University of Nevada, Las Vegas
Computer Science 456/656 Spring 2019

Answers to Practice Final for Spring 2019

The entire practice test is 725 points.

The actual final examination will be shorter.

1. [5 points each] True or False. If the question is currently open, write "O" or "Open."

   (i) **F** Every subset of a regular language is regular.

   (ii) **T** The intersection of any context-free language with any regular language is context-free.

   (iii) **T** The complement of every recursive language is recursive.

   (iv) **F** The complement of every recursively enumerable language is recursively enumerable.

   (v) **T** Every language which is generated by a general grammar is recursively enumerable.

   (vi) **T** The question of whether two context-free grammars generate the same language is undecidable.

   (vii) **T** There exists some proposition which is true but which has no proof.

   (viii) **T** The set of all binary numerals for prime numbers is in the class $\mathcal{P}$.

   (ix) **T** If L_1 reduces to L_2 in polynomial time, and if L_2 is $\mathcal{NP}$, and if L_1 is $\mathcal{NP}$-complete, then L_2 must be $\mathcal{NP}$-complete.

   (x) **F** Given any context-free grammar G and any string $w \in L(G)$, there is always a unique leftmost derivation of w using G.

   (xi) **F** For any deterministic finite automaton, there is always a unique minimal non-deterministic finite automaton equivalent to it.

   (xii) **T** The question of whether two regular expressions are equivalent is $\mathcal{NP}$-complete.

   (xiii) **T** The halting problem is recursively enumerable.

   (xiv) **F** The complement of every context-free language is context-free.

   (xv) **F** No language which has an ambiguous context-free grammar can be accepted by a DPDA.

   (xvi) **T** The union of any two context-free languages is context-free.

   (xvii) **F** The question of whether a given Turing Machine halts with empty input is decidable.

   (xviii) **T** The class of languages accepted by non-deterministic finite automata is the same as the class of languages accepted by deterministic finite automata.

(xix) **F** The class of languages accepted by non-deterministic push-down automata is the same as the class of languages accepted by deterministic push-down automata.

(xx) **T** The intersection of any two regular languages is regular.

(xxi) **F** The intersection of any two context-free languages is context-free.

(xxii) **T** If L_1 reduces to L_2 in polynomial time, and if L_2 is $\mathcal{NP}$, then L_1 must be $\mathcal{NP}$.

(xxiii) **T** Let $F(0) = 1$, and let $F(n) = 2^{F(n-1)}$ for $n > 0$. Then F is recursive.

(xxiv) **T** Every language which is accepted by some non-deterministic machine is accepted by some deterministic machine.

(xxv) **F** The language of all regular expressions over the binary alphabet is a regular language.

(xxvi) **T** Let π be the ratio of the circumference of a circle to its diameter. (That's the usual meaning of π you learned in kindergarten.) The problem of whether the n^{th} digit of π, for a given n, is equal to a given digit is decidable.

(xxvii) **T** There cannot exist any computer program that can decide whether any two C++ programs are equivalent.

(xxviii) **F** An undecidable language is necessarily $\mathcal{NP}$-complete.

(xxix) **T** Every context-free language is in the class $\mathcal{P}$-TIME.

(xxx) **T** Every regular language is in the class $\mathcal{NC}$

(xxxi) **F** Every function that can be mathematically defined is recursive.

(xxxii) **T** The language of all binary strings which are the binary numerals for multiples of 23 is regular.

(xxxiii) **F** The language of all binary strings which are the binary numerals for prime numbers is context-free.

(xxxiv) **F** Every bounded function from integers to integers is Turing-computable. (We say that f is *bounded* if there is some B such that $|f(n)| \leq B$ for all n.)

(xxxv) **F** The language of all palindromes over $\{0, 1\}$ is inherently ambiguous.

(xxxvi) **F** Every context-free grammar can be parsed by some deterministic top-down parser.

(xxxvii) **T** Every context-free grammar can be parsed by some non-deterministic top-down parser.

(xxxviii) **F** Commercially available parsers cannot use the LALR technique, since most modern programming languages are not context-free.

