1. **The 3 parts in defining a CFG**	Define the smallest thing Split it up into parts combine
2. **Accepted in Nondeterministic Polynomial Time**	If there is a NTM, N, accepting L such that TN is a member of O(n^r) for some r is a member of Nat
3. **Alphabet**	A finite set of objects called the symbols of the alpabet.
4. **Alternating TM (ATM)**	an extension of an NTM in which every state is either existential (∃) or universal (∀)
5. **ambiguous is if you have 2 or more parse trees defining it**	...
6. **Approximation for optimization problems**	There exist problems for which solutions sufficiently close to optimum can be found (and there exist problems without this property)
7. **Boolean circuit**	acyclic directed graph with 3 types of nodes (input, gates, output)
8. **Boolean circuit family**	C = {C0, C1, ... } can compute f if each Cn computes f restricted to inputs of size n
9. **Boolean expression**	a combination of variables, logical connectives, and parenthesis
10. **Bounded error PP class (BPP)**	languages accepted by poly-time PTM's accepting (rejecting) iff more than half the paths terminate in YES (NO) state Repeating a BPP machine rapidly increases the probability of accuracy L ∈ BPP iff ∃ a PTM M M runs in poly-time ∀ x ∀ x∈L, P(M(x) = 1) ≥ 2/3 (1-ε) ∀ x∉L, P(M(x) = 1) ≤ 1/3 (ε)
11. **CFG makes sure you pay attention to the constraints n >= when calculating it**	...
12. **Characteristic function of a language L member of sigma***	Fix distinct strings w1,w2 member of sigma* The total function Xl : sigma* -> sigma* defined by Xl(x) { w1 if x member of L w2 otherwise}
13. **Church turing**	they defined what an algorithm was
14. **Church-Turing hypothesis**	The intuitive notion of an algorithm is a halting Turing machine
15. **The Church-Turing Thesis**	Every effective procedure can be carried out by a Turing Machine
16. **CLASS-complete problem**	A problem that is CLASS-hard and an element of CLASS
17. **CLASS-hard problem with respect to reduction R**	every problem in CLASS reduces by R to this problem
18. **The class NP**	the class of all languages that can be recognized by polynomial time bounded non-deterministic TMs
19. **Clause**	a disjunction (or) of literals
20. **Clique Problem (complete subgraph problem)**	Does G have a complete subgraph with k vertices (k >= 1)
21. **Complete problem**	represents a complexity class in that it is at least as hard as any other problem in that same class

22. **Component Design**	Match components of two problems to relate them
23. **Configuration definition**	A string uqv such that v member of gamma*, q member of Q, v member of r+
24. **Conjunctive Normal Form**	An expression C1 ^ C2 ^ ... ^ Cn is in conjunctive normal form if C1...Cn are disjunctions of literals
25. **Conjunctive Normal Form (CNF)**	conjunctions of clauses
26. **Constant time**	Complexity of O(1), the size of the dataset doesn't matter, the algorithm will always take the same amount of time to execute. These algorithms scale perfectly.
27. **Context Free language closed under**	Star • Concatenation Union No intersect or complement
28. **Context Sensitive Grammar**	Each a -> B satisfies |a| <= |B| except S -> ^ Every context free language is also context sensitive
29. **Counting argument**	counts all possible configs
30. **Crash Avoidance**	For every TM, M, there is a TM, M', such that L(M)=L(M') and M' does not crash on any input. -Extend gamma with a new symbol # -M' writes # on the first square, inserts blank on the 2nd by shifting the input string to the right -moves the tape head to the 2nd square -executes the transitions on M
31. **Decidable in Polynomial Time/Space**	A language L is decidable in polynomial time if there is a TM deciding L such that τM/SM is a member of O(n^r) for some r is a member of N.
32. **Decidable (recursive)**	Language L is decidable if it is accepted by some TM, L, that on every input eventually reaches a halting config. We say M decides L.
33. **Decision problem**	A problem that only has answers "yes" and "no"
34. **Define a mapping reduction of the acceptance problem Atm to the halting problem HALT TM**	For every TM M, let M* be the following TM: M* = "On input x: 1. Run M on x. 2. If M accepts, accept. 3. If M rejects, enter an infinite loop." Thus: If M accepts input x, then M* accepts x If M explicitly rejects x, then M* never halts on x If M never halts on x, then M*never halts on x To summarize, M accepts x iff M* halts on x Let then f be the function defined by f(<M,w>)=<M*,w>. Is f computable? Yes Obviously <M,w> is in ATM iff f(<M,w>) is in HALT TM i.e. f is a mapping reduction of Atm to HALT TM So, since ATM is undecidable, HALT TM is undecidable as well.
35. **Define a Nondeterministic Turing Machine (NTM)**	- The reject state hr is dropped - delta has the form (Q-{ha}) x gamma -> 2^(Q x gamma x {L,R,S})

