSA encryption
Classical: Apparently worse than polynomial ("trial division" is exponential)
Quantum: Proportional to $n^2$
Quantum is an exponential improvement!
Quantum Computing
An interesting read...

The Emperor's New Mind
by Roger Penrose

Won the 1990 Science Book Prize

Central claim: Human consciousness is non-algorithmic, and quantum physics plays a key role in human consciousness.

So... are quantum computers essential to "real AI"?

Quantum Computing
So, is this real or just mathematical games?

In the past few years quantum computing has gotten a lot of attention due to practical advancements...

Quantum Computing
So, is this real or just mathematical games?

October 2012 announcement of Nobel Prize for Physics - for work that could help build quantum computers....
DNA Computing

**Basic idea:** DNA is just a set of instructions on how to build a living organism, and constructing that organism is "executing the code".

**So:** Can we synthesize instruction sequences in DNA to compute a solution to a non-biological problem?

**Why:** DNA has incredibly high storage density!

One cubic centimeter of DNA holds more information than a trillion CDs.

DNA Computing

 Are these real?

Yes, they can be built!

Existing DNA computers, like the one reported in 2008, are very simplistic ("two-pancake" problem, similar to "two-qubit" quantum computer):

- Used genetically engineered E. coli bacteria
- Not useful as computing systems yet, but interesting "proof of concept"

The potential (using real/realistic numbers):

- 1000 operations per second,
- With 100 billion in parallel,
- Gives 100 trillion operations per second.

Where do we go next... for impact

National Academy of Engineering selected 14 "Grand Challenge" problems - these make significant impacts on civilization!

1. Make solar energy economical
2. Provide energy from fusion
3. Develop carbon sequestration methods
4. Manage the nitrogen cycle
5. Provide access to clean water
6. Restore and improve urban infrastructure
7. Advance health informatics
8. Engineer better medicines
9. Reverse-engineer the brain
10. Prevent nuclear terror
11. Secure cyberspace
12. Enhance virtual reality
13. Advance personalized learning
14. Engineer the tools of scientific discovery

Challenges in red reflect strong computer science problems

Challenges in blue cannot be advanced without strong computational tools
Where do we go next... for impact

National Academy of Engineering selected 14 "Grand Challenge" problems - these make significant impacts on civilization!

1. Make solar energy economical
2. Provide energy from fusion
3. Develop methods for carbon sequestration
4. Manage the nitrogen cycle
5. Provide access to clean water
6. Restore and improve urban infrastructure
7. Advance health informatics
8. Engineer better medicines
9. Reverse-engineer the brain
10. Prevent nuclear terror
11. Secure cyberspace
12. Enhance virtual reality
13. Advance personalized learning
14. Engineer the tools of scientific discovery

Challenges in red reflect strong computer science problems
Challenges in blue cannot be advanced without strong computational tools

It's time for you to start thinking about how you can make the world a better place!

And computing is a great way to make a difference...