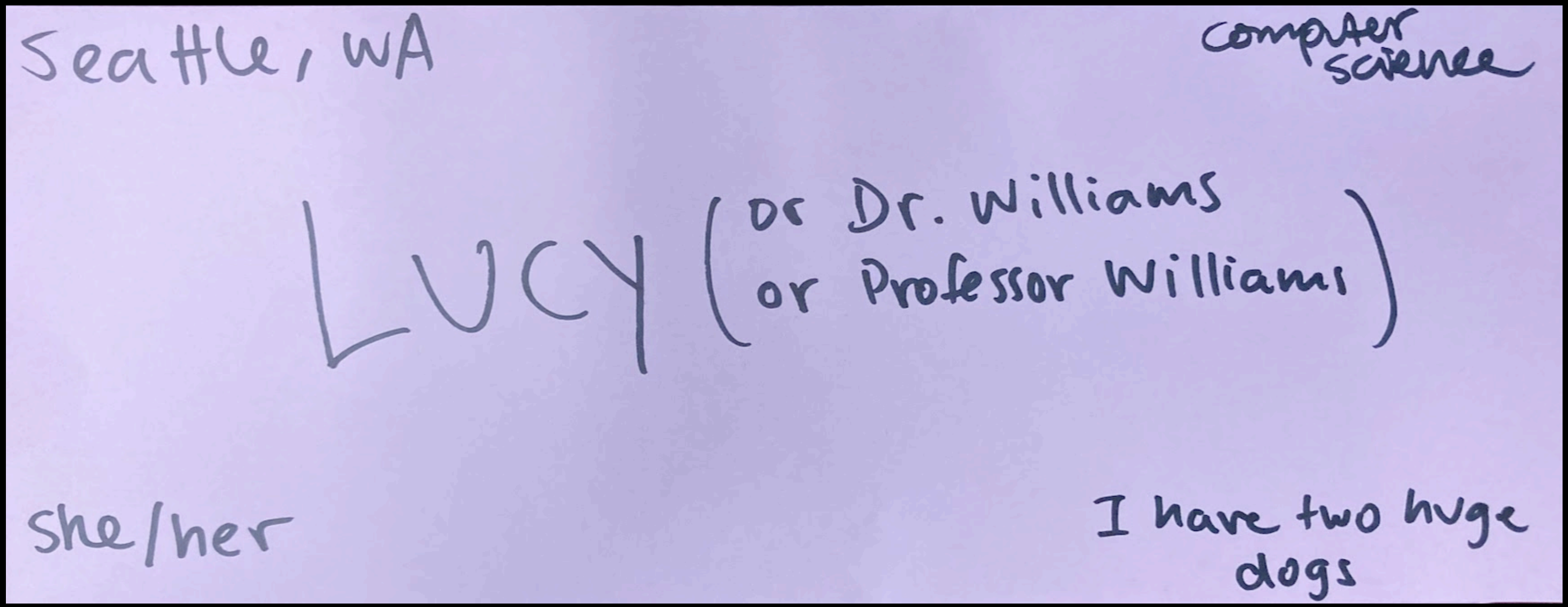
CSCI 332: Advanced Algorithms & Data Structures

After you sit down, please fold your paper hot dog style and write:

- What you'd like to be called
- Your hometown
- Your pronouns
- Your major/concentration
- A fun fact about you



Seattle, WA

computer science

LUCY (or Dr. Williams or Professor Williams)

she/her

I have two huge dogs

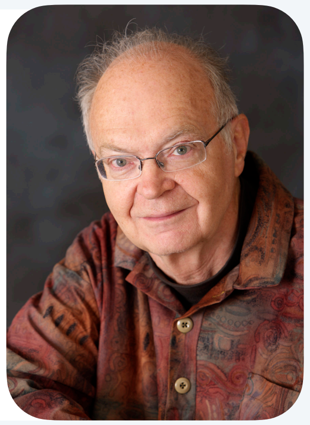
Introduce yourself to your neighbors!

Algorithm definition

Algorithm definition

" An algorithm is a finite, definite, effective procedure, with some input and some output. "

— Donald Knuth



But...

But...

"*Algorithmic problems form the heart of computer science, but they rarely arrive as cleanly packaged, mathematically precise questions. Rather, they tend to come bundled together with lots of messy, application-specific detail, some of it essential, some of it extraneous.*"

— *Kleinberg & Tardos*

CSCI 232 vs. CSCI 332

What were the focuses of CSCI 232?

