CHAPTER - 1

PRELIMINARIES
CHAPTER 1

1.1 SOME NECESSARY DEFINITIONS

GRAPH:

A Graph G is an ordered triple $(V(G), E(G), \Psi_G)$ consisting of a non-empty set $V(G)$ of vertices, a set $E(G)$ of edges and an incidence function $\Psi_G$, that associates with each edge of G of an unordered pair of vertices of G.

EDGES & VERTICES:

In a Graph G, if e is an edge and u, v are vertices such that $\Psi_G(e) = uv$, then e is said to join u and v, here the vertices u and v are called the ends of e.

Eg:

Let $G = (V(G), E(G), \Psi_G)$ be a graph with

$V(G) = \{v_1, v_2, v_3, v_4, v_5\}$,

$E(G) = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8\}$

and $\Psi_G$ is defined by

$\Psi_G(e_1) = v_1v_2, \ \Psi_G(e_2) = v_2v_3, \ \Psi_G(e_3) = v_1v_3, \ \Psi_G(e_4) = v_3v_4,$

$\Psi_G(e_5) = v_4v_5, \ \Psi_G(e_6) = v_3v_5, \ \Psi_G(e_7) = v_5v_5, \ \Psi_G(e_8) = v_1v_5$

Then the corresponding graph is defined as,

INCIDENCY:

In a Graph G, if $v_1$ is an end vertex of an edge e (=v_1v_2 or v_2v_1), then we say that vertex $v_1$ and edge e are Incident with each other

ADJACENCY:

Two vertices $v_1$ and $v_2$ of a graph G are said to be adjacent to each other, if there must be an edge which joins $v_1$ and $v_2$.

Similarly, two edges $e_1$ and $e_2$ are said to be adjacent edges, if they have a common end vertex.
Eg:

Adjacent vertices of \( v_1 \) are \( v_2, v_4, v_5 \).

Adjacent vertices of \( v_2 \) are \( v_1, v_3, v_5 \).

Adjacent vertices of \( v_3 \) are \( v_2 \) and \( v_4 \).

Adjacent edges of \( e_1 \) are \( e_7, e_6, e_4, e_2 \).

Adjacent edges of \( e_2 \) are \( e_1, e_4, e_3 \).

**LOOP, LINK AND PARALLEL EDGES:**

Let G be a Graph. An edge which joins the same vertex is called a Loop, an edge that joins two distinct vertices is called a Link. If two or more edges of G have the same end vertices, then these edges are called Parallel Edges (or) Multiple Edges.

**SIMPLE GRAPH:**

A Graph is said to be a Simple graph, if it has no Loops and no Parallel Edges.

**PLANAR GRAPH:**

A Graph G is said to be Planar graph, if it can be drawn on a plane or on a paper such that no two of its edges intersect anywhere in the graph, other than at their end vertices, otherwise it is called a Non-planar graph.

**FINITE, INFINITE GRAPH:**

A Graph is said to be Finite, if both its vertex set and edge sets are finite, otherwise it is called an Infinite graph.
TRIVIAL, NON-TRIVIAL GRAPHS AND ISOLATED VERTEX:

A Graph is said to be Trivial or Empty graph, if it has only vertices and no edges, otherwise it is called a Non-trivial graph. A vertex in a graph is said to be Isolated, if it has no incident edge.

DEGREE OF A VERTEX:

The number of edges incident with a vertex $v$ is called the degree of $v$ and is denoted by $d(v)$.

Eg:

![Graph Diagram]

Here $d(v_1) = 2$

PENDENT VERTEX:

A vertex of degree one is called a pendent vertex.

SUB-GRAph:

A Graph $H$ is said to be a Subgraph of $G$, if $V(H) \subseteq V(G)$, $E(H) \subseteq E(G)$ and $\Psi_H$ is the restriction of $\Psi_G$ to $E(H)$.

