

Contents

1	Week 2	2
1.1	Pushdown Automata (PDA)	2
1.1.1	Example	3
1.1.2	More examples	3
1.2	Turing Machines (TM)	3
1.2.1	What can't PDAs do?	3
1.2.2	Definition	4
1.2.3	Example	4
2	Week 3	5
2.1	Delta function layout	5
2.2	Output Turing Machine	5
2.2.1	Example	5
2.3	Language of a Turing Machine	5
2.3.1	Question	6
2.4	Multi-tape Turing Machines	6
2.4.1	Show a single tape TM can simulate a multi-tape TM	6
2.5	Single infinite tape TM	6
2.6	Enumerators	6
3	Week 4	7
4	Week 4 Tutorial	7
4.1	Diagonalization to prove \mathbb{R} does not map to \mathbb{N} (Not countable)	7
4.2	Proof that $\text{Eq}_{\text{BFA}} = \langle (\text{DFA1}, \text{DFA2}); \langle (\text{DFA1}) = \langle (\text{DFA2}) \rangle \rangle$	7
5	Week 5	8
5.1	Continuing.. . . .	8
5.2	Halting Problem	9
5.2.1	Turings Method	9
5.2.2	Method 2	9
6	Week 6	10
7	Week 7	10
7.1	Anything about the language is undecideable	11
7.2	Midterm	11
7.2.1	Past test, quesiton 7	12

8	Week 8	13
8.1	Moving Forward	13
8.2	The Lecture	13
9	Week 9	15
9.1	Example	15
9.2	Example	16
9.3	Question	16
9.4	Question	16
9.5	Complexity Reductions	17
10	Week 10	17
10.1	From last time	17
10.2	Toolbox of problems	17
10.3	Hardest NP Problem? H	18
11	Week 11	19
11.1	Last Time	19
11.2	Subset Sum	20
11.3	Knapsack Problem	20
11.4	NP-Hard	21
12	Week 12	21
12.1	Co-NP	21
12.2	Cook-Levin Thm - SAT is NP-Complete	22
12.3	The Exam	23

1 Week 2

1.1 Pushdown Automata (PDA)

- $L = \{ a^n b^n \mid n \geq 0 \}$
- 7 tuple, $M = (Q, \Sigma, \Gamma, \delta, s, F, \perp)$
 - Q : Finite set of states
 - Σ : Finite input alphabet
 - Γ : finite stack alphabet
 - δ : transitions
 - $\perp \in \Gamma$

- $S \in Q$
- $F \subseteq Q$
- $\delta \subseteq (Q \times (\sum U \{\epsilon\} \times \Gamma) \times (Q \times \Gamma^*))((p, a, A), (g, B_1 B_2 \dots B_k))$
 - When in state p , reading in " a ", and A is on top of the stack, the machine may move to state g , pop A off the stack and push $B_k B_{k-1} \dots B_1$ (B_1 is now on the stop of the stack.)

1.1.1 Example

- Stage 1: Push A's
- Stage 2: Pop A's
- Stage 3: Are we accepting? (This mean B's and A's must be equal)

1.1.2 More examples

- a) $L = \{ a^n b^n c^n \mid n \geq 0 \}$ (Impossible)
- b) $L = \{ ww^R \mid w \in \sum^* \}$ (Possible)
- c) $L = \{ ww \mid w \in \sum^* \}$ (Impossible)
- d) $L = \{ a^n b^m c^m d^n \mid n, m \geq 0 \}$ (Possible)

1.2 Turing Machines (TM)

1.2.1 What can't PDAs do?

- Infinite memory, but it's not "perfect"
- What if we do have perfect/infinite memory
- Does this allow for more accepted languages
- How do we formally define it

1.2.2 Definition

- We'll say a TM is an 8-Tuple
- $M = \{ Q, \Sigma, \Gamma, \delta, s, q_{\text{accept}}, q_{\text{reject}}, \square \}$
 - $q_{\text{accept}} \in Q$
 - $q_{\text{reject}} \in Q$
 - $q_{\text{accept}} \neq q_{\text{reject}}$
 - $\square \in \Gamma$
 - $\Sigma \subseteq \Gamma - \{ \square \}$: $Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$
 - $\delta(g_1, a) = (g_2, b, R)$
 - If I am in g_1 and it is an a , write a b and move right

Input is initially written on the tape

If at any point the TM enters state q_{accept} , it stops (halts) and accepts the input

If at any point the TM enters state q_{reject} , it stops (halts) and rejects the input

Can be used as language accepters, or as output machines

Does all input get accepted or rejected?

