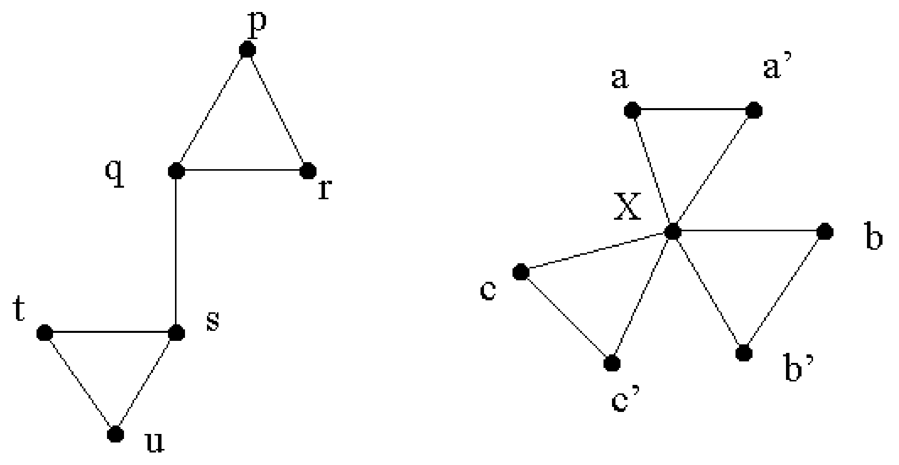
Feedback — Week 6: NP-Completeness and Undecidability

Help Center

You submitted this homework on **Mon 2 Nov 2015 1:37 AM CET**. You got a score of **6.00** out of **6.00**.

Question 1

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:

Your Answer		Score	Explanation
○ (a=TRUE, b=TRUE, c=TRUE, t=red, u=blue, p=red)			
● (a=TRUE, b=FALSE, c=TRUE, t=blue, r=red, p=green)	✓	1.00	
○ (a=FALSE, b=TRUE, c=TRUE, t=blue, u=red, p=blue)			
○ (a=TRUE, b=TRUE, c=TRUE, s=red, t=red, p=green)			
Total		1.00 / 1.00	

Question 2

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 exactly as we did for the reduction of 3SAT to Node Cover (see video 22 -- the last video -- starting at slide 10). 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?

Your Answer		Score	Explanation
○ {(1,1),(1,3),(2,1),(4,3)}			
○ {(1,1),(2,1),(2,3),(3,2),(4,3)}			
○ {(2,1),(3,2),(4,3)}			
● {(1,2), (2,1), (3,2), (4,3)}	✓	1.00	

Total	1.00 / 1.00

Question 3

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 1,000,000?
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 1000?

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

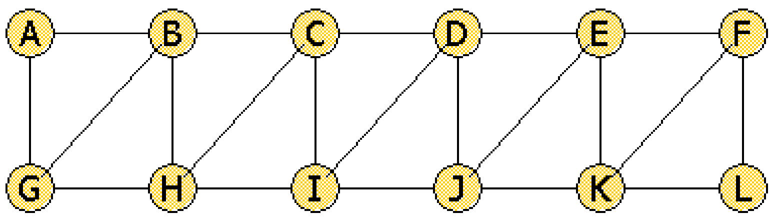
1. Proposition F_1: There is an algorithm A_1 that solves problem P_1 in O(nC) time.
2. Proposition F_2: There is an algorithm A_2 that solves problem P_2 in O(n) time.
3. Proposition F_3: P_3 is NP-complete.
4. Proposition F_4: There is an algorithm A_3 that solves problem P_4 in $O(m^2n^m)$ time for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

Choose a correct statement from the choices below:

Your Answer		Score	Explanation
○ F_4 must be FALSE. Such an algorithm A_3 cannot exist.			
○ If F_1 and F_2 are both TRUE, then P_1 is in P but we cannot conclude so about P_2.			
○ If F_1 and F_2 are both TRUE, then P_1 is not in P and P_2 is not in P.			
● There is nothing inconsistent between propositions F_3 and F_4.	✓	1.00	
Total		1.00 / 1.00	

Question 4

How large can an independent set be in the graph below?



Identify one of the maximum independent sets below.

Your Answer		Score	Explanation
[✓ {A,E,H,L}]	✓	1.00	
{B,D,F}			
{A,C,E,J,L}			
{A,D,H,J,L}			
Total		1.00 / 1.00	

Question 5

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:

Your Answer		**Score**	**Explanation**
"Is L(H) contained in L(G)?" is decidable.			
"Is Comp(L(G)) equal to (0+1)*?" is undecidable.			
[✓ "Is L(G) intersect L(H) equal to (0+1)*?" is undecidable.]	✓	1.00	Correct! The trick is to reduce "L(G) = (0+1)*" to this problem by choosing H to be a grammar whose language is (0+1)*.
"Is L(R) contained in L(G)?" is decidable.			
Total		1.00 / 1.00	

Question 6

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.

Your Answer		Score	Explanation
Allow the TM to run for only n^2 moves when the input is of length n.			
Allow the TM to use only n^2 tape cells when the input is of length n.			
Allow the TM to use only n^3 tape cells when the input is of length n.			
[✓ Allow a tape cell to change its symbol only once.]	✓	1.00	The trick is to simulate any given TM by writing its ID's in succession on a tape, perhaps using another tape to copy ID's. Then, you only need to change blanks to some other symbol, and then leave them unchanged forever.
Total		1.00 / 1.00	