SPANNING SUB-GRAph:

A Spanning subgraph of $G$ is a subgraph $H$ of $G$ with $V(H) = V(G)$. i.e., $H$ and $G$ have the same vertex set.

VERTEX INDUCED SUB-GRAph:

Let $G (V,E)$ be a graph and $V_1$ be a subset of $V$. Then the subgraph of $G$ whose vertex set is $V_1$ and edge set is the set of whose edges in $E$ whose both ends are in $V_1$ is called the vertex induced subgraph and is denoted by $<V_1>$.
WALK:
A Walk in a Graph G is a finite sequence \( W = v_0e_1v_1e_2v_2e_3\ldots\ldots v_{k-1}e_kv_k \), whose terms are alternatively vertices and edges such that the edge \( e_i \) has ends \( v_{i-1} & v_i \). We say that \( W \) is a Walk from \( v_0 \) to \( v_k \) or \( (v_0,v_k) \)-Walk. The Vertex \( v_0 \) is called the Origin of the Walk, while \( v_k \) is called the Terminus of the Walk \( W \). The vertices \( v_1,v_2,v_3,\ldots,v_{k-1} \) in the Walk are called its Internal vertices, the integer \( k \) is the number of edges in the Walk is called the length of the Walk \( W \).

TRAIL:
If the edges \( e_1,e_2,e_3,\ldots,e_k \) of the Walk \( W = v_0e_1v_1e_2v_2e_3\ldots\ldots v_{k-1}e_kv_k \) are distinct, then the Walk \( W \) is called a Trail.

PATH:
A Trail in which the vertices are also distinct is called a Path.

DIRECTED PATH:
A Directed Path in a Directed Graph G is a non-null alternating sequence of distinct vertices and edges such that each edge is oriented from the vertex preceding it to the vertex following it.

CYCLE:
A Walk \( W = v_0e_1v_1e_2v_2e_3\ldots\ldots v_{k-1}e_kv_0 \) with Origin and Terminus are the same is called a Cycle, if its Internal Vertices and edges are distinct. (OR) A non-trivial closed path of G is called a cycle of G.

CONNECTED GRAPH:
A Graph is said to be Connected, if there is a Path between every pair of vertices, otherwise it is said to be disconnected graph.

TREE:
A connected acyclic graph is a tree.
DOMINATING SET:
A Subset $D$ of $V$ is said to be a Dominating Set of $G$, if every vertex in $V \setminus D$ is adjacent to a vertex in $D$. A dominating set with minimum cardinality is said to be a minimum dominating set. The domination number $\gamma(G)$ of a graph $G$ is the minimum cardinality of a dominating set in $G$.

Eg:

![Graph G = (V,E)]

Dominating Sets are

$D = \{2, 5\}, D = \{1, 5\}, D = \{4, 5\}, D = \{3, 4, 5\}$

Domination Number $\gamma(G) = 2$

MINIMAL DOMINATING SET:
A dominating set $D$ is called a Minimal dominating set, if no proper subset of $D$ is a dominating set.

Eg:

![Graph G = (V,E)]

In the graph $G$, the sets $\{2, 5\}, \{1, 4, 5\}, \{1, 4, 6, 7\}$ are minimal dominating sets
CONNECTED DOMINATING SET:

A Dominating Set $D$ of $G$ is connected dominating set, if the induced subgraph $<V-D>$ is connected. The Connected domination number $\gamma_c(G)$ of a graph $G$ is the minimum cardinality of a connected dominating set.

Eg:

\[
\begin{align*}
\text{Connected Dominating Set} &= \{2, 5\} \\
\text{Connected Domination Number} &= 2
\end{align*}
\]

NON-SPLIT DOMINATING SET:

A dominating set $D$ of a graph $G(V,E)$ is a Non-split dominating set, if the induced subgraph $<V-D>$ is connected.

Eg:

\[
\begin{align*}
\text{Non-Split Dominating Set} D &= \{2, 6\} \\
\text{Induced Subgraph} <V-D> &\text{ Connected}
\end{align*}
\]
NON-SPLIT DOMINATION NUMBER:

The Non-split domination number $\gamma_{ns}(G)$ of a graph $G$ is the minimum cardinality of a non-split dominating set.

