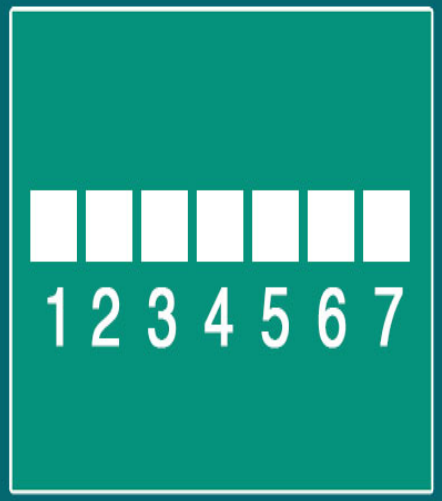
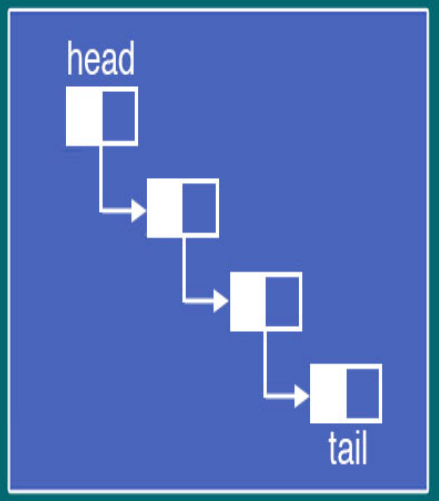
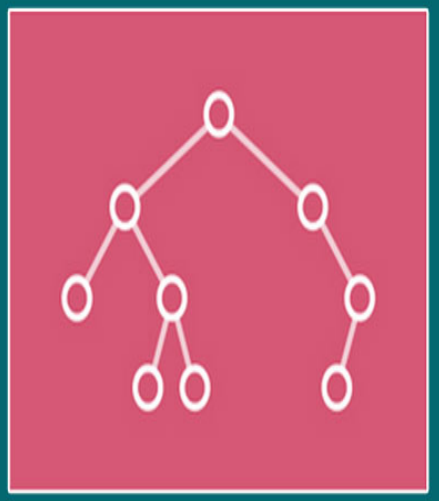
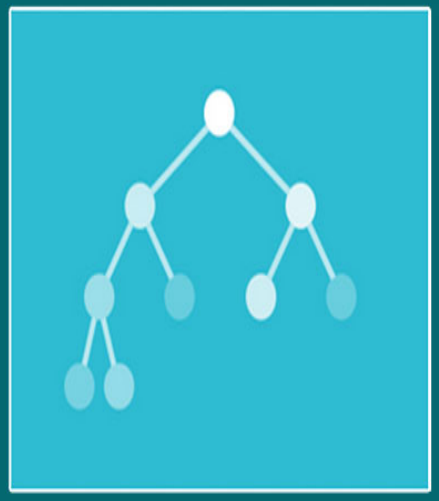
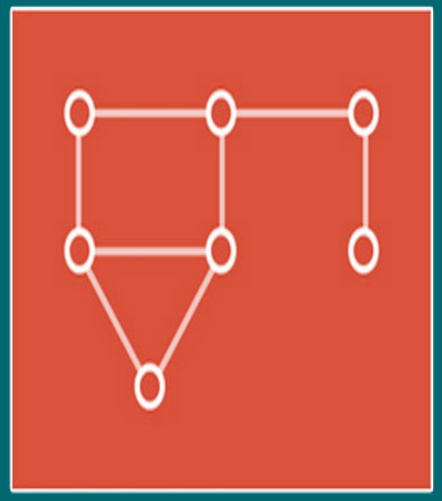


