Lecture 1

CS 161 Design and Analysis of Algorithms
Ioannis Panageas
Course staff

Instructor: Ioannis Panageas
Email: ipanagea at ics dot uci dot edu
Office hours: Wednesday 3:00-4:00pm

TA: Will Overman
Email: overmana at uci dot edu
Office hours: Tuesday 2:30-3:30pm

TA: Stelios Stavroulakis
Email: sstavrou at uci dot edu
Office hours: Wednesday 1:00-2:00pm
Course material

We will use canvas for announcements and homeworks. Slide materials will be posted on
https://panageas.github.io/algo2022/

We will be using Piazza for questions of general interest about the course material, the homework, and the tests
https://piazza.com/uci/spring2022/compsci161

Required Textbook
• Algorithm Design and Applications, by M. T. Goodrich and R. Tamassia.

Recommended Textbook
Grading

• **Homeworks**: 20%
  – There will be given 4 Homeworks to solve (**+5% bonus for using LaTeX!**).

• **Midterms**: 40%
  – There will be given 2 midterms, on Thursdays of week 5 and 9. Each midterm will contain topics from all taught previous weeks.

• **Final**: 40%
  – Material from all weeks.

**+1% bonus for Course Evaluation**
Letter Grades

- **Not** a straight scale nor straight curve
- 90% and up guaranteed some sort of A or A-
- 80% and up guaranteed at least B-
- 70% and up guaranteed at least C-
Submitting Assignments

- **Written assignments in Canvas**
  - Must be legible
    - If you have messy handwriting, **type** your homework!
    - **Bonus 5% for Latex!**
  - Must be **on-time**.
  - Deadline: Fridays 23:59pm (see syllabus)

- **Programming assignments in Gradescope**
  - Code must be in python and need to pass test cases
Exam Dates and Rules

- The exams are held on the **days listed (syllabus)**
  - See policy in syllabus for makeup exams
- Exams will not be excused for reasons within your control
Academic Integrity Policy

• If you need help, see:
  ○ Ioannis
  ○ Will or Stelios

• Plagiarism risks an F in the class and more.

• The following are examples of not okay:
  ○ Chegg GeeksForGeeks
  ○ CourseHero Quora
  ○ StackOverflow Github (generally)
Collaboration with classmates

● You can discuss some things freely with others:
  ○ What a problem is asking
  ○ How to do a non-homework or non-exam problem
  ○ How something from lecture worked
● You should never:
  ○ Show your homework assignment to someone else
  ○ Write your solutions from notes taken outside lecture / discussion
  ○ Seek homework solutions from outside sources -- especially online!
  ○ Tell a student specifically how to solve a homework problem
● Penalty for academic dishonesty: F in the course.
Commercial Note Taking

- It is **prohibited** to be **paid** to take notes
- It is **prohibited** to **sell** your notes from this class
- **Do not upload course materials**
  - Do not upload handouts
  - Do not upload returned exams
  - Do not upload lecture slides
- **Violations are violations** of student conduct code
To-Do This Week

● Read the syllabus
  ○ Treat it as though it’s a reading assignment.
  ○ Main document plus associated policy documents

● Review Prerequisites
  ○ Help is available all week, including at all discussion sections

● Programming Assignment 0
  ○ Get familiar with Gradescope
What is algorithm

- Algorithm is a procedure for solving a task

  e.g. how do you sort a cart of books in increasing order of the volume number? (i.e. volume 1, volume 2, volume 3....)

  - Bad algorithm: compare all books, put smallest volume in the beginning and repeat.
  
  - Clever algorithm: divide the cart into two, sort the first half, sort the second half, merge them.

Design and Analysis of Algorithms
What is algorithm

• Algorithm is a procedure for solving a task

e.g. How to find the best travelling time between from a station to any other station?

- Bad algorithm: manually find the travelling between each station.
- Clever algorithm: just record the travelling time between consecutive stations, then use the Dijkstra shortest path algorithm.
Case study: Finding a Celebrity

Since coming to UC Irvine, has anyone met a celebrity?
What is a celebrity?

