Practice Final Exam #1 Question

We can represent questions about context-free languages and regular languages by choosing a standard encoding for context-free grammars (CFG's) and another for regular expressions (RE's), and phrasing the question as recognition of the codes for grammars and/or regular expressions such that their languages have certain properties. Some sets of codes are decidable, while others are not.

In what follows, you may assume that G and H are context-free grammars with terminal alphabet {0,1}, and R is a regular expression using symbols 0 and 1 only. You may assume that the problem "Is L(G) = (0+1)*?", that is, the problem of recognizing all and only the codes for CFG's G whose language is all strings of 0's and 1's, is undecidable.

There are certain other problems about CFG's and RE's that are decidable, using well-known algorithms. For example, we can test if L(G) is empty by finding the pumping-lemma constant n for G, and checking whether or not there is a string of length n or less in L(G). It is not possible that the shortest string in L(G) is longer than n, because the pumping lemma lets us remove at least one symbol from a string that long and find a shorter string in L(G).

You should try to determine which of the following problems are decidable, and which are undecidable:

- Is Comp(L(G)) equal to (0+1)*? [Comp(L) is the complement of language L with respect to the alphabet {0,1}.]
- Is Comp(L(G)) empty?
- Is L(G) intersect L(H) equal to (0+1)*?
- Is L(G) union L(H) equal to (0+1)*?
- Is L(G) finite?
- Is L(G) contained in L(H)?
- Is L(G) = L(H)?
- Is L(G) = L(R)?
- Is L(G) contained in L(R)?
- Is L(R) contained in L(G)?

Then, identify the true statement from the list below:

a) "Is L(G) contained in L(H)?" is decidable.

b) "Is L(G) intersect L(H) equal to (0+1)*?" is undecidable.

c) "Is L(G) = L(H)?" is decidable.

d) "Is L(G) infinite?" is undecidable

Then, identify the true statement from the list below:

a) "Is L(G) contained in L(R)?" is decidable.

b) "Is L(R) contained in L(G)?" is decidable.

c) "Is L(G) infinite?" is undecidable.

d) "Is L(H) contained in L(G)?" is decidable.

Then, identify the true statement from the list below:

a) "Is L(G) finite?" is decidable.

b) "Is Comp(L(G)) empty?" is decidable.

c) "Is L(G) contained in L(H)?" is decidable.

d) "Is L(G) = L(R)?" is decidable

Then, identify the true statement from the list below:

a) "Is Comp(L(G)) equal to (0+1)*?" is undecidable.

b) "Is L(G) union L(H) equal to (0+1)*" is undecidable.

c) "Is Comp(L(G)) empty?" is decidable.

d) "Is L(R) contained in L(G)?" is decidable.

Then, identify the true statement from the list below:

a) "Is L(G) contained in L(H)?" is decidable.

b) "Is L(G) union L(H) equal to (0+1)*" is undecidable.

c) "Is L(H) contained in L(G)?" is decidable.

d) "Is L(R) contained in L(G)?" is decidable.

Then, identify the true statement from the list below:

a) "Is L(G) finite?" is decidable.

b) "Is L(G) infinite?" is undecidable.

c) "Is L(R) contained in L(G)?" is decidable.

d) "Is L(G) = L(H)?" is decidable.

Then, identify the true statement from the list below:

a) "Is L(G) contained in L(H)?" is decidable.

b) "Is L(G) union L(H) equal to (0+1)*" is undecidable.

c) "Is L(R) contained in L(G)?" is decidable.

d) "Is L(H) contained in L(G)?" is decidable.

Then, identify the true statement from the list below:

a) "Is L(G) = L(R)?" is decidable.

b) "Is L(G) intersect L(H) equal to (0+1)*?" is undecidable.

c) "Is L(H) contained in L(G)?" is decidable.

d) "Is L(G) infinite?" is undecidable.

Then, identify the true statement from the list below:

a) "Is L(H) contained in L(G)?" is decidable.

b) "Is L(G) finite?" is decidable.

c) "Is L(G) contained in L(H)?" is decidable.

d) "Is L(G) = L(H)?" is decidable.

Then, identify the true statement from the list below:

a) "Is L(G) finite?" is decidable.

b) "Is Comp(L(G)) equal to (0+1)*?" is undecidable. **Trick Here**

c) "Is L(G) = L(R)?" is decidable.

d) "Is L(H) contained in L(G)?" is decidable.

Correct Answers:

"Is L(G) finite?" is decidable.
"Is L(G) union L(H) equal to (0+1)*" is undecidable.
"Is L(G) contained in L(R)?" is decidable.

Whether a context free or regular grammar is infinite is decidable. This is from the pumping lemma.

Practice Final Exam #1 Question

The proof that the Independent-Set problem is NP-complete depends on a construction given in Theorem 10.18 (p. 460), which reduces 3SAT to Independent Sets. Apply this construction to the 3SAT instance: (u+v+w)(-v+-w+x)(-u+-x+y)(x+-y+z)(u+-w+-z)
Note that - denotes negation, e.g., -v stands for the literal NOT v. Also, remember that the construction involves the creation of nodes denoted [*i*,*j*]. The node [*i*,*j*] corresponds to the *j*th literal of the *i*th clause. For example, [1,2] corresponds to the occurrence of *v*.
After performing the construction, identify from the list below the one pair of nodes that does NOT have an edge between them.

a) [1,3] and [2,2]

b) [3,1] and [5,1]

c) [2,3] and [3,2]

d) [2,1] and [3,3]

After performing the construction, identify from the list below the one pair of nodes that does NOT have an edge between them.

a) [1,1] and [3,1]

b) [5,1] and [5,2]

c) [2,2] and [5,2]

d) [1,1] and [1,3]

After performing the construction, identify from the list below the one pair of nodes that does NOT have an edge between them.

a) [2,2] and [5,2]

b) [5,1] and [5,2]

c) [2,3] and [3,2]

d) [1,3] and [5,2]

Two keys to this problem:

1. **If two literals are in the same clause (e.g. "[1,1] and [1,3]" or "[5,1] and [5,2]"), then they have an edge between them.**
2. **Otherwise, two literals must be complements/negations of each other.**

Practice Final Exam #1 Question

A *k-clique* in a graph G is a set of *k* nodes of G such that there is an edge between every pair of nodes in the set. The problem *k*-CLIQUE is: Given a graph G and a positive integer *k*, does G have a *k*-clique? We can prove *k*-CLIQUE to be NP-complete by reducing the 3SAT problem to it. Consider the 3-CNF expression:

$E = (x_1' + x_2 + x_3)(x_1' + x_2' + x_3')(x_3 + x_4 + x_5')(x_1 + x_2 + x_4)$

[Note: *a*' denotes the negation NOT(*a*) of variable *a*.] Let G be the graph constructed from this expression as in Theorem 10.18 (p. 460). Then, let H be the *complement* of G, that is, the graph with the same nodes as G and an edge between two nodes if and only if G DOES NOT have an edge between those two nodes. Let us denote the vertices of H by using the corresponding (clause, literal) pair. The node (i,j) corresponds to the j^{th} literal of the i^{th} clause. Which of the following nodes form a maximum-sized clique in H?