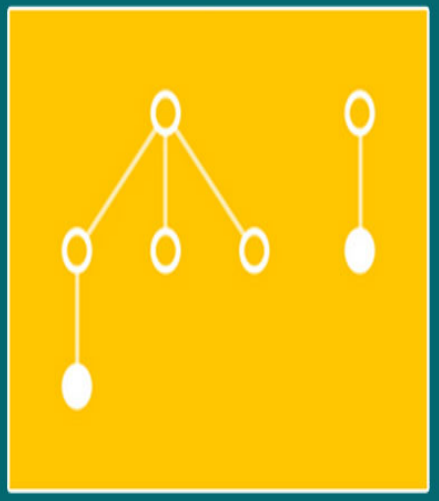


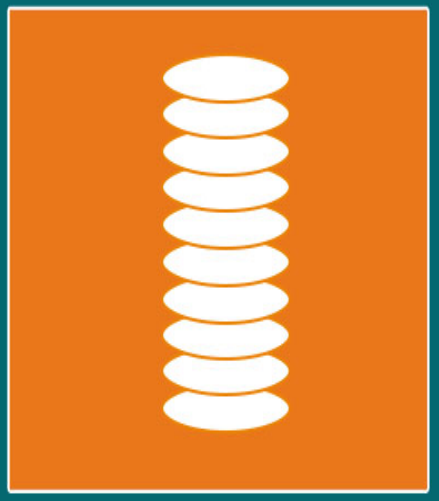


Data Structures and Algorithms









CMPSC 465

PennState College of Engineering

ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

Instructors:

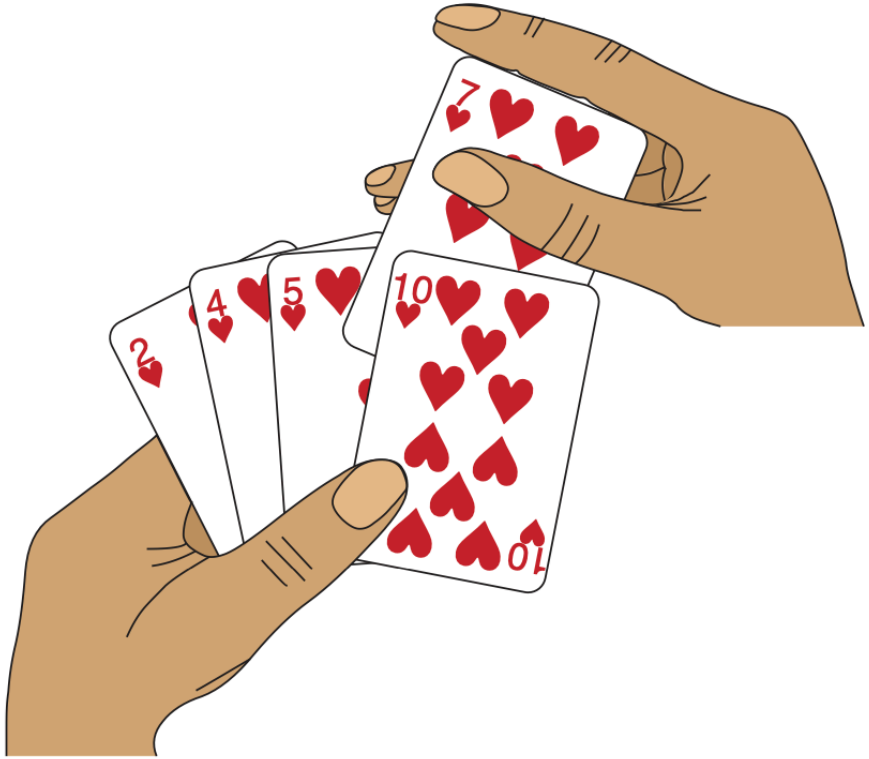
Mehrdad Mahdavi
Ph.D. in CS
Research: Machine Learning & Randomized Algorithms

Office hours: Wednesday 9:00 am - 10:00 am
Westgate W365

David Koslicki
Ph.D. in MATH
Research: Computational Biology, Theoretical Computer Science