36. **Define Mapping Reduction and Mapping Reducibility**	Let A and B be languages over an alphabet Sigma. We say that A is mapping reducible to B, written A<=mB, if there is a computable function f: Sigma*->Sigma* such that, for every w is in Sigma*, w is in A iff f(w) is in B. The function f is called a mapping reduction of A to B.
37. **Definition of Accept**	A TM accepts an input string iff, for this input, sooner or later it enters the accept state. Otherwise the string is considered rejected.
38. **Definition of acceptance for a Finite State Automata M**	M accepts the string u1 u2 ... un iff there is a sequence r1, r2, ..., rn, rn+1 of states such that: r1=s ri+1 = delta(ri,ui), for each i with 1<= i <= n rn+1 is in F
39. **Definition of a CFG**	A 4-tuple (V,Sigma,R,S) 1. V is a finite set called the variables; 2. Sigma is a finite set, disjoint from V, called the terminals; 3. R is a finite set of rules, with each rule being a pair of a variable and a string of variables and terminals; 4. S is an element of V called the start variable.
40. **Definition of a regular expression**	R is a Regular Expression (RE) iff R is one of the following: 1. a, where a is a symbol of the alphabet 2. The Empty Set 3. The Null Set 4. (R1)U(R2), where R1 and R2 are RE 5. (R1) o (R2), where R1 and R2 are RE 6. (R1)*, where R1 is a RE
41. **Definition of Asymptotic Upper Bound**	Let f and g be functions f,g: N->R+. Say that f(n) = O(g(n)) iff positive integers c and n0 exists such that for every integer n>=n0, f(n) Less than or equal to cg(n). When f(n) = O(g(n)), we say that g(n) is an asymptotic upper bound for f(n).
42. **Definition of Cartesian Product for sets S and T**	S X T={(s,t) | s is in S and t is in T}
43. **Definition of Concatenation**	L1 o L2 = {xy | x is in L1 and y is in L2}
44. **Definition of Decidable Relative**	Language A is decidable relative to language B iff there is an OTM with an oracle for B that decides A.
45. **Definition of Decider**	A Turing machine is said to be a decider iff it halts for every input.
46. **Definition of Halting Problem**	Let HALT TM = {<M,w> | M is a TM and M halts on input w}
47. **Definition of Oracle Turing Machine**	An oracle Turing machine (OTM) is a modified Turing machine that has the additional capability of querying an oracle. An oracle for a language B is an external device that is capable of reporting whether any string w is a member of B.
48. **Definition of Star**	L* = {x1...xk | k >= 0 and each xi is in L}

49.	**Definition of the class of languages P**	P is the class of languages that are decidable in polynomial time on a deterministic (single-tape) TM. In other words, P = TIME(n^1) U TIME(n^2) U TIME(n^3) U TIME(n^4) U ...
50.	**Definition of time complexity for nondeterministic machines**	Let M be a nondeterministic TM that is a decider (meaning that, on every input, each branch of computation halts). The running time or time complexity of M is the function f: N->N, where f(n) is the maximum number of steps that M uses on any branch of its computation on any input of length n.
51.	**Definition of Turing Reducibility**	Language A is Turing reducible to language B, written A<=TB, iff A is decidable relative to B.
52.	**Definition of Union**	L1 U L2 = {x | x is in L1 or x is in L2}
53.	**Definitions of computing, computability, and the graph of a function**	A function g: Sigma*-> Sigma* is said to be computable, iff there is a TMO M such that for every input w is in Sigma* M returns the output u with u=g(w). In this case we say that M computes g. The graph of such a function is the language {(w,u) | w is in Sigma*, u=g(w)}
54.	**Definitions of NP, coNP, EXPTIME**	NP is the class of languages decided by some nondeterministic polynomial time Turing machine. coNP = {L | L is the complement of some language in NP} EXPTIME = TIME(2^(n^(1))) U TIME(2^(n^(2))) U TIME(2^(n^(3))) U ...
55.	**Definitions of Running Time and Time Complexity**	Let M be a deterministic TM that halts for every input. The running time or time complexity of M is the function f: N->N, where f(n) is the maximum number of steps that M uses on any input of length n. If f(n) is the time complexity of M, we say that M runs in time f(n), or that M is an f(n) time machine.
56.	**Each CFG needs what in it to end it**	an empty string
57.	**Example of a Logarithmic algorithm**	Binary search, which increases the number of steps to complete by 1 every time the data set doubles. A binary search of 4 numbers may take t steps, but a binary search of 2,000,000 would be completed in 20t steps.
58.	**Example of Linear Algorithm**	A program that uses a loop to add up all the numbers of an input would have complexity O(n), a set of 5 numbers would take 5 steps and a set of 100 numbers would take 100 steps.
59.	**Examples of Language Properties**	- L(M) is finite - L(M) = sigma
60.	**Examples of things that are NOT language properties**	- M halts on every input - e(M) member of L(M)
61.	**Exponential Time**	Complexity pf O(n^x). When x > 0, increase at a rate even faster than polynomial complexity. 2 steps with 10 inputs would take 2^10 (1024) steps.
62.	**EXPTime**	The class of all languages decidable in exponential time - there is a TM deciding L such that τM is a member of O(2^n^r)
63.	**Formal definition of a finite automaton**	a 5-tuple (Q, Sigma, delta, s, F) Q -> finite set called the states Sigma -> finite set called the alphabet delta -> function of the type Q X Sigma -> Q called the transition function s -> an element of Q called the start state F -> a subset of Q called the set of accept states

