Gradiance Online Accelerated Learning

Zayd

- Home Page
- Assignments Due
- Progress Report
- Handouts
- Tutorials
- Homeworks
- Lab Projects
- Log Out

Help

Submission number: 74830
Submission certificate: JA025615
Submission time: 2014-04-24 00:54:53 PST (GMT - 8:00)

Number of questions: 5
Positive points per question: 3.0
Negative points per question: 1.0
Your score: 0

Based on Sections 10.2 and 10.3 of HMU.

1. 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 y1, y2,...

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) (y2+u)
c) (-y2+y1+v)
d) (-y3+x)

Answer submitted: **b)**

Your answer is incorrect.

Possible error: you may have grouped the expression so (vw)+x was processed first. Hint: remember that when you take the OR of two CNF expressions (i.e., AND's of clauses), introduce a new variable *y* that is added positively (as *y*) to each clause of the first expression and negatively (as NOT*y*) to each clause of the second expression. You should consult the reduction from SAT to CSAT in Section 10.3.3 (p. 452).

Question Explanation:

The first subexpression to which we apply the transformation is vw. The AND rule is simple: take the AND of the clauses for each side. That gives us (v)(w) as the CNF expression.

Next, we work on u+(vw). The rule for OR requires us to introduce variable y1. It is added positively to all the clauses on the left side and negatively to all clauses on the right side. That gives us (y1+u)(-y1+v)(-y1+w).

Finally, we apply the same transformation to (u+(vw))+x, introducing y2. The final answer is (y2+y1+u)(y2+-y1+v)(y2+-y1+w)(-y2+x).

The correct choice is: **a)**

---

2. 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) $(l+z_4+-z_5)$

b) $(g+h+-z_1)$

c) $(m+z_4+-z_5)$

d) $(d+y_1+-y_2)$

Answer submitted: **b)**

You have answered the question correctly.

Question Explanation:

(a+b) becomes $(a+b+x_1)(a+b+-x_1)$.

(c+d+e+f) becomes $(c+d+-y_1)(e+f+y_1)$.

(g+h+i+j+k+l+m) becomes $(g+h+-z_1)(i+z_1+-z_2)(j+z_2+-z_3)(k+z_3+-z_4)$ $(l+m+z_4)$.

---

3. The Boolean expression wxyz+u+v is equivalent to an expression in 3-CNF (a product of clauses, each clause being the sum of exactly three literals). Find the simplest such 3-CNF expression and then identify one of its clauses in the list below. Note: -*e* denotes the negation of *e*. Also note: we are looking for an expression that involves only u, v, w, x, y, and z, no other variables. Not all boolean expressions can be converted to 3-CNF without introducing new variables, but this one can.

a) (w+z+v)

b) (z+u+v)

c) (u+v+-y)

d) (x+y+u)

Answer submitted: **a)**

Your answer is incorrect.

One approach to solving the problem is to use the distributive law of OR over AND. That is, if *E, F*, and *G* are any boolean expressions, then EF+G is equivalent to (E+G)(F+G). Pick the right *E, F*, and *G*, and apply this law several times.

The definition of 3-CNF is in Section 10.3.1 (p. 448). Note that the reduction of SAT to CSAT in Theorem 10.13 (p. 452) and of CSAT to 3-SAT in Theorem 10.15 (p. 457) involve converting arbitrary boolean expressions to 3-CNF expressions, but usually introduce new variables, as they would in this case.

Question Explanation:

The simplest way to proceed is to use the distributing law of OR over AND, three times, to distribute u+v over wxyz. The result is (w+u+v)(x+u+v)(y+u+v)(z+u+v).

The correct choice is: **b)**

---

4. In the following expressions, - represents negation of a variable. For example, -x stands for "NOT x"), + represents logical OR, and juxtaposition represents logical AND (e.g., (x+y)(y+z) represents (x OR y) AND (y OR z).

Identify the expression that is satisfiable, from the list below.

a) (y+z)(y+-z)(-y+z)(-y+-z)
b) (-y+z)(-y+-z)(y)(z+x)
c) (y+z)(-z+x)(-y)(-x+y)
d) (y)(-y+-z)(-z+x)(-x+-y)

Answer submitted: **b)**

Your answer is incorrect.

Hint: Notice that the third clause tells us the expression can only be satisfied if y is true. But then, the first clause can only be satisfied if z is true. Can you complete the argument?

The satisfiability problem is defined in Section 10.2.1 (p. 438).

Question Explanation:

All choices fall into one of four categories (possibly with clauses reordered):

(y)(-y+-z)(-z+x)(-x+-y) is satisfiable. Let y=1 and z=x=0.

(y+z)(y+-z)(-y+z)(-y+-z) is not satisfiable. If y=1 and z=1, the fourth clause is false. If y=1 and z=0, the third clause is false. If y=0 and z=1, the second clause is false. If y=0 and z=0, the first clause is false.

(y)(-y+z)(-z+x)(-x+-y) is not satisfiable. The first clause forces y to be true if the whole expression is to be satisfied. Then, the second clause forces z to be true and the third clause forces x to be true. But then the fourth clause is false.

(y)(-y+z)(-y+-z)(z+x) is not satisfiable. The first clause forces y to be true if the whole expression is to be satisfied. Then, the second clause forces z to be true. But then the third clause is false.

The correct choice is: **d)**

---

5. 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 = F, c = T, d = T, e = T, C = 15
b) a = T, b = T, c = T, d = T, e = T, C = 15
c) a = T, b = F, c = F, d = T, e = F, C = 15
d) a = T, b = F, c = T, d = F, e = T, C = 15

Answer submitted: **b)**

Your answer is incorrect.

Hint: The edges in the cut come from two sets: the edges between complemented and uncompleted literals for the same variable and from the edges between literals in the same clause. Clause $(a + b + c)$ has all literals true and hence contributes nothing to the second set.

Question Explanation:

If each clause has at least two literals with different truth values, they end up in different partitions of the cut. The maximum contribution from a single clause to the total cut-size is hence 2. For 4 clauses, the total is 8. Complemented and uncomplemented literals for the same variable will necessarily be in different partitions of the cut and hence each such pair contributes 1 to the cut-size. There are 7 such pairs in E with a total contribution of 7. Hence maximum cut-size = 8 + 7 = 15.

The correct choice is: **c)**