Research Article Available online www.ijsrr.org ISSN: 2279–0543

International Journal of Scientific Research and Reviews

An Alternate C++ Programme for Total Dominator Chromatic Number of Paths and Cycles

J.Virgin Alangara Sheeba* and A.Vijayalekshmi

Department of Mathematics, S.T.Hindu College, Nagercoi629002, Tamilnadu,India.
Affiliated to Manonmaniam Sundaranar University, Abishekapatti, Thirunelveli-627012. TN, India

ABSTRACT

A total dominator coloring of a graph G=(V,E) without isolated vertices is a proper coloring together with each vertex in G properly dominates a color class. The total dominator chromatic number of G is a minimum number of color classes with additional condition that each vertex in G properly domi- nates a color class and is denoted by χ_{td} (G). In this paper we introduce an alternate C++ programme to find the total dominator chromatic number of paths and cycles.

MATHEMATICS SUBJECT CLASSIFICATION: **05C15, 05C69**

KEYWORDS: Total dominator chromatic number, path graph, cycle graph.

***Corresponding author**

J.Virgin Alangara Sheeba

*Research Scholar, Department of Mathematics,

S.T.Hindu College, Nagercoi629002, Tamilnadu, India.

Affiliated to Manonmaniam Sundaranar University,

Abishekapatti, Thirunelveli-627012.Tamil Nadu, India

Email:vijimath.a@gmail.com

INTRODUCTION

In this paper we only consider paths and cycles. Further details in graph the- ory can be found in F.Harrary[1].

Let G=(V,E) be a graph with minimum degree atleast one. For two vertices v_0 and v_n of a graph G,a v_0 - v_n walk is an alternate sequence of vertices and edges $v_0, e_1, v_1, ...,$ e_n, v_n such that the consecutive vertices and edges are incident. A path is a walk in which no vertex is repeated. The vertices v_0 and v_n are called the initial and terminal vertex respectively. A v_0 - v_n walk is closed if $v_0 = v_n$. A closed walk with no repeated vertices except the initial and terminal vertices is called a cycle. A path and cycle with n vertices are denoted by P_n and C_n respectively.$_{td}$ A proper coloring of G is an assignment of colors to the vertices of G, such that adjacent vertices have different colors. The smallest number of colors for which there exists a proper coloring of G is called a chromatic number of G, and is denoted by $\chi(G)$. A total dominator coloring (td-coloring) of G is a proper coloring of G with extra property that every vertex in G properly dominates a color class. The total dominator chro- matic number is denoted by $\chi_{td}(G)$ and is defined by the minimum number of colors needed in a total dominator coloring of G. This concept was introduced by A.Vijayalekshmi in [2]. This notion is also referred as a smarandachely k-dominator coloring of G,(k ≥ 1) and was introduced by A.Vijayalekshmi in [2]. For an integer k ≥ 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 χ^s (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 lass)if it is not dominated by any vertex of G and these color classes are also called repeated color classes.

The total dominator chromatic number of paths and cycles were found in [3]. We have the following observation from [3]

Theorem A [3]

Let G be P_n or C_n. Then

$$\chi_{td}(P_n) = \chi_{td}(C_n) = \begin{cases} 2\left\lfloor \frac{n}{4} \right\rfloor + 2 & if\ n \equiv 0(mod 4) \\ 2\left\lfloor \frac{n}{4} \right\rfloor + 3 & if\ n \equiv 1(mod 4) \\ 2\left\lceil \frac{n+2}{4} \right\rceil + 2 & Otherwise \end{cases}$$

MAIN RESULT

We have to find the total dominator chromatic number of paths and cycles by using C++ programme by removing the corresponding rows and columns of the repeated colors from the adjacency matrix. The C++ programme is successfully compiled and run on C++ platform. The runtime test is included.

