NP Completeness

In this chapter, we are concerned with the problem that can be solved by polynomial time and the problem for which no polynomial time algorithm is known.

The algorithm which is solved in polynomial time is called **polynomial time algorithm**, whose running time is $O(n^k)$ for some constant k and input size n. The problem that is solved in polynomial time is called **tractable or easy**. The problems, those require exponential time is **intractable or hard**.

Deterministic algorithm and non-deterministic algorithm:

In deterministic algorithm, for a given particular input, the computer will always produce the same output going through the same states.

Non-deterministic algorithms are algorithm that, even for the same input, can exhibit different behaviors on different executions. It is the algorithm whose output cannot be pre determined.

```
Algorithm  : NDSearch(A,n,key)
1.j = Choice()
2.if key = A[j]
3.      Write j
4.      Success()
5.else,failure()
```

Class P

Class P consist of those problems that are solvable in polynomial time. A problem is said to be polynomial bound, if there exist a polynomial bound algorithm for it.

Any set of decision problem with 'yes' or 'no' answer is polynomial bound.

Class NP: (Non-deterministic polynomial time)

Class NP consists of those problems that are verifiable in polynomial time. That means if we have somehow given a certificate of a solution and we would verify that the certificate is correct in polynomial time.

Decision Problem: vs Optimization problem:

1. decision problem has 'yes' or 'no; answer.

   where as in optimization problem, each solution has a value and we wish to find optimal value/solution.

2. If optimization problem is easy, its related decision problem is also easy.

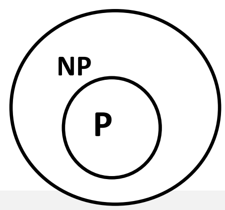
3. If decision problem is hard, its related optimization problem is also hard.

$\underline{P \subseteq NP?}$

Any problem in P is also in NP. Since if a problem is solved in polynomial time is also verifies in polynomial tome. As all NP problems are verified in polynomial time. So $P \subseteq NP$.

$\underline{P = NP?}$

Till Date it is unanswered.



Class NPC:

NPC is a class of problem whose status is unknown. That means no polynomial time algorithm has yet been discovered for any NP complete problems, nor any one has yet been able to prove that no polynomial algorithm can exists for any of them.

If any NPC problem can be solved in polynomial time, then every problem in NPC has a polynomial time algorithm.

A problem is in NPC, if **It is in NP** and It is **NP Hard**. This means, a problem (L) is in NPC if,

1. $L \in NP$
2. $L1 \leq_P L$ for all $L1 \in NP$

A problem L that satisfies property 2 but not necessarily property 1, then it is called **NP Hard**.

Verification Algorithm:

A verification algorithm is an algorithm (A) that takes two inputs.

1. An ordinary encoded input(X)
2. A certificate (Y)

The Algorithm A verifies the input string X, if there exist a certificate Y such that A(X,Y) = 1

Example: Hamiltonian cycle of an undirected graph G=(V,E) , Hamiltonian circuit is a simple cycle that contains each vertex in V visited only once except the start vertex.

As an optimization problem, we require to obtain a Hamiltonian cycle, which requires enumeration of all permutation of each vertices in V and check each permutation is a Hamiltonian cycle, which requires exponential time. i.e. O(n!)

As a decision problem, a list of vertices in the Hamiltonian cycle is given as certificate(Y), and we have to verify that the certificate is correct or not.

For this, check the list of vertices is a permutation of the vertices of V, and whether the consecutive edges along the cycle actually exist in the graph.

This verification can be implemented in the order of n i.e. O(n) .

Reducibility

A problem Q can be reduced to another problem Q1 if any instance of Q can be easily rephrased as an instance of Q1, solution to which provides a solution to the instance of Q.

Example: The problem ax + b = 0 can be converted to $0x^2$ + ax +b =0, whose solution provides the solution to ax + b = 0.