a) {(1,1),(2,2),(3,3),(4,2)}

b) {(1,2), (2,1), (3,2), (4,3)}

c) {(2,1),(3,2),(4,3)}

d) {(1,2),(2,2),(3,1),(4,3)}

Which of the following nodes form a maximum-sized clique in H?

a) {(1,1),(2,1),(2,3),(3,2),(4,3)}

b) {(1,1),(1,3),(2,1),(4,3)}

c) {(1,2),(2,1),(3,1),(3,2),(4,3)}

d) {(1,1), (2,1), (3,1), (4,3) }

Which of the following nodes form a maximum-sized clique in H?

a) (1,2),(2,1),(4,3),(3,2),(4,2)}

b) {(1,1), (2,1), (3,1), (4,3) }

c) {(1,1),(2,2),(3,3),(4,2)}

d) {(1,2),(2,2),(3,1),(4,3)}

After performing the construction, identify from the list below the one pair of nodes that does NOT have an edge between them.

a) [3,2] and [4,1]

b) [3,1] and [5,1]

c) [2,3] and [4,1]

d) [5,1] and [5,2]

Key to this problem – Maximum size is four. Can't be five then at least two are in the same triangle so when the graph is inverted, they can't be in the same clique. From there, look for two vertices with no edges.

Practice Final Exam #1 Question

Suppose there are three languages (i.e., problems), of which we know the following:

1. L1 is in P.
2. L2 is NP-complete.
3. L3 is not in NP.

Suppose also that we do not know anything about the resolution of the "P vs. NP" question; for example, we do not know definitely whether P=NP. Classify each of the following languages as (a) Definitely in P, (b) Definitely in NP (but perhaps not in P and perhaps not NP-complete) (c) Definitely NP-complete (d) Definitely not in NP:

- L1 [union] L2.
- L1 ∩ L2.
- L2cL3, where c is a symbol not in the alphabet of L2 or L3 (i.e., the*marked concatenation* of L2 and L3, where there is a unique marker symbol between the strings from L2 and L3).
- The complement of L3.

Based on your analysis, pick the correct, definitely true statement from the list below.

a) L1 [union] L2 is definitely in P.

b) L1 [union] L2 is definitely not in NP.

c) L1 [union] L2 is definitely in NP.

d) The complement of L3 is definitely not in NP.

Based on your analysis, pick the correct, definitely true statement from the list below.

a) The complement of L3 is definitely not NP-complete. (Wrong)

b) The complement of L3 is definitely not in P.

c) L1 [union] L2 is definitely NP-complete.

d) L1 ∩ L2 is definitely in P.

Based on your analysis, pick the correct, definitely true statement from the list below.

a) L1 ∩ L2 is definitely in P.

b) L2cL3 is definitely not in NP.

c) L2cL3 is definitely NP-complete.

d) L1 [union] L2 is definitely not in NP.

Based on your analysis, pick the correct, definitely true statement from the list below.

a) The complement of L3 is definitely in NP.

b) L1 [union] L2 is definitely not in NP.

c) The complement of L3 is definitely not NP-complete.

d) L1 [union] L2 is definitely in NP.

Based on your analysis, pick the correct, definitely true statement from the list below.

a) The complement of L3 is definitely not in NP.

b) L1 ∩ L2 is definitely in NP.

c) L2cL3 is definitely in NP.
d) L1 [union] L2 is definitely not in NP.

a) L2cL3 is definitely not in NP.
b) L1 [union] L2 is definitely not in NP.
c) The complement of L3 is definitely in NP.
d) L2cL3 is definitely in NP.

Based on your analysis, pick the correct, definitely true statement from the list below.

a) The complement of L3 is definitely not NP-complete.
b) L1 ∩ L2 is definitely NP-complete.
c) L1 [union] L2 is definitely in NP.
d) L1 [union] L2 is definitely NP-complete.

Summary right answers:

L2cL3 is definitely not in NP.
The complement of L3 is definitely not in P.
L1 [union] L2 is definitely in NP.
L1 ∩ L2 is definitely in NP.

Practice Final Exam #2 Question

The classes of languages P and NP are closed under certain operations, and not closed under others, just like classes such as the regular languages or context-free languages have closure properties. Decide whether P and NP are closed under each of the following operations.

1. Union.
2. Intersection.
3. Intersection with a regular language.
4. Concatenation.
5. Kleene closure (star).
6. Homomorphism.
7. Inverse homomorphism.

Then, select from the list below the true statement.

a) NP is not closed under intersection with a regular language.

b) NP is not closed under homomorphism.

c) NP is not closed under intersection.

d) P is not closed under intersection.

Then, select from the list below the true statement.

a) NP is not closed under union.

b) NP is not closed under concatenation.

c) NP is not closed under homomorphism.

d) NP is closed under homomorphism.

Then, select from the list below the true statement.

a) NP is closed under inverse homomorphism.

b) NP is not closed under Kleene closure.

c) NP is closed under homomorphism.

d) NP is not closed under union.

Then, select from the list below the true statement.

a) P is not closed under Kleene closure.

b) P is closed under homomorphism.

c) P is closed under Kleene closure.

d) NP is not closed under Kleene closure.

Need to figure this one out later.

Practice Final Exam #2 Question

Consider the descriptions of the following problems:

1. Problem P_1: Given a set S of positive integers $\{s_1, s_2, ..., s_n\}$ and an integer C as input, is there a subset T of S such that the integers in T add up to C?
2. Problem P_2: Given a set S of positive integers $\{s_1, s_2, ..., s_n\}$ as input, is there a subset T of S such that the integers in T add up to c_0?. Here c_0 is a positive integer constant.
3. Problem P_3: Given an undirected graph G and an integer K as input, is there a clique in G of size K?
4. Problem P_4: Given an undirected graph G as input does G contain a clique of size m? Here m is a positive integer constant.

Now consider some additional propositions about the above problems (These may be TRUE or FALSE):

1. Proposition F_1: There is an algorithm A_1 that solves problem P_1 in O(nC) time.
2. Proposition F_2: There is an algorithm A_2 that solves problem P_2 in O(n) time.
3. Proposition F_3: P_3 is NP-complete.
4. Proposition F_4: There is an algorithm A_3 that solves problem P_4 in $O(m^2n^m)$ time.

Choose a correct statement from the choices below:

a) P_2 is reducible to P_1.

b) If F_1 and F_2 are both TRUE, then P_1 is in P but we cannot conclude so about P_2.

c) If F_3 and F_4 are both TRUE, then P is not equal to NP.

d) If F_1 and F_2 are both TRUE, then P_1 is not in P and P_2 is not in P.