64. **Given functions f,g: N -> R+ what does it mean for f to be of order g?**	f if of order g if there are positive integers c and n0 such that f(n) <= c . g(n) for all n >= n0
65. **Grammar ending with something/ starting with somehting** **Odd grammar**	S -> Bb B-> ~~ S->bB B->~~ S->SSS
66. **Graph of partial function f: sigma -> sigma**	The set Graph(f) = {(v, w) member of sigma* X sigma* | f(v) = w}
67. **Growth Rates of Functions equivalencies**	- Loga(n) is a member of O(n) but n is not a member of O(loga(n)) - n^r is a member of O(b^n) but b^n is not a member of O(n^r) - b^n is a member of O(n!) but n! Is not a member of O(b^n)
68. **Hamilton Circuit Problem**	Does G have a path through all nodes such that v0=vn and v1...vn are pairwise distinct (visits every node once with no repeats)
69. **How to show context free**	You can create a CFG from it
70. **If L1 and L2 are languages such that L1 is reducible to L2, and L2 is decidable/semidecidable then**	L1 is also decidable/semidecidable
71. **If S is countable then the powerset (2^S) is uncountable**	True
72. **Implementation level**	Remember to have "M on input w: 1. check if it is in the form if not then reject 2. Mark everything 3. Check if everything is marked, if not then reject, else accept
73. **Independent Set**	a discrete subgraph; a set of nodes from a graph such that no two nodes are adajcent
74. **Inherently sequential problems**	problems for which no significant speedup can be achieved by parallel computation
75. **Instances**	In a set of questions each of which has the answer yes or no, questions are the instances of the decision problem
76. **Intractable Problem**	A problem that cannot be solved by any polynomial time bounded algorithm
77. **In what form does the Universal Turing Machine (U) take inputs?**	e(M)e(w) where M is a TM and w is the string over M's sigma
78. **Isometric**	Two graphs the same up to the renaming of the vertices
79. **Language property**	A property, P, of TMs is a language property if for every TM, M, with property P, and every TM, M', with the same language as M, M' has property P
80. **Languages of PDA can include what**	They can include expressions such as Union, Star ex. {b^n(aUb) ^n | n>=0 }

81.	**Las Vegas Algorithms**	Iterating random algorithms until a definite answer is reached (zero error probability, but may never answer; expected poly-time) A ∈ ZPP so A ∈ RP and ~A ∈ RP. Check A and ~A interleaved and answer if one accepts
82.	**Linear Bounded Automata**	Like a NTM except that: -the initial config for an input w is qo(w) where <,> member of gamma - sigma - the tape head cannot move left of < or right of > - < and > cannot be overwritten L is accepted by LBA if and only if L is context sensitive
83.	**Linear time**	Complexity of O(n), increases at exactly the same rate as the data set.
84.	**Literal**	a variable or its negation
85.	**literal**	A variable x or a negated variable xbar
86.	**L(M), a.k.a., The language recognized by M**	the set all strings that the machine M accepts
87.	**Logarithmic time**	Complexity of O(log(n)), increase their execution time at a slower rate than the rate at which the data set increases (n here stands for the size of the data set). These algorithms scale very well.
88.	**log-space uniform (Ubc-uniform)**	a circuit family C for which there exists a DLOG machine to construct every circuit in C
89.	**Monte Carlo Algorithms**	iterating an RTM with a RNG on the same input and say YES if any iterations accepts; else say NO. P(M(x) = 1 | x ∈ L) = (1-ε)^i P(M(x) = 0 | x ∈ L) = 1 - (1-ε)^i P(M(x) = 1 | x ∉ L) = 0 P(M(x) = 0 | x ∉ L) = 1
90.	**NP-Complete**	A language is NP-Complete if it is in NP-Hard and belongs to NP
91.	**NP-Hard**	A language L is NP-Hard if every language in NP is polynomial time reducible to L
92.	**NSA**	SAbar
93.	**N P**	Class of all languages that are accepted in nondeterministic polynomial time
94.	**Padding argument**	Adds padding to input string
95.	**Parallelizable problems**	problems that can be decomposed into smaller subproblems which can be executed by using multiple processors
96.	**Parallel RAM (PRAM)**	a network of RAMs running concurrently on a global, shared memory (CRCW, CREW, EREW)
97.	**Partition Problem**	Given a finite set A and its size set S, decide if there exists a partition such that the sum of the sizes of the two sets are equal
98.	**Polynomial Time**	Complexity of O(n^x). When x > 0, execution time increases at a greater rate than an algorithm with linear complexity If a function with quadratic complexity - O(n^2) - takes one step with one input value, increasing to 10 input values it will take 10^2 (100) steps. A function with cubic complexity - O(n^3) - would increase its execution time even quicker as the size of the input grows.

