Computer Science as a research discipline

A grand but brief and idiosyncratic tour
The idea of a research university

Research scholars teaching courses ... why would we do that?

Why not full-time teachers?

There is a tension ...

... but also a (larger?) benefit
Many sub-areas with varying methods ...
Some look like natural sciences, some like social sciences,
some like mathematics, some like engineering research.
Questions (examples):
What is the most efficient way to determine primality?
Is a probabilistic answer cheaper than an absolute answer?
Suppose that solutions to a problem can be verified quickly. Then, can the solutions themselves also be computed quickly? ($P = NP$?)
(Quick $1,000,000$ if you’ve got the answer)

Arguments and evidence: Proofs (mathematics)

Rich interactions with applied areas of computer science
AI researchers may or may not care about how human intelligence works

Example questions:
How can I learn to distinguish spam from real email?
How can I rank and group web search results?
How can I combine statistical reasoning with logic?
How quickly can I determine satisfiability of a propositional logic formula?
How can I make a car drive itself?

Methods:
    Some theory (proofs), lots of building and testing; contests (!)
Includes both user interface design & the underlying science of human perception and cognition (Strong overlap with cognitive psychology in methods and topics)

Example questions:
What visual search strategies do people use when looking for information on a web page?
What are the limitations of haptic (force feedback) devices for exploring information?
Is pitch or volume more effective in communicating a quantitative variable?

Methods and evidence often involve human subjects experiments
Example questions:
How can geographically distributed teams work together effectively?
What parts of software testing can be automated?
How can I organize software to improve reuse?

Methods: Varies depending on the question (wildly eclectic)
Applications research brings CS to another field, and another field to CS

Not just building stuff ...
   It’s only research if we learn something new from it. It’s only CS research if we learn something new in CS.

Example: Assistive interfaces for blind users
   CS angles: Requirements engineering (SE), interface design (HCI), collaborative design (SE, HCI)

Example: Neuroinformatics
   CS angles: high-performance computation, visualization (systems)
Relatively new (even within CS), but growing

Example questions:
How can I prevent address spoofing?
How can I be sure electronic voting is secure?
Are protocols based on prime factorization really secure?
How can I securely distribute software updates?
How can I guard against distributed denial-of-service attacks by zombie armies?
How can I distinguish human users from bots?
Distinct but intertwined research areas

Ranging from very theoretical (mostly proofs) to very experimental

Example questions:
(PL) What is the relation between a type system and a logic?
(CM) How can I add generics to a language like Java, without sacrificing type safety or efficiency?
(PL) How can I cleanly capture the semantics of continuations?
(CM) Does continuation-passing style help produce better code?
An area defined as much by methods (experimental computer science) as by topics ... but with lots of interplay

Example questions:
How can I (design | program | compile) programs to run efficiently on 1000s of processors?
How can I diagnose performance problems in big parallel programs?
How can I design software to limit battery drain in small mobile devices?
How can I cost-effectively distribute video to millions of people across the internet, even if a previously obscure video suddenly goes viral?
Eclectic, but not random. 
It’s not about the computer

Computational thinking weaves together the theoretical and experimental areas of computer science

Neither fish nor fowl nor platypus

As old as Euclid, as fundamental as Muḥammad ibn Mūsā al-Khwārizmī

But just over 50 years as an academic discipline