(xxxix) **F** The boolean satisfiability problem is undecidable.

(xl) **T** If anyone ever proves that $\mathcal{P} = \mathcal{NP}$, then all one-way encoding systems will be insecure.

(xli) **T** If a string w is generated by a context-free grammer G, then w has a unique leftmost derivation if and only if it has a unique rightmost derivation.

2. [10 points] If there is an easy reduction from L_1 to L_2, then L_2 is at least as hard as L_1.

3. [5 points each] For each language given, write "R" if the language is recursive, write "RE not R" if the language is recursively enumerable but not recursive, and write "not RE" if the language is not recursively enumerable.

   (a) **RE not R** The language consisting of all Pascal programs P such that P halts if given P as its input file.

   (b) **not RE** The language of all encodings of Turing Machines which fail to halt for at least one possible input string.

   (c) **R** The 0-1 Traveling Salesman Problem.

   (d) **not RE** The diagonal language.

   (e) **R** L_{sat}, the set of satisfiable boolean expressions.

4. [15 points] Draw the state diagram for a minimal DFA that accepts the language described by the regular expression `a*a(b+ab)*`

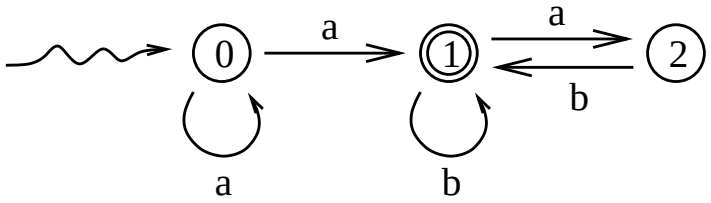

Figure 1: NFA for the regular expression `a*a(b+ab)*`

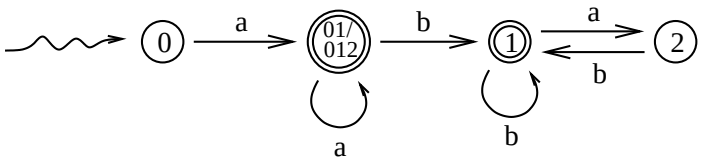

Figure 2: Minimal DFA for the regular expression `a*a(b+ab)*`. Dead state not shown.

5. [15 points] Write a regular expression for the language accepted by the NFA shown in Figure 3.
   I found this one, but there may be simpler ones: $\mathbf{b}^*(\mathbf{a}+\mathbf{c}+\mathbf{ba}^*\mathbf{c})(\mathbf{a}+\mathbf{c}+\mathbf{bb}^*(\mathbf{a}+\mathbf{c}+\mathbf{ba}^*\mathbf{c}))^*$

6. [20 points] Let L be the language of all binary numerals for positive integers equivalent to 2 modulo 3. Thus, for example, the binary numerals for 2, 5, 8, 11, 14, 17 ... are in L. We allow a binary numeral to have leading zeros; thus (for example) $001110 \in L$, since it is a binary numeral for 14. Draw a minimal DFA which accepts L.

8. [10 points] Consider the context-free grammar with start symbol S and productions as follows:

   $S \to s$

   $S \to bLn$

   $S \to wS$

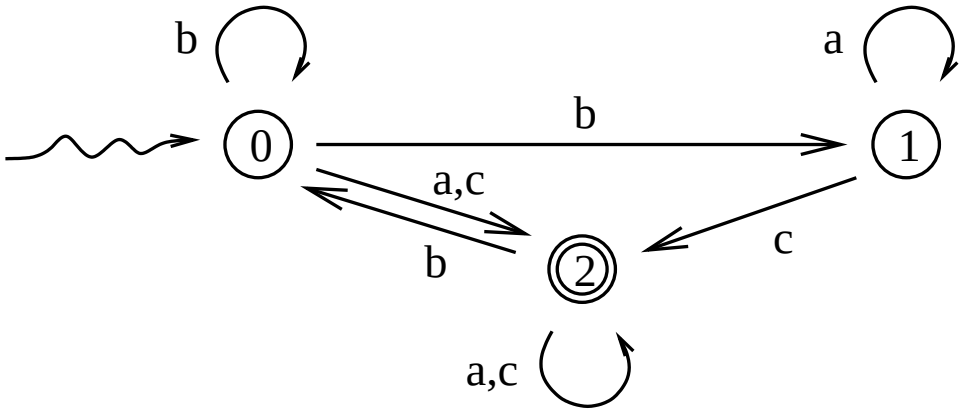

Figure 3: The NFA for Problems 5 and 18.

