An alternate C++ programme for the total dominator chromatic number of ladder graphs

A.Vijayalekshmi¹* and J.Virgin Alangara Sheeba²

Abstract
A total dominator coloring of a graph \( G = (V, E) \) without isolated vertices is a proper coloring of \( G \) in which each vertex of \( G \) is adjacent to every vertex of some color class. The total dominator chromatic number of \( G \) is the minimum number of colors among all total dominator colorings of \( G \) and is denoted by \( \chi_{td}(G) \). In this paper, we provide C++ programme for the total dominator chromatic number of Ladder graphs.

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

AMS Subject Classification
05C69, 68W25.

1. Introduction
In this paper we consider ladder graphs only. Further details in graph theory can be found in F.Harrary [4]. Let \( G = (V, E) \) be a graph with minimum degree at least one. For any two graphs \( G \) and \( H \), we define the cartesian product, denoted by \( G \times H \), to be the graph with vertex set \( V(G) \times V(H) \) and edges between two vertices \( (u_1, v_1) \) and \( (u_2, v_2) \) if either \( u_1 = u_2 \) and \( v_1v_2 \in E(H) \) or \( u_1u_2 \in E(G) \) and \( v_1 = v_2 \). A ladder graph can be defined as \( P_2 \times P_n \), where \( n \geq 2 \) and is denoted by \( L_n \).

A proper coloring of \( G \) is an assignment of colors to the vertices of a graph so that no two adjacent vertices have the same color. The chromatic number of a graph \( G \) is the smallest number of colors needed to a proper coloring of \( G \) and is denoted by \( \chi(G) \).

A total dominator coloring (td-coloring) of \( G \) is a proper coloring of the vertices of \( G \) in which each vertex of the graph is adjacent to every vertex of some color class and is denoted by \( \chi_{td}(G) \). This concept was introduced by Vijayalekshmi in [1]. This notion is also referred as a smarandachely \( k \)-dominator coloring of \( G, (k \geq 1) \) and was introduced by Vijayalekshmi in [2]. For an integer \( k \geq 1 \), a smarandachely \( k \)-dominator coloring of \( G \) is a proper coloring of \( G \), such that every vertex in a graph \( G \) properly dominates a \( k \) color class. 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 \( \chi_{tdk}(G) \).

In a proper coloring \( C \) of a graph \( G \), a color class of \( C \) is a set consisting of all those vertices assigned the same color. Let \( C \) be a minimum td-coloring of \( G \). We say that a color class is called a non-dominated color class (\( n-d \) color class) if it is not dominated by any vertex of \( G \) and these color classes are also called repeated color classes.

A ladder graph of order \( n \) is denoted by \( L_n \) and \( V(L_n) = p = 2n, n \geq 2 \). The total dominator chromatic number of ladder graphs was found in [3]. For more details on this theory and its applications, we suggest the reader to refer [5, 6].

2. Preliminaries
In this section, we recall the crucial theorem [3] which is...
very useful in our work.

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

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

### 3. Main Result

In this section we provide the source code of the C++ program to find the $td$-chromatic number of a ladder graphs. The program is successfully compiled and tested under C++ platform. The runtime test is also shown below.

#### Program source code

```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;
    while (inpt >= 5)
    {
        int N = (inpt + inpt), M = (inpt + inpt);
        int** a = new int*[N]; b = new int*[N];
        int** mat = new int*[N];
        int** cc = new int*[N];
        int** bb = new int*[N];
        for (int i = 0; i < N; ++i)
        {
            a[i] = new int[M]; b[i] = new int[M]; mat[i] = new int[M];
            cc[i] = new int[M];
        }
        int i, j, k, n, g, h, gg, hh, ii = 0, jj = 0, d = 0;
        n = (inpt + inpt);
        HANDLE p = GetStdHandle(STD_OUTPUT_HANDLE);
        for (i = 0; i < n; i++)
        {
            g = 4, h = 5, gg = 6, hh = 7;
            for (j = 0; j < n; j++)
            {
                a[i][j] = j;
                if (j == g || j == gg || j == 1 || j == 4 || j == 0 || j == h || j == hh)
                {
                    b[i][j] = 0;
                    if (j == g) { g = j + 6; }
                    if (j == gg) && j + 6 != n - 2) { gg = j + 6; }
                    if (j == h) { h = j + 6; }
                }
            }
        }
    }
}
```