Eg:

SPLIT DOMINATING SET:

A dominating set $D$ of $G$ is called a Split dominating set, if the vertex induced subgraph $<V-D>$ is disconnected.

Eg:
SPLIT DOMINATION NUMBER:

The Split domination number \( \gamma_s(G) \) of a graph \( G \) is the minimum cardinality of a split dominating set.

Eg:

\[ \text{Split Domination Number} = 2 \]

TOTAL DOMINATING SET:

A Total dominating set \( D \) is dominating set in which every vertex is adjacent to some vertex in it. The total domination number \( \gamma_t(G) \) of a graph \( G \) is the minimum cardinality of a total dominating set.

CONNECTED TOTAL DOMINATING SET:

A total dominating set \( D \) of a graph \( G \) is a Connected total dominating set, if the induced subgraph \(<D>\) is connected. The connected total domination number is the minimum cardinality of a connected total dominating set.

DIRECTED GRAPH (OR) DIGRAPH:

A Directed Graph or Digraph is a graph, each of whose edges has a direction.

DIRECTED CYCLE:

A directed cycle is a directed path whose origin and terminus is the same vertex.
**IN-NEIGHBOUR AND OUT-NEIGHBOUR:**

If \( V \) is a vertex set of a graph \( G \) and for any \( v \in V \) and \((u,v),(v,w)\in E\), \( u \) and \( w \) are called an In-neighbour and Out-neighbour of \( v \), denoted by \( N^-(v) \) and \( N^+(v) \) respectively.

**IN-DEGREE AND OUT-DEGREE:**

The In-degree and the Out-degree of \( v \) are the number of its in-neighbours and out-neighbours denoted by \( d^-(v) \) and \( d^+(v) \) respectively. The degree of \( v \) is \( d(v) = d^-(v) + d^+(v) \).

**BONDAGE NUMBER:**

The Bondage number \( b(G) \) of a non-empty graph \( G \) is the minimum cardinality among all sets of edges \( E_1 \) for which \( \gamma(G-E_1) > \gamma(G) \).

**Eg:**

![Interval Graph G](image)

Interval Graph G  
Dominating Set of G = \{4, 8\}  
Remove the edge \( e = (3, 4) \) from G

![Dominating Set of G-e](image)

Dominating Set of G-e = \{3, 4, 8\}
TOTAL BONDAGE NUMBER:

The Bondage number \( b_t(G) \) of a non-empty graph \( G \) is the minimum cardinality among all sets of edges \( E_1 \) for which \( \gamma_t(G-E_1) > \gamma_t(G) \).

STRONGLY DOMINATED VERTEX:

Let \( G = (V, E) \) be a graph and \( u, v \in V \). Then \( u \) strongly dominates \( v \) if (i) \( uv \in E \) (ii) \( \deg(u) \geq \deg(v) \).

STRONG DOMINATING SET:

A subset \( D_s \) of \( V \) is a strong dominating set of \( G \) if every vertex in \( V - D_s \) is strongly dominated by at least one vertex in \( D_s \).

STRONG DOMINATION NUMBER:

The strong domination number \( \gamma_{st}(G) \) of \( G \) is the minimum cardinality of a strong dominating set.

SPLIT STRONG DOMINATING SET:

A strong dominating set \( D_s \) of \( G \) is a split strong dominating set if the vertex induced subgraph \(< V - D_s >\) is disconnected.

\[
\begin{array}{cccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
\end{array}
\]

Graph G

Strong dominating set \( D_s = \{3, 7\} \)

Split strong dominating set = \{3, 4, 7\}
SPLIT STRONG DOMINATION NUMBER:

The split strong domination number $\gamma_{ss}(G)$ of $G$ is the minimum cardinality of a split strong dominating set.

