VARIANTS OF ROMAN AND ITALIAN DOMINATION IN WEIGHTED GRAPHS

A Project Report Submitted

in Partial Fulfilment of the Requirements

for the Degree of

BACHELOR OF TECHNOLOGY

in

Mathematics and Computing

by

Krishna Priyatam D

(Roll No. 170123028)

Sujana Maithili Chindam

(Roll No. 170123016)

to the

DEPARTMENT OF MATHEMATICS

INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI

GUWAHATI - 781039, INDIA

April 2021

CERTIFICATE

This is to certify that the work contained in this report entitled **"Roman and Italian Domination in Weighted Trees"** submitted by **Krishna Priyatam D** (**Roll No: 170123028**) and **Sujana Maithili Chindam** (**Roll No: 170123016**) to the Department of Mathematics, Indian Institute of Technology Guwahati towards the requirement of the course **MA498: Project II** has been carried out by them under my supervision.

It is also certified that, along with the literature survey, a few new results are established by the students under the project.

Turnitin percentage : 23%

Guwahati - 781 039 (Dr. Gautam K. Das)

April 2021 Project Supervisor

ABSTRACT

A Roman dominating function on an undirected graph $G = (V, E)$ is a function $f : V(G) \to \{0, 1, 2\}$ satisfying the condition that every vertex $u \in V$ for which $f(u) = 0$ is adjacent to at least one vertex $v \in V$ for which $f(v) = 2$. The weight of a Roman dominating function f is the value $f(V(G)) = \Sigma_{u \in V(G)} f(u)$. The minimum weight of a Roman dominating function on a graph G is called the Roman domination number $\gamma_R(G)$ of G.

A Perfect Roman dominating function on a graph G is a Roman dominating function in which every vertex with $f(u) = 0$ is adjacent to exactly one vertex with $f(v) = 2$.

A Total Roman dominating function on a graph G is a Roman dominating function in which every vertex u with $f(u) > 0$ is adjacent to another vertex v with $f(v) > 0$.

An Independent Roman dominating function on a graph G is a Roman dominating function in which the set $V_1 \cup V_2$ is an independent set, where $V_1 = \{v \in G : f(v) = 1\}$ and $V_2 = \{v \in G : f(v) = 2\}$. An Italian dominating function on an undirected graph $G = (V, E)$ is a function $f : V(G) \to \{0, 1, 2\}$ satisfying the condition that every vertex $u \in V$ for which $f(u) = 0$, it holds that $\Sigma_{v \in N(u)} f(v) \geq 2$ where $N(u) = \{v \in V | (u, v) \in E\}$. The weight of an Italian dominating function f is the value $f(V(G)) = \Sigma_{u \in V(G)} f(u)$. The minimum weight of an Italian dominating function on a graph G is called the Italian domination number $\gamma_I(G)$ of G.

A Perfect Italian dominating function on a graph G is an Italian dominating function in which for every vertex $u \in V$ with $f(u) = 0$, it holds that $\Sigma_{v \in N(u)} f(v) = 2$ where $N(u) = \{v \in V | (u, v) \in E\}$.

In our work, we introduce the concept of Roman and Italian Domination in weighted graphs, and give a linear time algorithm for finding the Roman, Perfect Roman, Total Roman domination number and Italian, Perfect Italian domination number for weighted trees. We also give a linear time algorithm for finding the Independent and Total Roman domination number, and the Perfect Italian domination number in block graphs.

Contents

Chapter 1

Introduction

The concept of domination in graph theory is very important and significant area, and there has been much research in this area since 1950s. The problem of finding the dominating sets is of practical interest in several areas, such as wireless networking, facility location problems, and modeling social networks.

A dominating set of a graph $G = (V, E)$ is a subset D of V such that every vertex not in D is adjacent to at least one vertex in D. The domination problem is to find the minimum size dominating set of a given graph. Suppose, that every vertex $v \in V$ has a cost $c(v)$ and every edge $e \in E$ has a cost $c(e)$. The weighted domination problem is to find a dominating set D such that its total cost $c(D) = \Sigma\{c(v) : \ v \in D\} + \Sigma\{c(u, v) : u \notin D, \ v \in D \ and \ (u, v) \in E\}$ is minimum.

Based on various constraints, conditions and requirements, many variants of the domination problem are present, i.e. independent domination, connected domination, total domination, dominating cycle, k-domination, edge domination, k-neighbor domination, etc. One such variant is called

the perfect domination. A perfect dominating set of an undirected graph $G = (V, E)$ is a subset $S \subseteq V$, such that every vertex $v \in V - S$ is adjacent to exactly one vertex in S.

Various studies had been made for finding polynomial time algorithms for different variants of domination problem for different classes of graphs. It is shown in [1] that the problems of independent perfect domination and total perfect domination are NP-complete for bipartite graphs and chordal graphs.A linear time algorithm for the domination number of a tree is given in [2], and a linear time algorithm for the domination number of a cactus is given in [3]. A linear time algorithm for the edge domination problem in a block graph is given in [4]. Linear time algorithms for the perfect domination, independent perfect domination, connected perfect domination and total perfect domination problems in weighted block graphs are given in [1]. Further, many other studies regarding the domination problem and its variants have been studied in [5–7].

One particular variant of the domination problem is the Roman domination problem which was initially studied in depth in [8, 9]. A Roman dominating function on an undirected graph $G = (V, E)$ is a function $f : V(G) \rightarrow \{0, 1, 2\}$ satisfying the condition that every vertex $u \in V$ for which $f(u) = 0$ is adjacent to at least one vertex $v \in V$ for which $f(v) = 2$. The weight of a Roman dominating function f is the value $f(V(G)) = \Sigma_{u \in V(G)} f(u)$. The minimum weight of a Roman dominating function on a graph G is called the Roman domination number $\gamma_R(G)$ of G. Algorithms for finding Roman domination number for cographs, AT-free graphs and graphs with a d-octopus are presented in [10]. It is given in [8] that the decision problem correspond-

ing to Roman Domination is NP-complete, even when restricted to bipartite, chordal, planar or split graphs.

A perfect Roman dominating function on an undirected graph $G = (V, E)$ is a Roman dominating function in which every vertex $u \in V$ with $f(u) = 0$ is adjacent to exactly one vertex with $f(v) = 2$. The minimum weight of a perfect Roman dominating function on a graph G is called the perfect Roman domination number $\gamma_R^P(G)$ of G. Decision problem associated with perfect Roman Domination number is NP-complete for bipartite graphs given in [11]. Bounds for perfect Roman Domination number in trees is given in [12].

An Independent Roman Dominating Function(IRDF) on any undirected graph $G = (V, E)$ is a Roman dominating function $f : V(G) \to \{0, 1, 2\}$ in which every vertex $u \in V$ for which $f(u) = 0$ is adjacent to at least one vertex $v \in V$ for which $f(v) = 2$, such that the set formed by the union of the vertices that are assigned the values 1 and 2, is an independent set. That is, let $V_i = \{v \in V : f(v) = i\}$, $i = 0, 1, 2$. Then, we call the function f as an Independent roman dominating function if and only if $V_1 \cup V_2$ is an independent set.

A total Roman Dominating Function(TRDF) on an undirected graph $G = (V, E)$ is a function $f : V(G) \to \{0, 1, 2\}$ satisfying the following conditions: (i) every vertex $u \in V$ for which $f(u) = 0$ is adjacent to at least one vertex $v \in V$ for which $f(v) = 2$ and (ii) the subgraph of G induced by the set of all vertices of positive assignment has no isolated vertices. The weight

of a total Roman dominating function f is the value $f(V(G)) = \Sigma_{u \in V(G)} f(u)$. The minimum weight of a total Roman dominating function on a graph G is called the total Roman domination number $\gamma_{tR}(G)$ of G.

Another variant of Roman Domination called Double Roman Domination was introduced in [13] and a linear time algorithm for finding the Double Roman Domination number in trees was proposed in [14].

The Italian domination problem is a generalization of the Roman domination problem and was first introduced in [15] where it was called as the Roman $\{2\}$-domination. An Italian dominating function on an undirected graph $G = (V, E)$ is a function $f : V(G) \to \{0, 1, 2\}$ satisfying the condition that every vertex $u \in V$ for which $f(u) = 0$ it holds that $\Sigma_{v \in N(u)} f(v) \geq 2$ where $N(u) = \{v \in V | (u, v) \in E\}$. The weight of an Italian dominating function f is the value $f(V(G)) = \Sigma_{u \in V(G)} f(u)$. The minimum weight of an Italian dominating function on a graph G is called the Italian domination number $\gamma_I(G)$ of G. Bounds for Italian domination number in trees is given in [16]. It is shown in [17] that the decision problem for Italian domination is NP-complete even when restricted to planar graphs, and also a linear algorithm for computing the Italian domination of a unicyclic graph is given in [17].

A perfect Italian dominating function on an undirected graph $G = (V, E)$ is an Italian dominating function in which for every vertex $u \in V$ with $f(u) = 0$ it holds that $\Sigma_{v \in N(u)} f(v) = 2$ where $N(u) = \{v \in V | (u, v) \in E\}$.

The minimum weight of a perfect Italian dominating function on a graph G is called the perfect Italian domination number $\gamma_I^P(G)$ of G. Perfect Italian domination number in trees has been studied in [18].

We extend the above definition of Roman(Italian) domination for weighted graphs. Let $G = (V, E)$ be a weighted graph with the weight function $w : V(G) \to \mathbb{R}$. In this case, we define the weight of a Roman(Italian) dominating function f to be $f(V(G)) = \Sigma_{u \in V(G)} w(u) \times f(u)$, where $f : V(G) \to \{0, 1, 2\}$ is the Roman(Italian) dominating function on the weighted graph G. The minimum weight of a Roman(Italian) dominating function on a weighted graph G is called the Roman(Italian) domination number $\gamma_{RW}(G)(\gamma_{IW}(G))$ of the weighted graph G.

In our report, we have written algorithms specific to two classes of graphs, Trees and Block Graphs. The structure of a block graph is intuitively similar to that of a tree. This can be seen from the fact that every block graph has a corresponding block cut-point graph, which is a tree. There has been significant work done in the area of domination for block graphs also. The algorithm for Independent Italian Dominaton in block graphs is given in [19], and the algorithm for Perfect Roman Domination in block graphs is given in [20].

Chapter 2

Roman Domination in Weighted Trees

2.1 Perfect Roman Domination in Weighted Trees

Definition 2.1.1. A Perfect Roman Dominating Function(PRDF) on an undirected graph $G = (V, E)$ is a Roman dominating function in which every vertex $u \in V$ with $f(u) = 0$ is adjacent to exactly one vertex with $f(v) = 2$. The minimum weight $\Sigma_{u \in V(G)} f(u)$ of a perfect Roman dominating function on a graph G is called the perfect Roman domination number denoted by $\gamma_R^P(G)$ of G.

Let $T = (V, E)$ be a weighted tree rooted at a vertex $r \in V$ with the weight function $w : V(T) \to \mathbb{R}$. In a tree, a parent of a vertex $v \in V$ is the

vertex $u \in V$ connected to v on the path to the root. Every vertex has a unique parent except the root which has no parent. A child of a vertex u is a vertex of which u is the parent. Let us denote the set of children of a vertex $u \in V$ by $C(u)$.

In this case, we define the weight of a perfect Roman dominating function f to be $f(V(T)) = \Sigma_{u \in V(T)} w(u) \times f(u)$, where $f : V(T) \rightarrow \{0, 1, 2\}$ is a perfect Roman dominating function on the weighted tree T. The minimum weight of a perfect Roman dominating function on a weighted tree T is called the Perfect Roman Domination Number(PRDN) of T and is denoted by $\gamma^P_{RW}(T)$.

We use the dynamic programming paradigm to solve the problem of finding the perfect Roman domination number of a weighted tree. Let u be a specific vertex in T, and let T' be the subtree rooted at u. Let $f : V(T') \rightarrow \{0, 1, 2\}$ be a PRDF for the subtree T'. For the vertex u, it is given that $f(u) \in \{0, 1, 2\}$. Hence it is useful to consider the following 4 subproblems:

- $dp^{0,1}_0[u] = \min\{\ f(V(T'))\ :\ f(u) = 0\ \ and\ \ f(v) \in \{0, 1\}$
  $for\ all\ \ v \in C(u)\}$
- $dp^2_0[u] = \min\{\ f(V(T'))\ :\ f(u) = 0,\ \ f(v) = 2\ for\ some\ v \in C(u),$
  $and\ f(x) \in \{0, 1\}\ , for\ all\ x \in C(u) \setminus \{v\}\}$
- $dp_1[u] = \min\{\ f(V(T'))\ :\ f(u) = 1\}$
- $dp_2[u] = \min\{\ f(V(T'))\ :\ f(u) = 2\}$

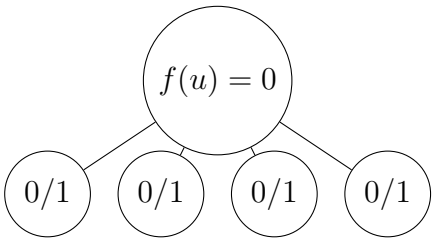

Figure 2.1: $dp_0^{0,1}[u]$

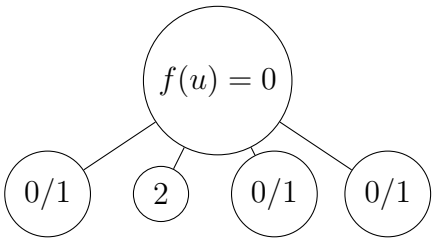

Figure 2.2: $dp_0^2[u]$

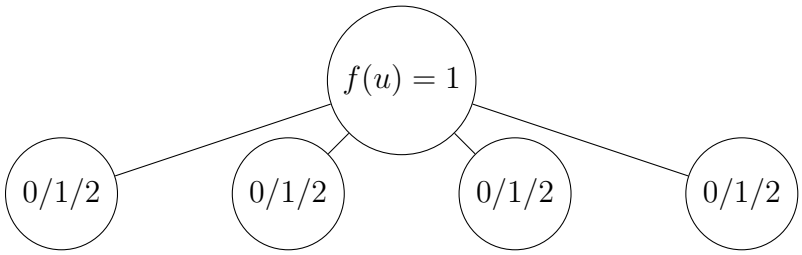

Figure 2.3: $dp_1[u]$

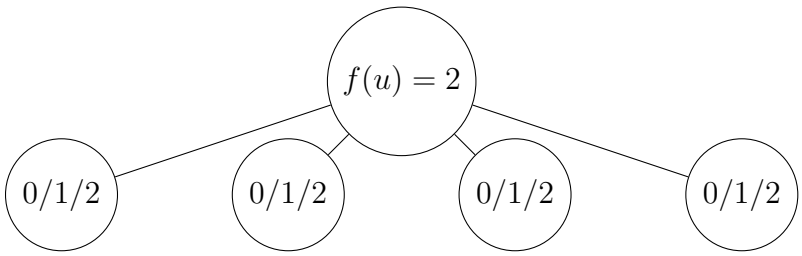

Figure 2.4: $dp_2[u]$

Here, $dp_1[u]$ is the perfect Roman domination number for the subtree T' with the condition that $f(u) = 1$. Similarly, $dp_2[u]$ is the perfect Roman domination number for the subtree T' with the condition that $f(u) = 2$. $dp_0^{0,1}[u]$ is the perfect Roman domination number for the subtree T' with the condition that $f(u) = 0$. $dp_1[u]$ is the weight of Roman domination Function for the subtree T' with the condition that $f(u) = 1$. In this case, we assume that for a vertex u with $f(u) = 0$ it has a parent t with $f(t) = 2$, i.e., the PRDF condition for the vertex u is satisfied by it's parent. $dp_0^2[u]$ is the perfect Roman domination number for the subtree T' with the condition that $f(u) = 0$, one child of u is assigned the value 2, and all the other children of u are assigned either 0 or 1 by the PRDF. In this case, we see that for a vertex u with $f(u) = 0$, it has exactly one child v such that $f(v) = 2$, i.e. the PRDF condition for vertex u is satisfied by one of it's children.

2.1.1 Algorithm

Theorem 2.1.2. *Let $T = (V, E)$ be a tree with a root $r \in V$. Let $u \in V$ be a non-leaf vertex of the tree. Then the following statements hold:*

- $dp_1[u] = 1 \times w(u) + \sum_{c \in C(u)} \min\{dp_0^2[c], dp_1[c], dp_2[c]\}$
- $dp_2[u] = 2 \times w(u) + \sum_{c \in C(u)} \min\{dp_0^{0,1}[c], dp_1[c], dp_2[c]\}$
- $dp_0^{0,1}[u] = \sum_{c \in C(u)} \min\{dp_1[c], dp_0^2[c]\}$
- $dp_0^2[u] = \min_{c \in C(u)} \{dp_2[c] + \sum_{c' \in C(u) \setminus \{c\}} \min\{dp_1[c'], dp_0^2[c']\}\}$

Proof. The proofs of the above statements are as follows:

- For a vertex u with $f(u) = 1$, there is no restriction for the values that's it's children can be assigned by the PRDF. Hence, for a subtree rooted at u, it's Perfect Roman Domination(PRD) number will be equal to the sum of the PRD numbers of the subtrees formed by each of it's children added to the contribution at vertex u, i.e. $w(u) \times 1$. Each of it's children can be assigned the value $\{0, 1, 2\}$ by the PRDF, hence we take the minimum of these 3 cases, for each subtree. Also, if one of the children of vertex u, let's say c, is assigned the value 0 by the PRDF, then for the vertex c to satisfy the PRD condition, one of the children of c should be assigned the value 2. Hence, we consider the case of $dp_0^2[c]$, while calculating the value of $dp_1[u]$.

- This case is similar to the previous case, except that for a vertex u with $f(u) = 2$, it's contribution to the PRD number is $w(u) \times 2$. Also, for a vertex u with $f(u) = 2$, if it has a child c such that $f(c) = 0$, then from the PRD condition, we can say that all the children of c will be assigned only either 0 or 1, by the PRDF. Hence, we consider the case of $dp_0^{0,1}$ while calculating the value of $dp_2[u]$.

- In this case, as the vertex u is assigned the value 0 by the PRDF, hence, for any child $c \in C(u)$, if $f(c) = 0$, then to satisfy the condition of PRD, one of the children c' of c must be assigned value 2 by the PRDF. Due to the constraint of $dp_0^{0,1}$, for each child, $c \in C(u)$, we consider only the cases $f(c) = 0$ or $f(c) = 1$.

- Similar to the previous case, in this case too, as $f(u) = 0$, if there is a child $c \in C(u)$, such that $f(c) = 0$, then to satisfy the condition

of PRD, one of the children c' of c must be assigned value 2 by the PRDF. Also, from the constraint of dp_0^2, exactly one of the children of u must be assigned 2 by the PRDF, to satisfy the PRD condition. Hence, to calculate the PRD number of the subtree rooted at u, we take the minimum of all the cases in which one child $c \in C(u)$ is assigned $f(c) = 2$, and the rest of them are assigned either 0 or 1 by the PRDF.

□

Algorithm 1 Perfect Roman Domination in weighted trees

Input A tree T fixed at a root r
Output Perfect Roman Domination number γ_{RW}^P of T

function PRD(T,u)
  for each $c \in T.C(u)$ **do** ▷ Depth First Search Traversal of Tree
    PRD(T,c)
  $dp_0^{0,1}[u] \leftarrow 0$
  $dp_0^2[u] \leftarrow \infty$
  $dp_1[u] \leftarrow 1 \times w(u)$
  $dp_2[u] \leftarrow 2 \times w(u)$
  if u is non-leaf **then** ▷ All nodes other than leaf
    for each $c \in T.C(u)$ **do**
      $dp_0^{0,1}[u] \leftarrow dp_0^{0,1}[u] + \min\{dp_1[c], dp_0^2[c]\}$
      $s \leftarrow dp_2[c]$
      for each $v \in T.C[u] - c$ **do**
        $s \leftarrow s + \min\{dp_0^2[v], dp_1[v]\}$
      $dp_0^2[u] \leftarrow \min\{dp_0^2[u], s\}$
      $dp_1[u] \leftarrow dp_1[u] + \min\{dp_1[c], dp_2[c], dp_0^2[c]\}$
      $dp_2[u] \leftarrow dp_2[u] + \min\{dp_1[c], dp_2[c], dp_0^{0,1}[c]\}$

function PRINT-PRD(T,r) ▷ Let r be the root of the tree T
  PRD(T,r)
  return $\min\{dp_0^2[r], dp_1[r], dp_2[r]\}$

We can make the above Algorithm 1 more efficient in terms of time complexity using the following lemma.