Choose a correct statement from the choices below:

a) If F_1 and F_2 are both TRUE, then P_1 is in P but we cannot conclude so about P_2.

b) If F_3 and F_4 are both TRUE, then P=NP.

c) P_2 is reducible to P_1.

d) If F_1 and F_2 are both TRUE, P_1 is in P and P_2 is in P.

Choose a correct statement from the choices below:

a) If F_3 and F_4 are both TRUE, then P=NP.

b) F_4 must be FALSE. Such an algorithm A_3 cannot exist.

c) F_2 must be FALSE. Such an algorithm A_2 cannot exist.

d) P_2 is reducible to P_1.

Practice Final Exam #2 Question

For the purpose of this question, we assume that all languages are over input alphabet {0,1}. Also, we assume that a Turing machine can have any fixed number of tapes.

Sometimes restricting what a Turing machine can do does not affect the class of languages that can be recognized --- the restricted Turing machines can still be designed to accept any recursively enumerable language. Other restrictions limit what languages the Turing machine can accept. For example, it might limit the languages to some subset of the recursive languages, which we know is smaller than the recursively enumerable languages. Here are some of the possible restrictions:

1. Limit the number of states the TM may have.
2. Limit the number of tape symbols the TM may have.
3. Limit the number of times any tape cell may change.
4. Limit the amount of tape the TM may use.
5. Limit the number of moves the TM may make.
6. Limit the way the tape heads may move.

Consider the effect of limitations of these types, perhaps in pairs. Then, from the list below, identify the combination of restrictions that allows the restricted form of Turing machine to accept all recursively enumerable languages.

a) Allow the TM to use only 2^n tape cells when the input is of length n.

b) Allow the TM to use only n^2 tape cells when the input is of length n.

c) Allow a tape cell to change its symbol only once.

d) Allow the TM to use only n^3 tape cells when the input is of length n.

Consider the effect of limitations of these types, perhaps in pairs. Then, from the list below, identify the combination of restrictions that allows the restricted form of Turing machine to accept all recursively enumerable languages.

a) Allow the TM to run for only n^{10} moves when the input is of length n.

b) Allow only two input symbols, 0 and 1, and one other tape symbol, B.

c) Allow the TM to use only n^3 tape cells when the input is of length n.

d) Allow tape heads to move only right or remain stationary on their tape.

Practice Final Exam #3 Question

There is a Turing transducer *T* that transforms problem P1 into probem P2. *T* has one read-only input tape, on which an input of length *n* is placed. *T* has a read-write scratch tape on which it uses O(S(n)) cells. *T* has a write-only output tape, with a head that moves only right, on which it writes an output of length O(U(n)). With input of length *n*, *T* runs for O(T(n)) time before halting. You may assume that each of the upper bounds on space and time used are as tight as possible.

A given combination of S(n), U(n), and T(n) may:

1. Imply that *T* is a polynomial-time reduction of P1 to P2.
2. Imply that *T* is NOT a polynomial-time reduction of P1 to P2.
3. Be impossible; i.e., there is no Turing machine that has that combination of tight bounds on the space used, output size, and running time.

What are all the constraints on S(n), U(n), and T(n) if *T* is a polynomial-time reducer? What are the constraints on feasibility, even if the reduction is not polynomial-time? After working out these constraints, identify the true statement from the list below.

a) $S(n) = \log_2 n$; $U(n) = n^2$; $T(n) = n^4$ is a polynomial-time reduction

b) $S(n) = n^2$; $U(n) = n^2$; $T(n) = n!$ is not physically possible.

c) $S(n) = n$; $U(n) = n^2$; $T(n) = n \log_2 n$ is a polynomial-time reduction

d) $S(n) = n^3$; $U(n) = n$; $T(n) = (1.01)^n$ is possible, but not a polynomial-time reduction.

Practice Final Exam #3 Question

Reducing MPCP to PCP

	List A	List B
1	01	010
2	11	110
3	0	01

Solution

	List A	List A
0	*0*1*	*0*1*0
1	0*1*	*0*1*0
2	1*1*	*1*1*0
3	0*	*0*1
4	$	*$

Tricks:

For everything but case 0, you place a trailing "*" after each character in list A and a preceding "*" in list B. For the case 0, you place a preceding star on the first character in list A and do the other two strings the same way. There is also a special case of "$" for list A and "*$" for list B.

	List A	List B
i	w_i	x_i
1	1	111
2	10111	10
3	10	0

Figure 9.12: An instance of PCP

Example 9.16: Suppose Fig. 9.12 is an MPCP instance. Then the instance of PCP constructed by the above steps is shown in Fig. 9.14. □

	List C	List D
i	y_i	z_i
0	$*1*$	$*1*1*1$
1	$1*$	$*1*1*1$
2	$1*0*1*1*1*$	$*1*0$
3	$1*0*$	$*0$
4	\$	$*$

Figure 9.14: Constructing an instance of PCP from an MPCP instance

Note this figure is wrong in the international edition. This figure is from the online version and is correct.

If we apply the reduction of MPCP to PCP described in Section 9.4.2 (p. 404), which of the following would be a pair in the resulting PCP instance.

a) (0*1*, *0*1*0)

b) (0*1*, *0*1*0*)

c) (0*1, *0*1*0)

d) (1*1, *1*1*0)

If we apply the reduction of MPCP to PCP described in Section 9.4.2 (p. 404), which of the following would be a pair in the resulting PCP instance.

a) (*0*1*, *0*1*0)

b) (*0*1, 0*1*0*)

c) (0*1, *0*1*0)

d) (*1*1, 1*1*0*)

If we apply the reduction of MPCP to PCP described in Section 9.4.2 (p. 404), which of the following would be a pair in the resulting PCP instance.

a) (1*1, *1*1*0)

b) (*1*1*, *1*1*0)

c) (*0*1*, *0*1*0)

d) (0*1*, *0*1*0*)

Practice Final Exam #3 Question

Which of the following problems about a Turing Machine *M* does Rice's Theorem imply is undecidable?

a) Is there some input that causes *M* to enter more than 100 states?

b) Is there some input that causes *M* to halt after no more than 500 moves?

c) Is the language of *M* a set of strings?

d) Does the language of *M* contain at least 10 strings?

Which of the following problems about a Turing Machine *M* does Rice's Theorem imply is undecidable?

a) Does *M* ever write the symbol 1 on its tape?

b) Does the language of *M* contain at least 10 strings?

c) Does *M* make more than 1000 moves when started with a blank tape?

d) Is the language of *M* a set of strings?

Which of the following problems about a Turing Machine *M* does Rice's Theorem imply is undecidable?

a) Is the language of *M* a set of strings?

b) Is the language of *M* not equal to itself?

c) Is the language of *M* a regular language?

d) Does *M* ever write the symbol 0 on its tape?