603/607
if (j == hh && j + 6 != n - 1) { hh = j + 6; }
bb[j][i] = 0;
}
else
{
b[i][j] = j;
bb[j][i] = j;
}

for (i = 0; i < n; i++)
{
for (j = 0; j < n; j++)
{
if (b[i][j] == 0)
{
SetConsoleTextAttribute(p, FOREGROUND_RED | FOREGROUND_INTENSITY);
}
else
{
SetConsoleTextAttribute(p, FOREGROUND_INTENSITY | FOREGROUND_INTENSITY);
if (b[i][j - 1] == 0 && b[i][j + 1] != 0)
{
    d = d + 1;
}
}
SetConsoleTextAttribute(p, FOREGROUND_INTENSITY | FOREGROUND_INTENSITY);
}
cout << endl << "The Adjacency Matrix for L" << n / 2 << endl << "with the repeating colours highlighted in red" << endl;
for (i = 0; i < n; i++)
{
if (i % 2 == 0)
{
for (j = 0; j < n; j++)
{
if (a[i][j] == i + 1 | a[i][j] == i - 1 | a[i][j] == i + 3)
{
    if (b[i][j] == 0 || bb[i][j] == 0)
    {
SetConsoleTextAttribute(p, FOREGROUND_RED | FOREGROUND_INTENSITY);
    }
else
    {
SetConsoleTextAttribute(p, FOREGROUND_INTENSITY | FOREGROUND_INTENSITY);
    }
mat[i][j] = 1;
cout << mat[i][j] << " ";
}
else
{
if (b[i][j] == 0 || bb[i][j] == 0)
{
SetConsoleTextAttribute(p, FOREGROUND_RED | FOREGROUND_INTENSITY);
}
}
else
{
    SetConsoleTextAttribute(p, FOREGROUND_INTENSITY | FOREGROUND_INTENSITY);
}
mat[i][j] = 0;
cout << mat[i][j] << " ";
}
}
else
{
    for (j = 0; j < n; j++)
    {
        if (a[i][j] == i + 1 | a[i][j] == i - 1 | a[i][j] == i - 3)
        {
            if (b[i][j] == 0 || bb[i][j] == 0)
            {
                SetConsoleTextAttribute(p, FOREGROUND_RED | FOREGROUND_INTENSITY);
            }
            else
            {
                SetConsoleTextAttribute(p, FOREGROUND_INTENSITY | FOREGROUND_INTENSITY);
            }
            mat[i][j] = 1;
cout << mat[i][j] << " ";
        }
        else
        {
            if (b[i][j] == 0 || bb[i][j] == 0)
            {
                SetConsoleTextAttribute(p, FOREGROUND_RED | FOREGROUND_INTENSITY);
            }
            else
            {
                SetConsoleTextAttribute(p, FOREGROUND_INTENSITY | FOREGROUND_INTENSITY);
            }
            mat[i][j] = 0;
cout << mat[i][j] << " ";
        }
    }
}
cout << "\n";
SetConsoleTextAttribute(p, FOREGROUND_INTENSITY | FOREGROUND_INTENSITY);
cout << "\n" "The Sub Matrix after removing the repeating colours are" << "\n";
for (i = 0; i < n; i++)
{
    for (j = 0; j < n; j++)
    {
        if (j == 0)
        {
            jj = 0;
        }
        if (a[i][j] == i + 1 | a[i][j] == i - 1 | a[i][j] == i - 3)
        {
            if (b[i][j] == 0 || bb[i][j] == 0)
            {
                SetConsoleTextAttribute(p, FOREGROUND_RED | FOREGROUND_INTENSITY);
            }
            else
            {
                SetConsoleTextAttribute(p, FOREGROUND_INTENSITY | FOREGROUND_INTENSITY);
            }
            mat[i][j] = 1;
cout << mat[i][j] << " ";
        }
        else
        {
            if (b[i][j] == 0 || bb[i][j] == 0)
            {
                SetConsoleTextAttribute(p, FOREGROUND_RED | FOREGROUND_INTENSITY);
            }
            else
            {
                SetConsoleTextAttribute(p, FOREGROUND_INTENSITY | FOREGROUND_INTENSITY);
            }
            mat[i][j] = 0;
cout << mat[i][j] << " ";
        }
    }
}
cout << "\n";
SetConsoleTextAttribute(p, FOREGROUND_INTENSITY | FOREGROUND_INTENSITY);
cout << "\n" "The Sub Matrix after removing the repeating colours are" << "\n";
for (i = 0; i < n; i++)
{
    for (j = 0; j < n; j++)
    {
        if (j == 0)
        {
            jj = 0;
        }
        if (a[i][j] == i + 1 | a[i][j] == i - 1 | a[i][j] == i - 3)
        {
            if (b[i][j] == 0 || bb[i][j] == 0)
            {
                SetConsoleTextAttribute(p, FOREGROUND_RED | FOREGROUND_INTENSITY);
            }
            else
            {
                SetConsoleTextAttribute(p, FOREGROUND_INTENSITY | FOREGROUND_INTENSITY);
            }
            mat[i][j] = 1;
cout << mat[i][j] << " ";
        }
        else
        {