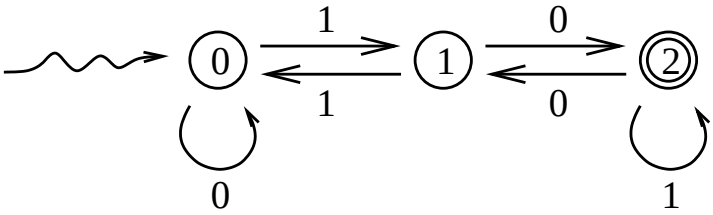

Figure 4: DFA which accepts the language of binary numerals equivalent to 2 modulo 3

$L \to \epsilon$

$L \to SL$

Write a leftmost derivation of the string *bswsbwsnn*

$$S \Rightarrow bLn \Rightarrow bSLn \Rightarrow bsLn \Rightarrow bsSLn \Rightarrow bswSLn \Rightarrow bswsLn \Rightarrow bswsSLn \Rightarrow bswsbLnLn \Rightarrow$$

$$bswsbSLnLn \Rightarrow bswsbwSLnLn \Rightarrow bswsbwsLnLn \Rightarrow bswsbwsnLn \Rightarrow bswsbwsnn$$

7. [20 points] Design a PDA that accepts the language of all palindromes over the alphabet $\{a, b\}$.

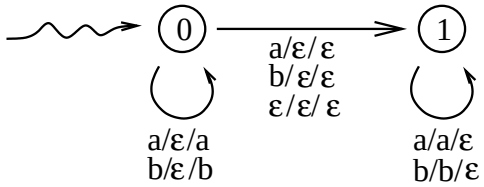

Figure 5: PDA which accepts, by empty stack, the palindromes over $\{\mathbf{a}, \mathbf{b}\}$

9. [5 points] What class of machines accepts the class of context free languages?

   Answer: Push-down automata.

10. [5 points] What class of machines accepts the class of regular languages?

    Answer: Deterministic or non-deterministic finite state automata.

11. [5 points] What class of machines accepts the class of recursively enumerable languages?

    Answer: Turing Machines.

12. [10 points] What is the Church-Turing Thesis, and why is it important?

    Answer: Every machine can be emulated by some Turing machine. It is important because if no Turing machine can perform a certain computation, then no machine can perform that computation.

13. [10 points] What does it mean to say that a language can be recursively enumerated in *canonical order*? What is the class of languages that can be so enumerated?

    Answer: We say that a language L can be enumerated in canonical order if there is some machine that outputs a (possibly infinite) sequence of strings so that the strings that are output are the strings that belong to L, and that the strings are output in canonical order.

    If w_1 and w_2 are strings, we say that w_1 comes before w_2 in canonical order if either w_1 is shorter than w_2, or w_1 and w_2 have the same length and w_1 comes before w_2 in lexical order.

14. [5 points] What does it mean to say that machines M_1 and M_2 are *equivalent*?

    Answer: It means that if M_1 and M_2 are given the same input, they will give the same output.

15. Give definitions: [10 points each]

    (a) Give a definition of the language class $\mathcal{NP}$-TIME.

    A language L is in the class $\mathcal{NP}$-TIME if there is some NTM (non-deterministic Turing machine) M and some integer k such that for any string $w \in L$, there is a computation of M with input w which halts in at most $|w|^k$ steps, and for any string $w \notin L$, there is no halting computation of M with input w.

    (b) Give the definition of a *polynomial time reduction* of a language L_1 to another language L_2.