- No, it can loop

1.2.3 Example

- Construct a TM M such that $L(M) = \{a^n b^n c^n \mid n \geq 0\}$
- Start by scanning the input from L to R to ensure it's of the form $a^* b^* c^*$
- Scan from R to L, overwrite exactly one c , b , then a with a special character
- Scan from L to R, overwrite exactly one a , b , then c repeat 2)
- If at any time, you encounter a letter out of order, reject the input

2 Week 3

2.1 Delta function layout

(1, a, R) - Move to state 1, if its an A move right

	a	b	c	-	\square
S	(1, a, R)	Reject	Reject	/	Accept
1	(1, a, R)	(2, b, R)	Reject	/	Reject
2	Reject	(2, b, R)	(3, c, R)	/	Reject
3	Reject	Reject	(3, c, R)	/	(4, \square , L)
4	Reject	Reject	(5, -, L)	(4, -, L)	Accept
5					
6					
7					
8					
9					
10					
11					

2.2 Output Turing Machine

2.2.1 Example

Let $\Sigma = \{a\}$. write a TM such that when M halts, the tape contains ww where w is the input. (Double the a's)

- $a = aa$
- $aaa = aaaaaa$

2.3 Language of a Turing Machine

- The collection or set of strings which M accepts $L(M)$
- A language L' is Turing-recognizable if \exists a Turing machine M, $L' = L(M)$
 - $w \in L' \Rightarrow$ if we run M on w, it accepts w
 - $w \notin L' \Rightarrow$ if we run M on w, it rejects or infinite loops
- A language L^- is Turing Decidable if \exists a TM, M sich $L^- = L(M)$ and M halts on all input
 - $w \in L^- \Rightarrow$ running M on w accepts
 - $w \notin L^- \Rightarrow$ running M on w rejects

2.3.1 Question

- L is recognizable $\Leftrightarrow L$ is decidable
- L is decidable $\Rightarrow \forall$ turing machines M , such that $L' = L(M) \Rightarrow$ if M is run on $w \in L$, M will halt? **NO**
 1. Let M be a TM which decided L'
 2. Make a new TM, M' , which mimics M except when M rejects, M' will loop

2.4 Multi-tape Turing Machines

- Just like a classic TM, but it has k tapes
- $\delta: Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$ is replaced with $\delta^k: Q \times \Gamma^k \rightarrow Q \times \Gamma^k \times \{L, R\}^k$

2.4.1 Show a single tape TM can simulate a multi-tape TM

- Encode multiple tapes on a single tape

$$\Gamma = \{a, b, \square\} \rightarrow \Gamma' = \{a, b, a', b', \square\}$$

2.5 Single infinite tape TM

- Same as a classic TM but infinite only in one direction
- Just add some left padding
- This is like a 2 tape TM, the 2nd tape is the 1st one flipped

2.6 Enumerators

- 2 tapes and a control
 - Control has the output tape with the following conditions
 - * Write Only
 - * Tape alphabet = Σ
 - * Only works to the right
 - Control has the work tape with the following conditions
 - * read/write

- * Basically a TM
- * No accept or reject states
- * Enumerate state
- Whenever the control reaches the enumerate state, it prints out whatever is on the output tape and clears it
- $L(E_)$ is the set of strings which W will eventually print out

3 Week 4

- I slept...

4 Week 4 Tutorial

4.1 Diagonalization to prove \mathbb{R} does not map to \mathbb{N} (Not countable)

- $d_{11}d_{12}d_{13}\dots d_{1n} \quad d_{21}\dots\dots d_{2n} \quad \dots\dots\dots d_{n1}\dots\dots d_{nn}$
- Construct e such that $e_{ii} = (d_{ii} + 2) \% 10$

4.2 Proof that $\text{Eq}_{\text{BFA}} = \langle (\text{DFA1}, \text{DFA2}); \langle (\text{DFA1}) = \langle (\text{DFA2}) \rangle \rangle$