Which of the following problems about a Turing Machine *M* does Rice's Theorem imply is undecidable?

a) Does *M* make more than three moves when started with a blank tape?

b) Does *M* ever write the symbol 1 on its tape?

c) Is the language of *M* context-free?

d) Is the language of *M* recursively enumerable?

Rice's Theorem says any non-trivial property of a language of a Turing Machine is undeciable. Attributes of the Turing machine structure are decidable.

Correct Answers:

Is the language of *M* context-free?
Is the language of *M* a regular language?
Does the language of *M* contain at least 10 strings?

Practice Final Exam #3 Question

Suppose a problem P_1 reduces to a problem P_2. Which of the following statements can we conclude to be TRUE based on the above?

a) If P_1 is non-RE, then it must be that P_2 is non-RE.

b) If P_2 is non-RE, then it must be that P_1 is non-RE.

c) If P_2 is undecidable, then it must be that P_1 is decidable.

d) If P_1 is undecidable, then it must be that P_2 is decidable.

Suppose a problem P_1 reduces to a problem P_2. Which of the following statements can we conclude to be TRUE based on the above?

a) If P_1 is undecidable, then it must be that P_2 is decidable.

b) If P_2 is decidable, then it must be that P_1 is decidable.

c) If P_2 is non-RE, then it must be that P_1 is non-RE.

d) If P_1 is decidable, then it must be that P_2 is decidable.

Suppose a problem P_1 reduces to a problem P_2. Which of the following statements can we conclude to be TRUE based on the above?

a) If P_1 is non-RE, then it must be that P_2 is non-RE.

b) If P_2 is undecidable, then it must be that P_1 is undecidable.

c) If P_1 is non-RE, then it must be that P_2 is RE.

d) If P_1 is RE, then it must be that P_2 is RE.

P2 is at least as hard as P1. This goes for language type (RE/non-RE) and decidability.

Practice Final Exam #3 Question

Reducing SAT to CSAT – The polynomial-time reduction from SAT to CSAT, as described in Section 10.3.3 (p. 452), needs to introduce new variables. The reason is that the obvious manipulation of a boolean expression into an equivalent CNF expression could exponentiate the size of the expression, and therefore could not be polynomial time.
Suppose we apply this construction to the expression

(u+(vw))+x

with the parse implied by the parentheses. Suppose also that when we introduce new variables, we use y_1, y_2,...
After constructing the corresponding CNF expression, identify one of its clauses from the list below. Note: logical OR is represented by +, logical AND by juxtaposition, and logical NOT by -.

Answer:

(u + vw)) + x

(y_1 + u) (y_1' + v)(y_1' + w)) + x

(y_1 + y_2 + u) (y_1' + y_2 + v)(y_1' + y_2 + w))(y_2' + x)

Summary:

1) AND – Create a new variable y1 to both terms and make in CNF.
2) OR – Create a new variable y2 or y3. Add it to the element on the left side of the "OR" and add its negation to the right side of the "OR".

Example 10.14: Let us show how the construction of Theorem 10.13 applies to a simple expression: $E = x\bar{y} + \bar{x}(y + z)$. Figure 10.7 shows the parse of this expression. Attached to each node is the CNF expression constructed for the expression represented by that node.

The leaves correspond to the literals, and for each literal, the CNF expression is one clause consisting of that literal alone. For instance, we see that the leaf labeled $\bar{y}$ has an associated CNF expression $(\bar{y})$. The parentheses are

$$(u + x)(u + \bar{y})(\bar{u} + \bar{x})(\bar{u} + v + y)(\bar{u} + \bar{v} + z)$$

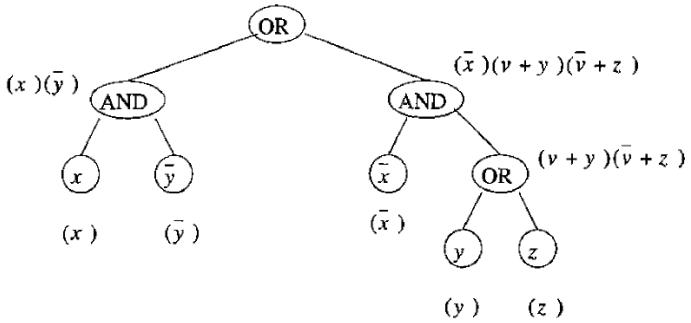


Figure 10.7: Transforming a boolean expression into CNF

unnecessary, but we put them in CNF expressions to help remind you that we are talking about a product of clauses.

For an AND node, the construction of a CNF expression is simply to take the product (AND) of all the clauses for the two subexpressions. Thus, for instance, the node for the subexpression $\bar{x}(y + z)$ has an associated CNF expression that is the product of the one clause for $\bar{x}$, namely $(\bar{x})$, and the two clauses for $y + z$, namely $(v + y)(\bar{v} + z)$.[4]

For an OR node, we must introduce a new variable. We add it to all the clauses for the left operand, and we add its negation to the clauses for the right operand. For instance, consider the root node in Fig. 10.7. It is the OR of expressions $x\bar{y}$ and $\bar{x}(y + z)$, whose CNF expressions have been determined to be $(x)(\bar{y})$ and $(\bar{x})(v + y)(\bar{v} + z)$, respectively. We introduce a new variable u, which is added without negation to the first group of clauses and negated in the second group. The result is

$$F = (u + x)(u + \bar{y})(\bar{u} + \bar{x})(\bar{u} + v + y)(\bar{u} + \bar{v} + z)$$

After constructing the corresponding CNF expression, identify one of its clauses from the list below. Note: logical OR is represented by +, logical AND by juxtaposition, and logical NOT by -.

a) (-y2+x)

b) (y3+y2+u)

c) (-y1+w)

d) (y3+-y2+-y1+w)

Practice Final Exam #4 Question

Use the construction from Theorem 10.15 (p. 457) to convert the following clauses:

1. (a+b)
2. (c+d+e+f)
3. (g+h+i+j+k+l+m)

to products of 3 literals per clause. In each case, the new clauses must be satisfiable if and only if the original clause is satisfiable. For the first clause, introduce variables x_1, x_2,... in that order from the left; for the second introduce y_1, y_2,... in that order from the left, and for the third introduce z_1, z_2,... in that order from the left. Use -w as shorthand for NOT w. Then identify, in the list below, the one clause that would appear among the clauses generated by the construction.

a) $(e+y_1+-y_2)$

b) $(k+z_2+-z_3)$

c) $(j+z_3+-z_4)$

d) $(c+d+-y_1)$

Need to solve this one explicitly.

Answer:

$(a + b + x_1')(a + b + x_1)$
$(c + d + y_1')(e + f + y_1)$

$(g + h + z_1')(i + z_1 + z_2') (j + z_2 + z_3') (k + z_3 + z_4') (l + m + z_4)$

Very Special Note: One of the Gradiance solutions is wrong. It lists $(j + x_2 + z_3')$ as a right answer which it is not.