    Answer: Let Σ_i be the alphabet of L_i, for each i. A polynomial time reduction of L_1 to L_2 is a function $R : \Sigma_1^* \to \Sigma_2^2$ such that there is a constant k and machine M where, for any $w \in \Sigma_1^*$

    i. M computes $R(w)$ in at most $|w|^k$ steps, and
    ii. $R(w) \in L_2$ if and only if $w \in L_1$

    (c) Give a definition of $\mathcal{NP}$*-complete language.*

    Answer: We say that a language L_1 is $\mathcal{NP}$-complete if:

i. L_1 is in the class $\mathcal{NP}$-TIME, and

ii. For any language L_2 in the class $\mathcal{NP}$-TIME, there is a polynomial time reduction of L_1 to L_2.

(d) Give a definition of a *decidable language*.

Answer: We say that a language L over an alphabet Σ is *decidable* if there is a machine M which has two possible outputs, 0 and 1, such that a computation of M with any input string w over Σ halts, and the output is 1 if $w \in L$,and is 0 if $w \notin L$.

16. [15 points] We say a binary string w over is *balanced* if w has the same number of 1's as 0's. Let L be the set of balanced binary strings. Give a context-free grammar for L.

Simple Answer: $S \to 0S1S \mid 1S0S \mid \varepsilon$. This grammar is ambiguous.

However, there is an unambiguous grammar. Recall the unambiguous grammar for the Dyck language: $S \to aSbS \mid \varepsilon$. Think of 0 and 1 as representing left and right parentheses, respecitively. Then $A \to 0A1A \mid \varepsilon$ is an unambiguous grammar for the Dyck language, where A is the start symbol, and $B \to 1B0B \mid \varepsilon$ is an unambiguous grammar for the inverse Dyck language, where B is the start symbol. Now consider w, a non-empty balanced binary string: w either begins with 0 or 1. Without loss of generality, w begins with 0. Find the unique 1 that pairs with that first 0, and let x be the string between those symbols; x will be a member of the Dyck language. We can uniquely write $w = 0x1y$ where x is a member of the Dyck language and $y \in L$. Thinking along those lines, we have an unambiguous grammar for L:

$S \to 0A1S \mid 1B0S \mid \varepsilon$

$A \to 0A1A \mid \varepsilon$

$B \to 1B0B \mid \varepsilon$

17. [10 points] Give a Chomsky Normal Form grammar for the language of all palindromes over the alphabet $\{a, b\}$.

$S \to XA \mid YB \mid AA \mid BB \mid a \mid b \mid \varepsilon$

$X \to AT$

$Y \to BT$

$T \to XA \mid YB \mid AA \mid BB \mid a \mid b$

$A \to a$

$B \to b$

18. [20 points] Construct a minimal DFA equivalent equivalent to the NFA shown in Figure 3.

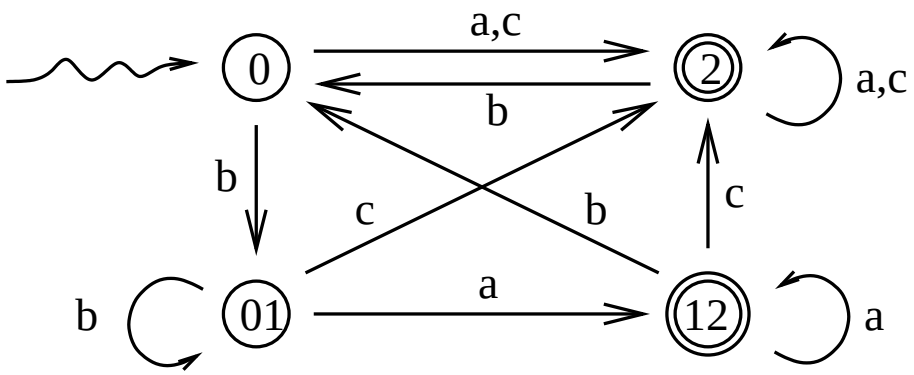

Figure 6: DFA Equivalent to NFA in Figure 3

19. [10 points] Consider the context-free grammar G, with start symbol S and productions as follows:

$S \to s$

$S \to bLn$

$S \to iS$

$S \to iSeS$

$L \to \epsilon$

$L \to LS$

Prove that G is ambiguous by giving two different leftmost derivations for some string.

