Topics for Final

- Open books and notes but no electronic aids
  - #s in “()” refer to homework problems; [] = Exercises
- Issues from prior exams/homeworks
  - Exam 1: (Prob. 2) Designing a DFA for some language
  - Exam 1 (Prob. 3): Pumping lemma for regular languages
  - Exam 2: (Prob. 4): Defining a CFL from a set description; pumping lemma for CFLs
  - Exam 2 (Prob.5) : Designing a TM
  - Other difficulties (from homeworks)
    - Induction proofs
    - NFAs to/from DFA to/from regexs
    - Showing closure properties via constructions
    - Estimating pumping length
    - ε rules in PDAs and equivalence to pushes and pops
- Special Topics: (possible subject of multiple choice questions)
  - Understand basics of combinators (S,K,I)
  - Understand general nature of Quantum Computing
  - Understand general nature of Micron Automata
- (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
    - <u>Always accepts</u> if input is in language
    - <u>Never accepts</u> if input is not in language
  - (p. 170) [V3:4.5, 4.10, 4.12, 4.14, 4.25] Language is **Turing-decidable** if some TM decides it
    - <u>Always accepts</u> if input in language
    - And <u>always rejects</u> any input not in language – NEVER LOOPS
  - (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)
    - (HW7.1) Be able to describe algorithm for decision
    - **(**pp. 194-197) decidable languages based on DFA/NFA (i.e. regular expressions)
    - (HW7.2) (pp. 198-200) decidable languages based on PDA (i.e. Context free)
  - (p. 201)(HW7.3) 4.2: **Undecidability**: cannot write algorithm to decide

    - May be recognizable or co-Turing recognizable, BUT NOT BOTH
    - First undecidable language: A_{TM} = {<M,w>|M 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 and LBA not covered
  - (p. 209) [HW7.4] **co-Turing recognizability** (complement is recognizable)
    - L is decidable iff recognizable and co-Turing recognizable
- (p. 215) Chap 5 Reducibility
  - [V3:5.5, V3:5.7] **Reduction**: transform any **instance** of Problem A into an instance of Problem B and use solver for B to solve instance of A
  - (p. 216) [V3:5.10 and 5.11, HW7.4, 7.6)]5.1 Undecidable problems from Language Theory
    - Be able to prove B is undecidable by showing reduction from a problem A (which is undecidable) to B
  - (p. 237) (HW7.5) 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
  - (V3:7.1c,d V3:7.2 c,d HW7.7) 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
    - [V3:7.8 HW7.8] 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
    - **NTIME(t(n))** = languages decidable by NTM in O(t(n)) time
    - (HW8.1, HW8.2, HW8.3) 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 all other 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
        - (HW8.4, 8.5) 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
    - [7.7,7.18] (HW8.1) Show NP closed under union, complementation, star
- (V3: 7.34) NP-Hard: from notes – simply remember all NP reduce to it but its not in NP