- $2 \cdot L(\text{DFA1}) = L(\text{DFA2}) \Rightarrow (L(\text{DFA1}) \cap L(\text{DFA2})) = \epsilon$ AND $(L(\text{DFA1}) \cup L(\text{DFA2})) = \epsilon$
- Checking if $L(\text{DFA}) = \epsilon$ is Decidable
 - Go through every state in the DFA
 - If you go to any state that is accepting with a string that is not empty, reject
 - * Otherwise accept.
- Full proof on pg 169 of tb
 - Introduction to theory of computation 2nd edition By M. Sipser

5 Week 5

5.1 Continuing..

- $A_{TM} = \{M\#w \mid M \text{ accepts } w\}$
 - The membership/acceptance problem
 - Thm: A_{TM} is a recognizable language
 - Proof: Construct a new TM U
 - * U = "on input $M\#_w$ where M is a TM and w is a string"
 1. Simulate M running on w
 2. If M accepts, U accepts
 3. If M rejects, U rejects
 - Universal Turing Machine (UTM)
 - Thm: M is undecidable
 - Proof: Assume that A_{TM} is decidable \Rightarrow there exist a TM D $D(M\#_w) = \{ \text{accept if M accepts W, reject if M does not accept W} \}$ P = "on input M, where M is a TM:"
 1. Run D on input $M\#_M$
 2. Output the opposite of D
 - * if D accepts, reject
 - * if D rejects, accept
 3. $P(M) = \{ \text{accept if M does not accept M, reject if M does accept M} \}$
 - * What if we run P on itself? $P(P) = \{ \text{accept if P rejects, reject if P accepts} \}$ This is a paradox \Rightarrow Contradiction $\therefore A_{TM}$ is undecidable
- Def $L_{bar} = \Sigma^* - L$
 - Thm: A_{TMBar} is unrecognizable
 - Proof: Will follow from
 - * Thm: A is decidable iff A and ABar are recognizable
 - * Proof:
 1. Show A is decidable \Rightarrow A and ABar are recognizable
 2. Show A is decidable \Rightarrow A is recognizable
 3. \Leftarrow A is recognizable $\Rightarrow \exists$ TM m such that $\forall w, M$ accept (and halt) w iff $w \in A$

- * Let M_1 and M_2 recognize A and A_{Bar} respectively $\Rightarrow A_W$
 M_1 halts if $w \in A$, M_2 halt is $w \notin A$
- * $M = \text{"Run } M_1 \text{ and } M_2 \text{ in parallel"}$
 1. If M_1 accepts, accept
 2. If M_2 accepts, reject
- * M is a decider for $A \Rightarrow A$ is decideable
- Thm (again): A_{TMBar} is unrecognizable
 - * if A_{TMBar} was recognizable $\Rightarrow A_{\text{TM}}$ is decidable, which it isn't

5.2 Halting Problem

5.2.1 Turings Method

- $\text{HP} = \{ M\#_w \mid M \text{ halts, on } w \}$
- Thm: The halting problem is undecidable
- Proof:
 1. Σ^* is countable
 2. The set of all TMs is countable
 3. Construct a TM P : $P(w) = \text{"on input } w, \text{ construct } M_w \text{ and we run } D \text{ on } M_w\#_w$
 - (a) If D rejects, accept
 - (b) if D accepts, loop
 4. Assume HP is decideable
 - $D(m\#_w) = \{ \text{accept if } M \text{ halts on } w, \text{ reject if } M \text{ loops on } w \}$
 - P is not in my table, \therefore contradiction.

5.2.2 Method 2

- Assume HP is decideable
 - Then \exists a decider for HP
- $D_1(M\#_w) = \{ \text{"accept if } M \text{ halts on } w, \text{ reject if } M \text{ loops on } w" \}$ D_2
 $= \text{"on input } M\#_w, M \text{ is a TM, } w \text{ is a string"}$

1. Run D_1 on input $M\#_w$
2. If D_1 rejects, reject
3. If D_1 accepts, simulate M on w 3.1) If M accepts, accept 3.2) If M rejects, reject

$D_2(M\#_w) = \{ \text{"if } M \text{ loops on } w \text{ reject, if } M \text{ rejects } w \text{ reject, if } M \text{ accepts } w \text{ accept"} \}$ D_2 decides A_{TM} $A_{TM} = \{ M\#_w \mid M \text{ accepts } w \}$
 Contradiction, \therefore HP is undecidable