There are infinitely many correct answers, but the shortest string with two leftmost derivations is $iises$.

$$S \Rightarrow iSeS \Rightarrow iiSeS \Rightarrow iiseS \Rightarrow iises$$

$$S \Rightarrow iS \Rightarrow iiSeS \Rightarrow iiseS \Rightarrow iises$$

22. Every language we have discussed this semester falls into one of these categories.

a. $\mathcal{NC}$.

b. $\mathcal{P}$ but not known to be $\mathcal{NC}$.

c. $\mathcal{NP}$ but not known to be $\mathcal{P}$ and not known to be $\mathcal{NP}$-complete.

d. Co-$\mathcal{NP}$ but not known to be $\mathcal{P}$.

e. $\mathcal{NP}$ and Co-$\mathcal{NP}$ but not known to be $\mathcal{P}$.

f. Known to be $\mathcal{NP}$-complete.

g. Recursive, but not known to be $\mathcal{NP}$.

h. RE (Recursively enumerable), but not recursive.

i. Co-RE, but not recursive.

j. Neither RE nor co-RE.

State which of the above categories each of the languages below falls into. [5 points each]

(i) __f__ Boolean satisfiability.

(ii) __f__ The 0-1 traveling salesman problem.

(iv) __h__ The halting problem.

(v) __i__ The diagonal language.

(vi) __f__ The clique problem.

(vii) __b__ Primality, where the input is written in binary.

(viii) __b__ The language generated by a given context-free grammar.

(ix) __a__ The language of all monotone increasing sequences of arabic numerals for positive integers. (For example, "1,5,23,41,200,201" is a member of that language.)

(x) __a__ The language accepted by a given DFA.

(xi) __e__ The 0/1 factoring problem, *i.e.* the set of all pairs of integers (n, m) such that n has a proper divisor which is at least m. (The input for an instance of this problem is the string consisting of the binary numeral for n, followed by a comma, followed by the binary numeral for m.)

(xiii) __g__ The set of all positions from which black can force a win in a game of generalized checkers.

(xiv) __g__ The set of all configurations of the children's game "Boxes" from which the first player can force a win. (I used to play that game as a child, and I never did figure out an optimal strategy. I don't feel bad about that anymore, now that I know the complexity class of that problem.)

(xv) __a__ The set of all configurations of the game "Nim" from which the first player can force a win.

(xvi) __j__ The set of all ordered pairs of positive numerals $(\langle n\rangle, \langle m\rangle) : m = \beta(n)$, where β is the busy beaver function.

(xix) __h__ The halting problem.

(xx) __b__ Primality.

(xxi) __i__ The context-free grammar equivalence problem.

(xxii) __f__ The independent set problem.

23. [30 points] The grammar below is an alternative unambiguous CF grammar for the Dyck language. Design an LALR parser for that grammar.

For your convenience, stack states are given on the right hand side of the production.

1 $S \to S_{1,3}\, a_2 S_3 b_4$

2 $S \to \epsilon$

	a	b	\$	S
0	$r2$		$r2$	1
1	$s2$		halt	
2	$r2$	$r2$		3
3	s2	$s4$		
4	$r1$	$r1$	$r1$	

24. For each of the following languages, state whether the language is regular, context-free but not regular, context-sensitive but not context-free, or not context-sensitive.

[5 points each]

25. **Reg** The set of all strings over the alphabet $\{a, b\}$ of the form $a^n b^m$.

26. **CF not Reg** The set of all strings over the alphabet $\{a, b\}$ of the form $a^n b^n$.

27. **CS not CF** The set of all strings over the alphabet $\{a, b, c\}$ of the form $a^n b^n c^n$.

28. **CF not Reg** The set of all strings over the alphabet $\{a, b, c\}$ which are **not** of the form $a^n b^n c^n$.

