CS5010 Algorithm Assignment 5

Q1 Complexity Theory

1.1 Which one of the following statements is true about NP-Complete and NP-Hard problems. Explain your answer in detail.

(a) To prove a language A belongs to class NP-Hard, we take a known NP-Hard problem B and reduce B to A

Answer:

The statement is not correct (false).

In order to prove a language A belongs to class NP-hard, we can certainly take a known NP-hard problem B and reduce B to A, but should be in polynomial time. Without the restriction, we can not safely say that A is NP-hard.

(b) NP-Complete is a subset of NP-Hard

Answer:

The statement is correct (true).

A dicision problem L is NP-hard if:

- Every problem in NP is reducible to L in polynomial time.

A dicision problem L is NP-complete if:

- L is in NP.
- Every problem in NP is reducible to L in polynomial time.

As from above, an NP-hard problem L is NP-complete if and only if L is also NP.

So it is safe to say that NP-complete is a subset of NP-hard.

(C) CIRUIT-SAT or circuit satisfiability was the first problem proven to be NP-Complete

Answer:

The statement is not correct (false).

SAT(Boolean satisfiability problem) is the first NP-complete problem proved by Cook.

(d) None of the above

(e) All of the above

The only true statement is (b)

1.2 Let us consider a problem A which is NP-Complete and A has a polynomial time reduction to another problem B. Which one of the following statements is True. Explain your answer in detail

(a) B belongs to class NP

Answer:

The statement is not correct (false).

There's no information that indicates B is NP, even NP-Complete A can be reduced to B in polynomial time.

(b) B belongs to class NP-hard

Answer:

The statement is correct (true).

Since A is NP-Complete, which means that NP can be reduced to A in polynomial time. According to the statement, A can be reduced into B in polynomial time. So NP can be reduced to B in polynomial time, which means that B is NP-hard.

(c) B belongs to class NP-complete

Answer:

The statement is not correct (false). As in (b), we can see that B is NP-hard, however we can not prove that B is NP, like in (a). A NP-Complete is NP-hard and also NP, so B is not a NP-Complete based on current conditions.

(d) All of the aboe

(e) None of the aboe

The only true statement is (b)

1.3 If A is a problem that belongs to NP, then which one of the following statements is true. Explain your answer in detail

(a) A may be undecidable

Answer:

The statement is correct (true).

NP is not P, so A, which is an NP, can not be decided in polynomial time, which can be regarded as undecidable. A might be P as well as NP.

(b) There is no polynomial time algorithm for A

Answer:

The statement is not correct (false).

Although A is NP, we do not know whether there exists a decision algorithm in polynomial time for A, that means, A is P.

(c) If A is NP-Hard, then A is also NP-Complete

Answer:

The statement is correct (true).

According to the defination, an NP-Complete problem is:

- NP-Hard
- NP

If A is NP-Hard, and the statement indicates that A is NP, so under such conditions, A is NP-Complete.

(d) If there is an algorithm to solve A deterministically in polynomial time, then it implies P = NP

Answer:

The Statement is correct s(true).

If the algorithm can find out the answer for A in polynomial time, which means that A is an P, for any instance of problem A, we can decide it in polynomial time. So P = NP.

The true statements are (a), (c), (d).

1.4 Let us consider two decision problems L1, L2. L1 has a polynomial reduction to 3-SAT. 3-SAT has a polynomial time reduction to L2. Assuming these reductions are correct, Which one of the following statements is true? Explain your answer in detail.

(a) L1 belongs to class NP, L2 belongs to class NP-Hard

(b) L2 belongs to class NP, L1 belongs to class NP-Hard

(c) Both L1 and L2 belong to class NP

(d) Both L1 and L2 belong to class NP-Hard

Answer:

Since we know that 3-SAT is NP-Complete, so L1 can be reduced to an NP-Complete, and an NP-Complete can be reduced to L2.

It is easy to prove that L2 is NP-Hard, since there's no information about whether L2 is NP, and every NP problem can be reduced to NP-Complete in polynomial time, and an NP-complete, here as 3-SAT, can be

reduced to L2 in polynomial time. This means that every NP problem can be reduced to L2 in polynomial time, but L2 may not be NP, so L2 is an NP-Hard problem.

Now we have L1 can be reduced to NP-Complete in polynomial time. As the defination says, every NP-problem can be reduced to NP-Complete in polynomial time. Also as the mapping rules indicates, one problem that can be reduced to an NP-Complete can not be outside of NP, every NP can be reduced to NP-Complete in polynomial time. So L1 is NP.

The only true statement is (a). L1 is NP and L2 is NP-Hard.

