C++ Programme for total dominator chromatic number of ladder graphs through simple transformations

J. Virgin Alangara Sheeba¹ and A. Vijayalekshmi²,*

Abstract
A total dominator coloring of a graph $G = (V, E)$ without isolated vertices, along with each vertex in $G$, is a proper coloring that dominates a color class. The total chromatic dominator number of $G$ is the minimum number of color classes with further assumption that each vertex in $G$ dominates a color class properly and is represented as $\chi_{td}(G)$. In this manuscript, we consider the chromatic total dominator number of ladder graphs through fundamental transformations via the program C++.

Keywords
Coloring, Total dominator coloring, Total dominator chromatic number.

AMS Subject Classification
05C69, 68W25.

1 Introduction ............................................................. 1480
2 Preliminaries ............................................................. 1481
3 Main Result ............................................................... 1481
4 Conclusion ............................................................... 1486
References ............................................................... 1487

1. Introduction

We mainly find ladder graphs in this manuscript. For additional information in graph theory and its applications, we suggest the reader to refer F. Harrary [4]. Allow $G = (V, E)$ to be a graph without isolated vertices. For any two graphs $G$ and $H$, we characterize the cartesian product, signified by $G \times H$, to be the graph with vertex set $\mathcal{V}(G) \times \mathcal{V}(H)$ and edges between two vertices $(u_1, v_1)$ and $(u_2, v_2)$ iff either $u_1 = u_2$ and $v_1v_2 \in E(H)$ or $u_1u_2 \in E(G)$ and $v_1 = v_2$.

In general, for $n \geq 2$, we characterize a ladder graph as $P_2 \times P_n$ and is signified by $L_n$ and $|\mathcal{V}(L_n)| = p = 2n, n \geq 2$.

A proper coloring of $G$ is an assignment of colors to the vertices of $G$, in a way that adjacent vertices have different colors. The smallest number of colors for which $G$ is properly colored is considered a chromatic number of $G$, and $\chi(G)$ is denoted. A total dominator coloring ($td$-coloring) of $G$ is a proper coloring of $G$ with additional axioms that is properly dominated color class by every vertex in $G$. Let $\chi_{td}(G)$ be the total dominator chromatic number and is defined by the minimum number of colors needed in a total dominator coloring of $G$. This principle was developed in [1] by Vijayalekshmi. This thought is often pointed to as a $G$, and is a Smarandachely $k$-dominator color and was presented in [2] by Vijayalekshmi. A Smarandachely $k$-dominator coloring of $G$ for an integer $k \geq 1$ is a proper coloring of $G$, so that each vertex in a $G$ graph properly dominates a color class of $k$. The smallest number of colors for which there exists a Smarandachely $k$-dominator coloring of $G$ is called the Smarandachely $k$-dominator chromatic number of $G$ and is denoted by $\chi_{td}^k(G)$.

Let $C$ be a minimum $td$-coloring of $G$. We say a color class is considered a non-dominated color class ($n - d$ color class) if no vertex of $G$ dominates it and these color classes are often considered repeated color classes.

We recommend the author to pertain to [3, 5, 6] for further information on this theory and its applications.
2. Preliminaries

In this segment, we remember the critical [3] theorem which is quite helpful in our research. For the subsequent observation the minimum dominator chromatic number of ladder graphs has been identified.

For every \( n \geq 2 \), the total dominator chromatic number of a ladder graph is

\[
\chi_{td}(C_n) = \begin{cases} 
2 \left\lfloor \frac{p}{6} \right\rfloor + 2, & \text{if } n \equiv 0 \pmod{6} \\
2 \left\lfloor \frac{p-2}{6} \right\rfloor + 4, & \text{if } n \equiv 0 \pmod{4} \\
2 \left\lfloor \frac{p-4}{6} \right\rfloor + 4, & \text{otherwise.}
\end{cases}
\]

3. Main Result

In this manuscript we obtain a C++ program which uses fundamental transformations to find the \( td \)-chromatic number of ladder graphs.