Office hours: Wednesday 9:00 am - 10:00 am
Westgate W205C

Sorting & searching



Database Management

Information Retrieval

Data Analysis and Statistic

File System Organization

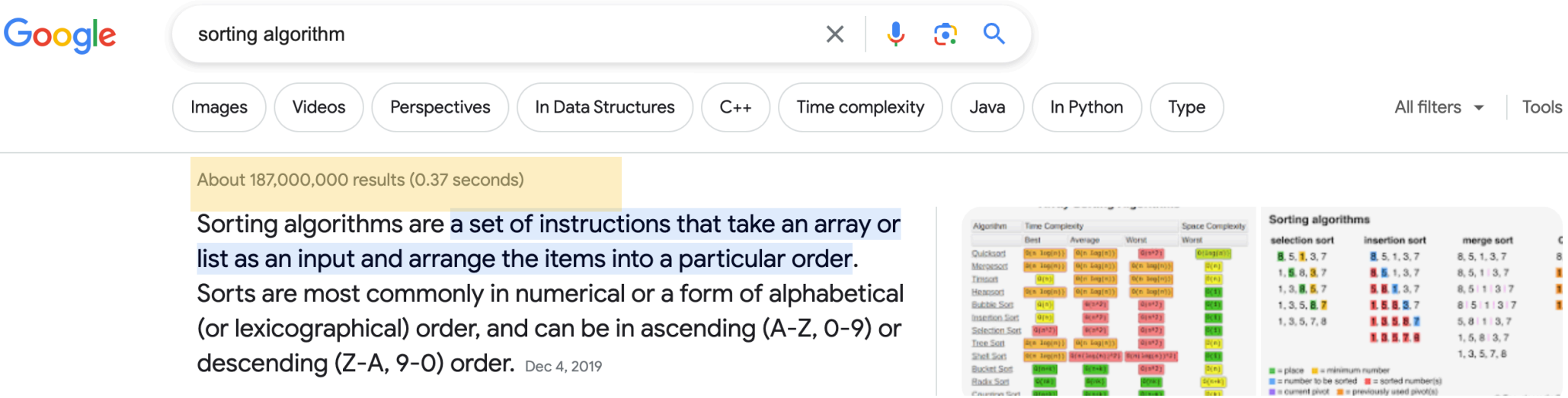
Task Scheduling

Web Search Engines

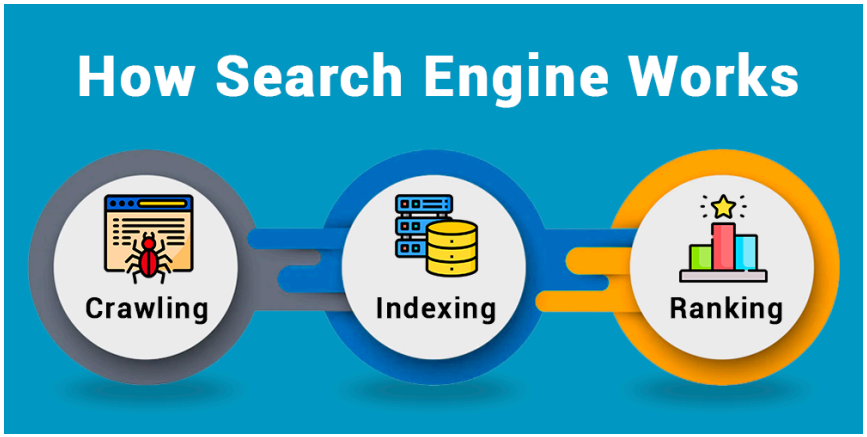
Network Routing

Artificial Intelligence & Game Playing

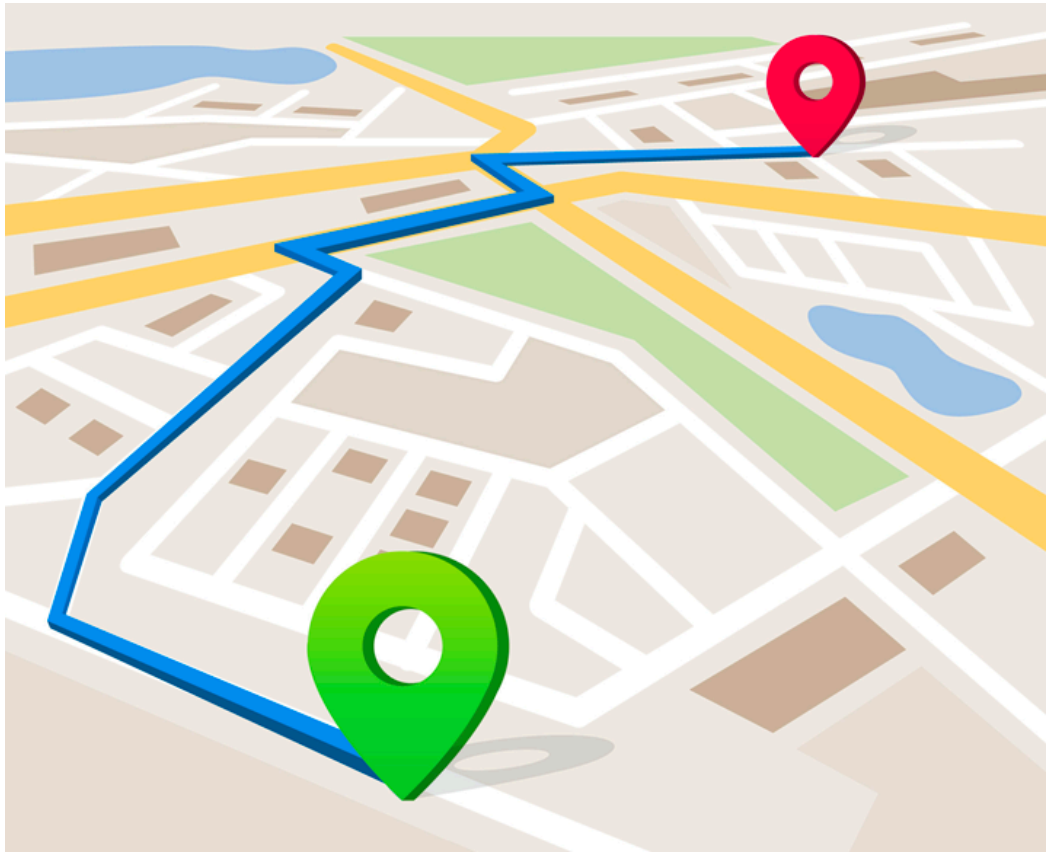