99.	**Precedence of operators in Regular Expressions**	*, o, U
100.	**Probabalistic TM**	An NTM that accepts its input iff more than half of all paths terminate in YES (accepts by majority vote, not existence of one path; repeating this machine does not increase the error probability)
101.	**Probabilistic poly-time class (PP = coPP)**	languages defined by poly-time PTMs
102.	**Proof: Accept-A is undecidable but semidecidable**	- Reduce MP to A-A. Suppose that A-A is decidable then the language of Y(A-A) is decidable - Therefore there exists a TM, TA, which accepts Y(A-A) and halts on all inputs - Let C be a TM which transforms an input e(M)e(w) into the code e(Mw) of a TM Mw - Machine Mw replaces its input with w and runs M on w. - We combine C with TA to obtain TMP which solves MP. CASE 1: w is a member of L(M) - Mw accepts every input so A is a member of L(Mw) - TA accepts e(Mw) and TMP accepts e(M)e(w) CASE 2: w is not a member of L(M) - Mw accepts no inputs so A is not a member of L(Mw) - TA rejects e(Mw) and TMP rejects e(M)e(w) But, MP is undecidable, so the assumption that A-A is decidable must be false. To show that A-A is semidecidable, construct a TM TA' which on input e(M) runs M on A and accepts e(M) if and only if M accepts A.
103.	**Proof: Complement of decidable language L is also decidable**	- Let M be a TM that accepts L and halts on every input - Assume M never crashes at the start of the tape - Modify M to a machine Mbar by swapping the accept and reject state - L(Mbar) = Lbar - Mbar halts on every input because M does so therefore Lbar is decidable
104.	**Proof: For every alphabet (sigma), there are uncountably many languages over sigma what are not semidecidable**	- Every semidecidable language over sigma is accepted by some TM (from definition) - Since TMs can be encoded as strings over {1,0} (which is countably infinite) the set of all TMs with input alphabet sigma is countable infinite too. - Therefore the set of all semidecidable languages over sigma is countable too - But 2^sigma* is uncountable - Property 1 of the countable sets Lemma follows that 2^sigma* - {L member of sigma* | L is semidecidable} is also uncountable

105. **Proof: If both L and its complement are semidecidable then L is decidable**	Let M and Mbar be TMs that accept L and Lbar respectively. The idea is to construct a 2-tape TM which copies its input to Tape 2 and then simulates M and Mbar in parallel (see notes for rest)
106. **Proof: NSA is not semidecidable**	- If NSA is semidecidable then there is a TM, M, such that L(M) = NSA - Consider e(M) - Case 1: e(M) member of L(M), then by assumption e(M) member of NSA. Then e(M) member of L(M). A contradiction - Case 2: e(M) not member of L(M). Then by NSA definition e(M) member of NSA. Another contradiction
107. **Proof: SA is not decidable**	NSA = SAbar = not semidecidable therefore SA cannot be decidable as the complement of a decidable language is decidable
108. **Proof: SA is semidecidable**	A TM, Tsa, which accepts SA works as follows: - Checks whether its input, w, is the code of some TM False: reject w True: Tsa constructs a TM, M, such that e(M) = w. If {0,1) is not a subset of M's input alphabet then reject w, else Tsa exectures M on w and accepts w if and only if M accepts w Therefore L(Tsa) = SA
109. **Proof: SAP is undecidable but semidecidable**	Using encoding e for TMs, we have Y(SAP) = {e(M) | M is a TM and e(M) member of L(M)} = SA
110. **Proof:** **- Show that the Accept-A problem is reducible either to PPP or its complement, PPPbar. PPP is decidable if and only if PPPbar is.** **- Let M(/) be a TM which rejects every input. If M(/) doesn't have P then reduce Accept-A to PPP, otherwise reduce Accept-A to PPPbar**	CASE 1 - M(/) doesn't have P. 1. Construct a reduction of A-A to PPP. Let M be a TM, an input to A-A. 2. Modify M to a TM, MA, which operates as follows: - If M rejects A, MA rejects input - If M loops on A, MA loops on inputs - If M erases A, MA erases the tape contents to the right of its original inputs, moves head back to first sequence and executes on its original input, a TM, MP, which has property P. - If M accepts A then L(MA) = L(MP). Hence MA has P - If M doesn't accept A, then L(MA) = (/). Hence L(MA) = L(M(/)). Since P is a language property and M(/) does not have P, MA does not have P either. CASE 2 - M(/) has P. 1. Construct a reduction of A-A to PPP analogously to Case 1, replacing MP with a machine MPbar that does not have P. - If M accepts A then L(MA) = L(MPbar) therefore MA does not have P - If M does not accept A then L(MA) = (/) and L(MA) = L(M(/)) therefore since M(/) has P, MA has P