Lemma 2.1.3. *For a non-leaf vertex $u \in V$, we can write*

$$dp_0^2[u] = \alpha + dp_0^{0,1}[u]$$

where $\alpha = \min_{c \in C(u)} \{dp_2[c] - min\{dp_1[c], dp_0^2[c]\}\}$

Proof. From Theorem 2.1.2, we can write,

$$\begin{aligned}
dp_0^2[u] &= \min_{c \in C(u)} \{dp_2[c] + \sum_{c' \in C(u) \setminus c} \min\{dp_1[c'], dp_0^2[c']\}\} \\
&= \min_{c \in C(u)} \{dp_2[c] - \min\{dp_1[c], dp_0^2[c]\} + \sum_{c' \in C(u)} \min\{dp_1[c'], dp_0^2[c']\}\} \\
&= \min_{c \in C(u)} \{dp_2[c] - \min\{dp_1[c], dp_0^2[c]\} + dp_0^{0,1}[u]\} \\
&= \min_{c \in C(u)} \{dp_2[c] - \min\{dp_1[c], dp_0^2[c]\}\} + dp_0^{0,1}[u] \\
&= \alpha + dp_0^{0,1}[u] \\
&\text{where } \alpha = \min_{c \in C(u)} \{dp_2[c] - \min\{dp_1[c], dp_0^2[c]\}\}
\end{aligned}$$

□

Algorithm 2 Perfect Roman Domination in weighted trees

Input A tree T fixed at a root r

Output Perfect Roman Domination number γ_{RW}^P of T

1: **function** PRD(T,u)
2: $dp_0^{0,1}[u] \leftarrow 0$
3: $dp_0^2[u] \leftarrow \infty$
4: $dp_1[u] \leftarrow 1 \times w(u)$

```
5:     dp_2[u] ← 2 × w(u)
6:     α ← ∞
7:     for each c ∈ T.C(u) do                ▷ Depth First search Traversal of Tree
8:         PRD(T,c)
9:         dp_0^{0,1}[u] ← dp_0^{0,1}[u] + min{dp_1[c], dp_0^2[c]}
10:        α ← min{α, dp_2[c] − min{dp_1[c], dp_0^2[c]}}
11:        dp_1[u] ← dp_1[u] + min{dp_1[c], dp_2[c], dp_0^2[c]}
12:        dp_2[u] ← dp_2[u] + min{dp_1[c], dp_2[c], dp_0^{0,1}[c]}
13:    if u is non-leaf then                 ▷ All nodes other than leaf
14:        dp_0^2[u] ← dp_0^{0,1}[u] + α
15:
16: function PRINT-PRD(T,r)                  ▷ Let r be the root of the tree T
17:    PRD(T,r)
18:    return min{dp_0^2[r], dp_1[r], dp_2[r]}
```

2.1.2 Time and Space Complexity Analysis

Lemma 2.1.4. *For a tree T with n vertices, the time and space complexities of Algorithm 1 are $O(n^2)$ and $O(n)$ respectively.*

Proof. We know that a Depth First Search (DFS) on a tree has a time complexity of $O(n)$, where n is the number of vertices in the tree. Along with that, in our PRD function, we have another loop on line 12 which runs a maximum of $n-2$ times, i.e. it takes $O(n)$ time. Hence, Algorithm 1 runs in $O(n^2)$ time. The space complexity is $O(n)$ since we create 4 arrays, $dp_0^{0,1}, dp_0^2, dp_1, dp_2$ of size n each. □

Lemma 2.1.5. *For a tree T with n vertices, the time and space complexities of Algorithm 2 are $O(n)$ and $O(n)$ respectively.*

Proof. The time complexity of Algorithm 2 is $O(n)$ as we are using Depth First Search (DFS) method to traverse the tree which takes $O(n)$ time. We

visit each vertex exactly once in this algorithm, and as the number of vertices are n, the total time complexity is $O(n)$. The space complexity is $O(n)$ since we create 4 arrays, $dp_0^{0,1}, dp_0^2, dp_1, dp_2$ of size n each, and a variable, α. $\square$

2.1.3 Proof of Correctness

Lemma 2.1.6. *Algorithm 1 computes Perfect Roman Domination number PRD correctly*

Proof. We use mathematical induction to prove the correctness of algorithm. We show that Algorithm 1 computes the PRD number correctly for all trees of height $h \in \mathbb{Z}^+$.

- Base Case: For $h = 1$,
  The tree contains a single node. Let us denote it by u. Now,
  $dp_0^{0,1}[u] = 0$, $dp_0^2[u] = \infty$, $dp_1[u] = 1 \times w(u)$, $dp_2[u] = 2 \times w(u)$
  $\gamma_{RW}^P = \min\{dp_0^2[u], dp_1[u], dp_2[u]\} = \min\{\infty, 1 \times w(u), 2 \times w(u)\} = 1 \times w(u)$.

- Induction Hypothesis: Let the correctness of the algorithm hold for all trees of height $h'(< h)$.

- Induction Step: Let T be a tree of height h and let us fix a vertex r to be the root of T. Let us have a weight function $w : V(T) \to R$ for the vertices of the tree T. For all $c \in C(r)$, the height of the subtree rooted at c is h' which is less than h. So, we have optimal values of $dp_0^{0,1}[c], dp_0^2[c], dp_1[c], dp_2[c]$ for all $c \in C(r)$. Therefore, from Theorem 2.1.2, we have the optimal values of $dp_0^{0,1}[r], dp_0^2[r], dp_1[r], dp_2[r]$.
  Now, the PRD number of the weighted tree will be $\gamma_{RW}^P(T) = \min\{dp_0^2[r], dp_1[r], dp_2[r]\}$.

This is because, from the PRDF, the vertex r can be assigned the value either 0 or 1 or 2, and if $f(r) = 0$, it is necessary that for exactly one child $c \in C(r)$, $f(c) = 2$ to satisfy the PRD condition, as r does not have any parent vertex. Hence, we see that the induction hypothesis is true for all trees of height h.

Hence, we see that Algorithm 1 computes the PRD number correctly for all trees of height $h \in \mathbb{Z}^+$. □

2.2 Roman Domination in Weighted Trees

Definition 2.2.1. A Roman Dominating Function(RDF) on an undirected graph $G = (V, E)$ is a function $f : V(G) \to \{0, 1, 2\}$ satisfying the condition that every vertex $u \in V$ for which $f(u) = 0$ is adjacent to at least one vertex $v \in V$ for which $f(v) = 2$. The weight of a Roman dominating function f is the value $f(V(G)) = \Sigma_{u \in V(G)} f(u)$. The minimum weight of a Roman dominating function on a graph G is called the Roman domination number $\gamma_R(G)$ of G.

Let us consider the weighted tree $T = (V, E)$ with a weight function $w : V \to \mathbb{R}$ that we have considered in the previous section. In the case of a weighted tree, we define the weight of a Roman dominating function f on a tree $T = (V, E)$ to be $f(V(T)) = \Sigma_{u \in V(T)} w(u) \times f(u)$, where $f : V(T) \to \{0, 1, 2\}$ is a Roman dominating function on the weighted tree T. The minimum weight of a Roman dominating function on a weighted tree T is called the Roman Domination Number(RDN) of T and is denoted by $\gamma_{RW}(T)$.

We use the dynamic programming paradigm to solve the problem of finding the Roman domination number of a weighted tree. Let u be a specific vertex in T, and let T' be the subtree rooted at u. Let $f : V(T') \to \{0, 1, 2\}$ be a RDF for the subtree T'. For the vertex u, it is given that $f(u) \in \{0, 1, 2\}$. Hence it is useful to consider the following 4 subproblems:

- $dp_0^{0,1,2}[u] = \min\{\ f(V(T'))\ :\ f(u) = 0\ \ and\ \ f(v) \in \{0, 1, 2\}$
  $for\ all\ \ v \in C(u)\}$
- $dp_0^2[u] = \min\{\ f(V(T'))\ :\ f(u) = 0,\ \ f(v) = 2\ for\ some\ v \in C(u),$
  $and\ f(x) \in \{0, 1, 2\}\ , for\ all\ x \in C(u) \setminus \{v\}\}$
- $dp_1[u] = \min\{\ f(V(T'))\ :\ f(u) = 1\}$
- $dp_2[u] = \min\{\ f(V(T'))\ :\ f(u) = 2\}$

Here, $dp_1[u]$ is the Roman Domination Number(RDN) for the subtree T' with the condition that $f(u) = 1$. Similarly, $dp_2[u]$ is the RDN for the subtree T' with the condition that $f(u) = 2$. $dp_0^{0,1,2}[u]$ is the RDN for the subtree T' with the condition that $f(u) = 0$, and all the children of u are assigned any value from $\{0, 1, 2\}$ by the RDF. We use this case assuming that for a vertex $u \in V$, it has a parent $t \in V$ with $f(t) = 2$, and hence, the children of u can be assigned any value by the RDF, as the RDF condition for the vertex u is already satisfied by it's parent vertex t. $dp_0^2[u]$ is the RDN for the subtree T' with the condition that $f(u) = 0$, one child of u is assigned the value 2, and all the other children of u are assigned any value from $\{0, 1, 2\}$ by the RDF. In this case, we see that for a vertex u with $f(u) = 0$, it has at least one child v such that $f(v) = 2$, i.e. the RDF condition for vertex u is satisfied by at least one of it's children.

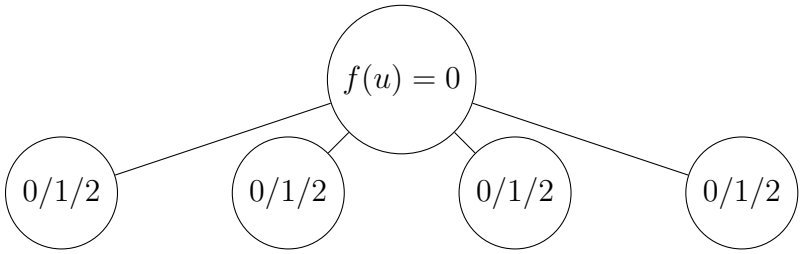

Figure 2.5: $dp_0^{0,1,2}[u]$

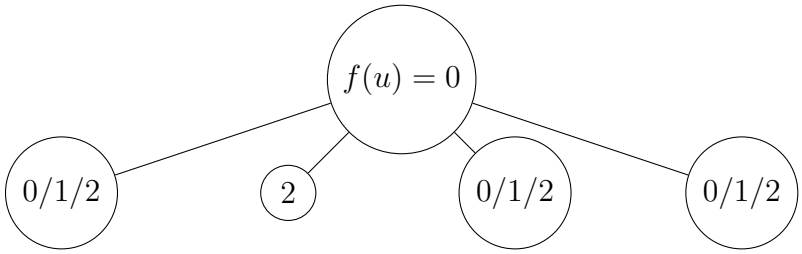

Figure 2.6: $dp_0^2[u]$

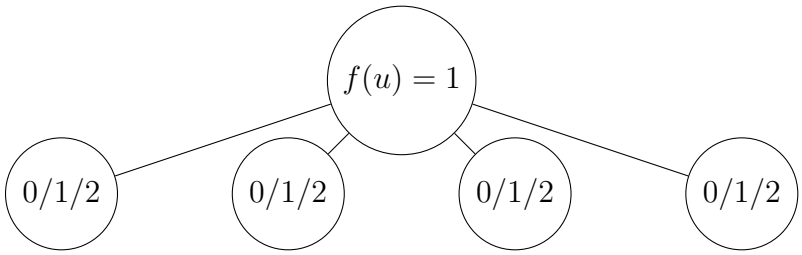

Figure 2.7: $dp_1[u]$

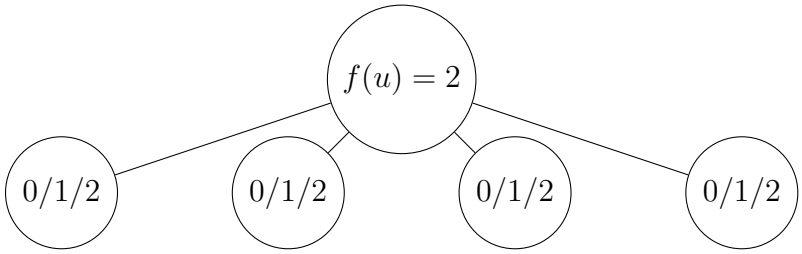

Figure 2.8: $dp_2[u]$

2.2.1 Algorithm

Theorem 2.2.2. *Let $T = (V, E)$ be a tree with a root $r \in V$. Let $u \in V$ be a non-leaf vertex of the tree. Then the following statements hold:*

- $dp_1[u] = 1 \times w(u) + \sum_{c \in C(u)} \min\{dp_0^2[c], dp_1[c], dp_2[c]\}$
- $dp_2[u] = 2 \times w(u) + \sum_{c \in C(u)} \min\{dp_0^{0,1,2}[c], dp_1[c], dp_2[c]\}$
- $dp_0^{0,1,2}[u] = \sum_{c \in C(u)} \min\{dp_2[c], dp_1[c], dp_0^2[c]\}$
- $dp_0^2[u] = \min_{c \in C(u)} \{dp_2[c] + \sum_{c' \in C(u) \setminus \{c\}} \min\{dp_2[c'], dp_1[c'], dp_0^2[c']\}\}$

Proof. The proofs of the above statements are as follows:

- For a vertex u with $f(u) = 1$, there is no restriction on the values of it's children that can be assigned by the RDF. Hence, for a subtree rooted at u, it's RDN will be equal to the sum of the RDNs of the subtrees formed by each of it's children added to the contribution at vertex u, i.e. $w(u) \times 1$. Each of it's children can be assigned the value $\{0, 1, 2\}$ by the RDF, hence we take the minimum of these 3 cases, for each subtree. Also, if one of the children of vertex u, let's say c, is assigned the value 0 by the RDF, then for the vertex c to satisfy the RDF condition, one of the children c' of c should be assigned the value 2. Hence, we consider the case of $dp_0^2[c]$, while calculating the value of $dp_1[u]$.

- This case is similar to the previous case, except that for a vertex u with $f(u) = 2$, it's contribution to the PRD number is $w(u) \times 2$. Also,

for a vertex u with $f(u) = 2$, if it has a child c such that $f(c) = 0$, then from the Roman domination condition, all the children of c can be assigned any value by the RDF. Hence, we consider the case of $dp_0^{0,1,2}$ while calculating the value of $dp_2[u]$.

- In this case, as the vertex u is assigned the value 0 by the RDF, hence, for any child $c \in C(u)$, if $f(c) = 0$, then to satisfy the condition of Roman domination, one of the children of c must be assigned 2 by the RDF. From the constraint of $dp_0^{0,1,2}$, for each child, $c \in C(u)$, we consider the minimum for the cases $f(c) = 0$, $f(c) = 1$ and $f(c) = 2$.

- Similar to the previous case, in this case too, as $f(u) = 0$, if there is a child $c \in C(u)$, such that $f(c) = 0$, then to satisfy the condition of Roman domination, one of the children of c must be assigned 2 by the RDF. Also, from the constraint of dp_0^2, at least one of the children of u must be assigned 2 by the RDF, to satisfy the Roman domination condition. Hence, to calculate the Roman domination number of the subtree rooted at u, we take the minimum of all the cases in which one child $c \in C(u)$ is assigned $f(c) = 2$, and the rest of them are assigned any value from $\{0, 1, 2\}$ by the RDF.

□

The outline of the algorithm is as follows. We first do a Depth First Search traversal on the given tree $T = (V, E)$, starting from the root $r \in V$. In the Depth First Search traversal, for a particular vertex, we first visit all the children of the vertex, and then come back to the vertex. Hence, when we reach a vertex v, then the entries corresponding to the children of v are

already filled in the 4 arrays, dp_1, dp_2, dp_0^2, and $dp_0^{0,1,2}$, and hence, from these, we can fill in the values of $dp_1[v]$, $dp_2[v]$, $dp_0^2[v]$, and $dp_0^{0,1,2}[v]$. After the traversal is complete, all the entries of the 4 arrays will be filled. Finally, as the root r can be assigned the values $\{0, 1, 2\}$ by the RDF, and if $f(r) = 0$, then at least one of the children of r should be assigned 2 to satisfy the RDF condition, we take the $\min\{dp_0^2[r], dp_1[r], dp_2[r]\}$ to find out the Roman domination number for the tree T.

We make some changes to the formula of $dp_0^2[u]$ for computational efficiency.

Lemma 2.2.3. *For a non-leaf vertex $u \in V$, we can write*

$$dp_0^2[u] = \alpha + dp_0^{0,1,2}[u]$$

where $\alpha = \min_{c \in C(u)} \{dp_2[c] - min\{dp_2[c], dp_1[c], dp_0^2[c]\}\}$

Proof. From Theorem 2.2.2, we can write,

$$\begin{aligned}
dp_0^2[u] &= \min_{c \in C(u)} \{dp_2[c] + \sum_{c' \in C(u) \setminus c} \min\{dp_2[c'], dp_1[c'], dp_0^2[c']\}\} \\
&= \min_{c \in C(u)} \{dp_2[c] - \min\{dp_2[c], dp_1[c], dp_0^2[c]\} + \sum_{c' \in C(u)} \min\{dp_2[c'], dp_1[c'], dp_0^2[c']\}\} \\
&= \min_{c \in C(u)} \{dp_2[c] - \min\{dp_2[c], dp_1[c], dp_0^2[c]\} + dp_0^{0,1,2}[u]\} \\
&= \min_{c \in C(u)} \{dp_2[c] - \min\{dp_2[c], dp_1[c], dp_0^2[c]\}\} + dp_0^{0,1,2}[u] \\
&= \alpha + dp_0^{0,1,2}[u] \\
&\text{where } \alpha = \min_{c \in C(u)} \{dp_2[c] - \min\{dp_2[c], dp_1[c], dp_0^2[c]\}\}
\end{aligned}$$

□

Algorithm 3 Roman Domination in weighted trees

Input A tree T fixed at a root r
Output Roman Domination number γ_{RW} of T

function RD(T,u)
  $dp_0^{0,1,2}[u] \leftarrow 0$
  $dp_0^2[u] \leftarrow \infty$
  $dp_1[u] \leftarrow 1 \times w(u)$
  $dp_2[u] \leftarrow 2 \times w(u)$
  $\alpha \leftarrow \infty$
  for each $c \in T.C(u)$ **do** ▷ Depth First Search traversal of tree
    RD(T,c)
    $dp_0^{0,1,2}[u] \leftarrow dp_0^{0,1,2}[u] + \min\{dp_2[c], dp_1[c], dp_0^2[c]\}$
    $\alpha \leftarrow \min\{\alpha, dp_2[c] - \min\{dp_2[c], dp_1[c], dp_0^2[c]\}\}$
    $dp_1[u] \leftarrow dp_1[u] + \min\{dp_1[c], dp_2[c], dp_0^2[c]\}$
    $dp_2[u] \leftarrow dp_2[u] + \min\{dp_1[c], dp_2[c], dp_0^{0,1,2}[c]\}$
  if u is non-leaf **then** ▷ All nodes other than leaf
    $dp_0^2[u] \leftarrow dp_0^{0,1,2}[u] + \alpha$

function PRINT-RD(T,r) ▷ Let r be the root of the tree T
  RD(T,r)
  return $\min\{dp_0^2[r], dp_1[r], dp_2[r]\}$

2.2.2 Time and Space Complexity Analysis

Lemma 2.2.4. *For a tree T with n vertices, the time and space complexities of Algorithm 3 are $O(n)$ and $O(n)$ respectively.*

Proof. The time complexity of Algorithm 3 is $O(n)$ as we are using Depth First Search method to traverse the tree which takes $O(n)$ time. We visit each vertex exactly once in the algorithm, and as the number of vertices are

n, the total time complexity is $O(n)$. The space complexity is $O(n)$ since we create 4 arrays, $dp_0^{0,1,2}, dp_0^2, dp_1, dp_2$ of size n each, and a single variable, α. □

2.2.3 Proof of Correctness

Lemma 2.2.5. *Algorithm 3 computes the Roman Domination number of a given tree $T = (V, E)$ correctly*

Proof. We use induction to prove the correctness of the algorithm. We show that Algorithm 3 computes the Roman domination number correctly for all trees of height $h \in \mathbb{Z}^+$.

- Base Case: For $h = 1$,
  The tree contains a single node. Let us denote it by u. Now,
  $dp_0^{0,1,2}[u] = 0$, $dp_0^2[u] = \infty$, $dp_1[u] = 1 \times w(u)$, $dp_2[u] = 2 \times w(u)$
  $\gamma_{RW} = \min\{dp_0^2[u], dp_1[u], dp_2[u]\} = \min\{\infty, 1 \times w(u), 2 \times w(u)\} = 1 \times w(u)$.

- Induction Hypothesis: Let the correctness of the algorithms hold on trees of height $h'(< h)$.

- Induction Step: Let T be a tree of height h and let us fix a vertex r to be the root of T. Let us have a weight function $w : V(T) \to R$ for the vertices of the tree T. For all $c \in C(r)$, the height of the subtree rooted at c, $h' < h$. So, we have optimal values of $dp_0^{0,1,2}[c], dp_0^2[c], dp_1[c], dp_2[c]$ for all $c \in C(r)$. Using these and the results from Theorem 2.2.2, we can find the optimal values of $dp_0^{0,1,2}[r], dp_0^2[r], dp_1[r], dp_2[r]$.
  Now, the Roman domination number of the weighted tree will be

$\gamma_{RW}(T) = \min\{dp_0^2[r], dp_1[r], dp_2[r]\}$. This is because, from the RDF, the vertex r can be assigned the value either 0 or 1 or 2, and if $f(r) = 0$, it is necessary that for at least one child $c \in C(r)$, $f(c) = 2$ to satisfy the Roman domination condition, as r doesn't have any parent vertex. Hence, the induction hypothesis is true for all trees of height h.

Hence, we see that Algorithm 3 computes the Roman domination number correctly for all trees of height $h \in \mathbb{Z}^+$. □

2.3 Total Roman Domination in Weighted Trees

Definition 2.3.1. A total Roman Dominating Function(TRDF) on an undirected graph $G = (V, E)$ is a function $f : V(G) \to \{0, 1, 2\}$ satisfying the following conditions: (i) every vertex $u \in V$ for which $f(u) = 0$ is adjacent to at least one vertex $v \in V$ for which $f(v) = 2$ and (ii) the subgraph of G induced by the set of all vertices of positive assignment has no isolated vertices. The weight of a total Roman dominating function f is the value $f(V(G)) = \Sigma_{u \in V(G)} f(u)$. The minimum weight of a Roman dominating function on a graph G is called the Roman domination number $\gamma_{tR}(G)$ of G.