Search engines



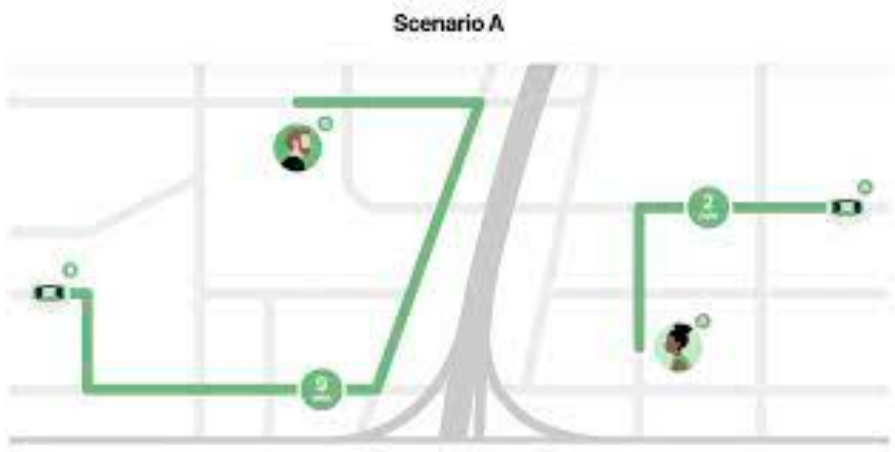
Search engines like Google or Bing rely heavily on data structures and algorithms to efficiently index and retrieve vast amounts of information. Algorithms like PageRank determine the relevance and ranking of web pages, while data structures such as inverted indexes enable quick keyword-based searches.

