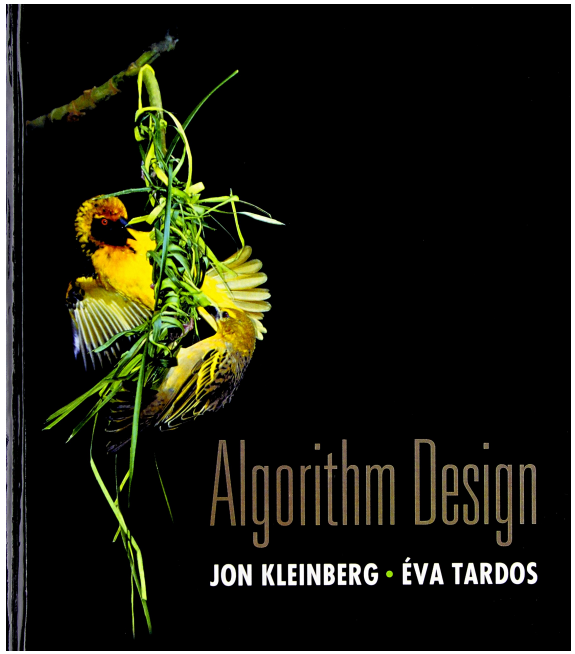
Basic Information

- Instructor: Hu Fu
- Office: ICICS-X539. Email: hufu@cs.ubc.ca
- Course website: http://fuhuthu.com/CPSC420F2019/index.html
  - Contains links to Piazza and Gradescope.
- Teaching Assistants:
  - Iliad Ramezani
  - Abner Turkieltaub abner7@cs.ubc.ca
  - Liam Vanderpoel liam1tvu@students.cs.ubc.ca

Course material

- We will cover selected topics from Chapters 7, 8, 11 and 13 from the textbook **Algorithm Design** by Kleinberg and Tardos.
- The textbook is required. It is the same one as required in CPSC 320.



Prerequisites

- Prerequisite: CPSC 320. You should be proficient with:
  - Asymptotic running time analysis (including big $O(\cdot)$ notations);
  - Basic data structures (stacks, queues, graphs etc.);
  - Basic graph algorithms (e.g. DFS, BFS, minimum spanning trees, shortest paths).

Prerequisites

- Prerequisite: CPSC 320. You should be proficient with:
  - Asymptotic running time analysis (including big $O(\cdot)$ notations);
  - Basic data structures (stacks, queues, graphs etc.);
  - Basic graph algorithms (e.g. DFS, BFS, minimum spanning trees, shortest paths).
- We briefly review some basic graph algorithms as warm up, but this should not be your first time seeing them.

Prerequisites

- Prerequisite: CPSC 320. You should be proficient with:
  - Asymptotic running time analysis (including big $O(\cdot)$ notations);
  - Basic data structures (stacks, queues, graphs etc.);
  - Basic graph algorithms (e.g. DFS, BFS, minimum spanning trees, shortest paths).
- We briefly review some basic graph algorithms as warm up, but this should not be your first time seeing them.
- Familiarity with basic probability theory will be helpful for Chapter 13. We will provide the basics.

Prerequisites

- Prerequisite: CPSC 320. You should be proficient with:
  - Asymptotic running time analysis (including big $O(\cdot)$ notations);
  - Basic data structures (stacks, queues, graphs etc.);
  - Basic graph algorithms (e.g. DFS, BFS, minimum spanning trees, shortest paths).
- We briefly review some basic graph algorithms as warm up, but this should not be your first time seeing them.
- Familiarity with basic probability theory will be helpful for Chapter 13. We will provide the basics.
- Let me know if we assumed in class something you haven't seen or have forgotten.

Course work

- Problem sets (30%) + Midterm (40%) + Final (30%)
- Midterm dates:
  - Oct 8, Tuesday, 7:30-9pm.
  - Nov 5, Tuesday, 7:30-9pm.
- Midterm location: DMP 310.

Course work

- Problem sets (30%) + Midterm (40%) + Final (30%)
- Midterm dates:
  - Oct 8, Tuesday, 7:30-9pm.
  - Nov 5, Tuesday, 7:30-9pm.
- Midterm location: DMP 310.
- All exams are open book.

Course work

- Problem sets (30%) + Midterm (40%) + Final (30%)
- Midterm dates:
  - Oct 8, Tuesday, 7:30-9pm.
  - Nov 5, Tuesday, 7:30-9pm.
- Midterm location: DMP 310.
- All exams are open book.
- We accommodate reasonable conflicts with the midterm times. Let me know ASAP.

Homework Policies

- There is a problem set roughly every two weeks.

Homework Policies

- There is a problem set roughly every two weeks.
- You are encouraged to form groups of up to three people for each assignment. Each group turns in only one solution. Every member of the group must completely understand the solution turned in.