Let us consider the weighted tree $T = (V, E)$ with a weight function $w : V \to \mathbb{R}$ that we have considered in the previous section. In the case of a weighted tree, we define the weight of a Roman dominating function f on a tree $T = (V, E)$ to be $f(V(T)) = \Sigma_{u \in V(T)} w(u) \times f(u)$, where $f : V(T) \to \{0, 1, 2\}$ is a total Roman dominating function on the weighted tree T. The minimum weight of a total Roman dominating function on a weighted tree T is called the total Roman Domination Number(TRDN) of T and is denoted

by $\gamma_{tRW}(T)$.

We use the dynamic programming paradigm to solve the problem of finding the total Roman domination number of a weighted tree. Let u be a specific vertex in T, and let T' be the subtree rooted at u. Let $f : V(T') \rightarrow \{0,1,2\}$ be a RDF for the subtree T'. For the vertex u, it is given that $f(u) \in \{0,1,2\}$. Hence it is useful to consider the following 6 subproblems:

- $dp_0[u] = \min\{\ f(V(T'))\ :\ f(u) = 0\ \ and\ \ f(v) \in \{0,1,2\}$
  $for\ all\ \ v \in C(u)\}$
- $dp_0^2[u] = \min\{\ f(V(T'))\ :\ f(u) = 0,\ \ f(v) = 2\ for\ some\ v \in C(u),$
  $and\ f(x) \in \{0,1,2\}\ , for\ all\ x \in C(u) \setminus \{v\}\}$
- $dp_1[u] = \min\{\ f(V(T'))\ :\ f(u) = 1\ \ and\ \ f(v) \in \{0,1,2\}$
  $for\ all\ \ v \in C(u)\}$
- $dp_1^{1,2}[u] = \min\{\ f(V(T'))\ :\ f(u) = 1,\ \ f(v) \in \{1,2\}\ for\ some\ v \in C(u),$
  $and\ f(x) \in \{0,1,2\}\ , for\ all\ x \in C(u) \setminus \{v\}\}$
- $dp_2[u] = \min\{\ f(V(T'))\ :\ f(u) = 2\ \ and\ \ f(v) \in \{0,1,2\}$
  $for\ all\ \ v \in C(u)\}$
- $dp_2^{1,2}[u] = \min\{\ f(V(T'))\ :\ f(u) = 2,\ \ f(v) \in \{1,2\}\ for\ some\ v \in C(u),$
  $and\ f(x) \in \{0,1,2\}\ , for\ all\ x \in C(u) \setminus \{v\}\}$

Here, $dp_0[u]$ is the total Roman Domination Number(TRDN) for the subtree T' with the condition that $f(u) = 0$, and all the children of u are assigned any value from $\{0,1,2\}$ by the TRDF . Similarly, $dp_1[u]$ and $dp_2[u]$ is

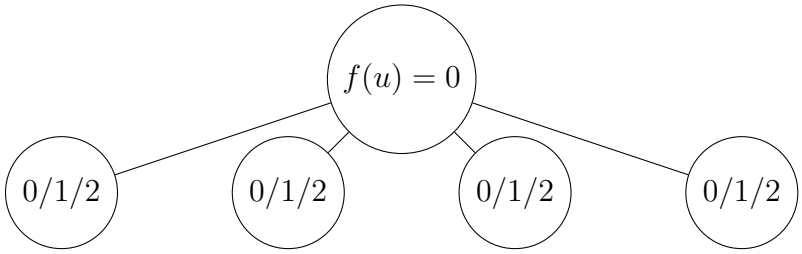

Figure 2.9: $dp_0[u]$

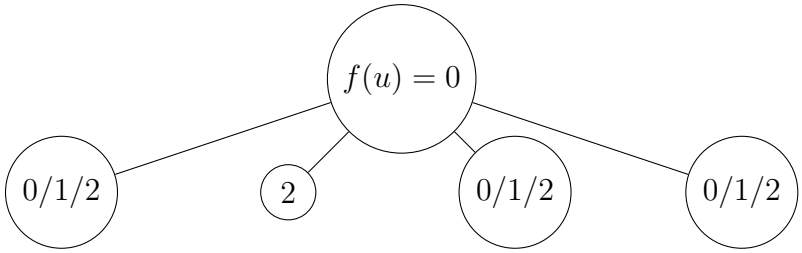

Figure 2.10: $dp_0^2[u]$

the TRDN for the subtree T' with the condition that $f(u) = 1$ and $f(u) = 2$ respectively, and all the children of u are assigned any value from $\{0, 1, 2\}$ by the TRDF. $dp_0^2[u]$ is the TRDN for the subtree T' with the condition that $f(u) = 0$, one child of u is assigned the value 2, and all the other children of u are assigned any value from $\{0, 1, 2\}$ by the TRDF. here, in $dp_0^2[u]$ the first condition of TRDF is satisfied by its children. $dp_1^{1,2}[u]$ is the TRDN for the subtree T' with the condition that $f(u) = 1$, one child of u is assigned the value 1 or 2, and all the other children of u are assigned any value from $\{0, 1, 2\}$ by the TRDF. Similarly, $dp_2^{1,2}[u]$ is the TRDN for the subtree T' with the condition that $f(u) = 2$, one child of u is assigned the value 1 or 2, and all the other children of u are assigned any value from $\{0, 1, 2\}$ by the TRDF. Here, in $dp_1^{1,2}[u]$ and $dp_2^{1,2}[u]$ the second condition of TRDF is satisfied by its children.

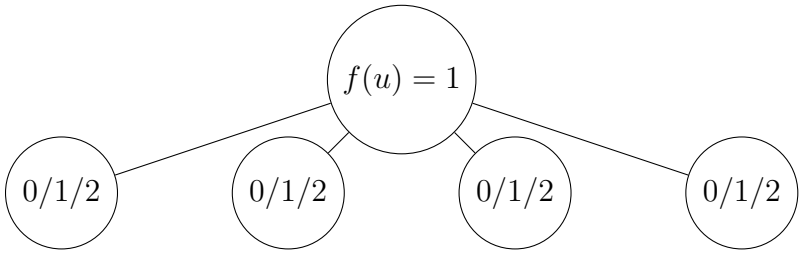

Figure 2.11: $dp_1[u]$

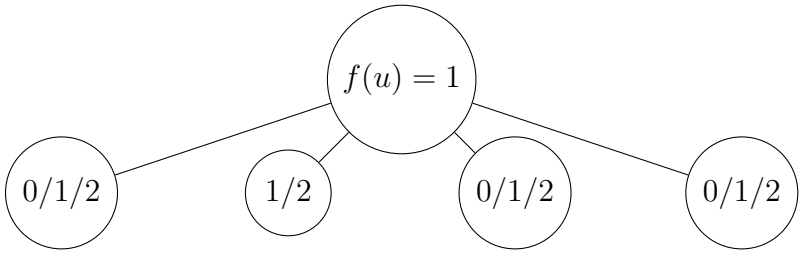

Figure 2.12: $dp_1^{1,2}[u]$

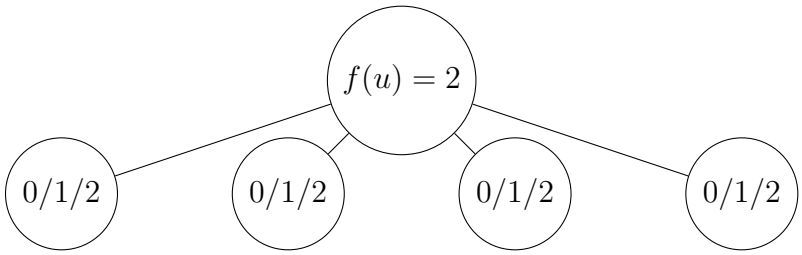

Figure 2.13: $dp_2[u]$

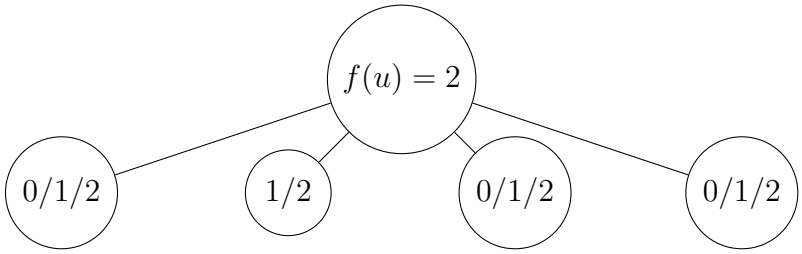

Figure 2.14: $dp_2^{1,2}[u]$

2.3.1 Algorithm

Theorem 2.3.2. *Let $T = (V, E)$ be a tree with a root $r \in V$. Let $u \in V$ be a non-leaf vertex of the tree. Then the following statements hold:*

- $dp_0[u] = \sum\limits_{c \in C(u)} \min\{dp_0^2[c], dp_1^{1,2}[c], dp_2^{1,2}[c]\}$
- $dp_0^2[u] = \min\limits_{c \in C(u)} \{dp_2^{1,2}[c] + \sum\limits_{c' \in C(u) \setminus \{c\}} \min\{dp_0^2[c'], dp_1^{1,2}[c'], dp_2^{1,2}[c']\}\}$
- $dp_1[u] = 1 \times w(u) + \sum\limits_{c \in C(u)} \min\{dp_0^2[c], dp_1[c], dp_2[c]\}$
- $dp_1^{1,2}[u] = 1 \times w(u) + \min\limits_{c \in C(u)} \{\min\{dp_1[c], dp_2[c]\} + \sum\limits_{c' \in C(u) \setminus \{c\}} \min\{dp_0^2[c'], dp_1[c'], dp_2[c']\}\}$
- $dp_2[u] = 2 \times w(u) + \sum\limits_{c \in C(u)} \min\{dp_0[c], dp_1[c], dp_2[c]\}$
- $dp_2^{1,2}[u] = 2 \times w(u) + \min\limits_{c \in C(u)} \{\min\{dp_1[c], dp_2[c]\} + \sum\limits_{c' \in C(u) \setminus \{c\}} \min\{dp_0[c'], dp_1[c'], dp_2[c']\}\}$

Proof. The proofs of the above statements are as follows:

- In this case, as the vertex u is assigned the value 0 by the TRDF, hence, for any child $c \in C(u)$, if $f(c) = 0$, then to satisfy the first condition of total Roman domination, one of the children of c must be assigned 2 by the TRDF else if $f(c) = 1$ or $f(c) = 2$, then to satisfy the second condition of total Roman domination, one of the children of c must be assigned 1 or 2 by the TRDF. From the constraint of dp_0, for each child, $c \in C(u)$, we consider the minimum for the cases $f(c) = 0$, $f(c) = 1$ and $f(c) = 2$.

- Similar to the previous case, in this case too, as $f(u) = 0$, if there is a child $c \in C(u)$, such that $f(c) = 0$, then to satisfy the first condition of

Roman domination, one of the children of c must be assigned 2 by the RDF. If $f(c) = 1$ or $f(c) = 2$, then to satisfy the second condition of total Roman domination, one of the children of c must be assigned 1 or 2 by the TRDF. Also, from the constraint of dp_0^2, at least one of the children say $c \in C(u)$ of u must be assigned 2 by the TRDF, to satisfy the total Roman domination first condition, one of the children of c must be assigned 1 or 2 by TRDF to satisfy total Roman domination second condition. Hence, to calculate the total Roman domination number of the subtree rooted at u, we take the minimum of all the cases in which one child $c \in C(u)$ is assigned $f(c) = 2$, and the rest of them are assigned any value from $\{0, 1, 2\}$ by the TRDF.

- In this case, as the vertex u is assigned the value 1 by the TRDF, hence, for any child $c \in C(u)$, if $f(c) = 0$, then to satisfy the first condition of total Roman domination, one of the children of c must be assigned 2 by the TRDF else if $f(c) = 1$ or $f(c) = 2$, as the second condition of total Roman domination is satisfied by its parent, children of c can be assigned any value by the TRDF. From the constraint of dp_1, for each child, $c \in C(u)$, we consider the minimum for the cases $f(c) = 0$, $f(c) = 1$ and $f(c) = 2$.

- Similar to the previous case, in this case too, as $f(u) = 1$, if there is a child $c \in C(u)$, such that $f(c) = 0$, then to satisfy the first condition of Roman domination, one of the children of c must be assigned 2 by the RDF. If $f(c) = 1$ or $f(c) = 2$, as the second condition of total Roman domination is satisfied by its parent, children of c can be assigned any

value by the TRDF. Also, from the constraint of $dp_1^{1,2}$, at least one of the children of u must be assigned 1 or 2 by the TRDF, to satisfy the total Roman domination second condition. Hence, to calculate the total Roman domination number of the subtree rooted at u, we take the minimum of all the cases in which one child $c \in C(u)$ is assigned $f(c) = 1$ or $f(c) = 2$, and the rest of them are assigned any value from $\{0, 1, 2\}$ by the TRDF.

- In this case, as the vertex u is assigned the value 2 by the TRDF, hence, for any child $c \in C(u)$, if $f(c) = 0$, as the first condition of total Roman domination is satisfied by its parent, children of c can be assigned any value by the TRDF else if $f(c) = 1$ or $f(c) = 2$, as the second condition of total Roman domination is satisfied by its parent, children of c can be assigned any value by the TRDF. From the constraint of dp_2, for each child, $c \in C(u)$, we consider the minimum for the cases $f(c) = 0$, $f(c) = 1$ and $f(c) = 2$.

- Similar to the previous case, in this case too, as $f(u) = 2$, if there is a child $c \in C(u)$, such that $f(c) = 0$, as the first condition of total Roman domination is satisfied by its parent, children of c can be assigned any value by the TRDF. If $f(c) = 1$ or $f(c) = 2$, as the second condition of total Roman domination is satisfied by its parent, children of c can be assigned any value by the TRDF. Also, from the constraint of $dp_2^{1,2}$, at least one of the children of u must be assigned 1 or 2 by the TRDF, to satisfy the total Roman domination second condition. Hence, to calculate the total Roman domination number of the subtree rooted at

u, we take the minimum of all the cases in which one child $c \in C(u)$ is assigned $f(c) = 1$ or $f(c) = 2$, and the rest of them are assigned any value from $\{0, 1, 2\}$ by the TRDF.

□

The outline of the algorithm is as follows. We first do a Depth First Search traversal on the given tree $T = (V, E)$, starting from the root $r \in V$. In the Depth First Search traversal, for a particular vertex, we first visit all the children of the vertex, and then come back to the vertex. Hence, when we reach a vertex v, then the entries corresponding to the children of v are already filled in the 6 arrays, $dp_0, dp_1, dp_2, dp_0^2, dp_1^{1,2}$ and $dp_2^{1,2}$, and hence, from these, we can fill in the values of $dp_0[v]$, $dp_1[v]$, $dp_2[v]$, $dp_0^2[v]$, $dp_2^{1,2}[v]$, and $dp_2^{1,2}[v]$. After the traversal is complete, all the entries of the 6 arrays will be filled. Finally, as the root r can be assigned the values $\{0, 1, 2\}$ by the TRDF, and if $f(r) = 0$, then at least one of the children of r should be assigned 2 to satisfy the TRDF first condition. If $f(r) = 1$ or $f(r) = 2$, then at least one of the children of r should be assigned 1 or 2 to satisfy the TRDF second condition. Hence we take the $\min\{dp_0^2[r], dp_1^{1,2}[r], dp_2^{1,2}[r]\}$ to find out the total Roman domination number for the tree T.

We make some changes to the formula of $dp_0^2[u]$ for computational efficiency.

Lemma 2.3.3. *For a non-leaf vertex $u \in V$, we can write*

$$dp_0^2[u] = \alpha + dp_0[u]$$

where $\alpha = \min_{c \in C(u)} \{dp_2^{1,2}[c] - \min\{dp_0^2[c], dp_1^{1,2}[c], dp_2^{1,2}[c]\}\}$

Proof. From Theorem 2.3.2, we can write,

$$
\begin{aligned}
dp_0^2[u] &= \min_{c \in C(u)} \{dp_2^{1,2}[c] + \sum_{c' \in C(u) \setminus c} \min\{dp_0^2[c'], dp_1^{1,2}[c'], dp_2^{1,2}[c']\}\} \\
&= \min_{c \in C(u)} \{dp_2^{1,2}[c] - \min\{dp_0^2[c], dp_1^{1,2}[c], dp_2^{1,2}[c]\} + \sum_{c' \in C(u)} \min\{dp_0^2[c'], dp_1^{1,2}[c'], dp_2^{1,2}[c']\}\} \\
&= \min_{c \in C(u)} \{dp_2^{1,2}[c] - \min\{dp_0^2[c], dp_1^{1,2}[c], dp_2^{1,2}[c]\} + dp_0[u]\} \\
&= \min_{c \in C(u)} \{dp_2^{1,2}[c] - \min\{dp_0^2[c], dp_1^{1,2}[c], dp_2^{1,2}[c]\}\} + dp_0[u] \\
&= \alpha + dp_0[u] \\
&\textit{where } \alpha = \min_{c \in C(u)} \{dp_2^{1,2}[c] - \min\{dp_0^2[c], dp_1^{1,2}[c], dp_2^{1,2}[c]\}\}
\end{aligned}
$$

□

Lemma 2.3.4. *For a non-leaf vertex $u \in V$, we can write*

$$dp_1^{1,2}[u] = \beta + dp_1[u]$$

where $\beta = \min_{c \in C(u)} \{\min\{dp_1[c], dp_2[c]\} - min\{dp_0^2[c], dp_1[c], dp_2[c]\}\}$

Proof. From Theorem 2.3.2, we can write,

$$
\begin{aligned}
dp_1^{1,2}[u] &= 1 \times w(u) + \min_{c \in C(u)} \{\min\{dp_1[c], dp_2[c]\} + \sum_{c' \in C(u) \setminus \{c\}} \min\{dp_0^2[c'], dp_1[c'], dp_2[c']\}\} \\
&= 1 \times w(u) + \min_{c \in C(u)} \{\min\{dp_1[c], dp_2[c]\} - \min\{dp_0^2[c], dp_1[c], dp_2[c]\} \\
&+ \sum_{c' \in C(u)} \min\{dp_0^2[c'], dp_1[c'], dp_2[c']\}\} \\
&= \min_{c \in C(u)} \{\min\{dp_1[c], dp_2[c]\} - min\{dp_0^2[c], dp_1[c], dp_2[c]\} + dp_1[u]\}
\end{aligned}
$$

$$= \min_{c \in C(u)} \{\min\{dp_1[c], dp_2[c]\} - min\{dp_0^2[c], dp_1[c], dp_2[c]\}\} + dp_1[u]$$

$$= \beta + dp_1[u]$$

$$where\ \beta = \min_{c \in C(u)} \{\min\{dp_1[c], dp_2[c]\} - min\{dp_0^2[c], dp_1[c], dp_2[c]\}\}$$

□

Lemma 2.3.5. *For a non-leaf vertex $u \in V$, we can write*

$$dp_2^{1,2}[u] = \gamma + dp_2[u]$$

$$where\ \gamma = \min_{c \in C(u)} \{\min\{dp_1[c], dp_2[c]\} - min\{dp_0[c], dp_1[c], dp_2[c]\}\}$$

Proof. From Theorem 2.3.2, we can write,

$$dp_1^{1,2}[u] = 2 \times w(u) + \min_{c \in C(u)} \{\min\{dp_1[c], dp_2[c]\} + \sum_{c' \in C(u) \setminus \{c\}} \min\{dp_0[c'], dp_1[c'], dp_2[c']\}\}$$

$$= 2 \times w(u) + \min_{c \in C(u)} \{\min\{dp_1[c], dp_2[c]\} - \min\{dp_0[c], dp_1[c], dp_2[c]\}$$

$$+ \sum_{c' \in C(u)} \min\{dp_0[c'], dp_1[c'], dp_2[c']\}\}$$

$$= \min_{c \in C(u)} \{\min\{dp_1[c], dp_2[c]\} - min\{dp_0[c], dp_1[c], dp_2[c]\} + dp_2[u]\}$$

$$= \min_{c \in C(u)} \{\min\{dp_1[c], dp_2[c]\} - min\{dp_0[c], dp_1[c], dp_2[c]\}\} + dp_2[u]$$

$$= \gamma + dp_2[u]$$

$$where\ \gamma = \min_{c \in C(u)} \{\min\{dp_1[c], dp_2[c]\} - min\{dp_0[c], dp_1[c], dp_2[c]\}\}$$

□

2.3.2 Time and Space Complexity Analysis

Lemma 2.3.6. *For a tree T with n vertices, the time and space complexities of Algorithm 4 are $O(n)$ and $O(n)$ respectively.*

Proof. The time complexity of Algorithm 4 is $O(n)$ as we are using Depth First Search method to traverse the tree which takes $O(n)$ time. We visit each vertex exactly once in the algorithm, and as the number of vertices are n, the total time complexity is $O(n)$. The space complexity is $O(n)$ since we create 6 arrays, $dp_0, dp_0^2, dp_1, dp_1^{1,2}, dp_2, dp_2^{1,2}$ of size n each, and a single variables, α, β, γ. □

2.3.3 Proof of Correctness

Lemma 2.3.7. *Algorithm 4 computes the total Roman Domination number of a given tree $T = (V, E)$ correctly*

Proof. We use induction to prove the correctness of the algorithm. We show that Algorithm 4 computes the Roman domination number correctly for all trees of height $h \in \mathbb{Z}^+$.