29. **not CS** The set of all strings over the alphabet $\{a\}$ of the form a^{n^2}.

30. [15 points] Draw a minimal DFA which accepts the language L over the binary alphabet $\Sigma = \{a, b, c\}$ consisting of all strings which contain either **a**ba or **c**aa as a substring.

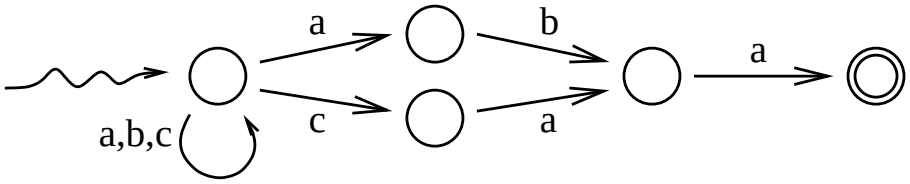

Figure 7: NFA for all strings containing either **aba** or **caa**

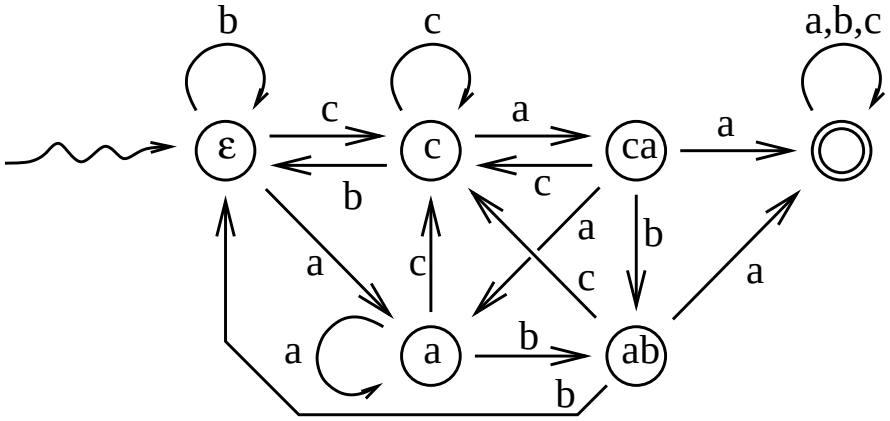

Figure 8: Minimal DFA for all strings containing either **aba** or **caa**

31. [10 points] State the pumping lemma for regular languages accurately. If you have all the right words but in the wrong order, that means you truly do not understand the lemma, and you might get no partial credit at all.

    For any regular language L, there is an integer $k \geq 1$ such that for any $w \in L$ such that $|w| \geq k$, there exist string x, y, and z such that: $w = xyz$, $|xy| \leq k$, $|y| \geq 1$, and for any integer $i \geq 0$, $xy^i z \in L$.

32. [20 points] In class, we demonstrated that a language is in the class $\mathcal{NP}$ if and only if it has a polynomial time *verifier*.

    What is a polynomial time verifier of a language? Your explanation should include the word "certificate," or as it is sometimes known, "witness."

A polynomial time verify for a language L is a machine M such that there is some k such that if M is input two strings, w and c, M will output $M(w,c)$, which is either 0 or 1, within $|w|^k$ steps, and such that if $w \in L$ there is some c, called a *witness* for w, such that $M(w,c) = 1$, and such that $M(w,c) = 0$ for all c if $w \notin L$.

33. These are reduction problems. I could give one of them on the test. The proof should be very informal.

(a) Find a $\mathcal{P}$-time reduction of 3-CNF-SAT to the independent set problem.