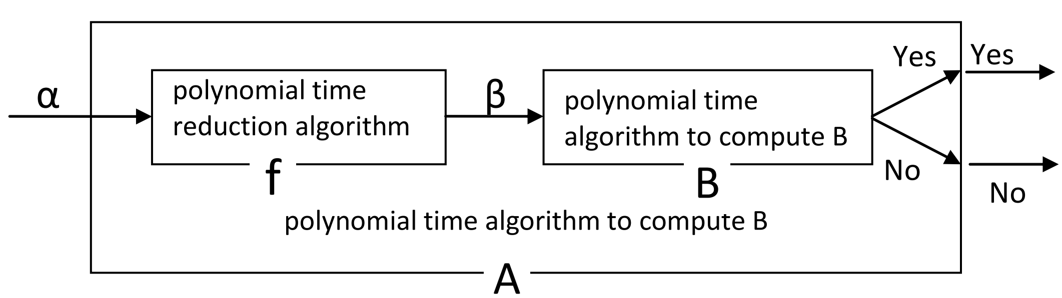
If a problem Q is reduced to another problem Q1, then Q is **no harder than** Q1

Polynomial time reducibility

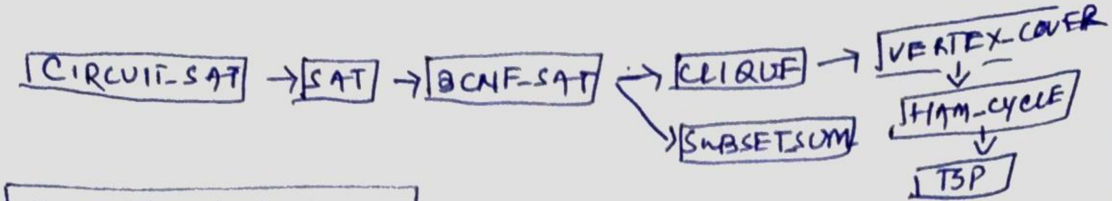
A problem Q is polynomial time reducible to Q1,(written as $Q \leq_p Q1$ if there exist a polynomial time computable function $f:\{0,1\}^* \rightarrow \{0,1\}$ such that for all x ϵ Q iff f(x) ϵ Q2.

The function f is called reducible function. The algorithm that computes f is called reduction algorithm. For a polynomial time reduction algorithm following condition must hold.

- ➔ Transformation takes polynomial time
- ➔ Answer for an instance of Q is YES if and only if answer for an instance of Q1 is YES



NP Complete problems ①

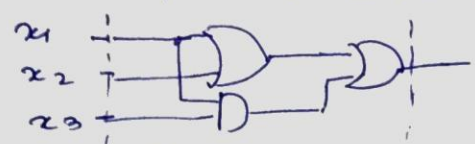


Circuit Satisfiability

A boolean combinational circuit consist of one or more boolean gates connected by wires.

A truth assignment for a boolean circuit is a set of boolean input. A circuit is satisfiable, if it has satisfying assignment i,e the inputs that cause the output of circuit to be 1.

Ex:



The assignment $\langle x_1 = 1, x_2 = 0, x_3 = 1 \rangle$ is satisfiable

so a circuit satisfiable problem is "Qi –
Given a boolean circuit composed of AND, OR, NOT Gates is satisfiable or not.

CIRCUIT-SAT = { <C> : C is a satisfiable boolean combinational circuit }

Proof : Circuit satisfiable problem ∈ NP.
We have to show that the problem is verifiable in Polynomial time. The verification algorithm has 2 input : 1 → Encoding of boolean circuit.
2 → certificate consisting the assignments (C)

For each logic gate, the algorithm checks the values in assignment on the encoded input. If the output of circuit is 1, the algorithm outputs 1. As the length of certificate is polynomial in the size of C, the algorithm runs in Polynomial time.

So circuit-sat ∈ NP

2) Formula satisfiability problem (SAT)

The objective of this problem is to determine whether a boolean formula (not circuit) is satisfiable.

A boolean formula ϕ consist of

1. n boolean variables $x_1, x_2 \dots x_n$
2. m boolean connectives $(\wedge, \vee, \sim, \rightarrow \dots \leftrightarrow$
3. Parenthesis

The problem is defined as -

$$SAT = \{ \phi : \phi \text{ is a satisfiable formula} \}$$

i.e A formula ϕ with truth assignment that evaluates 1.

Ex: $\phi = (x_1 \vee x_2) \wedge (x_1) \vee (\bar{x}_1 \vee x_2 \wedge \bar{x}_3)$

Above formula ϕ has satisfying Assignment $(x_1 = 1, x_2 = 1, x_3 = 0)$

General Alg. to determine any formula is satisfiable requires exponential time. As there are 2^n possible Assignment in a formula with n variable. Checking every Assignment requires $\Omega(2^n)$ time.

