Seeking evidence for basing the CS theory course on non-decision problems

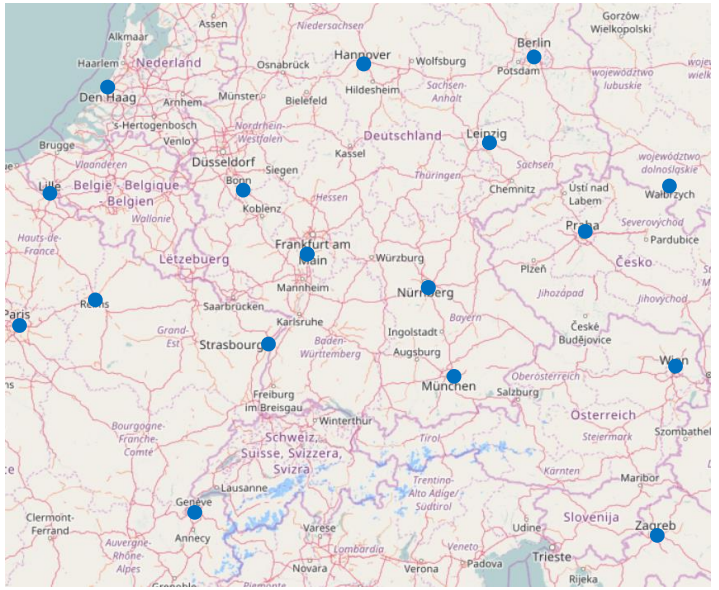
SIGCSE 2017 lightning talk

John MacCormick, Dickinson College

jmac@dickinson.edu

Which is more "useful": program *A* or program *B*?

Input: Input to both programs is a roadmap and a list of cities:



Output:

Program *A* outputs {

- "yes" if there's a driving route that visits each city and takes less than 100 hours
- "no" otherwise

Program *B* outputs {

- **a description of a suitable route** if there's a driving route that visits each city and takes less than 100 hours
- "no" otherwise

Which is more relevant for teaching: program *A* or program *B*?

Program *A* outputs { “**yes**” / “**no**”

Program *B* outputs { **a description of a suitable route** / “**no**”

Which is more relevant for teaching: program *A* or program *B*?

Program *A* outputs { “**yes**” / “**no**” }

- *Decision problem*.

Program *B* outputs { **a description of a suitable route** / “**no**” }

- *Non-decision problem*.

Which is more relevant for teaching: program *A* or program *B*?

Program *A* outputs { "`yes`" / "`no`" }

- *Decision problem*.
- Existing theory-of-computation courses usually focus on decision problems.

Program *B* outputs { **a description of a suitable route** / "`no`" }

- *Non-decision problem*.

Which is more relevant for teaching: program *A* or program *B*?

Program *A* outputs { "**yes**" / "**no**"

- *Decision problem*.
- Existing theory-of-computation courses usually focus on decision problems.

Program *B* outputs { **a description of a suitable route** / "**no**"

- *Non-decision problem*.

- This talk points to a way to teach the theory-of-computation course using non-decision problems.
- Students may achieve better learning because the content is perceived as relevant and practical.

Interested in a practical and relevant CS theory course? Get in touch!

Ways you can help and/or benefit:

1. Teach your theory course using the free beta version of a new textbook from Princeton University Press:

What Can Be Computed?: A Practical Guide to the Theory of Computation

   - Covers undergraduate computational and complexity theory using real Python programs and focusing on practical non-decision problems

2. Participate in an empirical analysis of student learning via this practical approach to CS theory. Collaborators and co-authors are needed!

How? Contact John MacCormick, jmac@dickinson.edu