111.	**Proof: The complete subgraph problem (clique problem) and the Hamilton Circuit problem are in NP**	- A NTM guesses nodes v1...vk in G and checks for each pain {vi,vj} whether this is on edge - The checking can be done in polynomial time in K + |G|. Where G is represented by a string of node numbers followed by a string of edge where edge are pairs of node numbers. - At mist, k^(2)/2 node pairs are checked which can be done with O(k^(2)/2 x |E|) quadratic time graph operations
112.	**Proof: The membership problem for semidecidable languages is undecidable but semidecidable**	- Reduce SAP to this problem so the decidability of MP implies decidability of SAP - Suppose MP is decidable, then Y(MP) is decidable -Construct a TM, Tsa, which solves SAP: Tsa accepts e(M) if and only if Tmp accepts e(M)e(e(M)), so only if e(M) member of L(M) - Tsa halts on all inputs because Tmp does - Tsa decides SA because Tsa solves SAP - But SAP is undecidable (a contradiction) - Therefore the assumption that MP is decidable must be false. MP is undecidable - Consider a universal TM, U. Y(MP) = L(U). MP is semidecidable.
113.	**Proof: There are uncountably many languages over every alphabet**	Since sigma* is infinite, 2^sigma* is uncountable by countable set Lemma 2 [If S is countable then the powerset (2^S) is uncountable]
114.	**Properties of encoding e**	1. There is an algorithm which for every string w member of {0,1}* decides whether w is the code of some TM and, if this is the case, constructs as machine M such that e(M) = w 2. If M and M' (some other TM) are TMs such that e(M) = e(M'), then L(M) = L(M') and they halt on the same inputs
115.	**Prove a language is Turing recognizable iff it is enumerable.**	Proof of if: Suppose E enumerates L. Construct a TM M that works as follows: M = "On input w: 1. Simulate E. Every time E prints a new string, compare it with w. 2. If w is ever printed, accept." Proof of only if: Suppose M recognizes L. Let s1,s2,s3,... be the lexicographic list of all strings over the alphabet of L. Construct an enumerator E that works as follows: E = " 1. Repeat the following for i=1,2,3,... 2. Simulate M for i steps on each of the inputs s1,s2,...,si. 3. If any computations accept, print out the corresponding sj."
116.	**Prove every nondeterministic TM has an equivalent deterministic TM.**	Simulate every possible branch of computation in a breadth-first manner.

117. **Prove every t(n) multitape Turing Machine has an equivalent O(t^2 (n)) time single tape TM**	Convert a multitape TM M into an equivalent single tape TM S that simulates M The simulation of each step of M takes O(k) steps in S, where k is the length of the active content of the tape of S specifically, S makes two passes through its tape; a pass may require shifting, which still takes O(k) steps). How big can k be? Not bigger than the number of steps M takes, multiplied by the (constant) number c of tapes. That is, k <= ct(n)). Thus, S makes O(t(n)) passes through the active part of its tape, and each pass takes (at most) O(t(n)) steps. Hence the complexity of S is O(t(n)) x O(t(n)) = O(t^2 (n)).
118. **Prove HALT TM is undecidable**	Assume, for a contradiction, that HALT TM is decidable. I.e. there is a TM R that decides HALT TM. Construct the following TM S: S = "On input <M,w>, an encoding of a TM M and a string w: 1. Run R on input <M,w>. 2. If R rejects, reject. 3. If R accepts, simulate M on w until it halts. 4. If M has accepted, accept; if M has rejected, reject." If M works forever on w, what will S do on <M,w>? Explicitly Reject If M accepts w, what will S do on input <M,w>? Accept If M explicitly rejects w, what will S do on <M,w>? Explicitly Reject Thus, S decides the language, which is impossible
119. **Prove If A<=mB and B is decidable, then A is decidable.**	Let DB be a decider for B and f be a reduction from A to B. We describe a decider DA for A as follows. DA= "On input w: 1. Compute f(w). 2. Run DB on input f(w) and do whatever DB does."
120. **Prove Let t(n) be a function, where t(n)>=n. Then every t(n) time single-tape nondeterministic TM has an equivalent 2^(O(t(n))) time single-tape TM.**	Convert a nondeterministic TM N into an equivalent deterministic TM D that simulates N by searching N's nondeterministic computation tree. Each branch of that tree has length at most t(n), and thus constructing and searching it takes O(t(n)) steps. And the number of branches is b^(O(t(n))), where b is the maximum number of legal choices given by N's transition function. But b<=2c for some constant c. So, the number of branches is in fact 2^(cO(t(n)))=2^(O(t(n))). Thus, the overall number of steps is O(t(n)) x 2^(O(t(n))) = 2^(O(t(n))).
121. **Prove that CLIQUE is in NP**	Here is a polynomial time NTM N deciding CLIQUE: N = "On input <G,k>: 1. Nondeterministically select a subset c of k nodes of G. 2. Test whether G contains all edges connecting nodes n c. 3. If yes, accept; otherwise reject."
122. **Prove that every multitape TM M has an equivalent single-tape TM S.**	Copy the contents of each tape in M onto the single tape of S, marking the boundaries between the tapes with a special character. Modify S so that it can move between the boundaries easily.