6 Week 6

- Sick

7 Week 7

- Last Time
 - $A \leq_m B$
 - ES and REG are both undecidable
- Given a TM M
 1. Accepts any string
 2. Accepts the string OOOO
 3. Accepts every string
 4. Accepts a finite set
 5. Accepts a CFL
 6. Has 100 or more states
 7. Has more than 100 steps on input w
 8. 1 through 5 are undecidable
 - About the language of the TM ($L(M)$)
 9. 6 through 7 are decidable
 - About the TM (M)

7.1 Anything about the language is undecidable

- Let S denote the set of all languages
- Let a property P be a subset of S
- A TM M has the property P if $L(M) \in P$
- Property
 - A TM accepts no strings $\Rightarrow P_{\text{empty}} = \{ \}$
 - A property is non trivial if $\exists M_1$ and M_2 such that $L(M_1) \in P$ and $L(M_2) \notin P$
 - $P_{\text{trivial}} = \{ L \mid L \text{ is recognizable} \}$
- Rice's Thm:
 - Let P be a non-trivial property of languages of Turing Machines
 - * $L_P = \{ M \mid L(M) \in P \}$ is undecidable

7.2 Midterm

- Thursday March 1st, 6:00-8:00pm
- IB120
- 5 Questions
- Topics
 - PDAs
 - Low Level TMs
 - * Define a TM, states, transitions, alphabet, etc
 - Models of Equivalence
 - * Ignore enumerators
 - Decideability and Recognizeability
 - A_{TM} /Halting Problems proofs
 - * Don't memorize the proof in full
 - * Know the idea, fill in the blanks/find the error
 - Reductions
 - * Definitions

* Proofs

- Show that language L is decideable/recognizeable using a reduction
- Define L is recognizable \exists a TM M such that $\forall w$ if $w \in L$ then M accepts w and if $w \notin L$ M loops or rejects w
- $HP = \{ M_{\#w} \mid M \text{ halts on } w \}$
 - $R(M_{\#w}) = \text{run } M \text{ on } w, \text{ if } M \text{ accepts} \rightarrow \text{accept. If } M \text{ rejects} \rightarrow \text{accept. If } M \text{ loops} \rightarrow \text{loop}$
 - $M_{\#w} \in HP$ iff M doesn't loop on w
- A is recognizable $\neg \rightarrow A'$ is unrecognizeable
- A is undecideable and A is recognizable $\rightarrow A'$ is unrecognizeable

7.2.1 Past test, quesiton 7

- $L'' = \{ M : L(M) = L_L \}$
- $L_L = \{ 0^n 1^m \mid n < m \}$
 - $011 \in L_L$
 - $0 \notin L_L$
 - $\epsilon \notin L_L$
 - $10 \notin L_L$
- is L'' decideable, recognizable?
- is L''' decidable, recognizable?
- HP, A_{TM}, HP', A_{TM}'
- $HP' \leq_m L$
- Show L'' is undecideable
 - $HP \leq_m L''$
 - $f(M_{\#w}) \in L'' \iff M_{\#w} \in HP$
 - $f(M_{\#w}) = \text{on input } x, \text{ set } x \text{ aside}$
 - * Run M on W

- * run a L_L recognizer on x
 - if it accepts \rightarrow accept
 - if it rejects \rightarrow reject
- $L(f(M_{\#W})) = \{ L_L \text{ if } M \text{ halts on } w, \text{ if } M \text{ loops on } W \}$
- $f(M_{\#W}) \in L'' \iff L(f(M_{\#W})) = L_L \iff M \text{ halts on } W \iff M_{\#W} \in HP$
- $\therefore L''$ is undecidable
- if $A \leq_m B$ then $A' \leq_m B'$
 - $HP' \leq_m L''$
 - $\therefore L''$ is unrecognizable

8 Week 8

- What can (or can't) computers do (effectiveness)
- So something using (relatively) little resources
 - Time and space

8.1 Moving Forward

- Look at low level TM
- Compare Models
- Classify problems / Hierarchy - Reductions

8.2 The Lecture

- Def Let M be a total deterministic TM, the time complexity of M is a function $f: \mathbb{N} \rightarrow \mathbb{N}$, $f(n)$ is the max number of steps M uses on any input of length n
 - M is a $f(n)$ TM
 - M runs in $f(n)$ time
 - Def $f, g: \mathbb{N} \rightarrow \mathbb{R}^T$, $f(n) = O(g(n))$ if $\exists n_0, c \forall n > n_0: f(n) < cg(n)$