To prove SAT is NP complete, Prove,

1. SAT $\in$ NP.
2. CIRCUIT-SAT $\leq_p$ SAT.

Proving SAT $\in$ NP.

Since a certificate consist of a satisfying assignment can be easily verified in polynomial time by replacing each variable with its value and then evaluating the formula to test the output is 1.

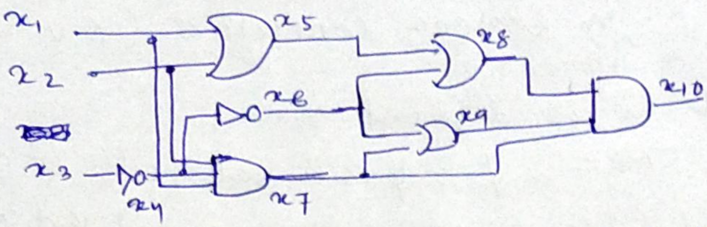
Hence proved.

④

Prove SAT is NP Hard i.e CIRCUIT_SAT $\le_p$ SAT

Any instance of circuit satisfiability problem can be reduced to an instance of formula satisfiability problem in polynomial time

Ex: let a circuit



For each wire x_i in circuit c, the formula ϕ has a variable x_i. The formula is produced by the reduction algorithm is the AND of circuit output variable with conjuction of clauses describing the operation of each gate. For above circuit the formula ϕ is -

$$
\begin{aligned}
\phi = x_{10} & \wedge (x_4 \leftrightarrow \neg x_3) \\
& \wedge (x_5 \leftrightarrow (x_1 \vee x_2)) \\
& \wedge (x_6 \leftrightarrow \neg x_4) \\
& \wedge (x_7 \leftrightarrow x_1 \wedge x_2 \wedge x_4) \\
& \wedge (x_8 \leftrightarrow (x_5 \vee x_6)) \\
& \wedge (x_9 \leftrightarrow (x_6 \vee x_7)) \\
& \wedge (x_{10} \leftrightarrow (x_8 \wedge x_9 \wedge x_7))
\end{aligned}
$$

SSG Mishra

(5)

3-CNF Satisfiability problem (3-CNF-SAT)

– 3rd Conjuctive Normal Form Satisfiability

3-CNF-SAT is NP complete if
1- 3-CNF-SAT $\in$ NP
2- SAT $\leq_p$ 3-CNF-SAT

A boolean formula is in conjuctive Normal form if it is expressed as an AND of clauses, each of which is the OR of one or more Literals.

A boolean formula is in 3-CNF if each clauses has exactly 3 distinct Literals

Ex: $(x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_4) \wedge (x_2 \vee \overline{x_3} \vee \overline{x_4}) \wedge (x_3 \vee x_4 \vee \overline{x_5})$

In 3CNF problem, we have to determine whether a boolean formula is satisfiable or not.

Proof: 3 CNF-SAT $\in$ NP.

A certificate consist of satisfying assignment to the formula can be easily verifiable in polynomial time by replacing each variable in the formula with its value and then evaluate the formula to test whether it outputs 1 or not.

Clique

A clique in an undirected graph $G=(V,E)$ is a subset $V' \subseteq V$ of vertices, each pair of which is connected by an edge in E.

In other words, clique is a complete subgraph of G. The size of clique is the number of vertices, it contains.

As optimization problem, the clique problem is to find a clique of maximum size.

As decision problem, the problem determines, whether a clique of a given size 'K' exists in the graph.

i.e. $CLIQUE = \{\langle G,K\rangle : G$ is a graph with a clique of size $K\}$

Proof: clique is NPC if

1. $CLIQUE \in NP$
2. $3\text{CNF-SAT} \leq_p CLIQUE$.

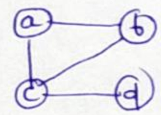
1. To show clique is in NP,

For a given graph $G = (V,E)$, we are given the set $V' \subseteq V$ of vertices as a certificate. Checking V' is a clique can be accomplished in polynomial time by checking whether for each pair $(u,v) \in V'$, the edge belongs to set E., which can be done in $O(n^2)$ time. - Proved.