123. **Prove that if a language A is mapping reducible to a language B and B is recognizable, then A is recognizable**	DB is a Turing Machine that recognizes B and f is a reduction from A to B. A Turing Machine DA recognizes A DA = "on input w: 1. compute f(w) 2. run DB on input f(w) and do whatever DB does"
124. **Prove that if a language L and its complement NOT L are both recognizable, then L is decidable**	The Turing Machine U recognizes the language L The Turing Machine NOT U recognizes the language NOT L let M = "on input w: 1. run U and NOT U on input w in parallel 2. if U reaches its accept state, accept if NOT U reaches its accept state, reject therefore, M decides L
125. **Prove that SUBSET-SUM is in NP**	Here is a polynomial time NTM N deciding SUBSET-SUM: N = "On input <S,t>: 1. Nondeterministically select a subset c of S. 2. Test whether the elements of c sum up to t. 3. If yes, accept; otherwise reject."
126. **Prove that the Complement of Atm (NOT Atm) is unrecognizable**	Atm = {<M,w> | M is a TM and M accepts string w} NOT Atm = {<M,w> | M is a TM and M does not accept string w} Suppose, for a contradiction, that NOT Atm is Turing-recognizable. That is, there is a TM U that recognizes NOT Atm. Thus: U recognizes Atm (slide 4.2.a) NOT U recognizes NOT Atm Let M = "On input w: 1. Run both U and NOT U on input w in parallel; 2. If U accepts, accept; if NOT U accepts, reject." It can be seen that M decides ATM, which is impossible
127. **Prove that the halting problem is recognizable**	HALT TM = {<M,w>| M is a TM and M halts on input w} the TM U (universal Turing Machine) recognizes HALT TM: U= "On input <M,w>, where M is a TM and w is a string: 1. Simulate M on input w 2. if M enters accept state, accept if M enters reject state, accept
128. **Prove that Th(N,+,x) is unrecognizable**	Suppose a TM M recognizes Th(N,+, x). Construct a TM D: D = "On input A, and arithmetic sentence, 1. Run M on both A and NOT A in parallel. 2. If M accepts A, accept; if M accepts NOT A, reject" Obviously D decides Th(N,+, *) , which is a contradiction

129.	**Prove the Universal Turing Machine is recognizable**	The following TM U, called the universal TM, recognizes ATM: U = "On input <M,w>, where M is a TM and w is a string: 1. Simulate M on input w. 2. If M ever enters its accept state, accept; if M ever enters its reject state, reject."
130.	**Prove the Universal Turing Machine is undecidable**	Suppose, for a contradiction, that ATM is decidable. That is, there is a TM H that decides ATM. Thus, that machine H behaves as follows: H(<M,w>)= accept if M accepts w reject if M does not accept w Using H as a subroutine, we can construct the following TM D: D = "On input <M>, where M is a TM: 1. Run H on input <M,<M>>. 2. Do the opposite of what H does. That is, if H accepts, reject, and if H rejects, accept." Thus, D(<M>)= accept if M does not accept <M> reject if M accepts <M> But then D(<D>) = accept if D does not accept <D> reject if D accepts <D> contradiction! To summarize: H accepts <M,w> exactly when M accepts w. D rejects <M> exactly when M accepts M. D rejects <D> exactly when D accepts <D>.
131.	**Pseudo-polynomial time algorithm**	An algorithm for a number problem that is poly-time bounded if the involved numbers are polynomially bounded in the input size
132.	**Pumping lemma for Context free grammars**	For every context-free language L, there is a number p, such that for every string s, which is in L, whose length is at least p, there are strings u, v, x, y, z, such that s = uvxyz and the following conditions are satisfied: 1.) for every i >= 0, uv^ixy^iz is in L; 2.) |vy| > 0; 3.) |vxy| <= p.
133.	**Pumping Lemma for Regular Expressions**	For every regular language L, there is a number p, such that for every string s, which is in L, whose length is at least p, there are strings x, y, z, such that s = xyz and the following conditions are satisfied: 1.) for every i >= 0, xy^iz is in L; 2.) |y| > 0; 3.) |xy| <= p.
134.	**Randomized Algorithms**	There exist problems for which the use of a random number generator allows for finding the exact solution most of the time
135.	**Randomized poly-time class (RP)**	class of problems defined by poly-time RTMs