- Def $f, g: \mathbb{N} \rightarrow \mathbb{R}^+$ $f(n) = o(g(n))$ if

$$\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = 0$$

g grows faster than f eventually

- Facts

- $f(n) = O(f(n))$
- $f(n) \neq o(f(n))$
- $f(n) = O(g(n)) \nrightarrow f(n) = o(g(n))$
- $f(n) = o(g(n)) \rightarrow f(n) = O(g(n))$

- Consider the language $L = \{0^n 1^n \mid n \geq 0\}$ $M_1 =$ on input w

1. Scan from L to R
2. Cross off first 0 encountered 2,1) Then cross off the first 1 encountered
3. When is reached, go back to the beginning
4. Repeat 2 4,1) If anything is found out of place, reject 4,2) If no 0 or 1 before , accept

- M is a $f(n) = o(n \log(n))$, then $L(M)$, is regular

- Def: $t: \mathbb{N} \rightarrow \mathbb{R}^+$, the time class $\text{TIME}(t(n))$ is the set of all languages that are decidable by a $O(t(n))$ TM.
 - * is $L \in \text{TIME}(N^3)$?
 - * is $L \in \text{TIME}(N)$?
 - * is $L \in \text{TIME}(n \log(n))$
- Thm Let $t(n)$ be a function where $t(n) \geq n$. Then every $t(n)$ time multitape TM has an equivalent $O(t^2(n))$ single tape TM
 - * How long can the single tape be at any moment in time? Assume $t(n) \geq n$
 - Each multitape can be at most $t(n)$ long
 - The single table can be at $\leq k \cdot t(n) + k - 1$ (Account for the boundry characters) $= O(t(n))$

- In the worst case for a single "step" take

$$O(t(n)) + kshifts$$

$$O(t(n)) + k * O(t(n))$$

$$O(t(n))$$

- Number of steps to simulate = $t(n)$
- \therefore Total simulation time $t(n) * O(t(n)) = O(t^2(n))$
- Let N be a total NTM, its running time $f(n)$ is the running time of its longest branch
- Thm Let $t(n) \geq n$, then every $t(n)$ NTM has an equivalent $2^{O(t(n))}$ time deterministic TM
- Def P is the class of languages that are decidable in polynomial time on a deterministic single-tape TM.

$$P = \bigcup_{k=0}^{\infty} TIME(n^k)$$

9 Week 9

- All deterministic computational models are polynomial equivalent
 - If you can code a program which runs in polynomial time, a single tape TM exists, which does the same thing

9.1 Example

$PATH = \{ \langle G, s, t \rangle \mid \text{There's a path from } s \text{ to } t \text{ in } G \}$

- Theorem $PATH \in P$
 - Proof A poly time algorithm M operates as follows $M(G, s, t) =$
 - * "Mark" s
 - * While at least one more node has been marked since last iteration
 - For each edge (a, b) if a is marked and b is not, mark b
 - If t is marked \rightarrow accept, else reject

9.2 Example

HPATH = $\{ \langle G, s, t \rangle \mid \text{there's a Hamiltonian path from } s \text{ to } t \}$

- Def A Hamiltonian path goes through every other node exactly once
- Def A verifier for a language L is an algorithm V where:
 - $L = \{ w \mid V \text{ accepts } \langle w, c \rangle \text{ for some } c \}$
 - c is called the certificate
 - A poly time verified runs in polynomial time in the length of w
 - A language L is polynomial time verifiable if it has a polynomial time verifier (This TM always halts)
- Def One NP is the class of languages which have polynomial time verifiers
- Def Two A language is in NP iff it is decided by some polynomial time NTM

9.3 Question

If a language is in P, it is also in NP?

- Def One
 - $V(\langle w, c \rangle) = \text{run } D_{\text{poly}} \text{ on } w$
 - $D_{\text{Poly}} = \text{decider for } L$
- Def Two
 - $\{ \text{TM} \} \subseteq \{ \text{NTM} \}$

9.4 Question

If a language is in NP, it is also in P?

- Well this is the $P = NP$ problem. We know sometimes, not if it is always.

