Topics for Final

- Open books and notes but no electronic aids
- Issues from prior exams/homeworks
  - Induction proofs
  - Showing closure properties via constructions
  - Estimating pumping length
  - ε rules in PDAs and equivalence to pushes and pops
  - Pumping lemmas
- (p. 176) Chap. 3.2 Variations of TMs
  - Multiple variations of TMs are possible
    - Add “S(tay)” to Left/Right directions
    - Multiple tapes
    - Bi-directional infinite tape
    - TM with a stack
    - Non-Deterministic TM: Transitions lead to a set of (QxΓx{L,R})
      - Computations follow a “tree” of possibilities
      - If some branch leads to an accept state, NTM accepts
  - None of these options lead to any more “capable” machine
    - May be faster but cannot compute anything standard TM can
  - Approach to proving this
    - (Easy) Show new machine can compute anything a 1 tape TM can
    - (Tougher) Show 1-tape TM can emulate any program for new machine
  - **Enumerators**: A TM that generates sequentially a set of strings from some language L in a way that guarantees that any string in L is eventually generated
- (p. 182) Chap. 3.4 Algorithms
  - **Algorithm**: ordered finite set of steps where each step does a finite operation
  - **Church-Turing Thesis**: any algorithm can be expressed as a TM (where any answer is left on tape)
  - Not all problems are solvable by a TM/algorithm
    - Example: Hilbert’s 10^{th} problem –integral root for a polynomial.
    - Recognizers may exist but not deciders
  - (p. 185) Terminology for describing TMs
    - **Formal Description**: all sets, all transitions
    - **Implementation level**: English prose on how the tape is processed by the TM
    - **High Level**: English prose description of algorithm (typically as composition of other algorithms)

- (p. 193) Chap. 4 Decidability
  - Language = set of strings
    - Machines can be encoded as strings (e.g. machine files for projects)
  - (p. 170) Language is **Turing-recognizable** if some TM **recognizes** it
    - Always accepts if input is in language
    - Never accepts if input is not in language
  - (p. 170) Language is **Turing-decidable** if some TM decides it
    - Always accepts if input in language
    - And always rejects any input not in language – NEVER LOOPS
  - TM is a **co-Turing recognizer** of L if TM recognizes the complement of L
  - (p. 194) **Acceptance problem** = is some specific string in a specific language?
  - (p. 194) **Decidable language**: algorithm exists to always determine yes or no (no loop)
    - Be able to describe algorithm for decision
    - Decidable languages based on DFA/NFA (i.e. regular expressions)
    - Decidable languages based on PDA (i.e. Context free)
  - (p. 201) 4.2: **Undecidability**: cannot write algorithm to decide
    - May be recognizable or co-Turing recognizable, BUT NOT BOTH
    - First undecidable language: $A_{TM} = \{\langle M,w \rangle | M \text{ accepts } w\}$
      - Proof by contradiction, Uses idea of diagonalization (do not need to understand details of p. 203-208 on diagonalization)
  - (pp. 220-226) Computational Histories (LBA **not** covered)
  - (p. 209) **co-Turing recognizability** (complement of a language is recognizable)
    - Complement of L = {w|w any string NOT in L}
  - L is decidable iff recognizable and co-Turing recognizable

- (p. 215) Chap 5 Reducibility
  - **Reduction of A to B**: transform any **instance** of Problem A into an instance of Problem B and use decider/solver for B to give correct answer for instance of A
  - (p. 216) 5.1 Undecidable problems from Language Theory
    - Be able to prove B is undecidable by showing reduction from problem A (which is undecidable) to B. If B is decidable then A must also, causing a contradictory
  - (p. 237**) Post Correspondence Problem** is undecidable – understand problem – do not need to recreate proof
  - (p. 234) 5.3 **Mapping Reducibility**: mapping from A to B is via a function

- (p. 275) Chap. 7 Time Complexity
  - Determine "Big O" time complexity of a function as function of size of input
  - (p. 279) **TIME(t(n))** = all languages decidable by O(t(n)) TM
  - (p. 282) Every t(n) time multi-tape TM has eqvt $O(t(n)^2)$ 1-tape TM
  - (p. 283) Running time of NTM = max # of steps in any possible path

- (p. 284) 7.2 **Class P**: polynomial time deciders
  - Show by designing deterministic TM decider in time $O(n^k)$ for some k
  - (p. 288) PATH = {<G,s,t>| there is a path from s to t}
  - (p. 289) RELPRIME = {<x,y>|x and y are relatively prime} Uses Euclidean alg
  - (p. 290) Every CFL is in P – uses dynamic programming
  - [7.6] Show P closed under union, concatenation, complement

- (p. 292) 7.3 **Class NP**: a NTM can produce, in poly time, a "certificate" which can be *checked* by a polynomial time verifier
  - NTM typically generates "all possible" solutions, and passes correct one to verifier to check.
  - Crystal Ball" guesses answer & verifier simply has to check in poly time
    - Essentially your brute-force SAT solver
  - Equivalent to being able to generate via an enumerator a possibly large but bounded number of certificates which can be fed to verifier
    - If one of these returns "verified" problem is solvable
  - **NTIME(t(n))** = languages decidable by NTM in O(t(n)) time
  - Proof technique:
    - Show NTM can generate a "certificate" (a.k.a a guess) in poly time
    - Show poly time NTM can verify
  - (p. 296) CLIQUE = {<G,k>|G has k vertices with edges to each other}
  - (p. 297) SUBSET-SUM = {<S,t>|some subset of S adds up to t}
  - SAT = {<wff>|wff is satisfiable}

- (p. 299) 7.4 NP-Complete: Subset of NP problems into which <u>all other</u> NP problems can be mapped
  - If poly time decider exists for any problem in NP-complete, then all of NP is in P
  - (p. 304) COOK-LEVIN Theorem: SAT is in NP-Complete because we can build a giant wff from a NTM and its input, that is satisfiable iff NTM accepts its input
    - Do not need to understand how wff is built, only that we can

    - To add other problems B to NP-complete
        - Show poly time mapping from all instances of some A (known to be in NP-Complete) into an instance of B
        - Show if decision for A exists then so does decision for B, & vice versa also
    - (p.302) 3SAT is poly time reducible to CLIQUE
  - (p. 311) Additional NP-Complete problems (Understand what problems are, not details of proof)
    - (p. 311) CLIQUE because of mapping from 3SAT
    - (p. 312) VERTEX-COVER = {<G,k>| some set of k vertices has all edges in G touching them) via Map from 3SAT
    - (p. 314) HAMPATH ={<G,s,t>|G directed graph: path from s to t touches all vertices once} via map from 3SAT
    - (p. 314) UHAMPATH ={<G,s,t>| G undirected}
    - (p. 320) SUBSET-SUM = {<S,t>|some subset of S adds up to t}
  - Other
    - Show NP closed under union, complementation, star
- (V3: 7.34) NP-Hard: from notes – simply remember all NP reduce to it but they are not in NP