136. **Random Turing Machine**	a restricted NTM with a clock such that the computation tree is a full binary tree with either YES (accepting) or NO (rejecting) in each leaf node. For every input, there must be either no accepting path or at least ε (1/2) of the paths are accepting
137. **Reducible**	Let L1,L2 be subsets of sigma*. L1 is reducible to L2 is there is a computable total function f: sigma* -> sigma* such that x member of L1 if and only if f(x) member of L2
138. **Regular languages closed under**	Union • Intersection • Complement • Star • Concatenation
139. **Relationship between Space and Time complexity of a TM**	For every TM, M: SM(n) <= τM(n) + 1
140. **Relativity Concept**	If a class is closed under a certain resource-bounded reduction, any larger class is also closed under that reduction
141. **Remeber language is in what notation and what do you need to include with it**	include an n and define it, also include set otation
142. **REMEMBER what does a^0 b^0 represent**	it represents empty, not 1
143. **Rice's Theorm**	Every nontrivial language property of Turing Machines is undecidable. That is, for each such property P, the Property P Problem is undecidable
144. **Self-Accepting definition**	SA = {e(M) | M is a TM and e(M) member of L(M)}
145. **Semidecidable Language (recursively enumerable)**	A language is semidecidable if there exists a TM that accepts it
146. **Show that PATH is in P**	A polynomial time algorithm M for PATH works as follows: On input <G,s,t>, where G is a directed graph with nodes s and t, 1. Place a mark on node s 2. Repeat until no additional nodes are marked: 3. Scan all edges of G. If an edge (a,b) is found going from a marked node a to an unmarked node b, mark node b. 4. if t is marked, accept. Otherwise, reject
147. **Show that RELPRIME is in P**	E = "On input <x,y>, where x and y are natural numbers, x>y: 1. Repeat until y=0. 2. Assign x <- x mod y. 3. Exchange x and y. 4. Output x." R = "On input <x,y>, where x and y are natural numbers: 1. Swap x and y if necessary so that x>y. 2. Run E on <x,y>. 3. If the result is 1, accept. Otherwise reject.
148. **Show that SAT is in NP**	- Use guess and check - NTM, N, checks if its input string is a boolean expression C in conjunctive normal form - Then, N guesses an assignment of truth values to variables and checks if this makes C true - If so, N accepts - Therefore N accepts is and only if C is satisfiable, and terminates after a polynomial number of steps

149. **Show that the Atm is Turing reducible to HALT TM**	For every TM M with input string w, consider the TM M*, which takes input <M,w> M* = "on input <M,w> 1. simulate M on w 2. if M reaches accept, accept 3. if M reaches reject, enter an infinite loop f is the computable function through which f(<M,w>) = <M*,w> Therefore, <M,w> is in Atm iff f(<M,w>) is HALT TM
150. **Show the nonregularity of B = {0n1n | n>=0}**	Proof by contradiction: Assume B is regular. Let then p be its pumping length. Select w is in B with |w| >= p. By the pumping lemma, w=xyz and y can be pumped, so that we must also have xyyz is in B. Case 1: y only has 0s. But then xyyz has more 0s than 1s and is not in B. Case 2: y only has 1s. But then xyyz has more 1s than 0s and is not in B. Case 3: y has both 0s and 1s. But then xyyz has a 1 followed by a 0 and is not in B. All cases lead to contradiction, therefore B is not regular
151. **Show the nonregularity of F = { ww | w is in {0,1}* }**	Proof by contradiction: Assume F is regular. Let then p be its pumping length. Observe that 0p10p1 is in F. By the pumping lemma, 0p10p1 =xyz and y can be pumped. By Condition 3, |xy|<=p. Therefore, y is entirely in the first 0p. Pumping y would produce a string that has only 0s is the first half, and two 1s in the second half. Obviously this string cannot be in F, which contradicts with the pumping lemma.
152. **Space Complexity of a TM**	SM(n) is the max number of tape squares M can visit on any input of length n
153. **Space Hierarchy Theorem**	DSPACE(o(f(n))) is a proper subset of DSPACE(f(n))
154. **String over Sigma**	A finite sequence of symbols from sigma
155. **Strongly NP-complete**	An NP-complete problem with no pseudo-poly time algorithm (assumes NP is not P)
156. **Theorm: There are arbitrarily complex languages**	Let f: N -> N be a total computable function. Then there is a decidable language L such that for every TM accepting L, τM is not bounded by f
157. **Theorm: There need not exits a fastest TM**	There is a decidable language L such that for every TM accepting L, there is a TM M' such that L(M') = L(M) and τM' is a member of O(log2(τM))
158. **Time Complexity of a TM**	τM(n) is the max number of transitions M can make on any input of length n
159. **Time-constructible function**	A function for which there exists a TM that halts after exactly f(|x|) steps for every input x
160. **Time Hierarchy Theorem**	DTIME(o(f(n)/log(f(n))) is a proper subset of DTIME(f(n))
161. **tractable**	A language, L, is tractable if L is a member of P
162. **Transition definition**	Uqv -> xry describes effect of a single move on a TM
163. **Translation**	Relations between complexity classes translate upwards (padding argument)
164. **Trivial**	A language property, P, of a TM is trivial if every or no TM has P
165. **Uniform circuit family**	a set of Boolean circuit C = {C0, C1, ...} such that (the encoding of) each Cn can be constructed from n by using an algorithm
166. **The union of countable sets is also countable**	True
167. **What are TMs with TMs as inputs used for**	To see if a language is decidable or not. To do this, TMs must be encoded as strings