Programme as follows

```cpp
#include "stdafx.h"
#include <Windows.h>
#include <conio.h>
#include <iostream>
using namespace std;

int main() {
    int inpt;
    cout << "Enter the Value of Ln" << endl;
    cin >> inpt;
    int N = inpt + inpt; int M = inpt + inpt;
    int** ary = new int*[N]; mat = new int*[N];
    int** mat1 = new int*[N];
    int** mat2 = new int*[N];
    int** mat3 = new int*[N];
    int** matsum = new int*[N];
    for (int i = 0; i < N; ++i)
    {
        ary[i] = new int[M]; mat[i] = new int[M]; mat1[i] = new int[M];
        mat2[i] = new int[M]; mat3[i] = new int[M]; matsum[i] = new int[M];
    }
    int k, l, sum;
    HANDLE p = GetStdHandle(STD_OUTPUT_HANDLE);
    SetConsoleTextAttribute(p, FOREGROUND_INTENSITY | FOREGROUND_INTENSITY);
    for (int i = 0; i < N; ++i)
    {
        ary[i][j] = i;
        cout << endl;
    }
    for (int j = 0; j < M; ++j)
    {
        ary[i][j] = j;
        cout << ary[i][j] << " ";
    }
    for (int i = 0; i < N; i++)
    {
        for (int j = 0; j < M; j++)
        {
            if (ary[j][i] == i + 1 | ary[j][i] == i - 1 | ary[j][i] == i + 3)
            {
                mat[i][j] = 1;
            }
        }
        cout << endl;
    }
}
```

In this section, We have to find the total dominator chromatic number of ladder graphs using C++ programme. The C++ programme is successfully compiled and run on C++ platform. The runtime test is included.
else
{
    mat[i][j] = 0;
    cout << mat[i][j] << " ";
}
else
{
    for (int j = 0; j < N; j++)
    {
        if (ary[j][i] == i + 1 | ary[j][i] == i - 1 | ary[j][i] == i - 3)
        {
            mat[i][j] = 1;
            cout << mat[i][j] << " ";
        }
        else
        {
            mat[i][j] = 0;
            cout << mat[i][j] << " ";
        }
    }
    cout << "\n";
}
cout << "\n" << "ADJACENCY MATRIX BY SUBSTRATING THE ROW ASSENDING VALUES" << "\n";
for (int i = 0; i < N; i++)
{
    int sum = 0;
    for (int j = 0; j < N; j++)
    {
        if (i >= 2 && i <= 5 && mat[i][j] == 1 && mat1[i - 2][j] == 1)
        {
            mat1[i][j] = mat[i][j] - mat[i - 2][j];
        }
        else if (i >= 6 && mat[i][j] == 1 && mat1[i - 2][j] == 1 && matsum[i - 4][0] != 1)
        {
            mat1[i][j] = mat[i][j] - mat1[i - 4][j];
        }
        else if (i >= 4 && mat[i][j] == 1 && mat1[i - 2][j] == 0 && matsum[i - 4][0] == 1)
        {
            mat1[i][j] = mat[i][j] - mat1[i - 4][j];
        }
        else
        {
            mat1[i][j] = mat[i][j];
        }
        sum = sum + mat1[i][j];
    }
    matsum[i][0] = sum;
}
for (int i = 0; i < N; i++)
{
    for (int j = 0; j < N; j++)
    {
        if (mat1[i][j] == 1)
```cpp
{
    SetConsoleTextAttribute(p, FOREGROUND_RED | FOREGROUND_INTENSITY);
    cout << mat1[i][j] << " ";
}
else
{
    SetConsoleTextAttribute(p, FOREGROUND_INTENSITY | FOREGROUND_INTENSITY);
    cout << mat1[i][j] << " ";
}
cout << "\n";

SetConsoleTextAttribute(p, FOREGROUND_INTENSITY | FOREGROUND_INTENSITY);
cout << "\n" << "\n" << "ADJACENCY MATRIX BY SUBSTRATING THE COLUMN VALUES";
if (N%3 == 0)
{
    N = N - 1;
    for (int i = N; i >= 0; i--)
    {
        for (int j = N; j >= 0; j--)
        {
            if (i >= 4 && mat1[i][j] == 1 && mat1[i - 4][j] == 1)
            {
                mat1[i - 4][j] = mat1[i][j] - mat1[i - 4][j];
            }
            else if (i >= 2 && mat1[i][j] == 1 && mat1[i - 2][j] == 1)
            {
                mat1[i - 2][j] = mat1[i][j] - mat1[i - 2][j];
            }
        }
    }
    N = N + 1;
    cout << "\n";
}
else
{
    for (int i = 0; i < N; i++)
    {
        for (int j = 0; j < N; j++)
        {
            if (mat1[i][j] == 1 && mat1[i][j+2] == 1)
            {
                mat1[i][j+2] = mat1[i][j+2] - mat1[i][j];
            }
            else if (mat1[i][j] == 1 && mat1[i][j + 4] == 1)
            {
                mat1[i][j + 4] = mat1[i][j + 4] - mat1[i][j];
            }
            else
            {
                mat1[i][j] = mat1[i][j];
            }
        }
    }
    cout << "\n";
}
```

