St. Francis Xavier University
Department of Computer Science

CSCI 541: Theory of Computing
Group Lecture and Report
Fall 2023

The group lecture and report are major assessments of this course. These assessments consist of three components: a lesson plan, the lecture itself, and an individual report. The group lecture is preferably completed in a group of four students, though *very limited* exceptions can be made if a group of four cannot be formed.

The lesson plan is worth a total of 10% of your final grade. The lecture component is worth a total of 20% of your final grade. The individual report is worth a total of 10% of your final grade. You must complete all of these components in order to pass the course.

Do not leave these assessments until the last minute! Start as soon as possible!

Lesson Plan

Due October 12, 2023 in lecture

The lesson plan is a short document meant to establish your group and your group's chosen lecture topic.

The lesson plan should include, at a minimum, the following information:

- The names of your group members
- One or two paragraphs giving a high-level introduction to your group's lecture topic
- A description of what each group member plans to contribute to the lecture
- A broad outline of the structure of your group's lecture (e.g., X minutes for introduction, Y minutes for background, etc.)
- A list of preliminary references that your group will use to create your lecture materials

A title page is not required.

When planning what each group member can contribute to the lecture, consider splitting the lecture topic into independent sections or subtopics and assigning one section or subtopic to each group member.

Existing survey articles or books are great preliminary references, as they often include a large amount of background information presented at an accessible level. If your group needs pointers to such preliminary references, let me know and I can try to help.

All references must be cited in a consistent style (e.g., IEEE citation style). Citations to articles and books are preferred over citations to lecture notes, Wikipedia, or websites.

Your group's lesson plan should be a 1–2 page double-spaced document written in 12-point text. This is a strict page limit, but references do not count toward the limit. The lesson plan will be marked in terms of completeness, organization, and adherence to guidelines.

Each group only needs to submit one lesson plan for all group members.

Lecture

Due date varies (last ∼3 weeks of lectures)

The lecture is a ∼50 minute presentation on your group's chosen topic. Your group will not be presenting for the entire length of time; you should budget ∼5 minutes for setup at the beginning and ∼5 minutes for questions and teardown at the end. Therefore, your actual lecture time will be closer to ∼40 minutes, and each group member should spend an approximately equal amount of time speaking (i.e., in a group of four, each group member speaks for ∼10 minutes).

Group members should take turns presenting so that no one is unfairly burdened with having to speak for the entire duration. This is an opportunity for individuals to use their strengths; for example, one person may cover the introduction, one person may work through an example, one person may choose to lead a live coding session, etc.

Your group's lecture should be presented at a level similar to other lectures in this course: targeted at a graduate student audience that is familiar with theoretical computer science. Your group can refer to past lecture notes as background material, but your group should ultimately develop your own notes or slides for use in the lecture itself.

Your group must submit your lecture materials (e.g., notes, slides, etc.) *at least one day before* the lecture is scheduled. Your lecture will be marked in terms of three broad categories: quality (e.g., focused on topic, targeted at an appropriate level), accuracy (e.g., correct summary of main results, material all related to topic), and organization (e.g., notes/slides are clearly written, lecture is well-paced).

All lecture material must be appropriately referenced, and citations must be included wherever necessary.

Lastly, remember that *quality is better than quantity.* A very well-produced lecture that takes ∼30 minutes is better to experience than a poorly-planned lecture that takes ∼40 minutes.

Report

Due December 5, 2023 in lecture

The report is an individual submission meant to complement the group lecture. In this report, you will write a brief survey of your lecture topic *in your own words* and give a short reflection on your group lecture experience.

On the first page of your report, you should include, at a minimum, the following information:

- The names of your group members and the topic of your lecture
- A list of the specific contributions you made in your group
- Approximately one paragraph discussing aspects of your lecture that you feel went well and aspects of your lecture that you feel could have been improved. Your opinion will not be shared with other group members.

In the rest of your report, you should include the following information:

- 3–4 pages summarizing your lecture topic, written in the style of a survey article or short lecture notes
- A complete list of references used in your report (not in your group lecture in general)

You must additionally include a title page that lists your name, report topic, and course details.

Note that your 3–4 page summary should be written *in your own words*; you must submit your own summary, and not one written collaboratively by all group members. If your group divided work by assigning specific sections or subtopics to individual members, this is a good opportunity to showcase your knowledge about the particular section or subtopic you focused on while designing your group lecture.

All references must be cited in a consistent style (e.g., IEEE citation style). Citations to articles and books are preferred over citations to lecture notes, Wikipedia, or websites.

Your report should be a 4–5 page double-spaced document written in 12-point text. This is a strict page limit, but references and the title page do not count toward the limit. The report will be marked primarily in terms of completeness, organization, and adherence to guidelines.

Since the report is due at the end of the term, the report deadline is firm. Late reports will not be accepted.

Suggested Topics

Below, I offer a selection of topics that your group can choose for your lecture. If your group wishes to study and present a topic related to this course that is not on this list, your group must check with me first to approve the proposed topic.

1. **Computational complexity of decision problems.**

   There are a number of natural decision problems we can ask about languages, and these problems may be either decidable or undecidable depending on which model of computation we're considering: finite automata, pushdown automata, linear-bounded automata, or Turing machines. The goal of this lecture will be to review known decidability results for each model of computation, and for those decision problems that are decidable, to discuss the computational complexity of those problems.