1. If e_i is a single literal, say (x),[5] Introduce two new variables u and v. Replace (x) by the four clauses $(x+u+v)(x+u+\overline{v})(x+\overline{u}+v)(x+\overline{u}+\overline{v})$. Since u and v appear in all combinations, the only way to satisfy all four clauses is to make x true. Thus, all and only the satisfying assignments for E can be extended to a satisfying assignment for F.

2. Suppose e_i is the sum of two literals, $(x+y)$. Introduce a new variable z, and replace e_i by the product of two clauses $(x+y+z)(x+y+\overline{z})$. As in case 1, the only way to satisfy both clauses is to satisfy $(x+y)$.

3. If e_i is the sum of three literals, it is already in the form required for 3-CNF, so we leave e_i in the expression F being constructed.

4. Suppose $e_i = (x_1+x_2+\cdots+x_m)$ for some $m \geq 4$. Introduce new variables $y_1, y_2, \ldots, y_{m-3}$ and replace e_i by the product of clauses

$$\begin{aligned}&(x_1 + x_2 + y_1)(x_3 + \overline{y_1} + y_2)(x_4 + \overline{y_2} + y_3)\cdots \\ &(x_{m-2} + \overline{y_{m-4}} + y_{m-3})(x_{m-1} + x_m + \overline{y_{m-3}})\end{aligned} \tag{10.2}$$

An assignment T that satisfies E must make at least one literal of e_i true; say it makes x_j true (recall x_j could be a variable or a negated variable).

Practice Final Exam #4 Question

In this question, L1, L2, L3, L4 refer to languages and M, M1, M2 refer to Turing machines. Let
L1 = {(M1,M2) | L(M1) is a subset of L(M2)},
L2 = {M | There exists an input on which TM M halts within 100 steps},
L3 = {M | There exists an input w of size less than 100, such that M accepts w},
L4 = {M | L(M) contains at least 2 strings}.
Decide whether each of L1, L2, L3 and L4 are recursive, RE or neither. Then identify the true statement below.

Decide whether each of L_1, L_2, L_3 and L_4 are recursive, RE or neither. Then identify the true statement below.

a) The complement of L_2 is recursive.

b) L_1 is recursively enumerable.

c) The union of L_2 and L_3 is not recursively enumerable.

d) The intersection of L_2 and L_3 is not recursively enumerable.

Decide whether each of L_1, L_2, L_3 and L_4 are recursive, RE or neither. Then identify the true statement below.

a) The complement of L_3 is recursively enumerable.

b) The union of L_2 and L_3 is not recursively enumerable.

c) The intersection of L_2 and L_3 is not recursively enumerable.

d) The complement of L_2 is recursive.

Decide whether each of L_1, L_2, L_3 and L_4 are recursive, RE or neither. Then identify the true statement below.

a) The complement of L_2 is not recursively enumerable.

b) The intersection of L_2 and L_3 is not recursively enumerable.

c) The complement of L_2 is recursive.

d) L_4 is recursive.

Decide whether each of L_1, L_2, L_3 and L_4 are recursive, RE or neither. Then identify the true statement below.

a) The complement of L_4 is not recursive.

b) L_2 is not recursive.

c) L_4 is recursive.

d) L_3 is recursive.

Decide whether each of L_1, L_2, L_3 and L_4 are recursive, RE or neither. Then identify the true statement below.

a) L_1 is recursively enumerable.

b) The complement of L_4 is not recursive.

c) L_4 is recursive.

d) The complement of L_3 is recursively enumerable.

Decide whether each of L_1, L_2, L_3 and L_4 are recursive, RE or neither. Then identify the true statement below.

a) L_3 is recursively enumerable.

b) The complement of L_2 is not recursively enumerable.

c) L_4 is recursive.

d) L_2 is not recursive.

Decide whether each of L_1, L_2, L_3 and L_4 are recursive, RE or neither. Then identify the true statement below.

a) The complement of L_4 is not recursive.

b) L_1 is recursively enumerable.

c) L_4 is not recursive but cannot be proved so by Rice's Theorem.

d) The union of L_2 and L_3 is not recursively enumerable.

Language #	Language Type
L1	Not Recursively Enumerable
L2	Recursive (Its Complement is Also Recursive)
L3	Recursively Enumerable
L4	Recursively Enumerable

Practice Final Exam #4 Question

We wish to perform the reduction of acceptance by a Turing machine to MPCP, as described in Section 9.4.3 (p. 407). We assume the TM *M* satisfies Theorem 8.12 (p. 346): it never moves left from its initial position and never writes a blank. We know the following:

1. The start state of *M* is *q*.
2. *r* is the accepting state of *M*.
3. The tape symbols of *M* are 0, 1, and B (blank).
4. One of the moves of *M* is $\delta(q,0) = (p,1,L)$.

Which of the following is DEFINITELY NOT one of the pairs in the MPCP instance that we construct for the TM *M* and the input 001?

a) (r1, r)

b) (0, 0)

c) (1, 1)

d) (#, #q001) should be (#, #q001#) since initial setup.

Which of the following is DEFINITELY NOT one of the pairs in the MPCP instance that we construct for the TM *M* and the input 001?

a) (#, #q001#)

b) (1q0, p11)

c) (r1, r)

d) (p##, #) should be r## since this is to represent the final state

Which of the following is DEFINITELY NOT one of the pairs in the MPCP instance that we construct for the TM *M* and the input 001?

a) (#, #q001#)

b) (r1, r)

c) (0r1, r)

d) (q0, 1p)

If we apply the reduction of MPCP to PCP described in Section 9.4.2 (p. 404), which of the following would be a pair in the resulting PCP instance.

a) ($, *$)

b) (*0*, *0*1)

c) ($, $*)

d) (*$, $)

Which of the following is DEFINITELY NOT one of the pairs in the MPCP instance that we construct for the TM*M* and the input 001?

a) (#, #q001#)

b) (q0, 1p)

c) (0r, r)

d) (0q0, p01)

Which of the following is DEFINITELY NOT one of the pairs in the MPCP instance that we construct for the TM *M* and the input 001?

a) (0r1, r)

b) (0q0, p01)

c) (#, #q001#)

d) (q0, 1p)

If we apply the reduction of MPCP to PCP described in Section 9.4.2 (p. 404), which of the following would be a pair in the resulting PCP instance.

a) (0*1*, *0*1*0)

b) (*0*, *0*1)

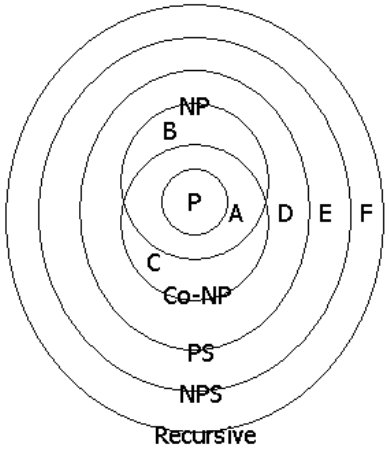
c) (*1*1*, *1*1*0)

d) (*1*1, 1*1*0*)

	List A	List B
0	#	#q001#
1	B	B
2	0	0
3	1	1
4	#	#
5	0q0	p01
6	1q0	p11
7	r	r
8	r1	r
9	r0	r
10	1r1	r
11	0r1	r
12	1r0	r
13	0r0	r
14	1r	r
15	0r	r
16	r##	#

Practice Final Exam #4 Question

In the diagram below we see certain complexity classes (represented as circles or ovals) and certain regions labeled A through F that represent the differences of some of these complexity classes.



