Week 9 - lecture 2

Introduction:

In this lecture , we discussed about the class of decision problems particularly about NP and NP complete problems and also understood them through example of the 3-SAT and clique problem .

Decision Problems :

These are the problems which are addressed as membership query in a language (set of strings) that is either the answer is YES or NO , that is either the query belongs to that language or not .

Hence these problems are easier to solve than non;decision problems since a !YES implies that the answer is NO .

Example : Given a subset of numbers , is the sum of number of that set equal to x ?

This only has a true or false answer and hence is a decision problem.

We can also convert every problem to a decision problem by asking if the proposed solution a part of the solution set of that language.

Class of P :

Here P stands for polynomial , hence class P is the class of problems that can be solved in polynomial time . That is say the input size is n , then the running time of the algorithm would be n^x where x will be a finite number .

More formally we can say that a problem W will be in class of P if there is a polynomial time algorithm (A) such that :

- if W(x) is true , A(x) is also true
- if W(x) is false ,A(x) is also false

Class of NP :

NP are class of problems which can be solved only in exponential time , that is no polynomial time algorithm exists , but given a proposed solution , we can verify the output in polynomial time .

Hence NP are a class of problems which have efficient verifiers that is there exists a polynomial time algorithm which can verify whether the output is correct or not .

Problem solved only in exponential time means unlike class of P , here b will be an exponent while calculating the complexity and not base.

Let us see two examples:

Clique Problem :

Clique = { G , k | G is an undirected graph with a k clique.

What is a clique ?

A clique is a subset of vertices of an undirected graph such that every pair of distinct vertices from the set are adjacent of have an edge connecting them .

There exists no polynomial time algorithm to find clique in a graph however given a solution we can verify whether it is correct or not in polynomial time .

The verifier V can be like :

For input << G , k>> ,c > :

- Check if c is a set of k nodes in G.
- Check if G contains all edges connecting nodes in c
- if both output true , return YES else return NO

Both the checks can be performed in polynomial time , hence the verifier works in polynomial time .

Hence clique problem is NP.

Similarly we can also show that the subset sum problem is also NP that is whether it is possible to partition the set S into two disjoint sets A and B such that sum of elements in A = sum of elements in B .

P vs NP problems :

All problems in class of P also belong to class of NP since if we can solve a problem in polynomial time , we can also verify the solution in polynomial time .

However we cannot comment anything about the relation P=NP since the known algorithms to solve NP class problems are exponential but that doesn't rule out the possibility of polynomial time solution for these problems since in future we might be able to find such polynomial time algorithms also for the class of NP problems .

Co-NP :

Co-NP are those problems for which these is a polynomial time algorithm which can verify that the problem is impossible to solve.

For example $\rightarrow$ 3-SAT is a NP problem , that is to find if we can get output as true of the given 3-SAT . The co-NP problem would be to find if it is possible to get output as false.

We don not know how different co-NP problems are from NP problems .

Polynomial Time Reduction :

Suppose we know the solution to a very hard problem B in polynomial time and we want to solve some problem A .

If we can convert the input of A to input of B in polynomial time , then we say that we solve the problem in polynomial time since we can first convert the input of A to input of B in polynomial time and then we can find the solution since we know the solution for B .

Hence $A <=_p B$ which means that problem A is at least as easy to solve as B .

Hence all we need to do is to find if there exists a function $f(x)$ which can take the input of A as input and output the corresponding input for B.

Let us see how we can do this through an example :

3-SAT is polynomial time reducible to Clique problem:

Here , we will try to draw equivalence between the 3-SAT and the Clique problem .

So we will try to show that if we have a solution for 3-SAT , it means we have solution for clique and similarly if we know solution for Clique problem , we can also solve the 3-SAT problem .

- Reducing 3-SAT to Clique problem :

Let us try to first construct a graph with 3.k nodes where k is the number of classes in our CNF .

We can draw edges between all nodes which satisfy following two conditions :

a) Nodes of same class are not connected

b) No edge is present between nodes with contradictory labels that is x cannot be connected with !x (not of x).

- 3-SAT to clique :

Suppose we know some solution to the 3-SAT problem for which the equation evaluates to true , then since we have k classes , that means that we have at least k variables which have value of true since we are ANDing all classes . Now from our graph construction we can see that those k variables make a clique of size k since in our graph construction we have already connected all such edges .

Hence solution to Clique problem is equivalent to getting solution for clique problem

- Clique to 3-SAT :

Suppose G has a k -clique . This means that we have selected k nodes that is one node from each of our k classes since nodes from same class do not have an edge between them . Now to all such nodes , we can assign true value and hence the expression will now evaluate to true since in each of the k classes we have a variable with true value and so the AND of all will also be true ..

Hence solution to clique problem can be converted to 3-SAT .

From the above example we saw how we can convert one Np problem to another .

NP - Complete :

A problem is NP - complete if :

- It is in the class of NP
- We can reduce all problems in class of NP in polynomial time to our NP - complete problem .

Hence if we can solve a NP complete problem in polynomial time , then P=NP since now we can reduce every NP problem to NP complete problem and then solve it in polynomial time .

We can observe that if say problem A is NP complete and $A <=_P B$ for B in NP , then B is also NP complete .

The SAT problem for example is NP - complete . It can be easily verified and it is also NP hard by Cook - Levin theorem .

Similarly 3-SAT is also NP complete problem since we can convert and SAT problem that is any k - SAT (k>3) problem to a 3-SAT problem by reducing the literals to CNF form as follows :

For a class with literals $(a_1, a_2 ... a_l)$, we can reduce it as :

$(a1 OR a2 OR z1) AND (!z1 OR a3 OR z2) AND ..... (!z_{l-3} OR a_{l-1} OR a_l)$

Hence any k-SAT problem can be reduced to 3-SAT and since SAT is NP complete , 3-SAT is also NP complete.