- Base Case: For $h = 1$,

  The tree contains a single node. Let us denote it by u. Now,

  $dp_0[u] = 0$, $dp_0^2[u] = \infty$, $dp_1[u] = 1 \times w(u)$, $dp_1^{1,2}[u] = \infty$, $dp_2[u] = 2 \times w(u)$, $dp_2^{1,2}[u] = \infty$

  $\gamma_{tRW} = \min\{dp_0^2[u], dp_1^{1,2}[u], dp_2^{1,2}[u]\} = \min\{\infty, \infty, \infty\} = \infty$.

- Induction Hypothesis: Let the correctness of the algorithms hold on trees of height $h'(< h)$.

- Induction Step: Let T be a tree of height h and let us fix a vertex r to be the root of T. Let us have a weight function $w : V(T) \to R$ for the vertices of the tree T. For all $c \in C(r)$, the height of the subtree rooted at c, $h' < h$. So, we have optimal values of $dp_0[c], dp_0^2[c], dp_1[c], dp_1^{1,2}[c], dp_2[c], dp_2^{1,2}[c]$ for all $c \in C(r)$. Using these and the results from Theorem 2.3.2, we can find the optimal values of $dp_0[r], dp_0^2[r], dp_1[r], dp_1^{1,2}[r], dp_2[r], dp_2^{1,2}[r]$. Now, the total Roman domination number of the weighted tree will be $\gamma_{tRW}(T) = \min\{dp_0^2[r], dp_1^{1,2}[r], dp_2^{1,2}[r]\}$. This is because, from the TRDF, the vertex r can be assigned the value either 0 or 1 or 2, and if $f(r) = 0$, it is necessary that for at least one child $c \in C(r)$, $f(c) = 2$ to satisfy the total Roman domination first condition, if $f(r) = 1$ or $f(r) = 2$, it is necessary that for at least one child $c \in C(r)$, $f(c) \in \{1, 2\}$ to satisfy the total Roman domination second condition, as r doesn't have any parent vertex. Hence, the induction hypothesis is true for all trees of height h.

Hence, we see that Algorithm 4 computes the Roman domination number correctly for all trees of height $h \in \mathbb{Z}^+$. □

Algorithm 4 total Roman Domination in weighted trees

Input A tree T fixed at a root r

Output total Roman Domination number γ_{tRW} of T

function TRD(T,u)
  $dp_0[u] \leftarrow 0$
  $dp_0^2[u] \leftarrow \infty$
  $dp_1[u] \leftarrow 1 \times w(u)$
  $dp_1^{1,2}[u] \leftarrow \infty$
  $dp_2[u] \leftarrow 2 \times w(u)$
  $dp_2^{1,2}[u] \leftarrow \infty$
  $\alpha \leftarrow \infty$
  $\beta \leftarrow \infty$
  $\gamma \leftarrow \infty$
  for each $c \in T.C(u)$ **do** ▷ Depth First Search traversal of tree
    TRD(T,c)
    $dp_0[u] \leftarrow dp_0[u] + \min\{dp_0^2[c], dp_1^{1,2}[c], dp_2^{1,2}[c]\}$
    $\alpha \leftarrow \min\{\alpha, dp_2^{1,2}[c] - \min\{dp_0^2[c], dp_1^{1,2}[c], dp_2^{1,2}[c]\}$
    $dp_1[u] \leftarrow dp_1[u] + \min\{dp_0^2[c], dp_1[c], dp_2[c]\}$
    $\beta \leftarrow \min\{\beta,\ \min\{dp_1[c], dp_2[c]\} - min\{dp_0^2[c], dp_1[c], dp_2[c]\}\}$
    $dp_2[u] \leftarrow dp_2[u] + \min\{dp_0[c], dp_1[c], dp_2[c]\}$
    $\gamma \leftarrow \min\{\gamma,\ \min\{dp_1[c], dp_2[c]\} - min\{dp_0[c], dp_1[c], dp_2[c]\}\}$
  if u is non-leaf **then** ▷ All nodes other than leaf
    $dp_0^2[u] \leftarrow dp_0[u] + \alpha$
    $dp_1^{1,2}[u] \leftarrow dp_1[u] + \beta$
    $dp_2^{1,2}[u] \leftarrow dp_2[u] + \gamma$

function PRINT-TRD(T,r) ▷ Let r be the root of the tree T
  TRD(T,r)
  return $\min\{dp_0^2[r], dp_1^{1,2}[r], dp_2^{1,2}[r]\}$

Chapter 3

Italian Domination in Weighted Trees

3.1 Perfect Italian Domination in Weighted Trees

Definition 3.1.1. A Perfect Italian dominating function PIDF on an undirected graph $G = (V, E)$ is an Italian dominating function in which for every vertex $u \in V$ with $f(u) = 0$ it holds that $\Sigma_{v \in N(u)} f(v) = 2$ where $N(u) = \{v \in V | (u, v) \in E\}$. The minimum weight $\Sigma_{u \in V(G)} f(u)$ of a perfect Italian dominating function on a graph G is called the perfect Italian domination number $\gamma_I^P(G)$ of G.

Let $T = (V, E)$ be a weighted tree rooted at a vertex $r \in V$ with the weight function $w : V(T) \to \mathbb{R}$. In a tree, a parent of a vertex $v \in V$ is the vertex $u \in V$ connected to v on the path to the root. Every vertex has a

unique parent except the root which has no parent. A child of a vertex u is a vertex of which u is the parent. Let us denote the set of children of a vertex $u \in V$ by $C(u)$.

In this case, we define the weight of a perfect Italian dominating function f to be $f(V(T)) = \Sigma_{u \in V(T)} w(u) \times f(u)$, where $f : V(T) \rightarrow \{0, 1, 2\}$ is a perfect Italian dominating function on the weighted tree T. The minimum weight of a perfect Italian dominating function on a weighted tree T is called the Perfect Italian Domination Number(PIDN) of T and is denoted by $\gamma_{IW}^{P}(T)$.

We use the dynamic programming paradigm to solve the problem of finding the Perfect Italian Domination Number(PIDN) of a weighted tree. Let u be a specific vertex in T, and let T' be the subtree rooted at u. Let $f : V(T') \rightarrow \{0, 1, 2\}$ be a PIDF for the subtree T'. For the vertex u, it is given that $f(u) \in \{0, 1, 2\}$. Hence it is useful to consider the following 5 subproblems:

- $dp_0^0[u] = \min\{\ f(V(T'))\ :\ f(u) = 0\ \ and\ \ \Sigma_{v \in C(u)} f(v) = 0\}$
- $dp_0^1[u] = \min\{\ f(V(T'))\ :\ f(u) = 0\ \ and\ \ \Sigma_{v \in C(u)} f(v) = 1\}$
- $dp_0^2[u] = \min\{\ f(V(T'))\ :\ f(u) = 0\ \ and\ \ \Sigma_{v \in C(u)} f(v) = 2\}$
- $dp_1[u] = \min\{\ f(V(T'))\ :\ f(u) = 1\}$
- $dp_2[u] = \min\{\ f(V(T'))\ :\ f(u) = 2\}$

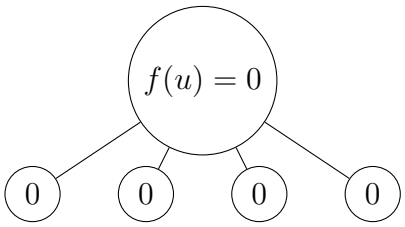

Figure 3.1: $dp_0^0[u]$

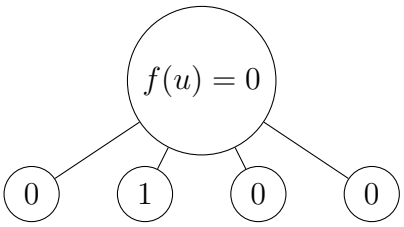

Figure 3.2: $dp_0^1[u]$

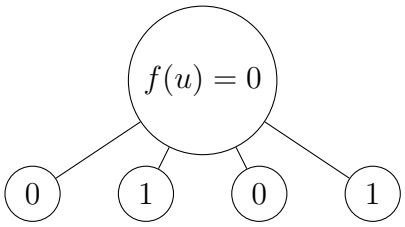

Figure 3.3: $dp_0^2[u]$ - Case 1

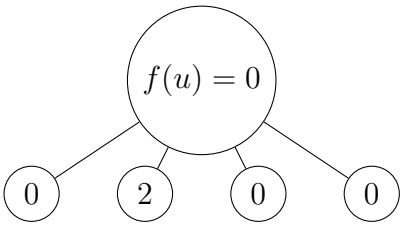

Figure 3.4: $dp_0^2[u]$ - Case 2

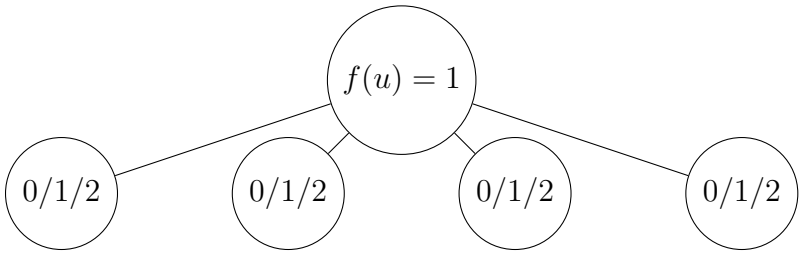

Figure 3.5: $dp_1[u]$

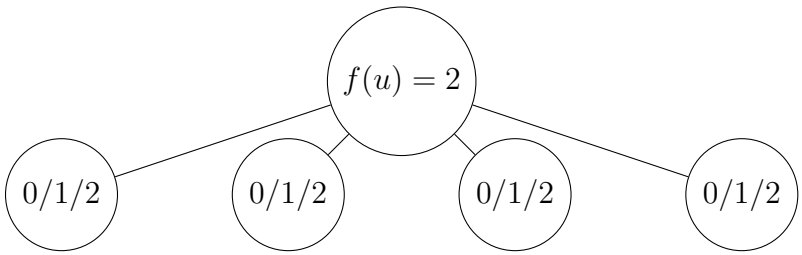

Figure 3.6: $dp_2[u]$

Here, $dp_0^0[u]$ is the Perfect Italian Domination Number for the subtree T' with the condition that $f(u) = 0$, and the sum of all the children of u is 0 this implies all the children are assigned 0 by the PIDF. In this case, we assume that for a vertex u with $f(u) = 0$ it has a parent t with $f(t) = 2$, i.e., the PIDF condition for the vertex u is satisfied only by it's parent.

$dp_0^1[u]$ is the Perfect Italian Domination Number for the subtree T' with the condition that $f(u) = 0$, and the sum of all the children of u is 1 this implies one child of u is assigned the value 1, and all the other children of u are assigned 0 by the PIDF. In this case, we assume that for a vertex u with $f(u) = 0$ it has a parent t with $f(t) = 1$ to satisfy the PIDF condition, i.e., the PIDF condition for the vertex u is satisfied by both it's parent and one of it's children.

$dp_0^2[u]$ is the Perfect Italian Domination Number for the subtree T' with the condition that $f(u) = 0$, and the sum of all the children of u is 2. We have 2 such cases one where two children of u are assigned the value 1, and all the other children of u are assigned 0 by the PIDF another where one child of u is assigned the value 2, and all the other children of u are assigned 0 by the PIDF. In this case, we see that for a vertex u with $f(u) = 0$, it has children sum $\Sigma_{v \in C(u)} f(v) = 2$, i.e the PIDF condition for vertex u is satisfied by its children.

$dp_1[u]$ is the Perfect Italian Domination Number for the subtree T' with the condition that $f(u) = 1$. Similarly, $dp_2[u]$ is the Perfect Italian Domination Number for the subtree T' with the condition that $f(u) = 2$.

3.1.1 Algorithm

Theorem 3.1.2. *Let $T = (V, E)$ be a tree with a root $r \in V$. Let $u \in V$ be a non-leaf vertex of the tree. Then the following statements hold:*

- $dp_1[u] = 1 \times w(u) + \sum_{c \in C(u)} \min\{dp_0^1[c], dp_1[c], dp_2[c]\}$
- $dp_2[u] = 2 \times w(u) + \sum_{c \in C(u)} \min\{dp_0^0[c], dp_1[c], dp_2[c]\}$
- $dp_0^0[u] = \sum_{c \in C(u)} \{dp_0^2[c]\}$
- $dp_0^1[u] = \min_{c \in C(u)} \{dp_1[c] + \sum_{c' \in C(u) \setminus \{c\}} \{dp_0^2[c']\}\}$
- $dp_0^2[u] = \min\{\alpha, \beta\}$ *where*
  $\alpha = \min_{c_1, c_2 \in C(u)} \{dp_1[c_1] + dp_1[c_2] + \sum_{c' \in C(u) \setminus \{c_1, c_2\}} \{dp_0^2[c']\}\}$
  $\beta = \min_{c \in C(u)} \{dp_2[c] + \sum_{c' \in C(u) \setminus \{c\}} \{dp_0^2[c']\}\}$

Proof. The proofs of the above statements are as follows:

- For a vertex u with $f(u) = 1$, there is no restriction on the values of children that can be assigned by the PIDF. Hence, for a subtree rooted at u, it's Perfect Italian Domination(PID) number will be equal to the sum of the PID numbers of the subtrees formed by each of it's children added to the contribution at vertex u, i.e. $w(u) \times 1$. Each of it's children can be assigned the value $\{0, 1, 2\}$ by the PIDF, hence we take a minimum of these 3 cases, for each subtree. Also, if one of the children of vertex u, let's say c, is assigned the value 0 by the PIDF, then for the vertex c to satisfy the PID condition, sum of the children

of c should be assigned the value 1. Hence, we consider the case of $dp_0^1[c]$, while calculating the value of $dp_1[u]$.

- This case is similar to the previous case, except that for a vertex u with $f(u) = 2$, it's contribution to the PID number is $w(u) \times 2$. Each of it's children can be assigned the value $\{0, 1, 2\}$ by the PIDF, hence we take the minimum of these 3 cases, for each subtree. Also, for a vertex u with $f(u) = 2$, if it has a child c such that $f(c) = 0$, then from the Italian Domination condition, we can say that sum of the children of c should be assigned the value 0 by the IDF. Hence, we consider the case of $dp_0^0[c]$ while calculating the value of $dp_2[u]$.

- In this case $dp_0^0[u]$, as the vertex u is assigned the value 0 by the PIDF, and sum of children of u is 0. Hence, for all children $c \in C(u)$, as $f(c) = 0$, then to satisfy the condition of PID, sum of the children of c must be assigned the value 2 by the PIDF. Hence, we consider the case of $dp_0^2[c]$ while calculating the value of $dp_0^0[u]$.

- In this case $dp_0^1[u]$ too, as $f(u) = 0$ and the sum of children of u is 1, if there is a child $c \in C(u)$, such that $f(c) = 0$, then to satisfy the condition of PID, the sum of the children of c must be assigned 2 by the PIDF. Hence, we consider the case $dp_0^2[c]$. Also, from the constraint of dp_0^1, exactly one of the children of u should be assigned 1 by the PIDF. Hence, to calculate the PID number of the subtree rooted at u, we take a minimum of all the cases in which one child $c \in C(u)$ is assigned $f(c) = 1$, and the rest of them are assigned 0 by the PIDF.

- Similarly, in the case $dp_0^2[u]$, as $f(u) = 0$ and the sum of children of u

is 2, if there is a child $c \in C(u)$, such that $f(c) = 0$, then to satisfy the condition of PID, sum of the children of c must be assigned the value 2 by the PIDF. Hence, we consider the case $dp_0^2[c]$. Also, from the constraint of dp_0^2, either two of the children of u should be assigned 1 by the PIDF or one of the children of u should be assigned 2 by the PIDF, and the remaining children should be assigned 0. Hence, we take minimum of these two cases.

□

Lemma 3.1.3. *For a non-leaf vertex $u \in V$, we can write*

$$dp_0^1[u] = s1 + dp_0^0[u]$$

where $s1 = \min_{c \in C(u)} \{dp_1[c] - dp_0^2[c]\}$

Proof. From Theorem 3.1.2, we can write,

$$\begin{aligned}
dp_0^1[u] &= \min_{c \in C(u)} \{dp_1[c] + \sum_{c' \in C(u) \setminus \{c\}} dp_0^2[c'] \} \\
&= \min_{c \in C(u)} \{dp_1[c] - dp_0^2[c] + \sum_{c' \in C(u)} dp_0^2[c'] \} \\
&= \min_{c \in C(u)} \{dp_1[c] - dp_0^2[c] + dp_0^0[u]\} \\
&= \min_{c \in C(u)} \{dp_1[c] - dp_0^2[c]\} + dp_0^0[u] \\
&= s1 + dp_0^0[u] \\
&\text{where } s1 = \min_{c \in C(u)} \{dp_1[c] - dp_0^2[c]\}
\end{aligned}$$

□

Algorithm 5 Perfect Italian Domination in weighted trees

Input A tree T fixed at a root r

Output Perfect Italian Domination number γ_{IW}^{P} of T

function PID(T,u)
  for each $c \in T.C(u)$ **do** ▷ Depth First Search Traversal of Tree
    PID(T,c)
  $dp_0^0[u] \leftarrow 0$
  $dp_0^1[u] \leftarrow \infty$
  $dp_0^2[u] \leftarrow \infty$
  $dp_1[u] \leftarrow 1 \times w(u)$
  $dp_2[u] \leftarrow 2 \times w(u)$
  if u is non-leaf **then** ▷ All nodes other than leaf
    for each $c \in T.C(u)$ **do**
      $dp_0^0[u] \leftarrow dp_0^0[u] + \{dp_0^2[c]\}$ ▷ Calculating $dp_0^0[u]$
      $s1 \leftarrow dp_1[c]$ ▷ Calculating $dp_0^1[u]$
      for each $v \in T.C[u] - c$ **do**
        $s1 \leftarrow s1 + \{dp_0^2[v]\}$
      $dp_0^1[u] \leftarrow \min\{dp_0^1[u], s1\}$
      for each $v1 \in T.C[u] - c$ **do** ▷ Calculating α
        $s2 \leftarrow dp_1[c] + dp_1[v1]$
        for each $v2 \in T.C[u] - c - v1$ **do**
          $s2 \leftarrow s2 + \{dp_0^2[v2]\}$
      $\alpha \leftarrow \min\{\alpha, s2\}$
      $s3 \leftarrow dp_2[c]$ ▷ Calculating β
      for each $v \in T.C[u] - c$ **do**
        $s3 \leftarrow s3 + \{dp_0^2[v]\}$
      $\beta \leftarrow \min\{\beta, s3\}$
      $dp_0^2[u] = \min\{\alpha, \beta\}$ ▷ Calculating $dp_0^2[u]$
      $dp_1[u] \leftarrow dp_1[u] + \min\{dp_1[c], dp_2[c], dp_0^1[c]\}$
      $dp_2[u] \leftarrow dp_2[u] + \min\{dp_1[c], dp_2[c], dp_0^0[c]\}$

function PRINT-PID(T,r) ▷ Let r be the root of the tree T
  PID(T,r)
  return $\min\{dp_0^2[r], dp_1[r], dp_2[r]\}$

Lemma 3.1.4. *For a non-leaf vertex $u \in V$, we can write*

$$\alpha = s1 + s2 + dp_0^0[u]$$

where,

$$\alpha = \min_{c_1,c_2 \in C(u)} \{dp_1[c_1] + dp_1[c_2] + \sum_{c' \in C(u)\setminus\{c_1,c_2\}} \{dp_0^2[c']\}\}$$
$$s1 = \min_{c_1 \in C(u)} \{dp_1[c_1] - dp_0^2[c_1]\}$$
$$s2 = \min_{c_2 \in C(u)\setminus\{c_1\}} \{dp_1[c_2] - dp_0^2[c_2]\}$$

Proof. From Theorem 3.1.2, we can write,

$$\begin{aligned}
\alpha &= \min_{c_1,c_2 \in C(u)} \{dp_1[c_1] + dp_1[c_2] + \sum_{c' \in C(u)\setminus\{c_1,c_2\}} \{dp_0^2[c']\}\} \\
&= \min_{c_1,c_2 \in C(u)} \{dp_1[c_1] - dp_0^2[c_1] + dp_1[c_2] - dp_0^2[c_2] + \sum_{c' \in C(u)} \{dp_0^2[c']\}\} \\
&= \min_{c_1,c_2 \in C(u)} \{dp_1[c_1] - dp_0^2[c_1] + dp_1[c_2] - dp_0^2[c_2] + dp_0^0[u]\} \\
&= \min_{c_1,c_2 \in C(u)} \{dp_1[c_1] - dp_0^2[c_1] + dp_1[c_2] - dp_0^2[c_2]\} + dp_0^0[u] \\
&= \min_{c_1 \in C(u)} \{dp_1[c_1] - dp_0^2[c_1]\} + \min_{c_2 \in C(u)\setminus\{c_1\}} \{dp_1[c_2] - dp_0^2[c_2]\} + dp_0^0[u] \\
&= s1 + s2 + dp_0^0[u] \\
&\textit{where } s1 = \min_{c_1 \in C(u)} \{dp_1[c_1] - dp_0^2[c_1]\} \\
&s2 = \min_{c_2 \in C(u)\setminus\{c_1\}} \{dp_1[c_2] - dp_0^2[c_2]\}
\end{aligned}$$

□

Lemma 3.1.5. *For a non-leaf vertex $u \in V$ we can write*

$$\beta = s3 + dp_0^0[u]$$

where,

$$\beta = \min_{c \in C(u)} \{dp_2[c] + \sum_{c' \in C(u) \setminus \{c\}} \{dp_0^2[c']\}\}$$
$$s3 = \min_{c \in C(u)} \{dp_2[c] - dp_0^2[c]\}$$

Proof. From Theorem 3.1.2, we can write,

$$\begin{aligned}
\beta &= \min_{c \in C(u)} \{dp_2[c] + \sum_{c' \in C(u) \setminus \{c\}} dp_0^2[c'] \} \\
&= \min_{c \in C(u)} \{dp_2[c] - dp_0^2[c] + \sum_{c' \in C(u)} dp_0^2[c'] \} \\
&= \min_{c \in C(u)} \{dp_2[c] - dp_0^2[c] + dp_0^0[u]\} \\
&= \min_{c \in C(u)} \{dp_2[c] - dp_0^2[c]\} + dp_0^0[u] \\
&= s3 + dp_0^0[u] \\
&\textit{where } s3 = \min_{c \in C(u)} \{dp_2[c] - dp_0^2[c]\}
\end{aligned}$$

□

We can use Lemma 3.1.3, Lemma 3.1.4 and Lemma 3.1.5 to make some changes to Algorithm 4 to make it more computationally efficient, and reduce it's time complexity.