CSCI 232 vs. CSCI 332

CSCI 232 vs. CSCI 332

CSCI 232. Implementation and consumption of classic algorithms.

CSCI 232 vs. CSCI 332

CSCI 232. Implementation and consumption of classic algorithms.

- Fundamental data structures (arrays, stacks, queues, etc.).

CSCI 232 vs. CSCI 332

CSCI 232. Implementation and consumption of classic algorithms.

- Fundamental data structures (arrays, stacks, queues, etc.).
- Sorting.

CSCI 232 vs. CSCI 332

CSCI 232. Implementation and consumption of classic algorithms.

- Fundamental data structures (arrays, stacks, queues, etc.).
- Sorting.
- Searching.

CSCI 232 vs. CSCI 332

CSCI 232. Implementation and consumption of classic algorithms.

- Fundamental data structures (arrays, stacks, queues, etc.).
- Sorting.
- Searching.
- Graph algorithms.

CSCI 232 vs. CSCI 332

CSCI 232. Implementation and consumption of classic algorithms.

- Fundamental data structures (arrays, stacks, queues, etc.).
- Sorting.
- Searching.
- Graph algorithms.
- String processing.

CSCI 232 vs. CSCI 332

CSCI 232. Implementation and consumption of classic algorithms.

- Fundamental data structures (arrays, stacks, queues, etc.).
- Sorting.
- Searching.
- Graph algorithms.
- String processing.
- Compression.

```
private static void sort(double[] a, int lo, int hi)  {
  if (hi <= lo) return;
  int lt = lo, gt = hi;
  int i = lo;
  while (i <= gt) {
    if      (a[i] < a[lo]) swap(a, lt++, i++);
    else if (a[i] > a[lo]) swap(a, i, gt--);
    else                   i++;
  }

  sort(a, lo, lt - 1);
  sort(a, gt + 1, hi);
}
```

Emphasizes critical thinking, problem-solving, and code.

CSCI 232 vs. CSCI 332

CSCI 332. Design and analysis of algorithms.

CSCI 232 vs. CSCI 332

CSCI 332. Design and analysis of algorithms.

- translate natural-language descriptions of computational problems into precisely formulated computational problems

CSCI 232 vs. CSCI 332

CSCI 332. Design and analysis of algorithms.

- translate natural-language descriptions of computational problems into precisely formulated computational problems
- recognize various real-world computational problems as instances of known computational problems

CSCI 232 vs. CSCI 332

CSCI 332. Design and analysis of algorithms.

- translate natural-language descriptions of computational problems into precisely formulated computational problems
- recognize various real-world computational problems as instances of known computational problems
- identify valid inputs and correct outputs to computational problems

CSCI 232 vs. CSCI 332

CSCI 332. Design and analysis of algorithms.

- translate natural-language descriptions of computational problems into precisely formulated computational problems
- recognize various real-world computational problems as instances of known computational problems
- identify valid inputs and correct outputs to computational problems
- trace through algorithm execution

CSCI 232 vs. CSCI 332

CSCI 332. Design and analysis of algorithms.

- translate natural-language descriptions of computational problems into precisely formulated computational problems
- recognize various real-world computational problems as instances of known computational problems
- identify valid inputs and correct outputs to computational problems
- trace through algorithm execution
- argue the correctness of an algorithm using a variety of proof techniques (direct proof, proof by induction, etc.)

CSCI 232 vs. CSCI 332

CSCI 332. Design and analysis of algorithms.

- translate natural-language descriptions of computational problems into precisely formulated computational problems
- recognize various real-world computational problems as instances of known computational problems
- identify valid inputs and correct outputs to computational problems
- trace through algorithm execution
- argue the correctness of an algorithm using a variety of proof techniques (direct proof, proof by induction, etc.)
- give counterexamples showing that an algorithm is incorrect

CSCI 232 vs. CSCI 332

CSCI 332. Design and analysis of algorithms.

- translate natural-language descriptions of computational problems into precisely formulated computational problems
- recognize various real-world computational problems as instances of known computational problems
- identify valid inputs and correct outputs to computational problems
- trace through algorithm execution
- argue the correctness of an algorithm using a variety of proof techniques (direct proof, proof by induction, etc.)
- give counterexamples showing that an algorithm is incorrect
- correctly describe an algorithm’s runtime from a variety of practical viewpoints

CSCI 232 vs. CSCI 332

CSCI 332. Design and analysis of algorithms.