- Within a group of people $G$, we say a person $p$ is a celebrity (famous) if:
  - Everyone knows who $p$ is
    - (celebrities must be known by everyone)
  - Person $p$ does not know who anyone else is

- **Goal**: Find a celebrity from $G$ if there exists one.
What is a celebrity?

- **Within a group of people** $G$, we say a person $p$ is a **celebrity** (famous) if:
  - Everyone knows who $p$ is
    - (celebrities must be known by everyone)
  - Person $p$ does not know who anyone else is

- **Goal**: Find a celebrity from $G$ if there exists one.

**Model the problem as a directed graph:**

- 0 knows 1, 0 knows 3, 0 knows 4
- 1 knows 2, 3 knows 2, 4 knows 3
Brute force approach

- Given a person $p$ we want to check if it is a celebrity
  - How efficiently can I check if person $p$ is a celebrity?
Brute force approach

- Given a person $p$ we want to check if it is a celebrity
  - How efficiently can I check if person $p$ is a celebrity?
  - Check all other group members if they know $p$ and also if $p$ does not know them.
Brute force approach

• Given a person \( p \) we want to check if it is a celebrity
  ○ How efficiently can I check if person \( p \) is a celebrity?
  ○ Check all other group members if they know \( p \) and also if \( p \) does not know them.

This gives \( 2n - 2 \) ”checks” where \( n \) is the group size.
Brute force approach

Given a person $p$ we want to check if it is a celebrity

- How efficiently can I check if person $p$ is a celebrity?
- Check all other group members if they know $p$ and also if $p$ does not know them.

This gives $2n - 2$ ”checks” where $n$ is the group size.

- We have to do the above for all possible persons $p$. 

Design and Analysis of Algorithms
Brute force approach

- Given a person $p$ we want to check if it is a celebrity
  - How efficiently can I check if person $p$ is a celebrity?
  - Check all other group members if they know $p$ and also if $p$ does not know them.

This gives $2n - 2$ ”checks” where $n$ is the group size.

- We have to do the above for all possible persons $p$.

Total ”checks” are $(2n - 2) \cdot n$ which gives $\Theta(n^2)$.
Brute force approach

- Given a person $p$ we want to check if it is a celebrity
  - How efficiently can I check if person $p$ is a celebrity?
  - Check all other group members if they know $p$ and also if $p$ does not know them.

  This gives $2n - 2$ ”checks” where $n$ is the group size.

- We have to do the above for all possible persons $p$.

  Total ”checks” are $(2n - 2) \cdot n$ which gives $\Theta(n^2)$.
Faster approach

- Put all the members in a list (arbitrary order)
  - Pick the first two members of the list, let $p, q$.
  - Check if $p$ knows $q$. 
Faster approach

- Put all the members in a list (arbitrary order)
  - Pick the first two members of the list, let $p, q$.
  - Check if $p$ knows $q$.

2 Cases:
1. $p$ knows $q$. Then $p$ is not a celebrity (remove $p$ from the list).
2. $p$ does not know $q$. Then $q$ is not a celebrity (remove $q$ from the list).
Faster approach

- Put all the members in a list (arbitrary order)
  - Pick the first two members of the list, let $p$, $q$.
  - Check if $p$ knows $q$.

2 Cases:
1. $p$ knows $q$. Then $p$ is not a celebrity (remove $p$ from the list).
2. $p$ does not know $q$. Then $q$ is not a celebrity (remove $q$ from the list).

- Repeat the above process. At every iterate, we remove one person.
Faster approach

- Put all the members in a list (arbitrary order)
  - Pick the first two members of the list, let $p, q$.
  - Check if $p$ knows $q$.
  - Repeat the above process. At every iterate, we remove one person.

After $n - 1$ ”iterates” we have one member in the list.

Check if this remaining person is a celebrity.
Faster approach

- Put all the members in a list (arbitrary order)
  - Pick the first two members of the list, let \( p, q \).
  - Check if \( p \) knows \( q \).

  - Repeat the above process. At every iterate, we remove one person.

After \( n - 1 \) "iterates" we have one member in the list.

Check if this remaining person is a celebrity.

Total "checks" are \( 2n - 2 \) which gives \( \Theta(n) \).