Consider a Boolean expression E in 3-CNF form. That is, Exp is the conjunction of k clauses: $Exp = C_1 * C_2 * \cdots C_k$, where each clause is the disjunction of three terms: $C_i = (t_{i,1} + t_{i,2} + t_{i,3})$ for each i, where each term is a Boolean variable or the negation of a Boolean variable Let $G = (V, E)$ be the graph where V is the set of vertices $v_{i,\ell}$ for $1 \leq i \leq k$ and $1 \leq \ell \leq 3$. There is thus a 1-1 correspondence between terms of Exp and vertices of G. We let $E = \{v_{i,\ell}, v_{j,m} : i = j$ or $t_{i,\ell}$ contradicts $t_{j,m}\}$. Let $Q_i = \{v_{i,1}, v_{i,2}, v_{i,3}\}$, the 3-clique corresponding to the clause C_i. Then (G, k) is an instance of the independent set problem. If G has an independent set I of size k, I must contain exactly one vertex in each Q_i. The terms corresponding to the members of I cannot contradict each other. Thus, we can assign a Boolean value to each of those terms. Any variables which are not used for any of the selected terms are assigned true, and thus Exp is satisfied by that assignment. Conversely, any satisfying assignment for Exp causes one term of each C_i to be true. The corresponding set of vertices of G must be independent, since those terms cannot contradict each other, and there are k of them since there is one in each Q_i.

(b) Find a $\mathcal{P}$-time reduction of the independent set problem to the subset sum problem.

Let (G, k) be an instance of the independent set problem, where $G = (V, E)$. Write $V = \{v_0, \ldots v_{n-1}\}$ and $E = \{e_0, \ldots e_{m-1}\}$. We can assume $k \geq 2$, since otherwise the independent set problem is trivial. We say that v_i *meets* e_j if v_i is one of the endpoints of e_j. Let $E(i) = \{j : v_i$ meets $e_j\}$, the set of indices of edges which contain v_i. Let $J = \{j : e_j$ does not meet any member of $I\}$

We construct an instance (W, S) of the subset sum problem as follows. Pick an integer $b > k$. We could pick $b = k + 1$, for instance. For each $0 \leq j < m$, let $y_j = b^j$. For each $0 \leq i < n$, let $x_i = \sum_{j \in E(i)} y_j + b^n$. Let $S = \sum_{j=0}^{m} y_j + kb^n$. Let W be a set of $n + m$ items of weights $x_0, \ldots x_{n-1}, y_0, \ldots y_{m-1}$.

The base b numeral for each x_i and each y_i consists of only zeros and ones, and the base b numeral for S is k followed by m ones.

Claim I: If there is an independent set of G of size k, there is a subset of W whose total weight is S. *Proof:* Let $I \subseteq V$ be an independent set of size k. Let W consist of the items of weight x_i for all $v_i \in I$, together with the items of weight y_j for all e_j which do not meet any member of I. Let w be the total weight of the items in W. let $\langle w \rangle$ be the base b numeral for w.

We now prove that, for any $0 \leq j < m$, the digit in the b^j place of $\langle w \rangle$ is 1. If e_j meets v_i, then b^j will be a term in x_i and Since I is independent, e_j does not meet any other vertex of I, and that digit is 1. On the other hand, if e_j does not meet any member of I, y_i is one of the weights of W, and thus the b^j place of $\langle w \rangle$ is 1.

The b^m place of $\langle w \rangle$ is k, since each member of I contributes a 1 on that place. Thus, $w = S$. ∎

Claim II: If there is a subset of W whose total weight is S, G has an independent set of size k.

Proof: Since the b^m place of $\langle W \rangle$ is k, W must contain there must be exactly k choices of i such that there is a 1 in the b^n place of x_i. Let I consist of the k corresponding vertices v_i of G. HERE HERE

If v_{i_1} and v_{i_2} are neighboring vertices of G, there is an edge, say e_j, between them and b^j is a term of both x_{i_1} and x_{i_2}. Thus, W cannot contain items corresponding to both those vertices, hence I is independent. ∎

By Claim I and Claim II, the reduction is correct.

(c) Find a $\mathcal{P}$-time reduction of the subset sum problem to the partition problem.