- translate natural-language descriptions of computational problems into precisely formulated computational problems
- recognize various real-world computational problems as instances of known computational problems
- identify valid inputs and correct outputs to computational problems
- trace through algorithm execution
- argue the correctness of an algorithm using a variety of proof techniques (direct proof, proof by induction, etc.)
- give counterexamples showing that an algorithm is incorrect
- correctly describe an algorithm’s runtime from a variety of practical viewpoints

CSCI 232 vs. CSCI 332

CSCI 332. Design and analysis of algorithms.

- translate natural-language descriptions of computational problems into precisely formulated computational problems
- recognize various real-world computational problems as instances of known computational problems
- identify valid inputs and correct outputs to computational problems
- trace through algorithm execution
- argue the correctness of an algorithm using a variety of proof techniques (direct proof, proof by induction, etc.)
- give counterexamples showing that an algorithm is incorrect
- correctly describe an algorithm’s runtime from a variety of practical viewpoints

CSCI 232 vs. CSCI 332

CSCI 332. Design and analysis of algorithms.

- translate natural-language descriptions of computational problems into precisely formulated computational problems
- recognize various real-world computational problems as instances of known computational problems
- identify valid inputs and correct outputs to computational problems
- trace through algorithm execution
- argue the correctness of an algorithm using a variety of proof techniques (direct proof, proof by induction, etc.)
- give counterexamples showing that an algorithm is incorrect
- correctly describe an algorithm’s runtime from a variety of practical viewpoints

$$\begin{aligned}
\sum_{i=1}^{n} \sum_{j=i+1}^{n} \frac{2}{j-i-1} &= 2\sum_{i=1}^{n} \sum_{j=2}^{n-i+1} \frac{1}{j} \\
&\leq 2n \sum_{j=1}^{n} \frac{1}{j} \\
&\sim 2n \int_{x=1}^{n} \frac{1}{x} dx \\
&= 2n \ln n
\end{aligned}$$

CSCI 232 vs. CSCI 332

CSCI 332. Design and analysis of algorithms.

- translate natural-language descriptions of computational problems into precisely formulated computational problems
- recognize various real-world computational problems as instances of known computational problems
- identify valid inputs and correct outputs to computational problems
- trace through algorithm execution
- argue the correctness of an algorithm using a variety of proof techniques (direct proof, proof by induction, etc.)
- give counterexamples showing that an algorithm is incorrect
- correctly describe an algorithm's runtime from a variety of practical viewpoints

Emphasizes critical thinking, problem-solving, and both open-ended problems and rigorous analysis.

CSCI 232 vs. CSCI 332

CSCI 332. Design and analysis of algorithms.

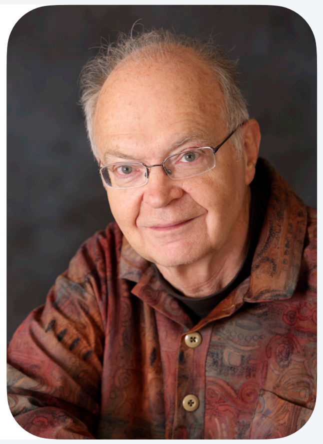
- translate natural-language descriptions of computational problems into precisely formulated computational problems
- recognize various real-world computational problems as instances of known computational problems
- identify valid inputs and correct outputs to computational problems
- trace through algorithm execution
- argue the correctness of an algorithm using a variety of proof techniques (direct proof, proof by induction, etc.)
- give counterexamples showing that an algorithm is incorrect
- correctly describe an algorithm's runtime from a variety of practical viewpoints

$$\begin{aligned} \sum_{i=1}^{n} \sum_{j=i+1}^{n} \frac{2}{j-i-1} &= 2\sum_{i=1}^{n} \sum_{j=2}^{n-i+1} \frac{1}{j} \\ &\leq 2n \sum_{j=1}^{n} \frac{1}{j} \\ &\sim 2n \int_{x=1}^{n} \frac{1}{x} dx \\ &= 2n \ln n \end{aligned}$$

Emphasizes critical thinking, problem-solving, and both open-ended problems and rigorous analysis.

Why study algorithms?

" *Algorithms are the life-blood of computer science… the common denominator that underlies and unifies the different branches.* " — *Donald Knuth*



Course logistics

In table groups, try to complete the syllabus quiz. Some of the questions are open-ended and may not have one single answer!

If your group comes up with a question you can’t answer (not necessarily one on the quiz), post it in #questions in Discord.

Matching med-school students to hospitals

How to match? What should we think about when designing an algorithm for this problem?