Algorithm 6 Perfect Italian Domination in weighted trees

Input A tree T fixed at a root r

Output Perfect Italian Domination number γ^P_{IW} of T

function PID(T,u)
  $dp^0_0[u] \leftarrow 0$
  $dp^1_0[u] \leftarrow \infty$
  $dp^2_0[u] \leftarrow \infty$
  $dp_1[u] \leftarrow 1 \times w(u)$
  $dp_2[u] \leftarrow 2 \times w(u)$
  $s1 \leftarrow \infty$
  $s2 \leftarrow \infty$
  $s3 \leftarrow \infty$
  for each $c \in T.C(u)$ **do** ▷ Depth First Search Traversal of Tree
    PID(T,c)
    $dp^0_0[u] \leftarrow dp^0_0[u] + \{dp^2_0[c]\}$
    $temp \leftarrow s1$
    $s1 \leftarrow \min\{s1, dp_1[c] - dp^2_0[c]\}$
    if $s1 = dp_1[c] - dp^2_0[c]$ **then**
      $s2 \leftarrow temp$
    else
      $s2 \leftarrow \min\{s2, dp_1[c] - dp^2_0[c]\}$
    $s3 \leftarrow \min\{s3, dp_2[c] - dp^2_0[c]\}$
    $dp_1[u] \leftarrow dp_1[u] + \min\{dp_1[c], dp_2[c], dp^1_0[c]\}$
    $dp_2[u] \leftarrow dp_2[u] + \min\{dp_1[c], dp_2[c], dp^0_0[c]\}$
  if u is non-leaf **then** ▷ All nodes other than leaf
    $dp^1_0[u] \leftarrow s1 + dp^0_0[u]$
    $\alpha \leftarrow s1 + s2 + dp^0_0[u]$
    $\beta \leftarrow s3 + dp^0_0[u]$
    $dp^2_0[u] = \min\{\alpha, \beta\}$

function PRINT-PID(T,r) ▷ Let r be the root of the tree T
  PID(T,r)
  return $\min\{dp^2_0[r], dp_1[r], dp_2[r]\}$

3.1.2 Time and Space Complexity Analysis

Lemma 3.1.6. *For a tree T with n vertices, the time and space complexities of Algorithm 4 are $O(n^3)$ and $O(n)$ respectively.*

Proof. We know that a Depth First Search on a tree has a time complexity of $O(n)$, where n is the number of vertices in the tree. Along with that, in our PID function, we have two more loops in line 16 and 18 which runs a maximum of $n-2$ and $n-3$ times each, i.e. it takes $O(n^2)$ time. Hence, the Algorithm 4 has a time complexity of $O(n^3)$. The space complexity is $O(n)$ since we create 5 arrays, $dp_0^0, dp_0^1, dp_0^2, dp_1, dp_2$ of size n each. □

Lemma 3.1.7. *For a tree T with n vertices, the time and space complexities of Algorithm 5 are $O(n)$ and $O(n)$ respectively.*

Proof. The time complexity of Algorithm 5 is $O(n)$ as we are using Depth First Search method to traverse the tree which takes $O(n)$ time. We visit each vertex exactly once in this algorithm, and as the number of vertices are n, the total time complexity is $O(n)$. The space complexity is $O(n)$ since we create 5 arrays, $dp_0^0, dp_0^1, dp_0^2, dp_1, dp_2$ of size n each, and a three variables, $s1, s2, s3$. □

3.1.3 Proof of Correctness

Lemma 3.1.8. *Algorithm 4 computes the Perfect Italian Domination Number(PIDN) of a given tree $T = (V, E)$ correctly*

Proof. We use mathematical induction to prove the correctness of algorithm. We show that Algorithm 4 computes the PID number correctly for all trees of height $h \in \mathbb{Z}^+$.

- Base Case: For $h = 1$,

  The tree contains a single node. Let us denote it by u. Now,
  $dp_0^0[u] = 0,\ dp_0^1[u] = \infty,\ dp_0^2[u] = \infty,\ dp_1[u] = 1 \times w(u),\ dp_2[u] = 2 \times w(u)$
  $\gamma_{IW}^P = \min\{dp_0^2[u], dp_1[u], dp_2[u]\} = \min\{\infty, 1 \times w(u), 2 \times w(u)\} = 1 \times w(u)$.

- Induction Hypothesis: Let the correctness of the algorithm hold for trees of height $h'(< h)$.

- Induction Step: Let T be a tree of height h and let us fix a vertex r to be the root of T. Let us have a weight function $w : V(T) \to R$ for the vertices of the tree T. For all $c \in C(r)$, the height of the subtree rooted at c is h' which is less than h. So, we have optimal values of $dp_0^0[c], dp_0^1[c], dp_0^2[c], dp_1[c], dp_2[c]$ for all $c \in C(r)$. Therefore, from Theorem 3.1.2, we have the optimal values of $dp_0^0[r], dp_0^1[r], dp_0^2[r], dp_1[r], dp_2[r]$.
  Now, the PID number of the weighted tree will be $\gamma_{IW}^P(T) = \min\{dp_0^2[r], dp_1[r], dp_2[r]\}$. This is because, from the PIDF, the vertex r can be assigned the value either 0 or 1 or 2, and if $f(r) = 0$, it is necessary that for sum of its children $\Sigma_{c \in C(r)} f(c) = 2$ to satisfy the PID condition, as r does not have any parent vertex. Hence, the induction hypothesis is true for all trees of height h.

Hence, we see that Algorithm 4 computes the PID number correctly for all trees of height $h \in \mathbb{Z}^+$. □

3.2 Italian Domination in Weighted Trees

Definition 3.2.1. An Italian Dominating Function(IDF) on an undirected graph $G = (V, E)$ is a function $f : V(G) \to \{0, 1, 2\}$ satisfying the condition that for every vertex $u \in V$ with $f(u) = 0$, it holds that $\Sigma_{v \in N(u)} f(v) \geq 2$. The weight of an Italian dominating function f is the value $f(V(G)) = \Sigma_{u \in V(G)} f(u)$. The minimum weight of an Italian dominating function on a graph G is called the Italian domination number $\gamma_I(G)$ of G.

Let us consider the weighted tree $T = (V, E)$ with a weight function $w : V \to \mathbb{R}$ that we have considered in the previous section. In the case of a weighted tree, we define the weight of an Italian dominating function f on a tree $T = (V, E)$ to be $f(V(T)) = \Sigma_{u \in V(T)} w(u) \times f(u)$, where $f : V(T) \to \{0, 1, 2\}$ is an Italian dominating function on the weighted tree T. The minimum weight of an Italian dominating function on a weighted tree T is called the Italian Domination Number(IDN) of T and is denoted by $\gamma_{IW}(T)$.

We use the dynamic programming paradigm to solve the problem of finding the Italian Domination Number(IDN) of a weighted tree. Let u be a specific vertex in T, and let T' be the subtree rooted at u. Let $f : V(T') \to \{0, 1, 2\}$ be a RDF for the subtree T'. For the vertex u, it is given that $f(u) \in \{0, 1, 2\}$. Hence it is useful to consider the following 5 subproblems:

- $dp_0^0[u] = \min\{\ f(V(T'))\ :\ f(u) = 0\ \ and\ \ \Sigma_{v \in C(u)} f(v) \geq 0\}$
- $dp_0^1[u] = \min\{\ f(V(T'))\ :\ f(u) = 0\ \ and\ \ \Sigma_{v \in C(u)} f(v) \geq 1\}$

- $dp_0^2[u] = \min\{\ f(V(T'))\ :\ f(u) = 0\ \ and\ \ \Sigma_{v \in C(u)} f(v) \geq 2\}$
- $dp_1[u] = \min\{\ f(V(T'))\ :\ f(u) = 1\}$
- $dp_2[u] = \min\{\ f(V(T'))\ :\ f(u) = 2\}$

$dp_0^0[u]$ is the IDN of the subtree T' with the condition that $\Sigma_{v \in C(u)} f(v) \geq 0$. In this case, we assume that for a vertex $u \in V$ with $f(u) = 0$, it has a parent vertex $t \in V$ with $f(t) = 2$, i.e. the IDF condition for the vertex u is satisfied by it's parent t, and hence, it's children can be assigned any value by the IDF.

$dp_0^1[u]$ is the IDN of the subtree T' with the condition that $\Sigma_{v \in C(u)} f(v) \geq 1$. In this case, we assume that for a vertex $u \in V$ with $f(u) = 0$, it has a parent vertex $t \in V$ with $f(t) = 1$, i.e the IDF condition for the vertex u is satisfied by both it's parent vertex and its children, and hence, the IDF should assign values to the child vertices in such a way that their total sum is greater than or equal to 1.

$dp_0^2[u]$ is the IDN of the subtree T' with the condition that $\Sigma_{v \in C(u)} f(v) \geq 2$. In this case, we assume that for a vertex $u \in V$ with $f(u) = 0$, the vertex u has no parent, or, it has a parent vertex $t \in V$ with $f(t) = 0$, i.e. the IDF condition for u is satisfied by it's children, and hence, the IDF should assign values to the child vertices in such a way that their total sum is greater than or equal to 2.

Here, $dp_1[u]$ is the Italian Domination Number(IDN) for the subtree T' with the condition that $f(u) = 1$. Similarly, $dp_2[u]$ is the IDN for the subtree T' with the condition that $f(u) = 2$.

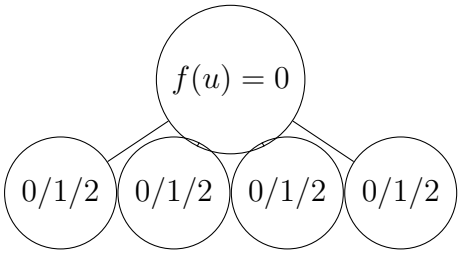

Figure 3.7: $dp_0^0[u]$

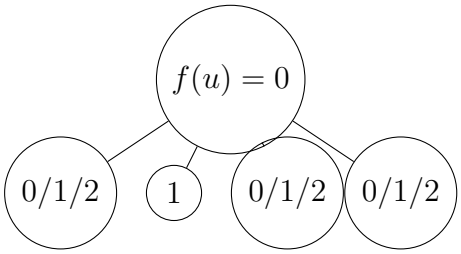

Figure 3.8: $dp_0^1[u]$

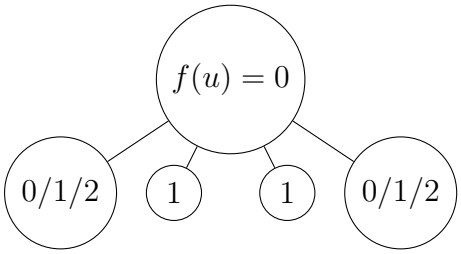

Figure 3.9: $dp_0^2[u]$ - Case 1

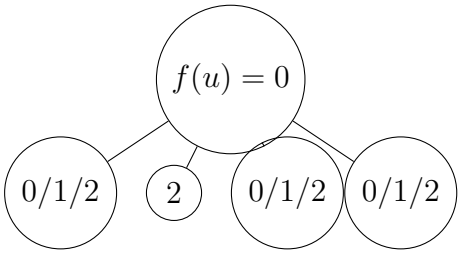

Figure 3.10: $dp_0^2[u]$ - Case 2

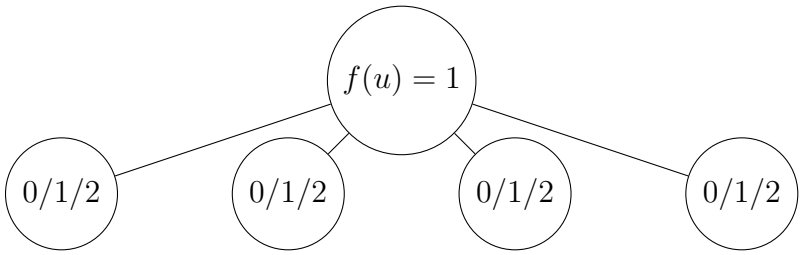

Figure 3.11: $dp_1[u]$

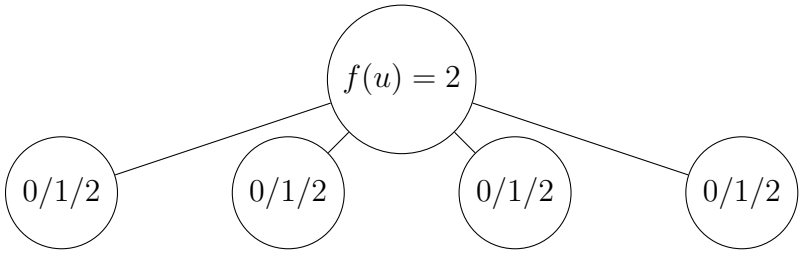

Figure 3.12: $dp_2[u]$

3.2.1 Algorithm

Theorem 3.2.2. *Let $T = (V, E)$ be a tree with a root $r \in V$. Let $u \in V$ be a non-leaf vertex of the tree. Then the following statements hold:*

- $dp_1[u] = 1 \times w(u) + \sum_{c \in C(u)} \min\{dp_0^1[c], dp_1[c], dp_2[c]\}$
- $dp_2[u] = 2 \times w(u) + \sum_{c \in C(u)} \min\{dp_0^0[c], dp_1[c], dp_2[c]\}$
- $dp_0^0[u] = \sum_{c \in C(u)} \min\{dp_2[c], dp_1[c], dp_0^2[c]\}$
- $dp_0^1[u] = \min_{c \in C(u)} \{dp_1[c] + \sum_{c' \in C(u) \setminus \{c\}} \min\{dp_2[c'], dp_1[c'], dp_0^2[c']\}\}$
- $dp_0^2[u] = \min\{\alpha, \beta\}$ *where*

  $\alpha = \min_{c_1, c_2 \in C(u)} \{dp_1[c_1] + dp_1[c_2] + \sum_{c' \in C(u) \setminus \{c_1, c_2\}} \min\{dp_2[c'], dp_1[c'], dp_0^2[c']\}\}$

  $\beta = \min_{c \in C(u)} \{dp_2[c] + \sum_{c' \in C(u) \setminus \{c\}} \min\{dp_2[c'], dp_1[c'], dp_0^2[c']\}\}$

Proof. The proofs of the above statements are as follows:

- For a vertex u with $f(u) = 1$, there is no restriction for the values that's it's children can be assigned by the IDF. Hence, for a subtree rooted at u, it's Italian Domination Number(IDN) will be equal to the sum of the IDNs of the subtrees formed by each of it's children added to the contribution at vertex u, i.e. $w(u) \times 1$. Each of it's children can be assigned the value $\{0, 1, 2\}$ by the IDF, hence we take the minimum of these 3 cases, for each subtree. Also, if one of the children of vertex u, let's say c, is assigned the value 0 by the IDF, then for the vertex c to satisfy the IDF condition, $\Sigma_{v \in C(c)} f(v) \geq 1$. Hence, we consider the case of $dp_0^1[c]$, while calculating the value of $dp_1[u]$.

- This case is similar to the previous case, except that for a vertex u with $f(u) = 2$, it's contribution to the IDN is $w(u) \times 2$. Each of it's children can be assigned the value $\{0, 1, 2\}$ by the IDF, hence we take the minimum of these 3 cases, for each subtree. Also, for a vertex u with $f(u) = 2$, if it has a child c such that $f(c) = 0$, then for the vertex c to satisfy the IDF condition, $\Sigma_{v \in C(c)} f(v) \geq 0$. Hence, we consider the case of $dp_0^0[c]$ while calculating the value of $dp_2[u]$.

- In this case $dp_0^0[u]$, the vertex u is assigned the value 0 by the IDF, and $\Sigma_{c \in C(u)} f(c) \geq 0$. Hence, for every $c \in C(u)$, the IDF can assign c any of the values $\{0, 1, 2\}$. For the vertex u, if it has a child $c \in C(u)$, such that $f(c) = 0$, then to satisfy the IDF condition for vertex c, $\Sigma_{v \in C(c)} f(v) \geq 2$. So, we consider the case of $dp_0^2[c]$ for the vertex c. Hence, to calculate the value of $dp_0^0[u]$, for each child c we take the minimum of the cases when $f(c) = 0$, $f(c) = 1$, and $f(c) = 2$.

- In the case of $dp_0^1[u]$, the vertex u is assigned the value 0 by the IDF, and $\Sigma_{c \in C(u)} f(c) \geq 1$. Hence, for one vertex in $c \in C(u)$, the IDF assigns the value 1, while for the remaining children, it can assign any of the values $\{0, 1, 2\}$. Similar to the above point, if for a vertex $c \in C(u)$, if $f(c) = 0$, then we consider the case of $dp_0^2[c]$ for that vertex. For calculating the value of $dp_0^1[u]$, we take a minimum of all the cases where one vertex is assigned 1 by the IDF, while the remaining vertices are assigned either 0, 1 or 2.

- In the case of $dp_0^2[u]$, the vertex u is assigned the value 0 by the IDF, and $\Sigma_{c \in C(u)} f(c) \geq 2$. Similar to the above point, if for a vertex $c \in C(u)$,

if $f(c) = 0$, then we consider the case of $dp_0^2[c]$ for that vertex. Now, if $\Sigma_{c \in C(u)} f(c) \geq 2$, then we have two cases. The first case is that the IDF assigns one vertex $c \in C(u)$ to the value 2, and then the remaining vertices are assigned any of the values $\{0, 1, 2\}$. The other case is that two of the vertices $c_1, c_2 \in C(u)$ are assigned the value 1, and the remaining vertices are assigned any of the values $\{0, 1, 2\}$. We calculate the possible IDN for the subtree rooted at u in both these cases, and then take the value of $dp_0^2[u]$ to be the minimum of these two cases.

□

Lemma 3.2.3. *For a non-leaf vertex $u \in V$, we can write*

$$dp_0^1[u] = s1 + dp_0^0[u]$$

where $s1 = \min_{c \in C(u)} \{dp_1[c] - \min\{dp_2[c], dp_1[c], dp_0^2[c]\}\}$

Proof. The proof is similar to the proof of Lemma 3.1.3 □

Lemma 3.2.4. *For a non-leaf vertex $u \in V$, we can write*

$$\alpha = s1 + s2 + dp_0^0[u]$$

where,

$$\alpha = \min_{c_1, c_2 \in C(u)} \{dp_1[c_1] + dp_1[c_2] + \sum_{c' \in C(u) \setminus \{c_1, c_2\}} \min\{dp_2[c'], dp_1[c'], dp_0^2[c']\}\}$$
$$s1 = \min_{c_1 \in C(u)} \{dp_1[c_1] - \min\{dp_2[c], dp_1[c], dp_0^2[c]\}\}$$
$$s2 = \min_{c_2 \in C(u) \setminus \{c_1\}} \{dp_1[c_2] - \min\{dp_2[c], dp_1[c], dp_0^2[c]\}\}$$

Proof. The proof is similar to the proof of Lemma 3.1.4 □

Lemma 3.2.5. *For a non-leaf vertex $u \in V$ we can write*

$$\beta = s3 + dp_0^0[u]$$

where,

$$\beta = \min_{c \in C(u)} \{dp_2[c] + \sum_{c' \in C(u) \setminus \{c\}} \min\{dp_2[c'], dp_1[c'], dp_0^2[c']\}\}$$
$$s3 = \min_{c \in C(u)} \{dp_2[c] - \min\{dp_2[c], dp_1[c], dp_0^2[c]\}\}$$

Proof. The proof is similar to the proof of Lemma 3.1.5 □

Algorithm 7 Italian Domination in weighted trees

Input A tree T fixed at a root r
Output Italian Domination number γ_{IW} of T

function ID(T,u)
  $dp_0^0[u] \leftarrow 0$
  $dp_0^1[u] \leftarrow \infty$
  $dp_0^2[u] \leftarrow \infty$
  $dp_1[u] \leftarrow 1 \times w(u)$
  $dp_2[u] \leftarrow 2 \times w(u)$
  $s1 \leftarrow \infty$
  $s2 \leftarrow \infty$
  $s3 \leftarrow \infty$
  for each $c \in T.C(u)$ **do** ▷ Depth First Search Traversal of Tree
    PID(T,c)
    $dp_0^0[u] \leftarrow dp_0^0[u] + \min\{dp_2[c], dp_1[c], dp_0^2[c]\}$
    $temp \leftarrow s1$
    $s1 \leftarrow \min\{s1, dp_1[c] - \min\{dp_2[c], dp_1[c], dp_0^2[c]\}\}$
    if $s1 == (dp_1[c] - \min\{dp_2[c], dp_1[c], dp_0^2[c]\})$ **then**
      $s2 \leftarrow temp$
    else
      $s2 \leftarrow \min\{s2, dp_1[c] - \min\{dp_2[c], dp_1[c], dp_0^2[c]\}\}$

```
        s3 ← min{s3, dp2[c] − min{dp2[c], dp1[c], dp0^2[c]}}
        dp1[u] ← dp1[u] + min{dp1[c], dp2[c], dp0^1[c]}
        dp2[u] ← dp2[u] + min{dp1[c], dp2[c], dp0^0[c]}
    if u is non-leaf then                          ▷ All nodes other than leaf
        dp0^1[u] ← s1 + dp0^0[u]
        α ← s1 + s2 + dp0^0[u]
        β ← s3 + dp0^0[u]
        dp0^2[u] = min{α, β}

function PRINT-ID(T,r)                             ▷ Let r be the root of the tree T
    ID(T,r)
    return min{dp0^2[r], dp1[r], dp2[r]}
```

3.2.2 Time and Space Complexity Analysis

Lemma 3.2.6. *For a tree T with n vertices, the time and space complexities of Algorithm 6 are $O(n)$ and $O(n)$ respectively.*

Proof. The time complexity of Algorithm 6 is $O(n)$ as we are using Depth First Search (DFS) method to traverse the tree which takes $O(n)$ time. We visit each vertex exactly once in this algorithm, and as the number of vertices are n, the total time complexity is $O(n)$. The space complexity is $O(n)$ since we create 5 arrays, $dp_0^0, dp_0^1, dp_0^2, dp_1, dp_2$ of size n each, and a three variables, $s1, s2, s3$. □

3.2.3 Proof of Correctness

Lemma 3.2.7. *Algorithm 6 computes Italian Domination number(IDN) of a given tree $T = (V, E)$ correctly.*

Proof. We use mathematical induction to prove the correctness of algorithm.

