week9_lec2

Main Ideas

- NP : Polynomial Time Verifiable Languages
  - Examples
    - Clique
    - Subset - Sum
  - NP vs Co-NP
- Polynomial Time Reduction
- NP Completeness

NP : Polynomial Time Verifiable Languages

A verifier for a language A is an algorithm V, where
$A = w | V\ accepts\ [w, c]\ for\ some\ string\ c$

A polynomial-time verifier runs in polynomial time in the length of w since we only measure the duration of a verifier in terms of the length of w.

A language A is polynomial verifiable if it has a polynomial time verifier.

All problems that can be verified in polynomial time are classified as NP(nondeterministic polynomial time).

Thus, NP is the class of languages that have polynomial time verifiers. Verifying whether a problem is NP class is simple, we just run the polynomial time verifier.

NP Language Examples :

Clique

$CLIQUE = ((G, K) | G\ is\ an\ undirected\ graph\ with\ K - Clique)$

A $k - clique$ means a complete sub-graph of size k.

Verifier for Clique:

- Input: ((G,K),c)

- Test whether c is a set of K nodes in G.
- Test whether G contains all edges connecting nodes in C.
- If both pass: Accept else reject

SUBSET-SUM

$SUBSET - SUM = ((S,T)|S = (x_1, x_2, ...x_k)\ and\ for\ some\ (y_1, y_2, ...y_l) \subseteq (x_1, x_2, ..x_k),\ we\ have \sum y_i = t)$

Verifier for SUBSET-SUM:

- On Input: ((S,T),c)
- Test whether c is a collection of numbers that sum to T.
- Test whether S contains all numbers in C.
- If both pass: Accept else reject.

NP v/s Co-NP

We observe that completeness of set, CLIQUE and SUBSET-SUM are not members of NP.

Co-NP contains the languages that are complements of languages in NP. This seems to be harder to verify than verifying the correctness of NP class problems.

Polynomial Time Reducibility

A function $f = \sum * \rightarrow \sum *$ is a polynomial time computable function if some polynomial time Turing Machine M exists that halts with just f(w) on its tape, when started on any input w.

Polynomial time reduction from A to B

Language A is polynomial time reducible to language B if a polynomial time computable function exists where for every w

The function f is called the polynomial time reduction of $AtoB$.

$w \in A \Leftrightarrow f(w) \in B$

Theorem :

$if\ A \leq_p B\ and\ B \in P,\ then\ A \in P$

Example :

3SAT is polynomial time reducible to CLIQUE

where $3SAT = \{< \phi > \mid \phi\ is\ a\ satisfiable\ 3\ cnf - formula\}.$

Proof — Let $\phi = (a_1 \vee b_1 \vee c_1) \wedge (a_2 \vee b_2 \vee c_2)... \wedge (a_k \vee b_k \vee c_k).$

We reduce this boolean formula into an undirected graph by grouping the nodes in G into k groups of three nodes, each called a triple $t_1, ..., t_k$.

Each of these triples corresponds to one clause of the boolean formula and each node in the triple corresponds to a literal in the corresponding clause.

In the resulting graph G, all nodes are connected by an edge except between nodes in the same triple and between complementary nodes. This can be visualized as a graph for better understanding.

Now, we need to show that a Boolean formula is satisfiable if and only if G has a $k - clique$. Suppose this holds. In the assignment at least one literal in each clause is true, so we select the corresponding node in each triple of the graph G. If more than one literal is true, pick one arbitrarily.

Now, the collection of all nodes chosen form a $k - clique$ because we chose one from each of the k triples.

Verifying, we get the following for each pair of selected nodes —

- Joined by an edge because no pair fits one of the exceptions previously mentioned.
- Not from the same triple because only one node was selected from each triple.
- Not conflicting labels, because the associated literals were both true in the assignment.

Thus, G contains a $k - clique$. Thus, ϕ is satisfiable.

NP- Completeness

A language B is NP-complete if:

- B is in NP classevery
- A in NP is polynomial time reducible to B

If B is NP-complete and $B \leq_P CforC$ in NP, then C is NP-complete.

Proof — We know that C is in NP, so we need to show that every A in NP is polynomial time reducible to C to satisfy the second condition.

Because B is NP-complete, every language in NP is polynomial time reducible to B and B is polynomial time reducible to C.

However, polynomial time reductions have the property of composition. Thus, if A is polynomial time reducible to B and B is polynomial time reducible to C, A is polynomial time reducible to C. Thus, the second condition is satisfied for C and thus, it is NP-complete.

COOK-LEVIN Theorem

$SAT\ is\ NP\ Complete$

$(a_1 \cup a_2 \cup a_3 \cup ...a_l)$

we can replace it with l-2 Clauses

$(a_1 \cup a_2 \cup z_1) \cap (\bar{z_1} \cup a_3 \cup z_1) \cap .....(\bar{z_{l-3}} \cup a_{l-1} \cup a_l)$