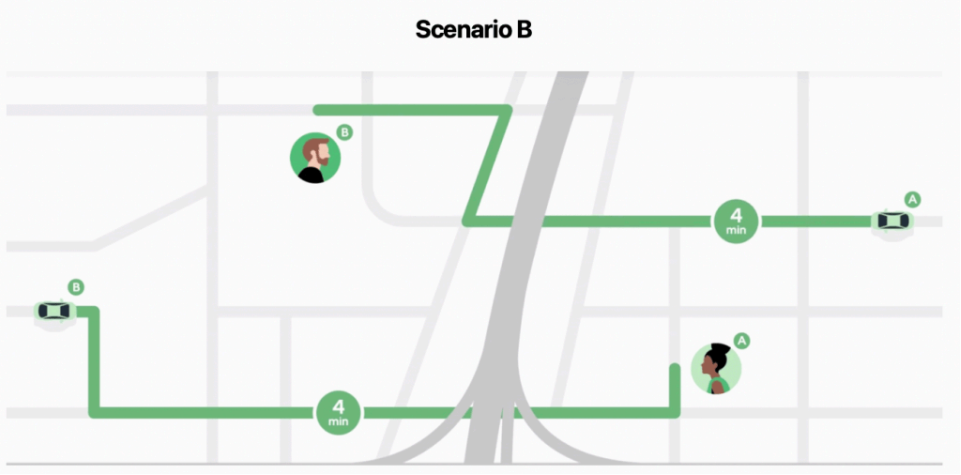


Transportation



Matching

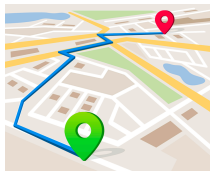


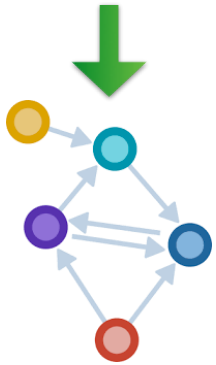


Three pillars

Three important problem-solving pillars:

- Data models, the abstractions used to describe problems
  (graphs, trees, lists, sequences, algebra, logic, etc)

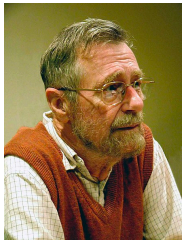




- Data structures, the programming-language constructs used to represent data models
  (arrays, stacks, queues, AVL trees, etc)

$$\begin{pmatrix} 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{pmatrix}$$

- Algorithms, a precise and unambiguous specification of a sequence of steps to obtain solutions by manipulating data as represented by the abstractions of a data model, by data structures, or by other means (dynamic programming, greedy algorithms, divide & conquer, etc)