We show that Algorithm 6 computes the Italian domination number correctly for all trees of height $h \in \mathbb{Z}^+$.

- Base Case: For $h = 1$,

  The tree contains a single node. Let us denote it by u. Now,
  $dp_0^0[u] = 0,\ dp_0^1[u] = \infty,\ dp_0^2[u] = \infty,\ dp_1[u] = 1 \times w(u),\ dp_2[u] = 2 \times w(u)$
  $\gamma_{IW} = \min\{dp_0^2[u], dp_1[u], dp_2[u]\} = \min\{\infty, 1 \times w(u), 2 \times w(u)\} = 1 \times w(u)$.

- Induction Hypothesis: Let the correctness of the algorithm hold for trees of height $h'(< h)$.

- Induction Step: Let T be a tree of height h and let us fix a vertex r to be the root of T. Let us have a weight function $w : V(T) \to R$ for the vertices of the tree T. For all $c \in C(r)$, the height of the subtree rooted at c is h' which is less than h. So, we have optimal values of $dp_0^0[c], dp_0^1[c], dp_0^2[c], dp_1[c], dp_2[c]$ for all $c \in C(r)$. Therefore, from Theorem 3.2.2, we have the optimal values of $dp_0^0[r], dp_0^1[r], dp_0^2[r], dp_1[r], dp_2[r]$. Now, the Italian domination number of the weighted tree will be $\gamma_{IW}(T) = \min\{dp_0^2[r], dp_1[r], dp_2[r]\}$. This is because, from the IDF, the vertex r can be assigned the value either 0 or 1 or 2, and if $f(r) = 0$, it is necessary that for sum of its children $\Sigma_{c \in C(r)} f(c) = 2$ to satisfy the IDF condition, as r does not have any parent vertex. Hence, we see that the induction hypothesis is true for all trees of height h.

Hence, we see that Algorithm 6 computes the Italian domination number correctly for all trees of height $h \in \mathbb{Z}^+$. □

Chapter 4

Roman Domination in Weighted Block Graphs

A graph $G = (V, E)$ is said to be a block graph, if every biconnected component, i.e., a block in the graph G is a clique.

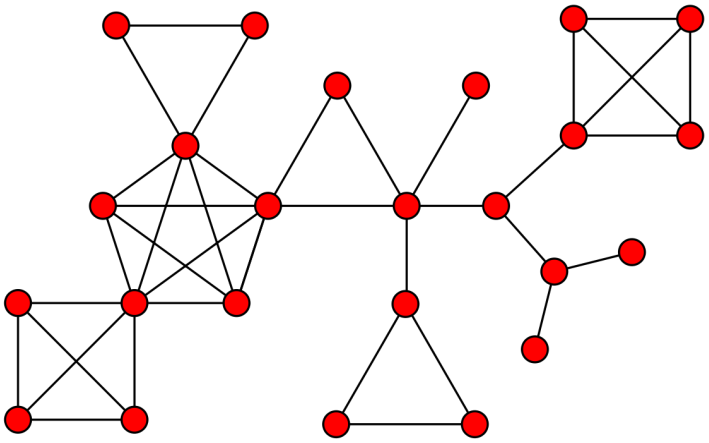

Figure 4.1: An example of block graph

Let us now go through some terminologies related to blocks and block graphs. A vertex x of G is said to be a cut-vertex, if removing the vertex x increases the number of connected components in the graph. A block in a graph G is a maximal biconnected subgraph of G. We can see that, in a block graph G, two blocks share at most one vertex, which is a cut-vertex of G. A block that contains only one cut-vertex, is called as a end block of G. We now define the block cut-point tree of a given block graph G.

Definition 4.0.1. A block cut-point tree of a given block graph $G = (V, E)$ is a bipartite tree $T(G) = (V', E')$ in which one set consists of the vertices b_i corresponding to each block B_i in G, and the other set consists of the cut-vertices of G, and the set of edges E' where $vb_i \in E'$, if and only if $v \in B_i$, where v is a cut-vertex, and b_i is a block-vertex in $T(G)$ corresponding to a block in G.

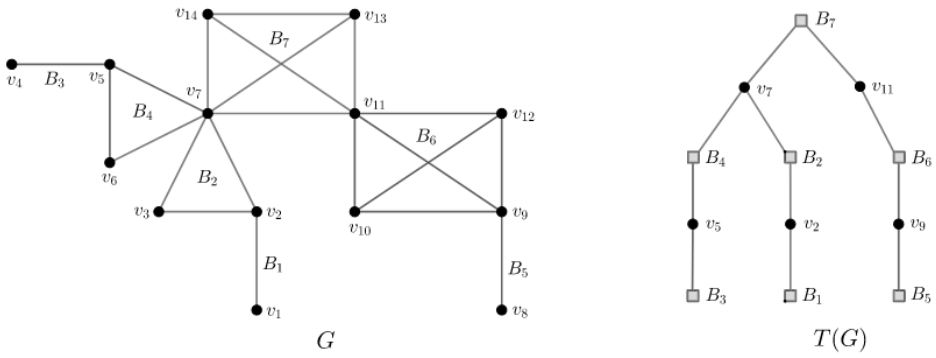

Figure 4.2: A block graph G and corresponding block cut-tree $T(G)$

4.1 Independent Roman Domination in Weighted Block Graphs

Definition 4.1.1. An Independent Roman Dominating Function(IRDF) on any undirected graph $G = (V, E)$ is a Roman dominating function $f : V(G) \to \{0, 1, 2\}$ in which every vertex $u \in V$ for which $f(u) = 0$ is adjacent to at least one vertex $v \in V$ for which $f(v) = 2$, such that the set formed by the union of the vertices that are assigned the values 1 and 2, is an independent set. That is, let $V_i = \{v \in V : f(v) = i\}$, $i = 0, 1, 2$. Then, we call the function f as an Independent roman dominating function if and only if $V_1 \cup V_2$ is an independent set.

Let us consider a weighted block graph $H = (V, E)$ with a weight function $w : V(H) \to \mathbb{R}$, which assigns a weight to each vertex of the block graph. Now, in this case, we define the weight of an independent roman dominating function f to be $f(V(H)) = \Sigma_{u \in V(H)} w(u) \times f(u)$, where $f : V(H) \to \{0, 1, 2\}$ is an Independent Roman dominating function on the weighted block graph H. The minimum weight of an Independent Roman dominating function on a weighted block graph H is called the Independent Roman Domination Number(IRDN) of H and is denoted by $\gamma^I_{RW}(H)$.

We now present an algorithm to find the Independent Roman Domination number of a connected block graph, $H = (V, E)$. The algorithm runs in $O(|V| + |E|)$ time.

For a graph H, let $F(H)$ be the set of all the functions which assign a value from $\{0, 1, 2\}$ to every vertex in H. Now, let us define the following sets corresponding to the graph H, and a specific vertex $v \in V(H)$:

- $F_0^2(v, H) = \{f \in F(H)$: f is an IRDF and $f(v) = 0\}$
- $F_0^0(v, H) = \{f \in F(H)$: $f_{H\setminus v}$ is an IRDF on $H - v$,$f(v) = 0$, and there is no $v\prime \in N_H(v)$ such that $f(v\prime) = 2\}$
- $F_1(v, H) = \{f \in F(H)$: f is an IRDF and $f(v) = 1\}$
- $F_2(v, H) = \{f \in F(H)$: f is an IRDF and $f(v) = 2\}$

Now, let us denote,

- $dp_0^2(v, H) = \min\{w(f) : f \in F_0^2(v, H)\}$
- $dp_0^0(v, H) = \min\{w(f) : f \in F_0^0(v, H)\}$
- $dp_1(v, H) = \min\{w(f) : f \in F_1(v, H)\}$
- $dp_2(v, H) = \min\{w(f) : f \in F_2(v, H)\}$

From this, we can see that for any arbitrary block graph H and $v \in V(H)$, $\gamma_{RW}^I(H) = min\{dp_0^0(v, H), dp_1(v, H), dp_2(v, H)\}$. Now, for a graph having only one vertex, i.e. a trivial graph, $dp_0^2(v, \{v\}) = \infty, dp_0^0(v, \{v\}) = 0, dp_1(v, \{v\}) = 1 \times w(v), dp_2(v, \{v\}) = 2 \times w(v)$.

4.1.1 Algorithm

For our algorithm, we view a block graph H as a graph rooted at a specific vertex of it. This can be done by visualizing it from the perspective of it's corresponding block-cutpoint tree $T(H)$. Now, we construct a method such that any connected block graph can be be constructed by applying this method repeatedly, starting from the trivial graphs. Let $H_1, H_2, \ldots, H_n$,

$n \geq 2$ be graphs rooted at $v_1, v_2, \ldots, v_n$, such that $v_i \in H_i$, $1 \leq i \leq n$. Now, we call the graph H as the composition of the graphs $\{H_1, H_2, \ldots, H_n\}$, if H is obtained by adding edges to $v_1, v_2, \ldots, v_n$, such that $\{v_1, v_2, \ldots, v_n\}$ is a clique.

Theorem 4.1.2. *Let $H_1, H_2, \ldots, H_n$, $n \geq 2$ be graphs rooted at $v_1, v_2, \ldots, v_n$, such that $v_i \in H_i$, $1 \leq i \leq n$. And, H is the graph rooted at v_1 obtained from the composition of the graphs $H_1, H_2, \ldots, H_n$, as shown above. Then, the following equalities hold:*

1. $dp_1(v_1, H) = dp_1(v_1, H_1) + \sum\limits_{i \in \{2,\ldots,n\}} dp_0^2(v_i, H_i)$

2. $dp_2(v_1, H) = dp_2(v_1, H_1) + \sum\limits_{i \in \{2,\ldots,n\}} \min\{dp_0^0(v_i, H_i), dp_0^2(v_i, H_i)\}$

3. $$dp_0^0(v_1, H) = \min \begin{cases} dp_0^0(v_1, H_1) + \min\limits_{i \in \{2,\ldots,n\}} \{dp_1(v_i, H_i) + \sum\limits_{j \in \{2,\ldots,n\} \setminus \{i\}} dp_0^2(v_j, H_j)\} \\ dp_0^0(v_1, H_1) + \sum\limits_{i \in \{2,\ldots,n\}}^{n} dp_0^2(v_i, H_i) \end{cases}$$

4. $$dp_0^2(v_1, H) = \min \begin{cases} dp_0^2(v_1, H_1) + \sum\limits_{i \in \{2,\ldots,n\}} dp_0^2(v_i, h_i) \\ dp_0^2(v_1, H_1) + \min\limits_{i \in \{2,\ldots,n\}} \{dp_1(v_i, H_i) + \sum\limits_{j \in \{2,\ldots,n\} \setminus \{i\}} dp_0^2(v_j, H_j)\} \\ dp_0^2(v_1, H_1) + A \\ dp_0^0(v_1, H_1) + A \end{cases}$$

   where $A = \min\limits_{i \in \{2,\ldots,n\}} \{dp_2(v_i, H_i) + \sum\limits_{j \in \{2,\ldots,n\} \setminus \{i\}} \min\{dp_0^0(v_j, H_j), dp_0^2(v_j, H_j)\}\}$

Proof. 1. Let $f \in F_1(v_1, H)$. Now, as $f(v_1) = 1$, from the condition of Independent Roman Domination, no $v \in \{v_2, \ldots, v_n\}$ must be assigned the value 1 or 2. Hence, $f(v_i) = 0$, for all $i \in \{2, \ldots, n\}$. Let f_{H_i} be

the restriction of f on H_i. As there is no $v \in \{v_1, \ldots, v_n\} \setminus \{v_i\}$ with $f(v) = 2$, if $f(v_i) = f_{H_r}(v_i) = 0$, then the roman domination condition for v_i must be satisfied by H_i itself. Hence, $f_{H_i} \in F_0^2(v_i, H_i)$.

2. Let $f \in F_2(v_1, H)$. Now, as $f(v_1) = 2$, from the condition of Independent Roman Domination, no $v \in \{v_2, \ldots, v_n\}$ must be assigned the value 1 or 2. Hence, $f(v_i) = 0$, for all $i \in \{2, \ldots, n\}$. Let f_{H_i} be the restriction of f on H_i. As $f(v_1) = 2$, for each $i \in \{2, \ldots, n\}$, $f_{H_i} \in F_0^0(v_i, H_i) \cup F_0^2(v_i, H_i)$. Hence, we take the minimum of these two cases.

3. Let $f \in F_0^0(v_1, H)$ and $f(v_1) = 0$. From the definition of $F_0^0(v_1, H)$, we see that there exists no $v \in \{v_2, \ldots, v_n\}$ such that $f(v) = 2$. Also, if there exists a $v \in \{v_2, \ldots, v_n\}$ with $f(v) = 1$, then from the condition of Independent Roman Domination, for all $u \in \{v_2, \ldots, v_n\} \setminus \{v\}$, $f(u) = 0$. Hence, there can be at most one vertex $v \in \{v_2, \ldots, v_n\}$ with $f(v) = 1$. Let f_{H_i} be the restriction of f on H_i. If $f(v_i) = 1$, then $f_{H_i} \in F_1(v_i, H_i)$, and for all $j \in \{2, \ldots, n\} \setminus \{i\}$, $f_{H_j} \in F_0^2(v_j, H_j)$. If $f(v_i) = 0$ for all $i \in \{2, \ldots, n\}$, then $f_{H_i} \in F_0^2(v_i, H_i)$.

4. Let $f \in F_0^2(v_1, H)$ and $f(v_1) = 0$. Let f_{H_1} be the restriction of f on H_1. As $f(v_1) = 0$, $f_{H_1} \in F_0^0(v_1, H_1) \cap F_0^2(v_1, H_1)$.

    - **Case 1:** $f_{H_1} \in F_0^0(v_1, H_1)$. Now, from the condition of Independent Roman Domination, exactly one $i \in \{2, \ldots, n\}$ exists such that $f(v_i) = 2$, and for all $j \in \{2, \ldots, n\} \setminus \{i\}$, $f(v_j) = 0$. Hence, $f_{H_i} \in F_2(v_i, H_i)$ and $f_{H_j} \in F_0^0(v_j, H_j) \cup F_0^2(v_j, H_j)$.

- **Case 2:** $f_{H_1} \in F_0^2(v_1, H_1)$. Now, from the condition of Independent Roman Domination, either for all the vertices $v \in \{v_2, \ldots, v_n\}$, $f(v) = 0$, or there exists exactly one vertex $v \in \{v_2, \ldots, v_n\}$ with either $f(v) = 1$ or $f(v) = 2$ and the remaining vertices $u \in \{v_2, \ldots, v_n\} \setminus \{v\}$ with $f(u) = 0$. If $f(v_i) = 0$ for all $i \in \{2, \ldots, n\}$, then $f_{H_i} \in F_0^2(v_i, H_i)$. If for some $i \in \{2, \ldots, n\}$, $f(i) = 1$, and for all $j \in \{2, \ldots, n\} \setminus \{i\}$, $f(v_j) = 0$, then $F_{H_i} \in F_1(v_i, H_i)$ and $F_{H_j} \in F_0^2(v_j, H_j)$. If for some $i \in \{2, \ldots, n\}$, $f(i) = 2$, and for all $j \in \{2, \ldots, n\} \setminus \{i\}$, $f(v_j) = 0$, then $F_{H_i} \in F_2(v_i, H_i)$. As $f(v_i) = 2$, $F_{H_j} \in F_0^2(v_j, H_j) \cap F_0^2(v_j, H_j)$ for all $j \in \{2, \ldots, n\} \setminus \{i\}$, as the Roman Domination condition for v_j is already satisfied by v_i.

Finally, we take the minimum of all these cases to establish the equality.

□

Algorithm 8 Independent Roman Domination in weighted block graphs

Input A connected block graph G

Output Independent Roman Domination number $\gamma^I_{RW}(G)$

$G_1 \leftarrow G$
$S \leftarrow \emptyset$
for $v \in V(G)$ **do**
    $dp^2_0(v, G[\{v\}]) \leftarrow \infty$
    $dp^0_0(v, G[\{v\}]) \leftarrow 0$
    $dp_1(v, G[\{v\}]) \leftarrow 1 \times w(v)$
    $dp_2(v, G[\{v\}]) \leftarrow 2 \times w(v)$

while G_1 has more than one vertex **do**
    Choose a block B with at most one cut vertex in G_1, where $V(B) =$
    $\{v_1, v_2, ..., v_k\}$ and v_1 is the only cut vertex if B has a cut vertex
    $S \leftarrow S \cup V(B)$
    Let H_i be the component in $G[S] - E(B)$ such that $V(B) \cap V(H_i) = v_i$
    for each $1 \leq i \leq k$
    Let H be the graph obtained from the disjoint union of $H_1, H_2, ..., H_k$
    by adding edges to make $\{v_1, v_2, ..., v_k\}$ a clique in H.
    Obtain the values of $dp^2_0(v_1, H), dp^0_0(v_1, H), dp_1(v_1, H), dp_2(v_1, H)$ by
    using 4.2.2
    $G_1 \leftarrow G_1 - \{v_2, v_3, ..., v_k\}$

$\gamma^I_{RW}(G) = min\{dp^0_2(v_r, G), dp_1(v_r, G), dp_2(v_r, G)\}$ where v_r is the only vertex in G_1
return $\gamma^I_{RW}(G)$

4.1.2 Time Complexity Analysis

Lemma 4.1.3. *Let B_i be the block considered at i^{th} iteration of the while loop and $V(B_i) = \{v_1, v_2, ..., v_k\}$. Computation of dp_0^0 takes $O(|V(B_i)|)$ time at the i^{th} iteration.*

Proof.

$$dp_0^0(v_1, H) = \min \begin{cases} \mathrm{dp}_0^0(v_1, H_1) + \min\limits_{i \in \{2,...,k\}} \{dp_1(v_i, H_i) + \sum\limits_{j \in \{2,...,k\} \setminus \{i\}} dp_0^2(v_j, H_j)\} \\ dp_0^0(v_1, H_1) + \sum\limits_{i \in \{2,...,k\}}^{n} dp_0^2(v_i, H_i) \end{cases}$$

We can modify case 1 as $dp_0^0(v_1, H_1) + \sum\limits_{j \in \{2,...,k\}} dp_0^2(v_j, H_j) + \min\limits_{i \in \{2,...,k\}} \{dp_1(v_i, H_i) - dp_0^2(v_i, H_i)\}$. Here, $\sum\limits_{j \in \{2,...,k\}} dp_0^2(v_j, H_j)$ and $\min\limits_{i \in \{2,...,k\}} \{dp_1(v_i, H_i) - dp_0^2(v_i, H_i)\}$ both take $O(|V(B_i)|)$ time for computation. Clearly case 2 takes $O(|V(B_i)|)$ time. So, $dp_0^0(v_1, H)$ can be computed in $O(|V(B_i)|)$ time. Similarly, we can prove for dp_0^2, dp_1, dp_2 takes $O(|V(B_i)|)$ time at the i^{th} iteration. □

Lemma 4.1.4. *For a graph G with n vertices and m edges, the time complexity of Algorithm 8 is $O(n + m)$.*

Proof. Let $\{B_1, B_2, .., B_r\}$ be the set of blocks and $\{c_1, c_2, .., c_r\}$ be the set of cut vertices of G. Assigning initial values $dp_0^2(v, \{v\}), dp_0^0(v, \{v\}), dp_1(v, \{v\}), dp_2(v, \{v\})$ in Algorithm 8 takes $O(n)$ time. Let $\{B_1, B_2, .., B_r\}$ be the inverted order of the order obtained by doing a breadth-first search traversal on the block cut-point tree of G. Breath-first search traversal takes $O(n+m)$ and construction of cut-point tree takes $O(n+m)$. We traverse using the order obtained in each iteration of while loop we choose a block with at most one cut-vertex in G_1. At each while loop iteration $dp_0^2, dp_0^0, dp_1, dp_2$ are calculated. From 4.1.3 we

know that computation of $dp_0^2, dp_0^0, dp_1, dp_2$ take $O(|V(B_i)|)$ time. We visit every block exactly only once by the while loop. Hence, time complexity of algorithm 8 is $O(n+m)$. □

4.2 Total Roman Domination in Weighted Block Graphs

Definition 4.2.1. A total Roman Dominating Function(TRDF) on an undirected graph $G = (V, E)$ is a function $f : V(G) \to \{0, 1, 2\}$ satisfying the following conditions: (i) every vertex $u \in V$ for which $f(u) = 0$ is adjacent to at least one vertex $v \in V$ for which $f(v) = 2$ and (ii) the subgraph of G induced by the set of all vertices of positive assignment has no isolated vertices. The weight of a total Roman dominating function f is the value $f(V(G)) = \Sigma_{u \in V(G)} f(u)$. The minimum weight of a Roman dominating function on a graph G is called the Roman domination number $\gamma_{tR}(G)$ of G.