            if (b[i][j] == 0 || bb[i][j] == 0)
            {
                SetConsoleTextAttribute(p, FOREGROUND_RED | FOREGROUND_INTENSITY);
            }
            else
            {
                SetConsoleTextAttribute(p, FOREGROUND_INTENSITY | FOREGROUND_INTENSITY);
            }
            mat[i][j] = 0;
cout << mat[i][j] << " ";
        }
    }
}
cout << "\n";
SetConsoleTextAttribute(p, FOREGROUND_INTENSITY | FOREGROUND_INTENSITY);
cout << "\n" "The Sub Matrix after removing the repeating colours are" << "\n";
for (i = 0; i < n; i++)
{
    for (j = 0; j < n; j++)
    {
        if (j == 0)
        {
            jj = 0;
        }
        if (a[i][j] == i + 1 | a[i][j] == i - 1 | a[i][j] == i - 3)
        {
            if (b[i][j] == 0 || bb[i][j] == 0)
            {
                SetConsoleTextAttribute(p, FOREGROUND_RED | FOREGROUND_INTENSITY);
            }
            else
            {
                SetConsoleTextAttribute(p, FOREGROUND_INTENSITY | FOREGROUND_INTENSITY);
            }
            mat[i][j] = 1;
cout << mat[i][j] << " ";
        }
        else
        {
            if (b[i][j] == 0 || bb[i][j] == 0)
            {
                SetConsoleTextAttribute(p, FOREGROUND_RED | FOREGROUND_INTENSITY);
            }
            else
            {
                SetConsoleTextAttribute(p, FOREGROUND_INTENSITY | FOREGROUND_INTENSITY);
            }
            mat[i][j] = 0;
cout << mat[i][j] << " ";
        }
    }
}
cout << "\n";
SetConsoleTextAttribute(p, FOREGROUND_INTENSITY | FOREGROUND_INTENSITY);
cout << "\n" "The Sub Matrix after removing the repeating colours are" << "\n";
for (i = 0; i < n; i++)
{
    for (j = 0; j < n; j++)
    {
        if (j == 0)
        {
            jj = 0;
        }
        if (a[i][j] == i + 1 | a[i][j] == i - 1 | a[i][j] == i - 3)
        {
            if (b[i][j] == 0 || bb[i][j] == 0)
            {
                SetConsoleTextAttribute(p, FOREGROUND_RED | FOREGROUND_INTENSITY);
            }
            else
            {
                SetConsoleTextAttribute(p, FOREGROUND_INTENSITY | FOREGROUND_INTENSITY);
            }
            mat[i][j] = 1;
cout << mat[i][j] << " ";
        }
        else
        {
            if (b[i][j] == 0 || bb[i][j] == 0)
            {
                SetConsoleTextAttribute(p, FOREGROUND_RED | FOREGROUND_INTENSITY);
            }
            else
            {
                SetConsoleTextAttribute(p, FOREGROUND_INTENSITY | FOREGROUND_INTENSITY);
            }
            mat[i][j] = 0;
cout << mat[i][j] << " ";
        }
    }
}
if (b[i][j] != 0 && bb[i][j] != 0)
{
    cc[ii][jj] = mat[i][j];
    jj = jj + 1;
}
if (j == n - 1 && bb[i][j] != 0)
{
    ii = ii + 1;
}
for (i = 0; i < ii; i++)
{
    for (j = 0; j < ii; j++)
    {
        cout << cc[i][j] << " ";
    }
    cout << "\n";
}
cout << "\n" << "Number of 2x2 sub Matrices are " << d << " " << "\n" << "Number of 3x3 sub Matrices are " << 0 << " " << "\n" << "\n";
for (i = 0; i < ii; i++)
{
    for (j = 0; j < ii; j++)
    {
        if (cc[i][j] == 1 || j == i)
        {
            SetConsoleTextAttribute(p, FOREGROUND_RED | FOREGROUND_INTENSITY);
            cout << cc[i][j] << " ";
        }
    else
    {
        SetConsoleTextAttribute(p, FOREGROUND_INTENSITY | FOREGROUND_INTENSITY);
        cout << cc[i][j] << " ";
    }
    cout << "\n";
}
SetConsoleTextAttribute(p, FOREGROUND_INTENSITY | FOREGROUND_INTENSITY);
cout << "\n" << "Total Dominator Chromatic Number is" << "\n" << " 2 + (2 * number of(2x2) matrices + 3 * number of(3x3) matrices " << "\n" << "TOTAL DOMINATOR CHROMATIC NUMBER IS " << 2 + (2 * d) << "\n";
}
return 0;
}
An alternate C++ programme for the total dominator chromatic number of ladder graphs — 607

return main();
}

4. Conclusion

In this paper, we provide C++ programme for the total dominator chromatic number of Ladder graphs in simplified and improved manner.

References