9.5 Complexity Reductions

- Def Language A is a polynomial time reduceable to language B, $A \leq_p B$, if \exists a polynomial time computable function f , where $\forall w, w \in A \iff f(w) \in B$
 - Typical use Given problem A, assume we have a black box which solves problem B. In polynomial time, make your instance of A look like an instance of problem B. Run your B black box on that instance, return an answer appropriately
- Example $LONG = \{ \langle G, s, t, d \rangle \mid \exists \text{ a path from } s \text{ to } t \text{ in } G, \text{ with at least distance of } d \}$
 - No cycles allowed, can't visit a node more than once
 - Prove that $HPATH \leq_p LONG$
 1. Assume I have a black box which solved LONG
 2. To solve $HPATH(G, s, t)$
 - (a) Set all edge weights to 1
 - (b) Call $LongSol(G, s, t, |V| - 1)$
 - * If yes \rightarrow yes, if no \rightarrow no
 - Thm If $A \leq_p B$, and $A \in P$, then $B \in P$
 - Caollary If $A \leq_p B$ and $A \notin P$, then $B \notin P$

10 Week 10

10.1 From last time

- $A \leq_p B$
 - If we can solve B, then we can solve A
 - Then B is at least as hard to solve as A is

10.2 Toolbox of problems

- Vertex Cover Given a graph $G=(V, E)$, a Vertex Cover, C, is a set of vertices such that $\forall (v_1, v_2) \in E, v_1 \in C, v_2 \in C$
 $VertexCover = \{ (G, K) \mid G \text{ has a vertex cover of size } k \text{ or less} \}$

- Independent-Set Given a graph $G=(V, E)$, an independent set is a set of vertices I , such that $\forall (v_1, v_2) \in E, \neg(v_1 \in I \text{ and } v_2 \in I)$
Independent-Set = $\{(G, K) \mid G \text{ has an independent set of size } K \}$
 - Prove: Independent Set \leq_P Vertex Cover Let C be a vertex cover of $G=(V, E)$. Consider any two nodes v_1 and v_2 in $V-C$.
What if $v_1, v_2 \in V-C$ and $(v_1, v_2) \in E \Rightarrow C$ is not a VC
 - * $\therefore v_1, v_2 \in V-C$ we know $(v_1, v_2) \notin E$
 - * $\therefore V-C$ is an independent set in G
 - * \therefore If there is a vertex cover of size $|V| - k$ in G , then there's an independent set of size k in G
 - IS(G, K) = call VertexCoverSol($G, |V|-k$)
 - * If accept \rightarrow accept (yes)
 - * If reject \rightarrow reject (no)
- Clique Given a graph $G=(V, E)$, a clique of the graph is a set S such that $S \subseteq V$ and $\forall v_1, v_2 \in S, (v_1, v_2) \in E$ or $v_1 = v_2$.
Clique = $\{(G, K) \mid G \text{ has a clique of size } k \text{ or more} \}$
 - Prove: Vertex cover \leq_P Clique Let the complement of $G=(V, E)$ be $G'=(V, E')$ where $E' = \{(v_1, v_2) : v_1, v_2 \in V \text{ and } v_1 \neq v_2 \text{ and } (v_1, v_2) \notin E\}$
Suppose G' has a clique $S \subseteq V$ where $|S| = |V| - k$. Let (v_1, v_2) be an edge in E , then $(v_1, v_2) \notin E'$, then one of v_1 or v_2 is not in S (otherwise S is not a clique in G') \Rightarrow At least one of v_1 or v_2 are in $V-S \Rightarrow (v_1, v_2)$ is covered by $V-S \Rightarrow$ All edge $\in E$ are covered by $V-S \Rightarrow V-S$ is a Vertex Cover $VC(G, K) = \dots$
 - * Create $G \leftarrow$ Polynomial time
 - * Call Clique($G', |V| - k$)
 - yes \rightarrow yes
 - no \rightarrow no
 - Prove: Clique \leq_P Independent Set
 1. Consider an independent set I in G' . Then in G , I is a clique
 $IS \leq_P VC \leq_P Clique \leq_P IS$