Let us consider a weighted block graph $H = (V, E)$ with a weight function $w : V(H) \to \mathbb{R}$, which assigns a weight to each vertex of the block graph. Now, in this case, we define the weight of an total roman dominating function f to be $f(V(H)) = \Sigma_{u \in V(H)} w(u) \times f(u)$, where $f : V(H) \to \{0, 1, 2\}$ is an total Roman dominating function on the weighted block graph H. The minimum weight of an total Roman dominating function on a weighted block graph H is called the total Roman Domination Number(TRDN) of H and is denoted by $\gamma_{tRW}(H)$.

We now present an algorithm to find the Independent Roman Domination number of a connected block graph, $H = (V, E)$. The algorithm runs in

$O(|V| + |E|)$ time.

For a graph H, let $F(H)$ be the set of all the functions which assign a value from $\{0, 1, 2\}$ to every vertex in H. Now, let us define the following sets corresponding to the graph H, and a specific vertex $v \in V(H)$:

- $F_0^{0,1}(v, H) = \{f \in F(H)$: f_{H-v} is an TRDF and $f(v) = 0, \quad \forall\ v\prime \in N_H(v)\ \ f(v\prime) \in \{0, 1\}\}$
- $F_0^2(v, H) = \{f \in F(H)$: f_H is an TRDF and $f(v) = 0$, for $v\prime \in N_H(v)\ \ f(v\prime) = 2$, and $f(x) \in \{0, 1, 2\}\ \ \forall\ x \in N_H(v) \setminus v\prime\}$
- $F_1^0(v, H) = \{f \in F(H)$: f_{H-v} is an TRDF and $f(v) = 1, \quad \forall\ v\prime \in N_H(v)\ \ f(v\prime) = 0\}$
- $F_1^{1,2}(v, H) = \{f \in F(H)$: f_H is an TRDF and $f(v) = 1$, for $v\prime \in N_H(v)\ \ f(v\prime) \in \{1, 2\}$, and $f(x) \in \{0, 1, 2\}\ \ \forall\ x \in N_H(v) \setminus v\prime\}$
- $F_2^0(v, H) = \{f \in F(H)$: f_{H-v} is an TRDF and $f(v) = 2, \quad \forall\ v\prime \in N_H(v)\ \ f(v\prime) = 0\}$
- $F_2^{1,2}(v, H) = \{f \in F(H)$: f_H is an TRDF and $f(v) = 2$, for $v\prime \in N_H(v)\ \ f(v\prime) \in \{1, 2\}$, and $f(x) \in \{0, 1, 2\}\ \ \forall\ x \in N_H(v) \setminus v\prime\}$

Now, let us denote,

- $dp_0^{0,1}(v, H) = min\{w(f) : f \in F_0^{0,1}(v, H)\}$
- $dp_0^2(v, H) = min\{w(f) : f \in F_0^2(v, H)\}$
- $dp_1^0(v, H) = min\{w(f) : f \in F_1^0(v, H)\}$
- $dp_1^{1,2}(v, H) = min\{w(f) : f \in F_1^{1,2}(v, H)\}$

- $dp_2^0(v, H) = min\{w(f) : f \in F_2^0(v, H)\}$
- $dp_2^{1,2}(v, H) = min\{w(f) : f \in F_2^{1,2}(v, H)\}$

From this, we can see that for any arbitrary block graph H and $v \in V(H)$, $\gamma_{tRW}(H) = min\{dp_0^2(v, H), dp_1^{1,2}(v, H), dp_2^{1,2}(v, H)\}$. Now, for a graph having only one vertex, i.e. a trivial graph, $dp_0^{0,1}(v, \{v\}) = 0$, $dp_0^2(v, \{v\}) = \infty$, $dp_1^0(v, \{v\}) = 1 \times w(v)$, $dp_1^{1,2}(v, \{v\}) = \infty$, $dp_2^0(v, \{v\}) = 2 \times w(v)$ and $dp_2^{1,2}(v, \{v\}) = \infty$.

4.2.1 Algorithm

For our algorithm, we view a block graph H as a graph rooted at a specific vertex of it. This can be done by visualizing it from the perspective of it's corresponding block-cutpoint tree $T(H)$. Now, we construct a method such that any connected block graph can be be constructed by applying this method repeatedly, starting from the trivial graphs. Let $H_1, H_2, \ldots, H_n$, $n \geq 2$ be graphs rooted at $v_1, v_2, \ldots, v_n$, such that $v_i \in H_i$, $1 \leq i \leq n$. Now, we call the graph H as the composition of the graphs $\{H_1, H_2, \ldots, H_n\}$, if H is obtained by adding edges to $v_1, v_2, \ldots, v_n$, such that $\{v_1, v_2, \ldots, v_n\}$ is a clique.

Theorem 4.2.2. *Let $H_1, H_2, \ldots, H_n$, $n \geq 2$ be graphs rooted at $v_1, v_2, \ldots, v_n$, such that $v_i \in H_i$, $1 \leq i \leq n$. And, H is the graph rooted at v_1 obtained from the composition of the graphs $H_1, H_2, \ldots, H_n$, as shown above. Then, the following equalities hold:*

1. $dp_0^{0,1}(v_1, H) = dp_0^{0,1}(v_1, H_1) + \sum_{i \in \{2,\ldots,n\}} \min\{dp_0^2(v_i, H_i), dp_1^{1,2}(v_i, H_i)\}$

2. $dp_0^2(v_1, H) = \min \begin{cases} dp_0^2(v_1, H_1) + \sum\limits_{i \in \{2,\ldots,n\}} \min\{dp_0^2(v_i, H_i),\ dp_1^{1,2}(v_i, H_i),\ dp_2^{1,2}(v_i, H_i)\} \\ dp_0^{0,1}(v_1, H_1) + S_0 \end{cases}$

3. $dp_1^0(v_1, H) = dp_1^0(v_1, H_1) + \sum\limits_{i \in \{2,\ldots,n\}} dp_0^2(v_i, H_i)$

4. $dp_1^{1,2}(v_1, H) = \min \begin{cases} dp_1^{1,2}(v_1, H_1) + \sum\limits_{i \in \{2,\ldots,n\}} \min\{dp_0^2(v_i, H_i),\ A_i,\ B_i\} \\ dp_1^0(v_1, H_1) + S_1 \end{cases}$

5. $dp_2^0(v_1, H) = dp_2^0(v_1, H_1) + \sum\limits_{i \in \{2,\ldots,n\}} \min\{dp_0^2(v_i, H_i), dp_0^{0,1}(v_i, H_i)\}$

6. $dp_2^{1,2}(v_1, H) = \min \begin{cases} dp_2^{1,2}(v_1, H_1) + \sum\limits_{i \in \{2,\ldots,n\}} \min\{A_i,\ B_i,\ C_i\} \\ dp_2^0(v_1, H_1) + S_2 \end{cases}$

where $S_0 = \min\limits_{i \in \{2,\ldots,n\}} \{dp_2^{1,2}(v_i, H_i) + \sum\limits_{j \in \{2,\ldots,n\} \setminus \{i\}} \min\{dp_0^2(v_j, H_j), dp_1^{1,2}(v_j, H_j), dp_2^{1,2}(v_j, H_j)\}\}$,

$S_1 = \min\limits_{i \in \{2,\ldots,n\}} \{\min\{A_i,\ B_i\} + \sum\limits_{j \in \{2,\ldots,n\} \setminus \{i\}} \min\{dp_0^2(v_j, H_j), A_j, B_j\}\}$,

$S_2 = \min\limits_{i \in \{2,\ldots,n\}} \{\min\{A_i,\ B_i\} + \sum\limits_{j \in \{2,\ldots,n\} \setminus \{i\}} \min\{A_j, B_j, C_j\}\}$,

$A_r = \min\{dp_1^0(v_r, H_r), dp_1^{1,2}(v_r, H_r)\}$,

$B_r = \min\{dp_2^0(v_r, H_r), dp_2^{1,2}(v_r, H_r)\}$,

and $C_r = \min\{dp_0^{0,1}(v_r, H_r), dp_0^2(v_r, H_r)\}$

Proof. 1. Let $f \in F_0^{0,1}(v_1, H)$ and $f(v_1) = 0$. From the definition of $F_0^{0,1}(v_1, H)$, we see that there exists no $v_i \in \{v_2, \ldots, v_n\}$ such that $f(v_i) = 2$. Also, if there exists a $v_i \in \{v_2, \ldots, v_n\}$ with $f(v_i) = 1$, then from the second condition of total Roman Domination, for $v_j \in \{v_2, \ldots, v_n\} \setminus \{v_i\}$, $f(v_j) \in \{1, 2\}$. Else if $f(v_i) = 0$, then from the first condition of total Roman Domination, for $v_j \in \{v_2, \ldots, v_n\} \setminus \{v_i\}$,

$f(v_j) = 2$. Hence, we take minimum of both the above cases for all $v_j \in \{v_2, \ldots, v_n\} \setminus \{v_i\}$.

2. Let $f \in F_0^2(v_1, H)$ and $f(v_1) = 0$. Let f_{H_1} be the restriction of f on H_1. As $f(v_1) = 0$, $f_{H_1} \in F_0^{0,1}(v_1, H_1) \cap F_0^2(v_1, H_1)$.

   - **Case 1:** $f_{H_1} \in F_0^2(v_1, H_1)$. Now, from the first condition of total Roman Domination for $v_i \in \{v_2, \ldots, v_n\}$ if $f(v_i) = 0$ then at least one neighbour of v_i must be assigned value 2 by TRD function. If $f(v_i) = 1$ or $f(v_i) = 2$ from second condition of total Roman Domination at least one neighbour of v_i must be assigned values 1 or 2.
   - **Case 2:** $f_{H_1} \in F_0^{0,1}(v_1, H_1)$. In this case we assume $F_0^2(v_1, H)$ condition is satisfied by some $v_i \in \{v_2, \ldots, v_n\}$ i.e $f(v_i) = 2$ and for all $v_j \in \{v_2, \ldots, v_n\} \setminus \{v_i\}$ v_j can be assigned values 0, 1, 2 by TRD function.

   Finally, we take the minimum of all these cases to establish the equality.

3. Let $f \in F_1^0(v_1, H)$ and $f(v_1) = 1$. From the definition of $F_1^0(v_1, H)$, we see that there exists no $v_i \in \{v_2, \ldots, v_n\}$ such that $f(v) \in \{1, 2\}$. Hence, $f(v_i) = 0$, for all $i \in \{2, \ldots, n\}$. Then from the first condition of total Roman Domination, for all $v_i \in \{v_2, \ldots, v_n\}$ as $f(v_i) = 0$, then v_i must have a neighbour v_j except $\{v_2, \ldots, v_n\}$ such that $f(v_j) = 2$.

4. Let $f \in F_1^{1,2}(v_1, H)$ and $f(v_1) = 1$. Let f_{H_1} be the restriction of f on H_1. As $f(v_1) = 1$, $f_{H_1} \in F_1^0(v_1, H_1) \cap F_1^{1,2}(v_1, H_1)$.

- **Case 1:** $f_{H_1} \in F_1^{1,2}(v_1, H_1)$. Now, in this case we are considering second condition of total Roman Domination is satisfied by all the other neighbours of v_1 except $\{v_2, \ldots, v_n\}$. Hence for any $v_i \in \{v_2, \ldots, v_n\}$ v_i can ve assigned values 0, 1 and 2 by TRD function.

- **Case 2:** $f_{H_1} \in F_1^0(v_1, H_1)$. In this case we assume second total condition is satisfied by some $v_i \in \{v_2, \ldots, v_n\}$ i.e $f(v_i) \in \{1, 2\}$ and for all $v_j \in \{v_2, \ldots, v_n\} \setminus \{v_i\}$ v_j can be assigned values 0, 1 and 2 by TRD function.

Finally, we take the minimum of all these cases to establish the equality.

5. Let $f \in F_2^0(v_1, H)$ and $f(v_1) = 2$. From the definition of $F_1^0(v_1, H)$, we see that there exists no $v_i \in \{v_2, \ldots, v_n\}$ such that $f(v_i) \in \{1, 2\}$. Hence, $f(v_i) = 0$, for all $i \in \{2, \ldots, n\}$. Then for all $v_i \in \{v_2, \ldots, v_n\}$ as $f(v_i) = 0$, then neighbours of v_i can be $\{0, 1, 2\}$.

6. Let $f \in F_2^{1,2}(v_1, H)$ and $f(v_1) = 2$. Let f_{H_1} be the restriction of f on H_1. As $f(v_1) = 2$, $f_{H_1} \in F_2^0(v_1, H_1) \cap F_2^{1,2}(v_1, H_1)$.

   - **Case 1:** $f_{H_1} \in F_2^{1,2}(v_1, H_1)$. Now, in this case we are considering second condition of total Roman Domination is satisfied by all the other neighbours of v_1 except $\{v_2, \ldots, v_n\}$. Hence for any $v_i \in \{v_2, \ldots, v_n\}$ v_i can ve assigned values 0, 1 and 2 by TRD function.

   - **Case 2:** $f_{H_1} \in F_2^0(v_1, H_1)$. In this case we assume second total condition is satisfied by some $v_i \in \{v_2, \ldots, v_n\}$ i.e $f(v_i) \in \{1, 2\}$

and for all $v_j \in \{v_2, \ldots, v_n\} \setminus \{v_i\}$ v_j can be assigned values 0, 1 and 2 by TRD function.

Finally, we take the minimum of all these cases to establish the equality.

□

Algorithm 9 total Roman Domination in weighted block graphs

Input A connected block graph G

Output total Roman Domination number $\gamma_{tRW}(G)$

$G_1 \leftarrow G$
$S \leftarrow \emptyset$
for $v \in V(G)$ **do**
    $dp_0^{0,1}(v, G[\{v\}]) \leftarrow 0$
    $dp_0^{2}(v, G[\{v\}]) \leftarrow \infty$
    $dp_1^{0}(v, G[\{v\}]) \leftarrow 1 \times w(v)$
    $dp_1^{1,2}(v, G[\{v\}]) \leftarrow \infty$
    $dp_2^{0}(v, G[\{v\}]) \leftarrow 2 \times w(v)$
    $dp_2^{1,2}(v, G[\{v\}]) \leftarrow \infty$

while G_1 has more than one vertex **do**
    Choose a block B with at most one cut vertex in G_1, where $V(B) =$
    $\{v_1, v_2, ..., v_k\}$ and v_1 is the only cut vertex if B has a cut vertex
    $S \leftarrow S \cup V(B)$
    Let H_i be the component in $G[S] - E(B)$ such that $V(B) \cap V(H_i) = v_i$
    for each $1 \leq i \leq k$
    Let H be the graph obtained from the disjoint union of $H_1, H_2, ..., H_k$
    by adding edges to make $\{v_1, v_2, ..., v_k\}$ a clique in H.
    Obtain the values of $dp_0^{0,1}(v_1, H)$,
    $dp_0^{2}(v_1, H), dp_1^{0}(v_1, H), dp_1^{1,2}(v_1, H), dp_2^{0}(v_1, H), dp_2^{1,2}(v_1, H)$ by
    using 2.3.2
    $G_1 \leftarrow G_1 - \{v_2, v_3, ..., v_k\}$

$\gamma_{RW}^{I}(G) = min\{dp_0^{2}(v_r, G), dp_1^{1,2}(v_r, G), dp_2^{1,2}(v_r, G)\}$ where v_r is the only vertex in G_1
return $\gamma_{tRW}(G)$

4.2.2 Time Complexity Analysis

Lemma 4.2.3. *Let B_i be the block considered at i^{th} iteration of the while loop and $V(B_i) = \{v_1, v_2, ..., v_k\}$. Computation of dp_0^2 takes $O(|V(B_i)|)$ time at the i^{th} iteration.*

Proof. .

$$dp_0^2(v_1, H) = \min \begin{cases} \mathrm{dp}_0^2(v_1, H_1) + \sum\limits_{i \in \{2,...,n\}} \min\{dp_0^2(v_i, H_i),\ dp_1^{1,2}(v_i, H_i),\ dp_2^{1,2}(v_i, H_i)\} \\ dp_0^{0,1}(v_1, H_1) + S_0 \end{cases}$$

where $S_0 = \min\limits_{i \in \{2,...,n\}} \{dp_2^{1,2}(v_i, H_i) + \sum\limits_{j \in \{2,...,n\} \setminus \{i\}} \min\{dp_0^2(v_j, H_j), dp_1^{1,2}(v_j, H_j), dp_2^{1,2}(v_j, H_j)\}\}$

Clearly case 1 takes $O(|V(B_i)|)$ time. We can modify case 2 as

$dp_0^{0,1}(v_1, H_1) + \sum\limits_{i \in \{2,...,n\}} \min\{dp_0^2(v_i, H_i),\ dp_1^{1,2}(v_i, H_i),\ dp_2^{1,2}(v_i, H_i)\}$

$+ \min\limits_{i \in \{2,...,k\}} \{dp_2^{1,2}(v_i, H_i) - \min\{dp_0^2(v_i, H_i),\ dp_1^{1,2}(v_i, H_i),\ dp_2^{1,2}(v_i, H_i)\}\}$. Here, second and third term of the modified expression both take $O(|V(B_i)|)$ time for computation. So, $dp_0^0(v_1, H)$ can be computed in $O(|V(B_i)|)$ time. Similarly, we can prove for $dp_0^{0,1}, dp_1^0, dp_1^{1,2}, dp_2^0, dp_2^{1,2}$ takes $O(|V(B_i)|)$ time at the i^{th} iteration. □

Lemma 4.2.4. *For a graph G with n vertices and m edges, the time complexity of Algorithm 9 is $O(n + m)$.*

Proof. Let $\{B_1, B_2, .., B_r\}$ be the set of blocks and $\{c_1, c_2, .., c_r\}$ be the set of cut vertices of G. Assigning initial values $dp_0^{1,2}(v, \{v\}), dp_0^2(v, \{v\}), dp_1^0(v, \{v\}), dp_1^{1,2}(v, \{v\}), dp_2^0(v, \{v\}), dp_2^{1,2}(v, \{v\})$ in Algorithm 8 takes $O(n)$ time. Let $\{B_1, B_2, .., B_r\}$ be the inverted order of the order obtained by doing a breadth-first search traversal on the block cut-point tree of G. Breath-first search traversal takes $O(n + m)$ and construction of cut-point tree takes $O(n + m)$.

We traverse using the order obtained in each iteration of while loop we choose a block with at most one cut-vertex in G_1. At each while loop iteration $dp_0^{0,1}, dp_0^2, dp_1^0, dp_1^{1,2}, dp_2^0, dp_2^{1,2}$ are calculated. From 4.2.3 we know that computation of $dp_0^{0,1}, dp_0^2, dp_1^0, dp_1^{1,2}, dp_2^0, dp_2^{1,2}$ take $O(|V(B_i)|)$ time. We visit every block exactly only once by the while loop. Hence, time complexity of algorithm 8 is $O(n+m)$. □

Chapter 5

Italian Domination in Weighted Block Graphs

5.1 Perfect Italian Domination in Weighted Block Graphs

Definition 5.1.1. A Perfect Italian dominating function PIDF on an undirected graph $G = (V, E)$ is an Italian dominating function in which for every vertex $u \in V$ with $f(u) = 0$ it holds that $\Sigma_{v \in N(u)} f(v) = 2$ where $N(u) = \{v \in V | (u, v) \in E\}$. The minimum weight $\Sigma_{u \in V(G)} f(u)$ of a perfect Italian dominating function on a graph G is called the perfect Italian domination number $\gamma_I^P(G)$ of G.

Let us consider a weighted block graph $H = (V, E)$ with a weight function $w : V(H) \to \mathbb{R}$, which assigns a weight to each vertex of the block graph. Now, in this case, we define the weight of a perfect italian dominating func-

tion f to be $f(V(H)) = \Sigma_{u \in V(H)} w(u) \times f(u)$, where $f : V(H) \rightarrow \{0, 1, 2\}$ is a Perfect Italian Dominating Function on the weighted block graph H. The minimum weight of a perfect italian dominating function on a weighted block graph H is called the Perfect Italian Domination Number(IRDN) of H and is denoted by $\gamma_{IW}^{P}(H)$.