for (int i = 0; i < N; i++)
{
for (int j = 0; j < N; j++)
{
if (mat1[i][j] == 1)
{
SetConsoleTextAttribute(p, FOREGROUND_RED | FOREGROUND_INTENSITY);
cout << mat1[i][j] << " ";
}
else
{
SetConsoleTextAttribute(p, FOREGROUND_INTENSITY | FOREGROUND_INTENSITY);
cout << mat1[i][j] << " ";
}
}
cout << "\n";
}
SetConsoleTextAttribute(p, FOREGROUND_INTENSITY | FOREGROUND_INTENSITY);
cout << "\n";
int ary2[] = { 0,1,4,5,2,3 }, aaa = 0;
for (int i = 0; i < N; i++)
{
for (int j = 0; j < N; j++)
{
if (ary2[aaa] > N-1)
{
ary2[aaa] = ary2[aaa]-2;
mat3[i][j] = mat1[ary2[aaa]][j];
}
mat3[i][j] = mat1[ary2[aaa]][j];
}
if (aaa < 5 )
{
ary2[aaa] = ary2[aaa] + 6;
aaa = aaa + 1;
}
else if (aaa = 5)
{
ary2[aaa] = ary2[aaa] + 6;
aaa = 0;
}
}
cout << "FINAL SUB MATRIXES AFTER SUBSTRACTING THE COLUMN FROM BOTTOM TO TOP" << "\n";
k = 0;
for (int i = 0; i < N; i++)
{
for (int j = 0; j < N; j++)
{
if (j % 2 == 0 && i % 2 == 0 && mat1[i][j] == 0 && mat1[i][j + 1] == 1 ||
mat1[i][j] == 1)
{
SetConsoleTextAttribute(p, FOREGROUND_RED | FOREGROUND_INTENSITY);
cout << mat1[i][j] << " ";
}
else if (j % 2 != 0 && i % 2 != 0 && mat1[i][j] == 0 && mat1[i][j - 1] == 1)
```cpp
{k = k + 1;
cout << mat1[i][j] << " ";
} else
{
SetConsoleTextAttribute(p, FOREGROUND_INTENSITY | FOREGROUND_INTENSITY);
cout << mat1[i][j] << " ";
}
cout << "\n";
SetConsoleTextAttribute(p, FOREGROUND_INTENSITY | FOREGROUND_INTENSITY);
cout << "\n" << "FINAL SUB MATRIXES AFTER INTERCHANGING THE COLUMN" << "\n";
for (int i = 0; i < N; i++)
{
for (int j = 0; j < N; j++)
{
if ( i% 2==0 && mat3[i][j] == 0 && mat3[i][j + 1] == 1 || mat3[i][j] == 1)
{
SetConsoleTextAttribute(p, FOREGROUND_RED | FOREGROUND_INTENSITY);
cout << mat3[i][j] << " ";
} else if (i % 2 != 0 && mat3[i][j] == 0 && mat3[i][j - 1] == 1)
{
SetConsoleTextAttribute(p, FOREGROUND_RED | FOREGROUND_INTENSITY);
cout << mat3[i][j] << " ";
} else
{
SetConsoleTextAttribute(p, FOREGROUND_INTENSITY | FOREGROUND_INTENSITY);
cout << mat3[i][j] << " ";
}
cout << "\n";
SetConsoleTextAttribute(p, FOREGROUND_INTENSITY | FOREGROUND_INTENSITY);
}
cout << "\n";
if (inpt % 3 == 0)
{
cout <<"\n" << "TOTAL DOMINATOR CHROMATIC NUMBER IS " << (2 * (k / 3)) + 2
<< "\n";
} else
{
cout << "\n" << "TOTAL DOMINATOR CHROMATIC NUMBER IS " << (2 * (k-1)/3) + 4
<< "\n";
}
system("Pause");
return 0;
for (int i = 0; i < N; ++i)
{
delete[] ary[i], ary, mat1[i], mat1, mat[i], mat, matsum[i], matsum, mat2[i], mat2,
mat3[i], mat3;
}
```
4. Conclusion

Within this manuscript, we treat the total dominator chro-
mantic number of ladder graphs in a simplified and enhanced fashion utilizing elementary transformations by C++ programme.

References