10.3 Hardest NP Problem? H

- $\forall A \in NP, A \leq_P H$

- Def A language B is NP-Complete if it satisfies two conditions
 - B is in NP
 - $\forall A \in \text{NP}, A \leq_p B$
- Corollary If A is NP-Complete and $A \leq_p B$ and B is in NP then B is NP-Complete
- How to prove B is NP-Complete
 - Prove B is in NP
 - Choose an appropriate NP-Complete problem A and show $A \leq_p B$
- Example $\text{SAT} = \{ \langle \phi \rangle \mid \phi \text{ is a satisfiable boolean formula} \}$
 - $\exists x \forall y \in \text{SAT}$
 $x \wedge x' \notin \text{SAT}$
 - Cook-Levin Thm SAT is NP-Complete, or $\text{SAT} \in \text{P}$ iff $\text{P}=\text{NP}$
 $3\text{SAT} = \{ \langle \phi \rangle \mid \phi \text{ is a satisfiable 3cnf boolean formula} \}$ Ex:
 $(x_1 \vee x_2 \vee x_3) \wedge (x'_1 \vee x_4 \vee x'_3) \wedge \dots \wedge (x'_1 \vee x'_{10} \vee x_{14})$ 3SAT
 is NP-Complete Circuit SAT is NP-Complete
 - Prove Independent Set is NP-Complete $3\text{SAT} \leq_p \text{Independent Set}$

11 Week 11

11.1 Last Time

- NP Complete Problems
 - B is in NP
 - Find a NP-Complete language A and show $A \leq_p B$
- Thm: 3SAT is NP-Complete
- Thm: $3\text{SAT} \leq_p \text{Independent Set}$

11.2 Subset Sum

- Def Subset-Sum = $\{ \langle S, t \rangle \mid \text{There exists a subset of } S \text{ which sums exactly to } t \}$
 - Ex $(\{1, 4, 6, 2, 3\}, 13) \in \text{Subset-Sum}$
 $(\{1, 4, 6, 2, 3\}, 17) \notin \text{Subset-Sum}$
- Thm: Subset-Sum is NP-Complete
- Def Partition = $\{ \langle S \rangle \mid S \text{ can be split into two equal subsets} \}$
- Partition \leq_p SubsetSum
- Thm: SubsetSum \leq_p Partition
 - Given an instance of SubsetSum S, t , create a new set S' , where s is the sum of all values of S and $S' = S \cup \{s - 2t\}$
 1. \exists a subset of S which sums to t , then there's a partition in S'
 2. if there's a partition in S' there's a subset sum S, t
- Proof
 1. What's the sum of S'
 - $s + s - 2t$
 - $s^2 - st$
 - $2(s - t)$

11.3 Knapsack Problem

- Def
 1. Given a set of items $I = \{(w_1, v_1), (w_2, v_2) \dots (w_n, v_n)\}$
 2. A max capacity C
 3. 2 possible goals
 - Maximize your value
 - Achieve a total value of at least V
 - * Find an $S \subseteq I$, $\sum_{w_i \in S} w_i \leq C$ and $\sum_{v_i \in S} v_i \geq V$
 4. Formal Def Knapsack = $\{ \langle I, C, V \rangle \mid \text{there exists a subset of items with at least total value } \geq V \text{ and total weight } \leq C \}$
- Thm Knapsack is NP-Complete

- Find an A such that $A \leq_p \text{knapsack}$
 - * Partition \leq_p Knapsack
 - * Given a set $S = \{a_1, a_2, \dots, a_n\}$
 1. $I = \{ (a_1, a_1), (a_2, a_2) \dots (a_n, a_n) \}$
 2. $C = \frac{1}{2^{\sum_{a \in S} a}}$ and $V = \frac{1}{2^{\sum_{a \in S} a}}$

11.4 NP-Hard

- Def A language/problem H is said to be NP-Hard if $\forall L \in \text{NP}, L \leq_p H$
- Prove H is NP-Hard Show any NP-Complete problem reduces to H
 - KnapsackMax = $\{ \langle I, C, V \rangle \mid V \text{ is the max value one can achieve} \}$
- Thm: HP is NP-Hard
- Proof: $3\text{SAT} \leq_p \text{HP}$
 1. Construct a TM S, which takes as input a boolean formula and brute force checks every possible configuration.
 - Can be constructed in polynomial time
 - 1.1) If it find a valid instantiation
 - Accept (halt)
 - 1.2) If it does not
 - Loop
 - Given Φ call $\text{HPsol}(S\#\Phi)$