We now present an algorithm to find the Independent Roman Domination number of a connected block graph, $H = (V, E)$. The algorithm runs in $O(|V| + |E|)$ time.

For a graph H, let $F(H)$ be the set of all the functions which assign a value from $\{0, 1, 2\}$ to every vertex in H. Now, let us define the following sets corresponding to the graph H, and a specific vertex $v \in V(H)$:

- $F_0^0(v, H) = \{f \in F(H)$: $f_{H \setminus v}$ is a PIDF on $H \setminus v$, $f(v) = 0$, and $f(N_H(v)) = 0\}$
- $F_0^1(v, H) = \{f \in F(H)$: $f_{H \setminus v}$ is a PIDF on $H \setminus v$, $f(v) = 0$, and $f(N_H(v)) = 1\}$
- $F_0^2(v, H) = \{f \in F(H)$: f is a PIDF on H, $f(v) = 0$, and $f(N_H(v)) = 2\}$
- $F_1(v, H) = \{f \in F(H)$: f is a PIDF on H, $f(v) = 1\}$
- $F_2(v, H) = \{f \in F(H)$: f is a PIDF on H, $f(v) = 2\}$

Now, let us denote,

- $dp_0^0(v, H) = \min\{w(f) : f \in F_0^0(v, H)\}$
- $dp_0^1(v, H) = \min\{w(f) : f \in F_0^1(v, H)\}$

- $dp_0^2(v, H) = \min\{w(f) : f \in F_0^2(v, H)\}$
- $dp_1(v, H) = \min\{w(f) : f \in F_1(v, H)\}$
- $dp_2(v, H) = \min\{w(f) : f \in F_2(v, H)\}$

From this, we can see that for any arbitrary block graph H and $v \in V(H)$, $\gamma_{IW}^P(H) = min\{dp_0^0(v, H), dp_1(v, H), dp_2(v, H)\}$. Now, for a graph having only one vertex, i.e. a trivial graph, $dp_0^2(v, \{v\}) = \infty, dp_0^1(v, \{v\}) = \infty, dp_0^0(v, \{v\}) = 0, dp_1(v, \{v\}) = 1 \times w(v), dp_2(v, \{v\}) = 2 \times w(v)$.

5.1.1 Algorithm

For our algorithm, we view a block graph H as a graph rooted at a specific vertex of it. This can be done by visualizing it from the perspective of it's corresponding block-cutpoint tree $T(H)$. Now, we construct a method such that any connected block graph can be be constructed by applying this method repeatedly, starting from the trivial graphs. Let $H_1, H_2, \ldots, H_n$, $n \geq 2$ be graphs rooted at $v_1, v_2, \ldots, v_n$, such that $v_i \in H_i$, $1 \leq i \leq n$. Now, we call the graph H as the composition of the graphs $\{H_1, H_2, \ldots, H_n\}$, if H is obtained by adding edges to $v_1, v_2, \ldots, v_n$, such that $\{v_1, v_2, \ldots, v_n\}$ is a clique.

Theorem 5.1.2. *Let $H_1, H_2, \ldots, H_n$, $n \geq 2$ be graphs rooted at $v_1, v_2, \ldots, v_n$, such that $v_i \in H_i$, $1 \leq i \leq n$. And, H is the graph rooted at v_1 obtained from the composition of the graphs $H_1, H_2, \ldots, H_n$, as shown above. Then, the following equalities hold:*

1. $dp_0^0(v_1, H) = dp_0^0(v_1, H_1) + \sum_{i \in \{2,\ldots,n\}} dp_0^2(v_i, H_i)$

2. $dp_0^1(v_1, H) = \min \begin{cases} dp_0^1(v_1, H_1) + \sum\limits_{i \in \{2,\ldots,n\}} dp_0^2(v_i, H_i) \\ dp_0^0(v_1, H_1) + \min\limits_{i \in \{2,\ldots,n\}} \{dp_1(v_i, H_i) + \sum\limits_{j \in \{2,\ldots,n\} \setminus \{i\}} dp_0^1(v_j, H_j)\} \end{cases}$

3. $dp_0^2(v_1, H) = \min \begin{cases} dp_0^2(v_1, H_1) + \sum\limits_{i \in \{2,\ldots,n\}} dp_0^2(v_i, H_i) \\ dp_0^1(v_1, H_1) + \min\limits_{i \in \{2,\ldots,n\}} \{dp_1(v_i, H_i) + \sum\limits_{j \in \{2,\ldots,n\} \setminus \{i\}} dp_0^1(v_j, H_j)\} \\ dp_0^0(v_1, H_1) + \min\limits_{i \in \{2,\ldots,n\}} \{dp_2(v_i, H_i) + \sum\limits_{j \in \{2,\ldots,n\} \setminus \{i\}} dp_0^0(v_j, H_j)\} \\ dp_0^0(v_1, H_1) + A \end{cases}$

   where $A = \min\limits_{i,j \in \{2,\ldots,n\}} \{dp_1(v_i, H_i) + dp_1(v_j, H_j) + \sum\limits_{k \in \{2,\ldots,n\} \setminus \{i,j\}} dp_0^0(v_k, H_k)\}$

4. $dp_1(v_1, H) = \min \begin{cases} dp_1(v_1, H_1) + \sum\limits_{i \in \{2,\ldots,n\}} \min\{dp_1(v_i, H_i), dp_2(v_i, H_i)\} \\ dp_1(v_1, H_1) + \min\limits_{i \in \{2,\ldots,n\}} \min\{dp_1(v_i, H_i) + \sum\limits_{j \in \{2,\ldots,n\} \setminus \{i\}} dp_0^0(v_j, H_j)\} \\ dp_1(v_1, H_1) + \sum\limits_{i \in \{2,\ldots,n\}} dp_0^1(v_i, H_i) \end{cases}$

5. $dp_2(v_1, H) = \min \begin{cases} dp_2(v_1, H_1) + \sum\limits_{i \in \{2,\ldots,n\}} \min\{dp_1(v_i, H_i), dp_2(v_i, H_i)\} \\ dp_2(v_1, H_1) + \sum\limits_{i \in \{2,\ldots,n\}} dp_0^0(v_i, H_i) \end{cases}$

Proof. 1. Let $f \in F_0^0(v_1, H)$ and $f(v_1) = 0$. Let f_{H_1} be the restriction of f on H_1. As $f \in F_0^0(v_1, H)$, $f_{H_1} \in F_)^0(v_1, H_1)$. Also, as $v \in \{v_2, \ldots, v_n\}$ are neighbors of v_1 in H, $f(v) = 0$ for all $v \in \{v_2, \ldots, v_n\}$. And for all $i \in \{2, \ldots, n\}$, $f_{H_i} \in F_0^2(v_i, H_i)$ to satisfy the Perfect Roman Domination condition.

2. Let $f \in F_0^1(v_1, H)$ and $f(v_1) = 0$. Let f_{H_1} be the restriction of f on

H_1. As $f \in F_0^1(v_1, H)$, $f_{H_1} \in F_0^0(v_1, H_1) \cup F_0^1(v_1, H_1)$.

- **Case 1:** If $f_{H_1} \in F_0^0(v_1, H_1)$, then, to satisfy the condition of $F_0^1(v_1, H)$, one of $\{v_2, \ldots, v_n\}$ must be assigned the value 1, while the rest are assigned the value 0. Hence, for an $i \in \{2, \ldots, n\}$, $f_{H_i} \in F_1(v_i, H_i)$ and for all $j \in \{2, \ldots, n\}$, $f_{H_j} \in F_0^1(v_j, H_j)$ inorder to satisfy the PRD condition.
- **Case 2:** If $f_{H_1} \in F_0^1(v_1, H_1)$, then as $\{v_2, \ldots, v_n\}$ are the neighbors of v_1, all of them have to be assigned 0 to satisfy the PRD condition for v_1. Hence, for $i \in \{2, \ldots, n\}$, $f_{H_i} \in F_0^2(v_i, H_i)$, to satisfy the PRD condition for v_i.

3. Let $f \in F_0^2(v_1, H)$ and $f(v_1) = 0$. Let f_{H_1} be the restriction of f on H_1. As $f \in F_0^2(v_1, H)$, $f_{H_1} \in F_0^0(v_1, H_1) \cup F_0^1(v_1, H_1) \cup F_0^2(v_1, H_1)$.

   - **Case 1:** If $f_{H_1} \in F_0^0(v_1, H_1)$, then, inorder to satisfy the PRD condition for v_1 in f, two neighbors of v_1 must be assigned 1, or one neighbor must be assigned 2, and the rest are assigned 0. In both the cases, for all the vertices $v_i \in \{v_2, \ldots, v_n\}$ which are assigned 0, $f_{H_i} \in F_0^0(v_i, H_i)$, inorder to satisfy the PRD condition. We then take the minimum of these 2 cases.
   - **Case 2:** If $f_{H_1} \in F_0^1(v_1, H_1)$, then, inorder to satisfy the PRD condition for v_1 in f, one neighbor of v_1 must be assigned 1, and the rest are assigned 0. For all the vertices $v_i \in \{v_2, \ldots, v_n\}$ which are assigned 0, $f_{H_i} \in F_0^0(v_i, H_i)$, inorder to satisfy the PRD condition.

- **Case 3:** If $f_{H_1} \in F_0^2(v_1, H_1)$, then all the neighbors $v \in \{v_1, \ldots, v_n\}$ of v_1 in H must be assigned the value 0 by f, inorder to satisfy the PRD condition. Then, for all the vertices $v_i \in \{v_1, \ldots, v_n\}$, $f_{H_i} \in F_0^2(v_i, H_i)$ to satisfy the PRD condition for f.

4. Let $f \in F_1(v_1, H)$ and $f(v_1) = 1$. So, the restriction of f on H_1, $f_{H_1} \in F_1(v_1, H_1)$. Then, we have the following cases:

   - **Case 1:** The rest of the vertices $v_i \in \{v_2, \ldots, v_n\}$ are either assigned 1 or 2. In this case, $f_{H_i} \in F_1(v_i, H_i)$ or $f_{H_i} \in F_2(v_i, H_i)$.
   - **Case 2:** If for some $v \in \{v_2, \ldots, v_n\}$, $f(v) = 0$, then at most one other vertex $v\prime \in \{v_2, \ldots, v_n\} \backslash \{v\}$ can be assigned the value 1, and the rest of the vertices should be assigned 0, to satisfy the PRD condition. If one vertex $v_i \in \{v_2, \ldots, v_n\}$ exists with $f(v_i) = 1$, then $f_{H_i} \in F_1 v_i, H_i$, and for all the vertices $v_j \in \{v_2, \ldots, v_n\} \setminus \{v_i\}$ $f(v_j) = 0$ and $f_{H_i} \in F_0^0(v_j, H_j)$ to satisfy the PRD condition for f.
   - **Case 3:** If all the vertices $v_i \in \{v_2, \ldots, v_n\}$ are assigned the value 0, then $f_{H_i} \in F_0^1(v_i, H_i)$, inorder to satisfy the PRD condition.

5. Let $f \in F_2(v_1, H))$ and $f(v_1) = 2$. So, the restriction of f on H_1, $f_{H_1} \in F_2(v_1, H_1)$. Then, we have the following cases:

   - **Case 1:** There exists at least one $v_i \in \{v_2, \ldots, v_n$ with $f(v_i) = 0$. Then, inorder to satisfy the PRD condition for v_i in f, all the other vertices $\{v_2, \ldots, v_n\} \setminus \{v_i\}$ should also be assigned 0. Hence, for all the vertices $v_i \in \{v_2, \ldots, v_n\}$, $f_{H_i} \in F_0^0(v_i, H_i)$.

- **Case 2:** For all the vertices $v_i \in \{v_2, \ldots, v_n$, $f(v_i) = 1$ or $f(v_i) = 2$. Hence, $f_{H_i} \in F_2(v_i, H_i)$ or $f_{H_i} \in F_2(v_i, H_i)$.

□

Algorithm 10 Perfect Italian Domination in weighted block graphs

Input A connected block graph G
Output Perfect Italian Domination number $\gamma^I_{RW}(G)$

$G_1 \leftarrow G$
$S \leftarrow \emptyset$
for $v \in V(G)$ **do**
    $dp^2_0(v, G[\{v\}]) \leftarrow \infty$
    $dp^1_0(v, G[\{v\}]) \leftarrow \infty$
    $dp^0_0(v, G[\{v\}]) \leftarrow 0$
    $dp_1(v, G[\{v\}]) \leftarrow 1 \times w(v)$
    $dp_2(v, G[\{v\}]) \leftarrow 2 \times w(v)$

while G_1 has more than one vertex **do**
    Choose a block B with at most one cut vertex in G_1, where $V(B) =$
    $\{v_1, v_2, ..., v_k\}$ and v_1 is the only cut vertex if B has a cut vertex
    $S \leftarrow S \cup V(B)$
    Let H_i be the component in $G[S] - E(B)$ such that $V(B) \cap V(H_i) = v_i$
    for each $1 \leq i \leq k$
    Let H be the graph obtained from the disjoint union of $H_1, H_2, ..., H_k$
    by adding edges to make $\{v_1, v_2, ..., v_k\}$ a clique in H.
    Obtain the values of $dp^0_0(v_1, H), dp^1_0(v_1, H), dp^2_0(v_1, H), dp_1(v_1, H), dp_2(v_1, H)$
    by using 5.1.2
    $G_1 \leftarrow G_1 - \{v_2, v_3, ..., v_k\}$

$\gamma^P_{IW}(G) = min\{dp^2_0(v_r, G), dp_1(v_r, G), dp_2(v_r, G)\}$ where v_r is the only vertex in G_1
return $\gamma^P_{IW}(G)$

5.1.2 Time Complexity Analysis

Lemma 5.1.3. *Let B_i be the block considered at i^{th} iteration of the while loop and $V(B_i) = \{v_1, v_2, ..., v_k\}$. Computation of dp_0^2 takes $O(|V(B_i)| + |E(B_i)|)$ time at the i^{th} iteration.*

Proof.

$$dp_0^2(v_1, H) = \min \begin{cases} \mathrm{dp}_0^2(v_1, H_1) + \sum\limits_{i \in \{2,...,n\}} dp_0^2(v_i, H_i) \\ \mathrm{dp}_0^1(v_1, H_1) + \min\limits_{i \in \{2,...,n\}} \{dp_1(v_i, H_i) + \sum\limits_{j \in \{2,...,n\} \setminus \{i\}} dp_0^1(v_j, H_j)\} \\ \mathrm{dp}_0^0(v_1, H_1) + \min\limits_{i \in \{2,...,n\}} \{dp_2(v_i, H_i) + \sum\limits_{j \in \{2,...,n\} \setminus \{i\}} dp_0^0(v_j, H_j)\} \\ \mathrm{dp}_0^0(v_1, H_1) + A \end{cases}$$

where $A = \min\limits_{i,j \in \{2,...,n\}} \{dp_1(v_i, H_i) + dp_1(v_j, H_j) + \sum\limits_{k \in \{2,...,n\} \setminus \{i,j\}} dp_0^0(v_k, H_k)\}$
Clearly case 1 takes $O(|V(B_i)|)$ time. We can modify case 2 and case 3 similarly to 4.1.3 so both take $O(|V(B_i)|)$ time. In case 4: $dp_0^0(v_1, H_1) + A$, A can be modified as $\sum\limits_{k \in \{2,...,n\}} dp_0^0(v_k, H_k)\} + \min\limits_{i,j \in \{2,...,n\}} \{dp_1(v_i, H_i) - dp_0^0(v_i, H_i) + dp_1(v_j, H_j) - dp_0^0(v_j, H_j)\}$. Here, $\sum\limits_{k \in \{2,...,n\}} dp_0^0(v_k, H_k)\}$ takes $O(|V(B_i)|)$ time and $\min\limits_{i,j \in \{2,...,n\}} \{dp_1(v_i, H_i) - dp_0^0(v_i, H_i) + dp_1(v_j, H_j) - dp_0^0(v_j, H_j)\}$ takes $O(|E(B_i)|)$ time as there is an edge between every pair of vertices in $v_i, v_j \in \{v_2, v_3, \ldots, v_n\}$. Hence case 4 takes $O(|V(B_i)| + |E(B_i)|)$ time. Similarly, we can prove for $dp_0^0, dp_0^1, dp_1, dp_2$ takes $O(|V(B_i)|)$ time at the i^{th} iteration. □

Lemma 5.1.4. *For a graph G with n vertices and m edges, the time complexity of Algorithm 8 is $O(n + m)$.*

Proof. Let $\{B_1, B_2, .., B_r\}$ be the set of blocks and $\{c_1, c_2, .., c_r\}$ be the set of cut vertices of G. Assigning initial values $dp_0^0(v, \{v\}), dp_0^1(v, \{v\}), dp_0^2(v, \{v\})$,

$dp_1(v, \{v\}), dp_2(v, \{v\})$ in Algorithm 10 takes $O(n)$ time. Let $\{B_1, B_2, .., B_r\}$ be the inverted order of the order obtained by doing a breadth-first search traversal on the block cut-point tree of G. Breath-first search traversal takes $O(n+m)$ and construction of cut-point tree takes $O(n+m)$. We traverse using the order obtained in each iteration of while loop we choose a block with at most one cut-vertex in G_1. At each while loop iteration $dp_0^0, dp_0^1, dp_0^2, dp_1, dp_2$ are calculated. From 5.1.3 we know that computation of $dp_0^0, dp_0^1, dp_1, dp_2$ take $O(|V(B_i)|)$ time and computation of dp_0^2 takes $O(|V(B_i)|+|E(B_i)|)$. We visit every block exactly only once by the while loop. Hence, time complexity of algorithm 10 is $O(n + m)$. □

Chapter 6

Conclusion and Future Work

Overall, we see that Domination in Graphs is a major topic in which much work has been done in the past, and is also being done presently. Among this, the topics of Roman, Italian Domination, and their variants are subject of much discussion. We've seen that for some classes of graphs, finding out the Roman/Italian Domination number is NP-Complete, while, in others, it's possible in polynomial time. We have provided the algorithms for Perfect Roman Domination, Perfect Italian Domination, Total Roman Domiantion for Trees, and Independent Roman Domination, Independent Italian Domination, Total Roman Dominatoin, for Block Graphs. There are some other classes of graphs, and some other variants of dominating functions, for which we can attempt to find the solution of the correspnding decision problem.

Bibliography

[1] Chain-Chin Yen and R.C.T. Lee. The weighted perfect domination problem and its variants. *Journal of Discrete Applied Mathematics*, 66(2):147–160, 1996.

[2] E.Cockayne and S.Goodman. A linear algorithm for the domination number of a tree. *Journal of Information Processing Letters*, 4:41–44, 1975.

[3] Renu Laskar S.T. Hedetniemi and John Pfaff. A linear algorithm for finding a minimum dominating set in a cactus. *Discrete Applied Mathematics*, 13(2):287–292, 1986.

[4] SF. Hwang and G.J. Chang. A linear time algorithm for the edge domination problem of a block graph. *Manuscript*, (2).

[5] G.J. Chang. Labeling algorithm for domination problems in sun-free chordal graphs. *Discrete Applied Mathematics*, 22(2):21–34, 1988.

[6] G.J. Chang and G.L. Nemhauser. The k-domination and k-stability problems on sun-free chordal graphs. *SIAM Journal on Algebraic Discrete Methods*, (2):332–345, 2006.

[7] S.M. Hedetniemi R. Laskar, J. Pfaff and S.T. Hedetniemi. On the algorithmic complexity of total domination. *SIAM Journal on Algebraic Discrete Methods*, 5(2):420–425, 1984.

[8] Ernie J Cockayne and Paul A Dreyer. Roman domination in graphs. *Journal of Discrete Mathematics*, 278:11–22, 2004.

[9] Ian Stewart. Defend the roman empire! *Scientific American*, 281:136–138, 1999.

[10] Mathieu Liedloff and Ton Kloks. Efficient algorithms for roman domination on some classes of graphs. *Journal of Discrete Applied Mathematics*, 156:3400–3415, 2008.

[11] Mahsa Darkooti and Abdollah Alhevaz. On perfect roman domination number in trees: complexity and bounds. *Journal of Combinatorial Optimization*, 38:712—-720, 2019.

[12] Michael A. Henning, William F. Klostermeyer, and Gary MacGillivray. Perfect roman domination in trees. *Journal of Discrete Applied Mathematics*, 236:235–245, 2018.

[13] Robert A. Beeler and Teresa W. Haynes. Double roman domination. *Journal of Discrete Applied Mathematics*, 211:23–29, 2016.

[14] Xiujun Zhang and Zepeng Li. Double roman domination in trees. *Journal of Information Processing Letters*, 134:31–34, 2018.

[15] Mustapha Chellali and Teresa W. Haynes. Roman 2-domination. *Journal of Discrete Applied Mathematics*, 204:22–28, 2016.

[16] Michael A. Henning and William F. Klostermeyer. Italian domination in trees. *Journal of Discrete Applied Mathematics*, 217:557–564, 2017.

[17] Abolfazl Poureidi and Nader Jafari Rad. On the algorithmic complexity of roman 2-domination (italian domination). *Iranian Journal of Science and Technology, Transactions A: Science*, 44:791–799, 2020.

[18] Michael A. Henning and Teresa W. Haynes. Perfect italian domination in trees. *Journal of Discrete Applied Mathematics*, 260:164–177, 2019.

[19] Changhong Lu Decheng Wei. Independent italian domination on block graphs. 2019.

[20] D. Pradhan S. Banerjee, J. Mark Keil. Perfect roman domination in graphs. *Theoretical Computer Science*, 796:1–21, 2019.