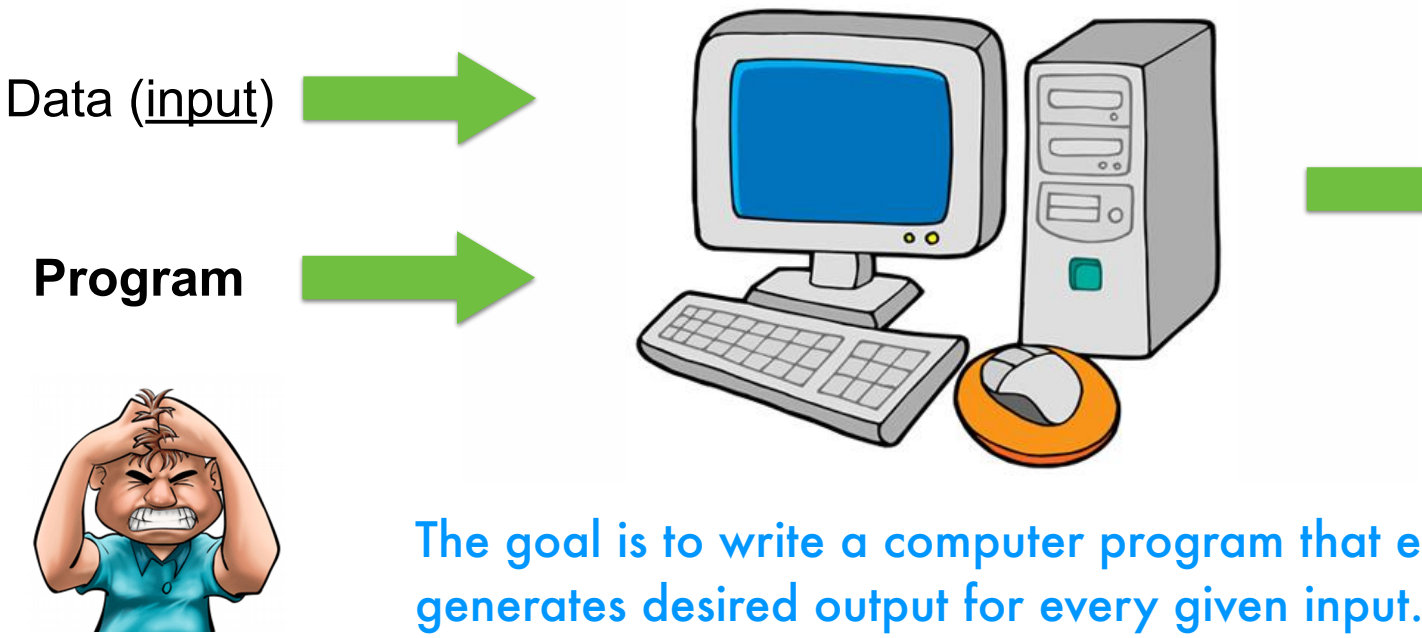
Dijkstra's algorithm

Algorithmic problem solving

We are given a well defined description of a problem to map a given input to its corresponding output!

Input-output relations are known and well defined!

E.g., sorting, shortest path, matching, scheduling, etc.



The goal is to write a computer program that efficiently generates desired output for every given input.

CMPSC 465 Data Structures & Algorithms

Algorithms: Finite set of unambiguous step-by-step instructions to solve a problem

Informally, an **algorithm** is any well-defined computational procedure that takes some value, or set of values, as **input** and produces some value, or set of values, as **output** in a finite amount of time. An algorithm is thus a sequence of computational steps that transform the input into the desired output.

Examples:

1. Algorithms for addition, subtraction, multiplication and division
2. Sorting numbers (bubble sort, insertion sort, quicksort, etc.)
3. Finding shortest route in a Map

Data Structures: Ways of storing and manipulating information/data

Using the appropriate data structure(s) is an important part of algorithm design

Examples: Arrays, Linked Lists, Hash tables, Heaps, Trees, etc.

Algorithms have a long history...

