Algorithms Analysis and Design

Week 9 - Diary

Ayan Agrawal (2020101034)

Lecture 15 : NP-Completeness

Problems for the class :

- Class P and NP
- Polynomial Time Reducibility
- 3-SAT Problem and its relation with the Clique problem
- NP Complete problems

Class P and NP

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**.

Class 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 NP :

The problems that have solutions which take more than polynomial time, maybe exponential time but have polynomial time verifiable verifiers are usually placed in the NP Class.

By polynomial time verifiers, we mean, in a hypothetical situation if we get a solution for an NP Problem, then the verifier that would verify this problem would take only polynomial time to do so. Such kind of problems are placed in the NP Class. Formally a verifier for any language can be stated as follows :

$$A = \{w|\ V \text{ accepts } < w, c > \text{for some string } c\ \}.$$

A is the language and V is the algorithm used by the verifier.

Clique Problem :

The CLIQUE Problem is one of the most famous NP problems till date, this is a very simple problem to understand, it states that given a graph, find the maximum sized clique in the graph G.

$$\text{Clique} = \{\text{G , k} \mid \text{G is an undirected graph with a k clique}\} .$$

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.

To prove that it is of the NP Class, we need to prove that the verifier algorithm V for the same is of the order polynomial complexity. The proof for the same goes as follows :

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.
- we return **YES** for both output to be true else **NO**.

Both the checks can be performed in polynomial time , hence the verifier works in polynomial time. Hence clique problem is NP.

P vs NP problems :

If we can solve a problem in polynomial time then definitely the solution can be verified in polynomial time, so every problem in class P is also belonging to class of NP problems.

However, we cannot remark on the relationship $P = NP$ because the known algorithms for solving NP class problems are exponential. However, this does not eliminate the possibility of polynomial time solutions for these problems because such polynomial duration algorithms may be found in the future for the class of NP problems.

Co-NP problems :

These are the problems in which a polynomial time algorithm exists to verify that the problem is unsolvable.

Example :

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 tells if it is possible to get output as false. We don't know how different Co-NP problems belong to class of NP problems .

Polynomial Time Reduction :

Any language A is polynomial reducibility to a language B if we can find a function f with polynomial complexity such that for each $w \in A$, a mapping to set B exists, such that $f(w)$ belongs to B. So, if a language B of P class then language A should belong to class of P as well because a function will allow us to convert from language A to B in polynomial complexity.

We can say that $A <=_P B$. The function f is called the polynomial time reduction of A to B. Formally we can say that :

If $A <=_P B$ and $B \in P$, then $A \in P$. " On input w :

1. Compute $f(w)$.
2. Run M on input $f(w)$ and output whatever M outputs. "

Here M is a Turing Machine having polynomial complexity that exists and halts with just $f(w)$ on the tape, when started with any input w.

3-SAT Problem and its relation with the Clique problem :

We try to draw a relation between 3-SAT and Clique problem here. We'll 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's try to construct a graph with $3.k$ nodes, k is the # of classes in CNF.

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

1. Nodes of same class are not connected
2. 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 t a 3-SAT problem for which equation evaluates to be true, then as we have k classes, that means we have at least k variables which have value as true since we are ANDing all classes . Now from our graph construction, we can see that those k variables make a size k clique since in our graph construction we have already connected all such edges .

- **Clique problem to 3-SAT :**

  Suppose graph G has a k -clique which implies we have selected k nodes that is one node from each of our k classes since nodes from same class cannot be connected. We can assign "true" value to all such nodes 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 ing of all will also be true.

From the above shown example, we learnt how we can convert from a NP- problem to another.

NP-Complete :

For any language B to be NP -Complete, following criteria must be followed :

1. B is in NP.
2. every A in NP is polynomial time reducible to B.

Also, if B is NP Complete and $B <=_P C$ for C in NP, then C is NP-Complete.

Proof :

We know language B is NP -complete which means for every $A \in NP$, A can be converted to B by the means of some function (polynomial time reducible).

Also by previous definition, if $B <=_P C$, then there exists some f which is a polynomial time reducible operation from B to C, thus, combining two polynomial reducible functions, for every language A in NP, we can convert them to C in polynomial time, thus C is NP -Complete.

Now, SAT is NP-complete problem by the Cook-Levin Theorem, we can use this only for proving that 3-SAT problem is also NP-complete.

For any clause containing, l literals, like : $(a_1 \vee a_2 \vee a_3 \ldots\ldots \vee a_l)$, we can replace with $l-2$ literals as follows :

$(a_1 \vee a_2 \vee z_1) \wedge (z_1' \vee a_3 \vee z_2) \wedge (z_2' \vee a_4 \vee z_3) \wedge \ldots\ldots \wedge (z_{l-3}' \vee a_{l-1} \vee a_l)$.

Hence any SAT Problem can be converted into 3-SAT thus by the previous proof, we find that 3-SAT is NP- Complete.