The state of our knowledge regarding the existence of problems in the regions A-F is imperfect. In some cases, we know that a region is nonempty, and in other cases we know that it is empty. Moreover, if P=NP, then we would know more about the emptiness or nonemptiness of some of these regions, but still would not know everything.

Decide what we know about the regions A-F currently, and also what we would know if P=NP. Then, identify the true statement from the list below.

Region A	Today	**Not known**
	P = NP	Empty (I think)
Region B	Today	**Not known**
	P = NP	Empty (I think)
Region C	Today	**Not known**
	P = NP	Empty (I think)
Region D	Today	**Not known**
	P = NP	**Not known**
Region E	Today	**Definitely Empty (Savitch's Theorem)**
	P = NP	**Definitely Empty (Savitch's Theorem)**
Region F	Today	**Not Empty**
	P = NP	Not Empty(I think)

Co-NP – Problems whose complement are in NP.
PS – Polynomial Space
NPS – Non-Polynomial Space
Savitch's Theorem – PS = NPS

Decide what we know about the regions A-F currently, and also what we would know if P=NP. Then, identify the true statement from the list below.

a) Region C is definitely empty.

b) If P=NP, it would still not be known whether region B is empty.

c) Region B is definitely empty.

d) Region E is definitely empty.

Decide what we know about the regions A-F currently, and also what we would know if P=NP. Then, identify the true statement from the list below.

a) If P=NP, then region D is definitely not empty.

b) If P=NP, it would still not be known whether region A is empty.

c) Region F is definitely not empty.

d) Region D is definitely not empty.

Decide what we know about the regions A-F currently, and also what we would know if P=NP. Then, identify the true statement from the list below.

a) If P=NP, then region D is definitely not empty.

b) Region C is definitely not empty.

c) Region E is definitely empty.

d) If P=NP, it would still not be known whether region A is empty.

Decide what we know about the regions A-F currently, and also what we would know if P=NP. Then, identify the true statement from the list below.

a) It is not known whether region F is empty.

b) Region F is definitely empty.

c) If P=NP, it would still not be known whether region A is empty.

d) Region F is definitely not empty.

Practice Final Exam #4 Question

Consider the grammar G1:

S → ε | aS | aSbS

Which of the following is correct (for a choice to be correct, all propositions must be correct)?

a) a) G1 generates all and only the strings of a's and b's such that every string has at least as many a's as b's. b) The inductive hypothesis to prove it is: For n < k, it holds: Any word in G1 of length n, is such that all its prefixes contain more a's than b's or as many a's as b's.

b) For any word w with every prefix having at least as many a's as b's, there is a unique b in w such that w can be written as aw'bw" --- hence w can be generated from shorter words using the production S → aSbS.

c) The string aaabbbababaabba is not generated by the grammar.

d) **For any word w with every prefix having at least as many a's as b's, either there is a unique b in w such that w can be written as aw''bw''' with w'' and w''' being such that every prefix has as many a's as b's--hence can be generated from shorter words of the grammar using the rule S → aSbS-- or w can be written as w=aw' with w' having every prefix with as many a's as b's -- hence can be generated by the rule S → aS.**

Which of the following is correct (for a choice to be correct, all propositions must be correct)?

a) The string aaba is not generated by the grammar.

b) a) G1 generates all and only the strings of a's and b's such that every string has at least as many a's as b's. b) The inductive hypothesis to prove it is: For n < k, it holds: Any word in G1 of length n, is such that all its prefixes contain more a's than b's or as many a's as b's.

c) a) G1 generates all and only the strings of a's and b's such that every prefix has at least as many a's as b's. b) The inductive hypothesis to prove it is: For n < k, it holds that: For any word in G1, any prefix of length n, is such that all its prefixes contain at least as many a's as b's.

d) **a) G1 generates all and only the strings of a's and b's such that every prefix has at least as many a's as b's. b) The following inductive hypothesis will prove it: For n less than k, G1 generates all and only the strings of a's and b's of length n such that every prefix has at least as many a's as b's.**

Which of the following is correct (for a choice to be correct, all propositions must be correct)?

a) The string aaabbbababaabba is not generated by the grammar.

b) There are strings that have as many a's as b's and are not generated by the grammar G1.

c) For any word w with every prefix having at least as many a's as b's, there is a unique b in w such that w can be written as aw'bw" --- hence w can be generated from shorter words using the production S → aSbS.

d) a) G1 generates all and only the strings of a's and b's such that every prefix has at least as many a's as b's. b) The inductive hypothesis to prove it is: For n < k, it holds that: For any word in G1, any prefix of length n, is such that all its prefixes contain at least as many a's as b's.

Consider the grammar G1:

$S \rightarrow \varepsilon \mid aS \mid aSbS$

Which of the following is correct (for a choice to be correct, all propositions must be correct)?

a) a) G1 generates all and only the strings of a's and b's such that every string has at least as many a's as b's. b) The inductive hypothesis to prove it is: For n < k, it holds: Any word in G1 of length n, is such that all its prefixes contain more a's than b's or as many a's as b's.

b) a) G1 generates all and only the strings of a's and b's such that every prefix has at least as many a's as b's. b) The following inductive hypothesis will prove it: For n < k, it holds that: Any word in G1 of length n, is such that all its prefixes contain at least as many a's as b's.

c) a) G1 generates all and only the strings of a's and b's such that every prefix has at least as many a's as b's. b) The inductive hypothesis to prove it is: For n < k, it holds that: For any word in G1, any prefix of length n, is such that all its prefixes contain at least as many a's as b's.

d) **a) G1 generates all and only the strings of a's and b's such that every prefix has at least as many a's as b's. b) The following inductive hypothesis will prove it: For n less than k, the following two assertions hold: (i) Any word in G1 of length n, is such that all its prefixes contain at least as many a's as b's. (ii) Any word of length n such that every prefix contains at least as many a's as b's is generated by G1.**

Which of the following is correct (for a choice to be correct, all propositions must be correct)?

a) a) G1 generates all and only the strings of a's and b's such that every string has at least as many a's as b's. b) The inductive hypothesis to prove it is: For n < k, it holds: Any word in G1 of length n, is such that all its prefixes contain more a's than b's or as many a's as b's.

b) a) G1 generates all and only the strings of a's and b's such that every prefix has at least as many a's as b's. b) The inductive hypothesis to prove it is: For n < k, it holds that: For any word in G1, any prefix of length n, is such that all its prefixes contain at least as many a's as b's.

c) The string aaabbbabbaabbaaabb is not generated by the grammar.

d) The string aaabbbababaabba is not generated by the grammar.