Division
Babylonia

2500 BC

Arithmetic
Egypt

1550 BC

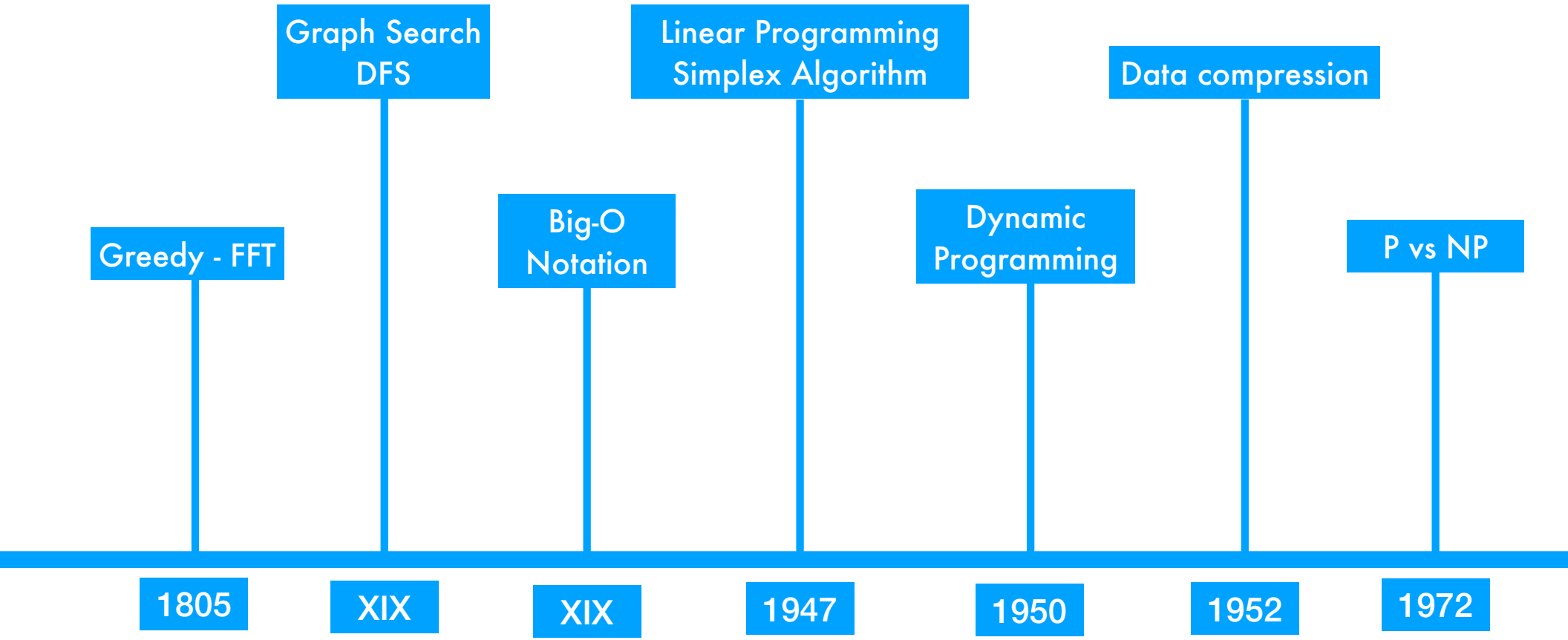
Euclid's algorithm
Greece

300 BC

Arithmetic with Indo-Arabic numbers
Al-Khwārizmī *(Algorithmi)*

825 AD

In this class: classical influential algorithms



Why study algorithms and data structures?

- Analyzing correctness and resource usage (e.g. memory)
- Feasibility (what can and cannot be done efficiently)
  - NP-completeness
- Have lead the way in the IT revolution (Google, Mapquest, Akamai)
- Computation is fundamental to understanding the world
  - social networks, brains, black holes, evolution
- **Interviews!**
- **Lots of FUN!**

Course objectives:

1. Classical algorithms
2. Analysis of algorithms: correctness, running time and space/memory utilization
3. Standard design techniques:

   divide-and-conquer, greedy, dynamic programming, etc.
4. Design algorithms using the building blocks presented in this class
5. Formulate and use appropriate and efficient data structures
6. Introduction the Complexity Theory

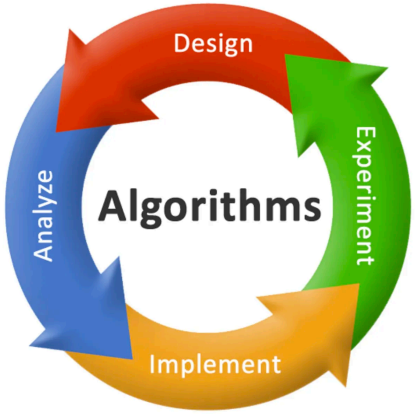
Course objectives:

Theoretical study: **proofs** of correctness and performance

1. Classical algorithms
2. Analysis of algorithms: correctness, running time and space/memory utilization
3. Standard design techniques:

   divide-and-conquer, greedy, dynamic programming, etc.
4. Design algorithms using the building blocks presented in this class
5. Formulate and use appropriate and efficient data structures
6. Introduction the Complexity Theory

Three questions will drive us:



1. Is the algorithm/data structure correct?
2. Is it efficient? (time and space)
3. Can we do better?

If computers were infinitely fast and computer memory were free, would you have any reason to study algorithms? YES! We still need to be certain that our solution method terminates and does so with the correct answer. But computing time is a bounded resource, which makes it precious.

Other important considerations: functionality, robustness, energy efficiency, modularity, maintenance times, etc. are beyond the scope of this course.

How to measure running times?