COMPLEMENT OF A GRAPH:

The complement of a graph $G$ is the graph $\overline{G}$ with the same vertices as $G$. An edge exists in $\overline{G}$ if and only if it does not exist in $G$, in other words, two vertices adjacent in $\overline{G}$ if and only if they are not adjacent in $G$.

COMPLEMENTARY TREE DOMINATING SET:

A dominating set $S \subseteq V$ of a graph $G$ with vertex set $V(G)$ and edge set $E(G)$ is a complementary tree dominating set if the induced subgraph $\langle V - S \rangle$ is a tree.

COMPLEMENTARY TREE DOMINATION NUMBER:

Complementary tree domination number $\gamma_{ctd}(G)$ is the minimum cardinality of a complementary tree dominating set of $G$.

INTERVAL GRAPH:

Let $I = \{I_1, I_2, I_3, I_4, \ldots, I_n\}$ be any interval family, where each $I_i$ is an interval on the real line and $I_i = [a_i, b_i]$ for $i = 1, 2, 3, 4, \ldots, n$. Here $a_i$ is called the left end point labeling and $b_i$ is the right end point labeling of $I_i$. Without loss of generality we assume that all the end points of the intervals in $I$ are distinct numbers between 1 and $2n$. Two intervals $i$ and $j$ are said to be intersect each other, if they have non-empty intersection. Two intervals are said to overlap, if they have non-empty intersection and neither one of them contains the other.

In general an undirected graph $G = (V, E)$ is an interval graph (IG), if the vertex set $V$ can be put into one-to-one correspondence with a set of intervals $I$ on the real line $R$, such that two vertices are adjacent in $G$, if and only if their corresponding intervals
have non-empty intersection. That is if \( i = [a_i, b_i] \) and \( j = [a_j, b_j] \), then \( i \) and \( j \) intersect means either \( a_j < b_i \) or \( a_i < b_j \).

\[ \begin{align*}
  a_i & \rightarrow b_i \\
  a_j & \rightarrow b_j
\end{align*} \]

\text{or} \quad
\[ \begin{align*}
  a_j & \rightarrow b_j \\
  a_i & \rightarrow b_i
\end{align*} \]

\text{CIRCULAR - ARC GRAPH:}

Let \( A = \{A_1, A_2, \ldots, A_n\} \) be a Circular-arc family on a Circle, where each \( A_i \) is an arc. Without loss of generality assume that the end points of all arcs are distinct and no arc covers the entire Circle. Denote an arc \( i \) that begins at \( p \) and ends at point \( q \) in the clockwise direction by \((p, q)\). Define \( p \) to be the head and \( q \) to be the tail of the arc \( i \) and now \( i \) is denoted by \( i = (p, q) \).

Two arcs \( j \) and \( i \) are said to intersect each other if they have non-empty intersection. The continuous part of the circle that begins with an end point \( c \) and ends with \( d \) in the clockwise direction is referred to as segment \((c, d)\) of the circle. We use “arc” to refer to a member of \( A \) and “segment” to refer to a part of the circle between two endpoints. A point on the circle is said to be in arc \((p, q)\) if it is contained in segment \((p, q)\). An arc \((p, q)\) of \( A \) is also referred as the segment \((p, q)\). An arc \( i = (a, b) \) is said to be contained in another arc \( j = (c, d) \) if segment \((a, b)\) is contained in the segment \((c, d)\). An arc family \( A \) is said to be proper if no arc in \( A \) is contained in another arc.

Let \( G(V, E) \) be a graph. Let \( A = \{A_1, A_2, \ldots, A_n\} \) be a family of arcs on a Circle. Then \( G \) is called a Circular-arc graph, if there is a one-to-one correspondence between \( V \) and \( A \) such that two vertices in \( V \) are adjacent if and only if their corresponding arcs in \( A \) intersect. We denote this circular-arc graph by \( G[A] \). If circular-arc family is proper then the corresponding graph is called a proper circular-arc graph.