Matching med-school students to hospitals

Input: a set of preferences among hospitals and med-school students

favorite ↓ least favorite ↓

	1st	2nd	3rd
Atlanta	Xavier	Yolanda	Zeus
Boston	Yolanda	Xavier	Zeus
Chicago	Xavier	Yolanda	Zeus

hospitals' preference lists

favorite ↓ least favorite ↓

	1st	2nd	3rd
Xavier	Boston	Atlanta	Chicago
Yolanda	Atlanta	Boston	Chicago
Zeus	Atlanta	Boston	Chicago

students' preference lists

Matching med-school students to hospitals

Input: a set of preferences among hospitals and med-school students

favorite ↓ least favorite ↓

	1st	2nd	3rd
Atlanta	Xavier	Yolanda	Zeus
Boston	Yolanda	Xavier	Zeus
Chicago	Xavier	Yolanda	Zeus

hospitals' preference lists

favorite ↓ least favorite ↓

	1st	2nd	3rd
Xavier	Boston	Atlanta	Chicago
Yolanda	Atlanta	Boston	Chicago
Zeus	Atlanta	Boston	Chicago

students' preference lists

Output: a (perfect) matching of hospitals to students…

Matching med-school students to hospitals

Input: a set of preferences among hospitals and med-school students

favorite ↓ least favorite ↓

	1st	2nd	3rd
Atlanta	Xavier	Yolanda	Zeus
Boston	Yolanda	Xavier	Zeus
Chicago	Xavier	Yolanda	Zeus

hospitals' preference lists

favorite ↓ least favorite ↓

	1st	2nd	3rd
Xavier	Boston	Atlanta	Chicago
Yolanda	Atlanta	Boston	Chicago
Zeus	Atlanta	Boston	Chicago

students' preference lists

Output: a (perfect) matching of hospitals to students…

{ Atlanta–Zeus, Boston–Yolanda, Chicago–Xavier }

Matching med-school students to hospitals

Input: a set of preferences among hospitals and med-school students

favorite ↓ least favorite ↓

	1st	2nd	3rd
Atlanta	Xavier	Yolanda	Zeus
Boston	Yolanda	Xavier	Zeus
Chicago	Xavier	Yolanda	Zeus

hospitals' preference lists

favorite ↓ least favorite ↓

	1st	2nd	3rd
Xavier	Boston	Atlanta	Chicago
Yolanda	Atlanta	Boston	Chicago
Zeus	Atlanta	Boston	Chicago

students' preference lists

Output: a (perfect) matching of hospitals to students…

{ Atlanta–Zeus, Boston–Yolanda, Chicago–Xavier }

…that is self-reinforcing: no mutually beneficial side deals!

Matching med-school students to hospitals

Input: a set of preferences among hospitals and med-school students

favorite ↓ least favorite ↓

	1st	2nd	3rd
Atlanta	Xavier	Yolanda	Zeus
Boston	Yolanda	Xavier	Zeus
Chicago	Xavier	Yolanda	Zeus

hospitals' preference lists

favorite ↓ least favorite ↓

	1st	2nd	3rd
Xavier	Boston	Atlanta	Chicago
Yolanda	Atlanta	Boston	Chicago
Zeus	Atlanta	Boston	Chicago

students' preference lists

Output: a (perfect) matching of hospitals to students…

{ Atlanta–Zeus, Boston–Yolanda, Chicago–Xavier }

…that is self-reinforcing: no mutually beneficial side deals!

With your table: can any hospitals/students that are not currently matched make a mutually beneficial deal?

Matching med-school students to hospitals

Input: a set of preferences among hospitals and med-school students

favorite ↓ least favorite ↓

	1st	2nd	3rd
Atlanta	Xavier	Yolanda	Zeus
Boston	Yolanda	Xavier	Zeus
Chicago	Xavier	Yolanda	Zeus

hospitals' preference lists

favorite ↓ least favorite ↓

	1st	2nd	3rd
Xavier	Boston	Atlanta	Chicago
Yolanda	Atlanta	Boston	Chicago
Zeus	Atlanta	Boston	Chicago

students' preference lists

Output: a (perfect) matching of hospitals to students…

{ Atlanta–Zeus, Boston–Yolanda, Chicago–Xavier }

…that is self-reinforcing: no mutually beneficial side deals!

With your table: can any hospitals/students that are not currently matched make a mutually beneficial deal?

On your syllabus quiz, tally up how many at your table think:

1) a stable matching always exists
2) there is an optimally efficient algorithm for this problem