2. **The Immerman–Szelepscényi theorem.**

   In class, we discussed the statement of the Immerman–Szelepscényi theorem, but we did not give a proof of the theorem. The goal of this lecture will be to present a *complete* and *understandable* proof of the theorem, including all necessary background material. The lecture can also discuss the history of the two "LBA problems" as well as a discussion of progress on the as-yet-unsolved first LBA problem. Comfort with rigorous proofs and mathematical expertise are strongly recommended for this topic.

3. **PSPACE-complete decision problems.**

   In class, we proved that the TQBF problem was PSPACE-complete, and we mentioned that a number of other decision problems are also PSPACE-complete. The goal of this lecture is to survey examples of PSPACE-complete decision problems and to describe (at a high level) the proofs of PSPACE-completeness for some problems.

4. **P-complete decision problems.**

   The goal of this lecture is to survey examples of P-complete decision problems and to describe (at a high level) the proofs of P-completeness for some problems. The lecture can also discuss connections between problems that are in the class P (that is, problems that can be solved efficiently and sequentially) and problems that are in the class NC (that is, problems that can be solved efficiently using parallelism).

5. **Provably intractable decision problems.**

   The goal of this lecture is to present a survey of decision problems that belong to classes outside of PSPACE, to survey relationships between complexity classes outside of PSPACE, and to study problems that require at least exponential time or space (or both). In particular, this could include problems that require doubly exponential time, triply exponential time, and so on, leading to an upper limit of the class of all decidable languages.

6. **The polynomial hierarchy.**

   The polynomial hierarchy generalizes the classes NP and coNP. The class NP can be formulated as the class of all languages that can be described using existential quantifiers applied to a polynomial-time predicate. If we alternate increasing numbers of existential and universal quantifiers, we add additional levels to the polynomial hierarchy. The goal of this lecture is to define the class PH and to discuss, for example, decision problems at varying levels of the hierarchy and consequences of a hierarchical collapse.

7. **Alternation.**

   A nondeterministic computation is existential, in that if *some* accepting computation branch exists, then the entire computation is accepting. We can generalize the notion of nondeterminism to use both existentialism and universality: if we introduce universality, where *all* computation branches must accept, we obtain the notion of alternation. The goal of this lecture is to define alternating Turing machines and to survey basic results about complexity classes as they relate to alternation.

8. **Approximability.**

   While certain decision problems are known to be intractable, in some cases it is possible for us to make a tradeoff between accuracy and efficiency using approximation: we can get a less-accurate answer more efficiently. The goal of this lecture is to discuss basic results about approximation, to survey decision problems that can be approximated, and to discuss the limitations of approximation and the notion of inapproximable problems.

9. **Average-case complexity.**

   Typically, computer scientists focus on the worst-case performance of algorithms, and base their analyses on this worst case. Many hardness results in complexity theory also rely on a worst-case analysis. The goal of this lecture is to introduce average-case analysis and average-case complexity, and to show how one can define the hardness of a problem "on average". A knowledge of probability theory is recommended for this topic.

10. **Communication complexity.**

    Suppose that two people—Alice and Bob—want to evaluate a function $f(a, b)$, where Alice has knowledge of a but not b, and Bob has knowledge of b but not a. Alice and Bob wish to share their information using the least number of bits possible, and minimizing this number of bits in the worst case for varying values of a and b is the core problem of communication complexity. The goal of this lecture is to present the basic definitions of communication complexity and to survey some of the major results in this area.

11. **Complexity of number-theoretic problems.**

    Many computational problems have deep connections to number theory, such as the Euclidean algorithm, factoring, and primality testing. The goal of this lecture is to review some of the most well-known number-theoretic problems, discuss algorithms for these problems, and survey their complexity. The lecture could also cover more advanced topics such as randomized algorithms for certain problems.

12. **Cryptography.**

    The entire study of public-key cryptography is predicated on the assumption that encodings of information can be computed efficiently, but decoding the encoded information is computationally intractable without the appropriate "key". The goal of this lecture is to discuss cryptography from a complexity theory perspective. Thus, the focus should not solely be on particular cryptographic protocols, but instead on the complexity of cryptography (e.g., the existence of one-way functions, cryptography under the assumption that $\mathsf{P} = \mathsf{NP}$, post-quantum cryptography, etc.)

13. **Kolmogorov complexity.**

    The Kolmogorov complexity of a word w is a measure of the smallest algorithm or program that outputs w and nothing else. "Smallest" in this case refers to the length of the algorithm or program in terms of characters or bits. The goal of this lecture is to define Kolmogorov complexity, present some examples, and review the main results in this area. The lecture could also discuss the notion of randomness as it relates to Kolmogorov complexity.

14. **Overhead-free computation.**

    The definition of a Turing machine that we used in class allowed the tape alphabet to be arbitrarily large and allowed us to use as many work tape cells as we desired. This has a dramatic effect on the complexity of the computation of a Turing machine in that multiplicative constants are ignored; even if a computation takes "polynomial time", this time may be something unreasonable like $2^{1000} \cdot n$. Overhead-free computation attempts to remedy this issue by placing restrictions on the Turing machine: it may only use input symbols, and it may only write to tape cells that are nonempty. The goal of this lecture is to present the notion of overhead-free computation, to explain how overhead-free Turing machines differ from the usual model, and to survey known results in this area.