12 Week 12

12.1 Co-NP

- Def If L is NP, then L' is co-NP
- Examples VertexCover' = $\{ \langle G, k \rangle \mid \text{in } G \text{ there is not a Vertex Cover of size } k \text{ or less} \}$
SubsetSum = $\{ \langle S, t \rangle \mid \text{Every non-empty subset of } S \text{ does not sum to } t \}$

- Does $NP = co-NP$?
 - Well, is $P \subseteq Co-NP$? *Yes. Assuming $P \neq NP$, we know of languages that are in NP and $co-NP$, but not in P .*
- What is the $co-NP$ problem to SAT ? $SAT' = \{ \langle \Phi \rangle \mid \Phi \text{ cannot be satisfied} \}$
 - Φ is a contradiction if it is always false
 - * $\text{Ex. } x \wedge x'$
 - Φ is a tautology if it is always evaluates to true
 - * $\text{Ex. } x \vee x'$

12.2 Cook-Levin Thm - SAT is NP-Complete

- Show all NP problems reduce to SAT
- Show if a problem is decided by a n^k NTM then it can be reduced to SAT
- Config $x_0x_1 \dots x_{k-1}q_ix_kx_{k+1} \dots x_m$
 - Create a TM with this on the tape, the head will be located at state q_i
- Let $N = \{Q, \Sigma, \Gamma, \delta, \dots\}$ be a n^k NTM
 - Come up with a boolean formula Φ which is satisfiable iff \exists a sequence of configurations which leads to an accept configuration
- There are n^{2k} cells
 - Polynomial in n
- δ, Σ, Γ , and Q are independent of n
- Let $x_{i,j,c}$ be a boolean variable where
 1. $1 \leq i, j \leq n^k$
 2. $c \in \Gamma \cup \{ \# \} \cup Q$
- $X_{i,j,c}$ is true if c is written in cell i, j
 - eg, if $x_{2,3,q_1}$ is true, q_1 is in cell 2, 3

- What if $x_{i,j,a}$ and $x_{i,j,b}$ were both true $\Phi = \Phi_{\text{cell}} \wedge \Phi_{\text{start}} \wedge \Phi_{\text{accept}} \wedge \phi_{\text{move}}$
- $\Phi_{\text{cell}} = \bigwedge_{1 \leq i,j \leq n^k} (V_c \in C \wedge x_{i,j,c} \wedge c_1, c_2 \in C \text{ and } c_1 \neq c_2 \rightarrow ((x_{i,j,2_1})' \vee (x_{i,j,c_2})'))$
- Φ_{start} = the first row is a valid start configuration
 - $\Phi_{\text{start}} = x_{1,1,\#} \wedge x_{1,2,q_{\text{start}}} \wedge x_{1,3,w_1} \wedge x_{1,n+2,w_n} \wedge x_{1,n+3,\square} \wedge x_{1,n^k+3,\#}$
- Φ_{accept} = One of the configurations is in the accept state
 - $\Phi_{\text{accept}} = \bigvee_{1 \leq i,j \leq n^k} x_{i,j,q_{\text{accept}}}$
- Φ_{move} = Each configuration legally follows from the last
 - There are multiple "legal" configurations
 - For each cell, we can look at a "window"
 - Let $\delta(q_3, a) \dots$
 - * (q_3, b, R)
 - * (q_4, c, L)
 - How many moves are there? $= |C|^6$
 - $\Phi_{\text{move}} = \bigwedge_{1 \leq i,j \leq n^k} \bigvee_{a_1 \dots a_6} \text{is a legal window } (x_{i,j,a_1} \wedge x_{i,j+1,a_2} \wedge \dots \wedge x_{i+1,j,a_5} \wedge x_{i-1,j+1,a_6})$

12.3 The Exam

- April 16th, 5 - 8pm
- Office Hours
 - April 12th and 14th, 3-5
- Lots of reductions
- 7 Questions
 1. 10 T/F, answer 7
 - Anything
 - 2 & 3) Computability

- What does it mean to be decidable/recognizable or some mapping reduction

4 - 7) Complexity

- Prove reduceability
 - * Vertex cover \leq_p clique, etc
 - Prove something in in NP, NP Hard, etc
 - Ranging difficulty from trivial to hard
- No PDA material on the exam
 - Study from assignments and lecture notes