Correct Answers:

There are strings that have as many a's as b's and are not generated by the grammar G1.

a) G1 generates all and only the strings of a's and b's such that every prefix has at least as many a's as b's. b) The following inductive hypothesis will prove it: For n less than k, the following two assertions hold: (i) Any word in G1 of length n, is such that all its prefixes contain at least as many a's as b's. (ii) Any word of length n such that every prefix contains at least as many a's as b's is generated by G1.

a) G1 generates all and only the strings of a's and b's such that every prefix has at least as many a's as b's. b) The following inductive hypothesis will prove it: For n less than k, G1 generates all and only the strings of a's and b's of length n such that every prefix has at least as many a's as b's.

For any word w with every prefix having at least as many a's as b's, either there is a unique b in w such that w can be written as aw''bw''' with w'' and w''' being such that every prefix has as many a's as b's-- hence can be generated from shorter words of the grammar using the rule S → aSbS-- or w can be written as w=aw' with w' having every prefix with as many a's as b's -- hence can be generated by the rule S → aS.

The string aaabbbabbaabbaaabb is not generated by the grammar.

Practice Final Exam #5 Question

The NOT-ALL-EQUAL 3SAT problem is defined as follows: Given a 3-CNF formula F, is there a truth assignment for the variables such that each clause has at least one true literal and at least one false literal? The NOT-ALL-EQUAL 3SAT problem is NP-complete.
This question is about trying to reduce the NOT-ALL-EQUAL 3SAT problem to the MAX-CUT problem defined below to show the latter to be NP-complete.
A cut in an undirected graph G=(V,E) is a partitioning of the set of nodes V into two disjoint subsets V1 and V2. The size of a cut is the number of edges $e = (u,v)$ where u is in V1 and v is in V2. The MAX-CUT problem is defined as follows: Given an undirected graph G=(V,E) and a positive integer k, does G have a cut of size k or more?
Given a 3CNF expression E, we create the graph G = (V,E) using the transformation given by Theorem 10.18 in Section 10.4.2 on p. 460 of the text. Then given an assignment A, create a cut C in G by partitioning the set of nodes V as follows: the nodes corresponding to the uncomplemented literals are in set V1 and those corresponding to the complemented variables are in set V2.
For variable a, let a' denote NOT(a). Let
$E = (a + b + c)(a + b' + c)(a' + b' + d)(c' + d' + e)$
be an instance of NOT-ALL-EQUAL 3SAT. Suppose a cut separates the true nodes from false nodes according to some truth assignment applied to E. How many edges between nodes corresponding to the literals in the same clause are cut? How many other edges are cut? Find out how the cut-size can be computed for an arbitrary instance of NOT-All-EQUAL 3SAT. Then for the instance E, determine in which of the cases below, the cut-size C corresponds to the satisfiable assignment given.

a) a = F, b = T, c = F, d = F, e = T, C = 13
b) a = T, b = F, c = T, d = F, e = T, C = 15
c) a = T, b = F, c = F, d = T, e = F, C = 15
d) a = T, b = F, c = F, d = T, e = F, C = 8

a) a = T, b = T, c = F, d = T, e = T, C = 15
b) a = F, b = T, c = F, d = F, e = T, C = 13
c) a = T, b = F, c = F, d = T, e = F, C = 7
d) a = T, b = F, c = T, d = F, e = T, C = 15

How many edges between nodes corresponding to the literals in the same clause are cut? How many other edges are cut? Find out how the cut-size can be computed for an arbitrary instance of NOT-All-EQUAL 3SAT. Then for the instance E, determine in which of the cases below, the cut-size C corresponds to the satisfiable assignment given.

a) a = T, b = T, c = F, d = T, e = T, C = 15
b) a = T, b = T, c = T, d = T, e = T, C = 15
c) a =T, b = T, c = F, d = T, e = T, C = 8
d) a = F, b = T, c = T, d = F, e = T, C = 7

Hints:

Number of cuts always appears to be 15. From there, find the set where for each clause, one literal (or its negation) is false and the other true.

Practice Final Exam #5 Question

Let us denote a problem X as NP-Easy if it is polynomial-time reducible to some problem Y that is in NP. Let us denote as NP-Equivalent, the class of problems that are both NP-Easy and NP-Hard. Let A,B,C,D and E be problems such that A is NP-Hard, B is NP-Complete, C is NP-Equivalent, D is NP-Easy and E is in NP. Which of the following statements is TRUE?

a) If C is in P then so is A.

b) If C is in P then P=NP.

c) If P=NP then A is in P.

d) If E is in P then so is A.

Which of the following statements is TRUE?

a) If B is in P then so is A.

b) If D is in P then so is A.

c) If C is in P then P=NP.

d) If P = NP then B is not in P

Which of the following statements is TRUE?

a) If D is in P then P=NP

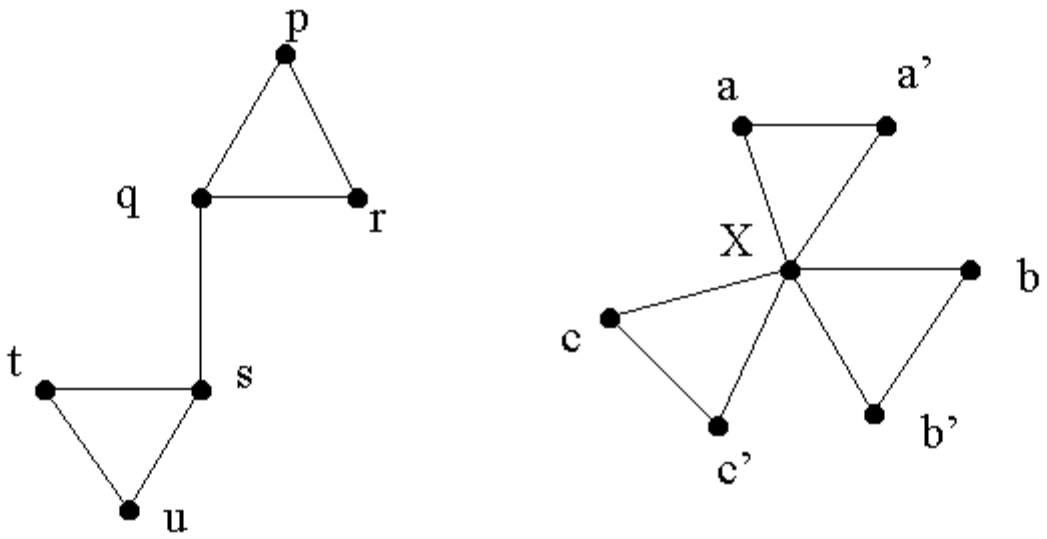
b) If D is in P then so is A.

c) If P = NP then B is not in P

d) If P=NP then D is in P.