Issue: The running time may depend on input size

Idea: Parametrize the running time by the size of the input

Worst-case analysis: (usually)

- $T(n)$ = maximum time of algorithm on any input of size n

Average-case: (sometimes)

- $T(n)$ = expected time of algorithm over all inputs of size n
- Requires assumption about distribution of inputs

Amortized: (sometimes)

- $T(m)$ = total time over m steps (operations)/ m

- **Best-case:** (never)
- Cheat with a slow algorithm that works well on some inputs

CMPSC 465: class organization/staff

- Two Lecture Sections:
  - 001 - Tu Th 12:05PM - 1:20PM (Keller Bldg 104)
  - 002 - Tu Th 3:05PM - 4:20PM (Keller Bldg 104)
  - Same staff, same assignments. Will essentially run as one big course!
- Two Instructors (Dr. Mahdavi & Dr. Koslicki)
- Nine Teaching Assistants
- 10 - 12 Graders
- 350+ students
- Eight recitation sections (Wednesdays) [please check syllabus]

Teaching Assistants:

Qiyu Chen (Recitation TA)
Vishnu Dasu (Head TA)
Sun Tingyang (Recitation TA)
Ayush Goyal (Head TA)
Neha Rathod (Recitation TA)
Fatemeh Rahbari (Recitation TA)
Sujeeth Reddy Vummanthala (Recitation TA)
Siyuan Hong (Recitation TA)

Brandon Edmunds (Content TA)

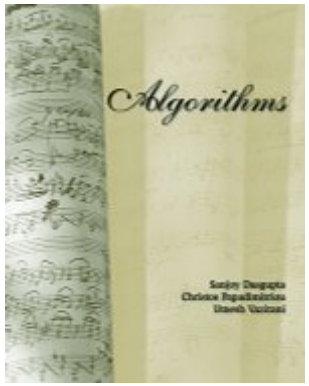
TA's office hours TBA!

COVID 19 or other illnesses policies:

If you are sick:

- Focus on your health and on getting better. Do not stress out about the class!
- You can skip up to two HW's without penalty (we will automatically drop two lowest scores at the end of semester)
- You can skip up to three quizzes without penalty (we will automatically drop three lowest scores)
- We will provide recordings for all lectures so that you can keep up
- No need for makeup HW's or quizzes (Not possible per class policy)
- If your symptoms last longer than three or four weeks, feel free to reach out to us for further accommodations

Course Info:



Textbook: Algorithms (Dasgupta, Papadimitriou and Vazirani)

Class Resources:

- Gradescope (for turning in your homework)
- Canvas (for class resources)
- Campuswire (for all communications)

Recitations:

- On Wednesdays
- Attendance highly encouraged
- Short quiz will be given each time (13% of your grade - we will drop 3)
- Handout with carefully selected problems

Homeworks:

- Problem set posted every Thursday evening (after the current HW is due)
- Due the following Thursday at 10 pm
- Submit through Gradescope **(Class code on Canvas)**
- Homework accounts for 13% of your grade
- We will drop the two HWs with the lowest score, so no exceptions on deadlines
- **Check the syllabus for important details** (e.g., submitting/formatting)

Collaboration:

- You are welcome to work on your own or in groups of up to 5 students
- **Every student most write her/his own solutions in her/his own words! — Important: most common AI violation in this course.**
- All collaborators & references should be acknowledged

Exams:

- Three midterm exams dropping the lowest score - 27% of your grade each
    - Midterm 1 (02/08)- in class
    - Midterm 2 (03/14)- in class (non-cumulative)
    - Midterm 3 (04/11) - in class (non-cumulative)
- One cumulative Final Exam - 20% of your grade

Getting help QA:

- Homework questions? Campuswire
- …other questions? Campuswire
- Course logistics? Campuswire and your recitation's TA
- Re-grading? Gradescope
- Contacting the Instructors? Campuswire or office hours

  Email responses from the Instructors may be slow due to the size of class

- Instructors' office hours: For major concerns and high-level questions
- TA's office hours: You are welcome to attend any/all. Homework questions OK.