Usually we deal with intervals/arcs instead of vertices. Further if there are \( n \) intervals/arcs in the existing interval/arc family, then we denote its corresponding vertex set by \( \{1, 2, \ldots, n\} \). So alternatively, depending on the convenience, we use intervals/arcs as vertices and vice versa.
1.2 A BRIEF SURVEY OF ALGORITHMS:

In this thesis we find various types of dominating sets using different types of Algorithms. Now we discuss in detail about algorithms and their analysis. What is an algorithm? An algorithm is a step by step specification on how to perform a certain task. The steps in the algorithm must be simple, unambiguous and be followed in a prescribed order. Further, we will insist that algorithm to be effective. That is, it must always solve the problem in a finite number of steps.

After writing the algorithm, the next crucial task is to analyze the algorithm. Analyzing algorithm is an intellectual activity. It gives people (experts) a chance to exhibit their skills by devising new ways of doing the same task even faster. This tendency has a large payoff in computing time. We need to make explicit our assumptions about the kind of computer we expect the algorithm to be executed on. The assumptions we make can have important consequences with respect to how fast a problem can be solved. This has given scope to consider faster computers and the need for faster computers has increased in recent years. As a consequence there has been considerable interest in demising parallel algorithms for solving various computational problems. We will discuss them in detail later. After deciding the kind of computers on which the algorithm can be executed, the next task is to determine which operations are employed and what are their relative costs. Once we do this, we next determine the sufficient number of data sets which cause the algorithm to exhibit all possible patterns of behavior.

After having this type of physical analysis of the algorithm the next type of analysis we execute is the correctness of the algorithm. This analysis can be done by tracing the algorithm, reading the algorithm for logical correctness, implementing the algorithm, and testing it on some data, or using mathematical techniques to prove it correct. Another type of analysis is of the simplicity of the algorithm. Perhaps the algorithm can be expressed in a simpler way so that it is easier to implement and perform other analyses on the algorithm. However, the simplest and most straightforward way of solving a problem is sometimes not the best one. Usually this occurs when the simplest approach involves the use of too much computer time or
space. Thus it is important to be able to analyze the time and space requirements of an algorithm to see if it is within acceptable limits. Time and space analyses are also important for comparisons of algorithms to determine the best one.

The next task of the analysis of the algorithm is to determine the exact amount of time required to execute it. Generally it is not possible to perform a simple analysis of an algorithm to determine the exact amount of time required to execute it. The first complication is that the exact amount of time will depend on the implementation of the algorithm and actual machine. But one likes to have this analysis which is not dependent on the language or machine that might be used to implement it. Such types of algorithms are useful in general.

Yet another complication arises in doing a timing analysis in that the time requirements will normally depend on the amount of input. For example, an algorithm that sums the values in a vector can be expected to require more time for a vector of size 100,000 values than for one of size 100. As a result the estimate for the time required by an algorithm is usually expressed as a function of the size of the input. Thus if the input has size n, and when there are n values in the vector, then the time is expressed as \( (n) \) and space as \( S(n) \), where T and S are functions of amount of data.

We now discuss the rates of growth for some functions that are often used to express T and S. The functions that involve n as an exponent, for example, \( 2^n \), \( n^n \) and \( n! \), are called exponential functions. Any algorithm whose execution time grows proportionally to exponential function is too slow and will be practical for small input sizes. Functions whose growth is less than or equal to \( n^c \) for some constant c, for example \( n^3 \), \( n^2 \), \( n \log_2 n \), \( n\) and \( \log_2 n \) are said to be polynomial. Algorithm with polynomial time can solve reasonable-sized problems if the constant in the exponent is small. Algorithms whose execution time is greater than polynomial time are said to be non-polynomial or NP-Complete. Algorithms whose execution time proportional to \( n \) are said to be linear. For problems involving large amounts of data, it is necessary to find an algorithm whose execution time grows linearly, proportional to \( n \), or sublinearly, for instance, proportional to \( \log_2 n \).