Practice Final Exam #5 Question

The graph k-coloring problem is defined as follows: Given a graph G and an integer *k*, is G k-colorable?, i.e. can we assign one of *k* colors to each node of G such that no edge has both of its ends colored by the same color. The graph 3-coloring problem is NP-complete and this fact can be proved by a reduction from 3SAT.



As a part of the reduction proof, we assume that there are three colors GREEN, RED and BLUE. For variable *a*, let *a*' denote NOT(a). We associate with each variable *a*, a "gadget" consisting of 3 nodes labeled *a*, *a*' and X. This gadget is shown at the right in the diagram above. X is common for all variables and is labeled blue. If *a* is TRUE (respectively FALSE) in a particular assignment, node *a* is colored green (respectively red) and node *a*' is colored red (respectively green).

With each clause we associate a copy of the gadget at the left in the diagram above, defined by the graph H = (V,E) where V = {*p*,*q*,*r*,*s*,*t*,*u*} and E = {(*p*,*q*), (*q*,*r*), (*r*,*p*), (*s*,*t*), (*t*,*u*), (*u*,*s*), (*q*,*s*)}. Nodes *t*, *u* and *r* from this copy of the gadget are connected to the nodes constituting the literals for the clause, in left to right order.

Consider a clause in the 3-CNF expression: *a* + *b*' + *c*. We must color the above gadget using the colors red, blue and green in such a way that no two adjacent nodes get the same color. In each choice below, you are given an assignment for the variables *a*, *b*, and *c* and a possible assignment of colors to some nodes in the gadget above. Indicate the choice of colors that is valid for the given truth assignment:

a) **(a=TRUE, b=FALSE, c=TRUE, t=blue, r=red, p=blue) (This is an error but Gradiance marks it right).**

b) (a=FALSE, b=TRUE, c=FALSE, t=green, u=blue, p=green)

c) (a=FALSE, b=TRUE, c=FALSE, t=green, r=green, p=blue**)**

d) (a=FALSE, b=TRUE, c=FALSE, s=blue, q=red, p=green)

Consider a clause in the 3-CNF expression: *a* + *b*' + *c*. We must color the above gadget using the colors red, blue and green in such a way that no two adjacent nodes get the same color. In each choice below, you are given an assignment for the variables *a*, *b*, and *c* and a possible assignment of colors to some nodes in the gadget above. Indicate the choice of colors that is valid for the given truth assignment:

a) (a=FALSE, b=TRUE, c=FALSE, t=green, u=blue, p=green)

b) (a=TRUE, b=FALSE, c=TRUE, t=blue, r=red, p=blue)

c) (a=FALSE, b=TRUE, c=TRUE, t=blue, u=red, p=blue)

d) (a=FALSE, b=TRUE, c=FALSE, t=red, r=green, p=blue)

Consider a clause in the 3-CNF expression: $a + b' + c$. We must color the above gadget using the colors red, blue and green in such a way that no two adjacent nodes get the same color. In each choice below, you are given an assignment for the variables *a*, *b*, and *c* and a possible assignment of colors to some nodes in the gadget above. Indicate the choice of colors that is valid for the given truth assignment:

a) (a=TRUE, b=FALSE, c=TRUE, t=red, u=blue, p=red)

b) (a=FALSE, b=TRUE, c=FALSE, t=red, r=green, p=blue)

c) (a=FALSE, b=TRUE, c=FALSE, s=blue, q=red, p=green)

d) (a=FALSE, b=TRUE, c=FALSE, t=green, u=blue, p=red)

Consider a clause in the 3-CNF expression: $a + b' + c$. We must color the above gadget using the colors red, blue and green in such a way that no two adjacent nodes get the same color. In each choice below, you are given an assignment for the variables *a*, *b*, and *c* and a possible assignment of colors to some nodes in the gadget above. Indicate the choice of colors that is valid for the given truth assignment:

a) (a=TRUE, b=FALSE, c=TRUE, t=red, u=blue, p=red)

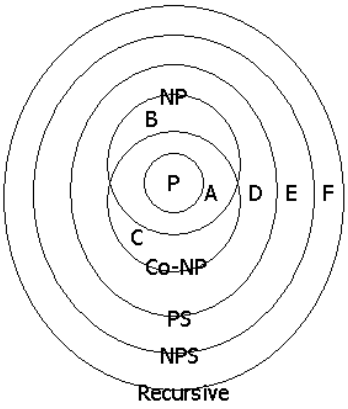
b) (a=TRUE, b=TRUE, c=TRUE, t=red, u=blue, p=red)

c) (a=FALSE, b=TRUE, c=FALSE, t=red, r=green, p=blue)

d) (a=FALSE, b=TRUE, c=FALSE, t=green, u=blue, p=red)

Practice Final Exam #5 Question

Consider the following problems:
SP (Shortest Paths): given a weighted, undirected graph with nonnegative integer edge weights, given two nodes in that graph, and given an integer limit k, determine whether the length of the shortest path between the nodes is k or less.
WHP (Weighted Hamilton Paths): given a weighted, undirected graph with nonnegative integer edge weights, and given an integer limit k, determine whether the length of the shortest Hamilton path in the graph is k or less.
TAUT (Tautologies): given a propositional boolean formula, determine whether it is true for all possible truth assignments to its variables.
QBF (Quantified Boolean Formulas): given a boolean formula with quantifiers for-all and there-exists, such that there are no free variables, determine whether the formula is true.
In the diagram below are seven regions, P and A through F.



Place each of the four problems in its correct region, on the assumption that NP is equal to neither P nor co-NP nor PS.

Notes:
Shortest Path (SP) (e.g. Djikstra's Algorithm) is P (polynomial time)
Weighted Hamiltonian Path (e.g. Travelling Salesman) B (NP Hard/NP Complete)
Tautologies: Region: Region C.
Quantified Boolean Formulas: Region D

Easy way to remember the answer: The order in the list in the question corresponds to P -> B -> C -> D.

Place each of the four problems in its correct region, on the assumption that NP is equal to neither P nor co-NP nor PS.

a) Problem TAUT is in region P.
b) Problem WHP is in region D.
c) Problem TAUT is in region C.
d) Problem SP is in region B.

Place each of the four problems in its correct region, on the assumption that NP is equal to neither P nor co-NP nor PS.

a) Problem TAUT is in region F.
b) Problem SP is in region D.

c) Problem QBF is in region D.

d) Problem TAUT is in region P.