Vertex cover problem

vertex cover of an undirected graph $G = (V, E)$ is a subset $V' \subseteq V$ such that if $(u,v) \in E$, then $u \in V'$ or $v \in V'$ or both. The size of a vertex cover is the number of vertices in it.

Ex:



vertex cover = $\{a, c\}$ size 2
or $\{b, c\}$ size 2.

As an optimization problem, vertex cover problem is to find a vertex cover of minimum size in a given graph.

As a decision problem, we have to determine whether a graph has a vertex cover of a given size 'k'.

VERTEX-COVER = $\{\langle G, k\rangle$: Graph G has a vertex cover of size k $\}$

Proof: VERTEX-COVER is NPC, if

1. VERTEX-COVER $\in$ NP
~~2. VERTEX~~
2. CLIQUE $\leq_p$ VERTEX-COVER

To show vertex cover $\in$ NP

Given a graph G and an integer k, the certificate is the vertex cover $V' \subseteq V$. We need to verify $|V'| = k$ and for each edge $(u,v) \in E$, $u \in V'$ or $v \in V'$ or both. This verification can be done in polynomial time, i.e $O(V'E)$ times.

Hamiltonian Cycle:

Hamiltonian cycle of an undirected graph G=(V,E) , Hamiltonian circuit is a simple cycle that contains each vertex in V visited only once except the start vertex.

As an optimization problem, we require to obtain a Hamiltonian cycle, which requires enumeration of all permutation of each vertices in V and check each permutation is a Hamiltonian cycle, which requires exponential time. i.e. O(n!)

As a decision problem, a list of vertices in the Hamiltonian cycle is given, and we have to verify that the certificate is correct or not.

HAM-CYCLE= {<G>: G is a Hamiltonian cycle}

HAM-CYCLE is NP Complete if

1. $HAM-CYCLE \in NP$
2. $VERTEX-COVER \leq_p HAM-CYCLE$

HAM CYCLE is in NP

Given in instance G(V,E) and a certificate containing list of vertices. We have to verify that the list of vertices is a permutation of the vertices of V, and whether the consecutive edges along the cycle actually exist in the graph. This verification can be implemented O(n) time . So HAM-CYCLE is in NP

Travelling salesman problem

Given a complete graph with n vertices (n cities). A salesman ~~who~~ wishes to make a hamiltonian tour (i.e visiting each city exactly once and finishing the tour at the city, he starts from).

Let $c[i,j]$ is the cost to ~~the~~ travell from city i to city j. The objective of the problem is to find a tour, whose total cost is minimum.

$TSP = \{\langle G, c, k\rangle, G = (V,E)$ is a complete graph, c is a cost function for the G and the cost of the tour is atmost $k\}$

As optimization problem, we have to find a tour with minimum cost.

As verification problem, we are given a certificate consist of n cities and ~~we~~ a cost k and we have to verify that the certificate is correct.

TSP is NP complete if.

1. $TSP \in NP$.
2. ~~TSP~~ HAM-CYCLE $\leq_p$ TSP.

TSP is in NP.

A certificate is given as the sequence of n vertices in the tour. Verification algoritham checks that the sequence contains each vertex exactly once, sums up the edge costs and check whether the sum is atmost k. This will be done in polynomial time. So $TSP \in NP$.

Subset Sum Problem

Given a set S of n distinct numbers and a target T greater than 0, the objective of the subset sum problem is to find a subset S1 of set S whose elements sums to T.

Example: Given n=5 , M = 35 , W={5,10,15,18,30},

The subset S1 = {5, 30} .

The problem can be defines as –

$$SUBSET-SUM = \left\{ < S,T >: \text{there exist a subset } \; S1 \subseteq S \text{ such that } \; T = \sum_{s \in S1} s \right\}$$

SUBSET-SUM is NP Complete if

3. $SUBSET-SUM \in NP$
4. $3CNF-SAT \leq_p SUBSET-SUM$

<u>To show SUBSET SUM is in NP:</u>

Given an instance <S,T> of the problem, let the subset S1 is a certificate, we can verify whether $T = \sum_{s \in S1} s$ in O(n) time. As verification algorithm for the problem takes polynomial time, the problem is in NP.