168. **What does it imply if L1 is polynomial time reducible to L2**	L2 is a member of P or NP implies L1 is a member of P or NP respectively
169. **What does it mean for a decision problem P to be in P/PSpace**	The language Y(P) is in P/PSpace when polynomial time encoding/decoding is assumed
170. **What does the stack not need to be for PDA**	it doesnt need to be empty when reaching final state, but the input does need to
171. **What does U do?**	Simulates M on input w where it: - accepts e(M)e(w) if M accepts w - rejects e(M)e(w) if M rejects w - diverges on e(M)e(w) if M diverges on w U is known as an interpreter for TMs
172. **What happens if there is there is more than one possible sequence for a NTM given input string?**	Input string is executed if there exists a transition sequence ending in ha. For every NTM, N, there is a TM, D, such that L(N) = L(D). D tries all branches of N in breath first. If D finds an accepting state it accepts otherwise it rejects
173. **What happens if you get to final state but still ahve stuff leftover?**	Don't accept,
174. **What is delta in a TM definition**	(Q - {ha, hr})x gamma -> Q x gamma x {L, R, S} is the transition function
175. **What is E(P) where P is a decision problem?**	language Y(P) U N(P) consists of all encoded instances
176. **What is gamma in a TM definition**	Tape alphabet including blank symbol
177. **What is q0 in a TM definition**	q0 member of Q is the initial state
178. **What is Q in a TM definition**	Finite set of states, including ha and hr
179. **What is sigma in a TM definition**	Input alphabet
180. **What is the format for pop and push**	push is the very last thing, pop is the middle thing , read is the first thing
181. **What is the membership problem?**	Given a TM, M, and a string w. Does M accept w?
182. **What is the relationship between a languages characteristic function and its decidability**	L is decidable if and only if it's char. Function is computable
183. **What is the transition function for a multitape TM?**	Delta: (Q-{ha,hr} x gamma^n -> Q x gamma^n x {L,R,S}^n where n>=2
184. **What is the variable, temrinal , start**	The start is the first thing, Variables include the start state Terminal is the end thing. Terminal does not include empty sttring
185. **What is Y(P) where P is a decision problem?**	Language of encoded yes instances
186. **What machine matches context sensitive languages**	Linear-bounded automaton
187. **What machine matches unrestricted / semidecidable languages?**	TM
188. **What tuple defines a TM?**	(Q, sigma, gamma, q0, delta)
189. **When can language L be enumerated by some multitape TM?**	If and only if L is semidecidable

190. **When does a multitape TM enumerate language L?**	1. The computation begins with all tapes blank 2. The tape head on tape 1 never moves left 3. At each point in the computation, the contents of the tape 1 is in the form [blank]#w1...#wn...#v 4. Every w member of L will eventually appear as one of the strings wi on the tape 1
191. **When is a decision problem P decidable?**	If Y(P) is decidable
192. **When is a decision problem P semidecidable?**	If Y(P) is semidecidable
193. **When is an encoding reasonable?**	- Y(P) n N(P) >< (/) - E(P) is decidable - There exists a decoding algorithm which given a code w member of E(P) constructs an instance I such that enc(I) = w
194. **When is a partial function f:(sigma)^k -> sigma turing computable?**	If there exists a TM such that fm = f
195. **When is a partial function f: sigma -> sigma computable?**	If and only if Graph#(f) is semidecidable
196. **When is set S countably infinite?**	If there exists a bijective function {0,1,2...} -> S
197. **When is the total function f: sigma -> sigma computable?**	If and only if Graph#(f) is decidable
198. **ZPP**	RP ∩ coRP; the randomized poly-time class with zero error probability
199. **P (fancy)**	The class of all languages decidable in polynomial time