Consider an instance of the subset sum problem, which consists of a sequence $x_1, \ldots x_m$ and a number S. That instance has a solution if and only if there is some subsequence of $\{x_i\}_{i=1}^m$ whose sum is S. Let $X = \sum_{i=1}^m x_i$. If $S = \frac{1}{2}X$ we are done, since the terms of the sequence can be partitioned into those in the subsequence and those not; each set has total weight S. If $S > \frac{1}{2}X$, let $x_{m+1} = 2S - X$. Then $2S$ is the sum of the sequence $\{x_i\}_{i=1}^{m+1} x_i$. That sequence can be partitioned into two subsequences with equal totals if and only if it has a subsequence whose total is S. The complement of such a subsequence also has total S. One of those subsequences does ot contain x_{m+1}, and that one is a solution to the instance of the subset sum problem. If $S < \frac{1}{2}X$, let $x_{m+1} = X - 2S$. Then $2(X - S)$ is the sum of the sequence $\{x_i\}_{i=1}^{m+1} x_i$. That sequence can be partitioned into two subsequences with equal totals if and only if it has a subsequence whose total is $X - S$. The complement of such a subsequence also has total $X - S$. One of those subsequences, say Y, contains x_{m+1}. Delete x_{m+1} from Y to obtain a subsequence of the original sequence of weight $X - S - (X - 2S) = S$.

34. [20 points] Prove that every recursively enumerable language is accepted by some Turing machine.

Proof: Let M be a machine which enumerates L, and let w_i be the i^{th} string written by M. The following program accepts L:

read w;

for($i = 1, \mathbf{true}, i++$)

    if ($w = w_i$)

        halt;

By the Church-Turing thesis, there is a Turing Machine which emulates the program and hence accepts L. ∎

35. [20 points] Prove that every language accepted by a Turing machine is recursively enumerable.

Proof: Let Σ be the alphabet of L. Let w_i be the i^{th} string of Σ^* in the canonical order.

The following program enumerates L:

for($t = 1, \mathbf{true}, t++$)

for($i = 1, i \leq t, i++$)

        if (M accepts w_i withn t steps)

            write w_i;

If $w_i \notin L$, it is not accepted by M and hence will never be written. If $w_i \in L$, then w_i will be accepted by M in some finite number of steps, say T steps. Let $t = \max i, T$. Then w_i will be written during the t^{th} iteration of the outer loop. ∎

36. [20 points] Give a general grammar for the language $\left\{a^{2^n}\right\}$

$S \to LaR$

$L \to LD$

$Da \to aaD$

$DR \to R$

$L \to \varepsilon$

$R \to \varepsilon$

37. [20 points] Prove that the halting problem is undecidable.

We say that a Turing machine accepts its input if it halts with that input.

Let HALT $= \{\langle M \rangle w : M \text{ accepts } w\}$

Let $L_d = \{\langle M \rangle : \langle M \rangle \langle M \rangle \notin \text{ HALT}\}$, the diagonal language.

Lemma 1 *The diagonal language is not recursively enumerable.*

Proof: By contradiction. Assume that L_d is R.E.

L_d is accepted by some Turing machine, M_d.

Claim I: For any Turing machine M, $\langle M \rangle \langle M \rangle \in$ HALT if and only if $\langle M \rangle \notin L_d$.

Claim I holds by the definition of L_d.

Claim II: For any Turing machine M, $\langle M_d \rangle \langle M \rangle \in$ HALT if and only if $\langle M \rangle \in L_d$.

Claim II holds by the definition of M_d.

Instantiating both claims, we have the following two statements:

S1. $\langle M_d \rangle \langle M_d \rangle \in$ HALT if and only if $\langle M_d \rangle \in L_d$.

S2. $\langle M_d \rangle \langle M_d \rangle \in$ HALT if and only if $\langle M_d \rangle \notin L_d$.

Combining S1 and S2, we have:

$\langle M_d \rangle \in L_d$ if and only if $\langle M_d \rangle \notin L_d$. Contradiction. Thus, L_d is not recursively enumerable. ∎

Theorem 1 *HALT is undecidable.*

Proof: By contradiction. Assume HALT is decidable. There is a program that decides, for any Turing machine M, whether M accepts $\langle M \rangle$. Therefore the diagonal language is decidable, hence recursively enumerable, which contradicts Lemma 1. ∎