Homework Policies

- There is a problem set roughly every two weeks.
- You are encouraged to form groups of up to three people for each assignment. Each group turns in only one solution. Every member of the group must completely understand the solution turned in.
- Group members are encouraged to discuss their approaches and polish each other's writing. "Division of labor" is bad for your practice. *Everyone should make an attempt at every problem.*

Homework Policies

- There is a problem set roughly every two weeks.
- You are encouraged to form groups of up to three people for each assignment. Each group turns in only one solution. Every member of the group must completely understand the solution turned in.
- Group members are encouraged to discuss their approaches and polish each other's writing. "Division of labor" is bad for your practice. *Everyone should make an attempt at every problem.*
- Some problems are more challenging than others. Discussion is welcome, including during office hours. Start thinking about the problems early; don't wait till the last day or two. Allow yourself time to think and to seek help.

Homework Policies cont.

- If, by the end of the semester, the majority of your assignments are typeset with LaTeX, your lowest assignment mark will be dropped. (A LaTeX template is provided on course website.)

Homework Policies cont.

- If, by the end of the semester, the majority of your assignments are typeset with LaTeX, your lowest assignment mark will be dropped. (A LaTeX template is provided on course website.)
- We use Gradescope for assignments. Entry code: 9ZJ5RJ. **If you form a group, make sure to submit as a group.**

Homework Policies cont.

- If, by the end of the semester, the majority of your assignments are typeset with LaTeX, your lowest assignment mark will be dropped. (A LaTeX template is provided on course website.)
- We use Gradescope for assignments. Entry code: 9ZJ5RJ. **If you form a group, make sure to submit as a group.**
- **If you work with someone outside your group or use some outside source, you must acknowledge them in your write-up.**

Difficulty and Curving

- Besides a few basic problems (drills) and challenging ones (often given as bonus questions), we try to assign most problems a little beyond your comfort zone.

Difficulty and Curving

- Besides a few basic problems (drills) and challenging ones (often given as bonus questions), we try to assign most problems a little beyond your comfort zone.
- Problem solving is like sports. One needs progressive training.

Difficulty and Curving

- Besides a few basic problems (drills) and challenging ones (often given as bonus questions), we try to assign most problems a little beyond your comfort zone.
- Problem solving is like sports. One needs progressive training.
- Occasionally the problems may be on the hard side, then we curve the grades according to class statistics.

Difficulty and Curving

- Besides a few basic problems (drills) and challenging ones (often given as bonus questions), we try to assign most problems a little beyond your comfort zone.
- Problem solving is like sports. One needs progressive training.
- Occasionally the problems may be on the hard side, then we curve the grades according to class statistics.
- We publish class performance statistics throughout the semester.

More on assignments and exams

- This course is "proof based." If a question asks you to design an algorithm, you should prove the correctness of your algorithm, unless the problem states otherwise.

More on assignments and exams

- This course is "proof based." If a question asks you to design an algorithm, you should prove the correctness of your algorithm, unless the problem states otherwise.
- When the question asks for a certain running time (e.g. polynomial time, or $O(n^2)$), you should analyze the running time of your algorithm, unless the problem states otherwise.

More on assignments and exams

- This course is "proof based." If a question asks you to design an algorithm, you should prove the correctness of your algorithm, unless the problem states otherwise.
- When the question asks for a certain running time (e.g. polynomial time, or $O(n^2)$), you should analyze the running time of your algorithm, unless the problem states otherwise.
- You are almost never asked to write pseudo-code. Use it as a last resort only if you can't express your ideas in other ways. (It suffices to say, e.g., perform BFS.)

More on assignments and exams

- This course is "proof based." If a question asks you to design an algorithm, you should prove the correctness of your algorithm, unless the problem states otherwise.
- When the question asks for a certain running time (e.g. polynomial time, or $O(n^2)$), you should analyze the running time of your algorithm, unless the problem states otherwise.
- You are almost never asked to write pseudo-code. Use it as a last resort only if you can't express your ideas in other ways. (It suffices to say, e.g., perform BFS.)
- "Talk to humans" rather than "talk to compilers".

More on assignments and exams

- This course is "proof based." If a question asks you to design an algorithm, you should prove the correctness of your algorithm, unless the problem states otherwise.
- When the question asks for a certain running time (e.g. polynomial time, or $O(n^2)$), you should analyze the running time of your algorithm, unless the problem states otherwise.
- You are almost never asked to write pseudo-code. Use it as a last resort only if you can't express your ideas in other ways. (It suffices to say, e.g., perform BFS.)
- "Talk to humans" rather than "talk to compilers".
- In one word, train yourself in clear writing (and speaking) when communicating mathematical ideas.

How to do well in this course

- Focus on the ideas, structures, and the way we approach problems, rather than the end results.

How to do well in this course

- Focus on the ideas, structures, and the way we approach problems, rather than the end results.
- Grasp mathematical definitions and statements rigorously.

How to do well in this course

- Focus on the ideas, structures, and the way we approach problems, rather than the end results.
- Grasp mathematical definitions and statements rigorously.
- Do exercises. Memorizing the textbook and slides cannot replace solving a few problems.