1.5 Let us look at the following statements: (i) The problem of cycle detection in an undirected graph is in P, (ii) The problem of cycle detection in an undirected graph is in NP, (iii) If a problem A is NP-Complete, it implies there exists a non-deterministic polynomial time solution for A. Given the above statements, which one of the following statements is true. Explain your answer in detail.

(a) (i), (ii)

(b) (i), (iii)

(c) (ii), (iii)

(d) (i), (ii), (iii)

Answer:

The correct choice is (a)

For (i):

For any graph, to determine whether there's a cycle in the graph, we only have to go over the edges and add every node that the edge connects into a visited list, if the node has already been added before, there exists a cycle, if no node appears more than once at the end of the iteration, there's no cycle.

This solution takes O(n) time, which is polynomial time. So we can actually solve (decide) the problem in polynomial time, so the problem is P.

For (ii):

Any proposed solution provided can be easily determined by going through the solution. Thus, the problem is NP (quick checkable).

For (iii):

NP-Complete only means that all NP problems can be reduced to it in polynomial time, but this does not mean that there must exist a solution in NP time for NP-Complete. Take halting problem as an example, there's no clue that whether the program would finish running or run forever, so there may not be a NP solution for NP-Complete problem.

Q2 Vertex Cover

Vertex cover is a graph problem defined as follows. Given a graph G(V,E) and a positive integer K, you have to find if there is a subset of vertices V' of size at most K such that every edge in the graph G is connected to some vertex in V' (that is, all edges are covered)

Using NP reductions, prove that the Vertex Cover problem is NP Complete. For this problem, you can assume that Clique, 3-CNF-SAT, SAT and CIRCUIT-SAT are given as NP complete problems

Solution:

The instance of the problem is a graph G(V,E) and a positive integer K, and the problem is to check whether a vertex cover of size at most K exists in G. Since an NP Complete problem, by definition is a problem which is both NP and NP-Hard, first we prove that this problem is NP.

1. Prove the Vertex Cover is NP.

   The certificate for the vertex cover problem is a subset V' of V, which contains the vertices in the vertex cover. We can check whether the set V' is a vertex cover of size k using the following strategy (for a graph G(V, E)):

   ```
   input V, E and K
   set cnt = 0
   for vertex in V:
       remove all edges adjacent to vertex from E
       increase cnt by 1
       if cnt == K and E is empty:
           the solution is correct.
   If comes to here
       The solution is incorrect.
   ```

   For any given V, E and K and solution, we can verify it with the above procedure, so we can verify the solution in polynomial time.

   Thus Vertex Cover is NP.

2. Prove the Vertex Cover is NP-Hard

   Now we consider how to prove Vertex is NP-Hard. In order to prove this, we can find that if any known NP-Complete problem can be reduced to Vertex Cover in polynomial time.

   Now we can use Clique, which is known as NP-Complete, to prove the Vertex Cover is NP-Hard.

   we consider the problem of finding out whether there is a clique of size k in the given graph. Therefore, an instance of the clique problem is a graph G (V, E) and a non-negative integer k, and we need to check for the existence of a clique of size k in G.

   Now, we need to show that any instance (G, k) of the Clique problem can be reduced to an instance of the vertex cover problem. Consider the graph G' which consists of all edges not in G, but in the complete graph using all vertices in G. Let us call this the complement of G. Now, the problem of

finding whether a clique of size k exists in the graph G is the same as the problem of finding whether there is a vertex cover of size |V| – k in G'. We need to show that this is indeed the case.

Assume that there is a clique of size k in G. Let the set of vertices in the clique be V'. This means |V'| = k. In the complement graph G', let us pick any edge (u, v). Then at least one of u or v must be in the set V – V'. This is because, if both u and v were from the set V', then the edge (u, v) would belong to V', which, in turn would mean that the edge (u, v) is in G. This is not possible since (u, v) is not in G. Thus, all edges in G' are covered by vertices in the set V – V'.

Now assume that there is a vertex cover V'' of size |V| – k in G'. This means that all edges in G' are connected to some vertex in V''. As a result, if we pick any edge (u, v) from G', both of them cannot be outside the set V''. This means, all edges (u, v) such that both u and v are outside the set V'' are in G, i.e., these edges constitute a clique of size k.

Thus, we can say that there is a clique of size k in graph G if and only if there is a vertex cover of size |V| – k in G', and hence, any instance of the clique problem can be reduced to an instance of the vertex cover problem. Thus, vertex cover is NP Hard.

Since vertex cover is in both NP and NP Hard classes, it is NP Complete.