PROGRAM AS FOLLOWS

```
#include "stdafx.h"

#include <Windows.h>

#include <conio.h>

#include <iostream>

using namespace std; int

opt =1;

int main()

{

cout << "\n";

cout << "please choose the below options" << "\n" << "\n" << "1. Path" << "\n" <<          "2. Cycle" << "\n" << "3.

Exit" << "\n";

cin >> opt;

while (opt <= 3&& opt >0)

{

if (opt == 1|| opt == 2) //---- if option 1or 2

{
```

```
int inpt;
if (opt == 1)
{
cout << "Enter the Value of Pn" << endl;
cin >> inpt;
}
else
{

cout << "Enter the Value of Cn" << endl;
cin >> inpt;
}
while (inpt >= 11)
{
// dimensions
int N=inpt; // matrix row
int M=inpt; // matrix column
// dynamic allocation
int** a=new int*[N]; //logic matrix int** b=new int*[N]; //logic
matrix
int** mat =new int*[N]; //adjacency matrix
int** cc =new int*[N]; //adjacency matrix without repeating colors int** bb =new int*[N];//logic matrix
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];
bb[i] =new int[M];
}
// variables int i, j, k; int n;
```

```
int g, h; int ii =0; int jj =0; int
d=0;
n=inpt;
HANDLE p=GetStdHandle(STD_OUTPUT_HANDLE);
if (n %4== 1)
{
for (i =0; i<n; i++)
{
g=0;
h=3;
for (j =0; j<n; j++)
{
a[i][j] =j;
if (j == g&& g<= n-1)
{
b[i][j] =0; g=j+4; bb[j][i] =0;
}
else
if (j == h&& h<= n-3&& n-1>= h)
{
b[i][j] =0; h=j+4; bb[j][i] =0;
}
else
{
b[i][j] =j;
bb[j][i] =j;
}
}
}
```

```
}
else
{
for (i =0; i<n; i++)
{
g=0;
h=3;
for (j =0; j<n; j++)
{
a[i][j] =j;
if (j == g&& g<= n-3&& n-1!= g)
{
b[i][j] =0;
g=j+4; bb[j][i] =0;
}
else
if (j == h&& h<= n-1)
{
b[i][j] =0;
h=j+4;
bb[j][i] =0;
}
else
{
b[i][j] =j;
bb[j][i] =j;
}
}
}
}
```

```
// -----END FORMING 2ROW MATRIXES------ for (i =0; i<1;

i++)
{
for (j =0; j<n; j++)
{
if (b[i][j] == 0)
{
SetConsoleTextAttribute(p, FOREGROUND_RED |FOREGROUND_INTENSITY);
//cout << b[i][j] << " ";
}
else
{
SetConsoleTextAttribute(p, FOREGROUND_INTENSITY |FOREGROUND_INTENSITY);
//cout << b[i][j] +1<< " ";
if (b[i][j -1] == 0&& b[i][j +1] != 0)
{
d=d+1;
}
}
}
//cout << endl;
SetConsoleTextAttribute(p, FOREGROUND_INTENSITY |FOREGROUND_INTENSITY);
}
cout << "\n";
if (opt == 1)
{
cout << "The Adjacency Matrix for P" << n<< "\n" << "with the repeating colors highlighted in red" << "\n" <<
```

```
"\n"; // GETTING VALUE FROM USER
}
else if (opt == 2)
{
cout << "The Adjacency Matrix for C" << n<< "\n" << "with the repeating colors highlighted in red" << "\n" <<
"\n"; // GETTING VALUE FROM USER
}

// -----LOGIC TO FORM MATRIX------ if (opt == 1)
{
for (i =0; i<n; i++)
{
for (j =0; j<n; j++)
{
if (b[i][j] == 0|| bb[i][j] == 0)
{
if (a[i][j] == i+1|a[i][j] == i-1)
{
mat[i][j] =1;
SetConsoleTextAttribute(p, FOREGROUND_RED |FOREGROUND_INTENSITY);
cout << mat[i][j] << " ";
}
else
{
mat[i][j] =0;
SetConsoleTextAttribute(p, FOREGROUND_RED |FOREGROUND_INTENSITY);
cout << mat[i][j] << " ";
}
}
else
```

```
{
if (a[i][j] == i+1|a[i][j] == i-1)
{
mat[i][j] =1;
SetConsoleTextAttribute(p, FOREGROUND_INTENSITY |FOREGROUND_INTENSITY);
cout << mat[i][j] << " ";
}
else
{
mat[i][j] =0;
SetConsoleTextAttribute(p, FOREGROUND_INTENSITY |FOREGROUND_INTENSITY);
cout << mat[i][j] << " ";
}
}
}
cout << "\n";
}
}
else if (opt == 2)
{
for (i =0; i<n; i++)
{
for (j =0; j<n; j++)
{
if (b[i][j] == 0|| bb[i][j] == 0)
{
if (a[i][j] == i+1|a[i][j] == i-1|a[i][j] == i+(n -1) |
    a[i][j] == i-(n -1))
{
```

```
mat[i][j] = 1;
 SetConsoleTextAttribute(p, FOREGROUND_RED |FOREGROUND_INTENSITY);
cout << mat[i][j] << " ";
}
else
{
mat[i][j] = 0;
SetConsoleTextAttribute(p, FOREGROUND_RED |FOREGROUND_INTENSITY);
cout << mat[i][j] << " ";
}
}
else
{
if (a[i][j] == i + 1 | a[i][j] == i - 1 | a[i][j] == i + (n - 1) |
   a[i][j] == i - (n - 1))
{
mat[i][j] = 1;
SetConsoleTextAttribute(p, FOREGROUND_INTENSITY |FOREGROUND_INTENSITY);
cout << mat[i][j] << " ";
}
else
{
mat[i][j] = 0;
SetConsoleTextAttribute(p, FOREGROUND_INTENSITY |FOREGROUND_INTENSITY);
cout << mat[i][j] << " ";
}
}
}
cout << "\n";
}
```

```
}
cout << "\n";
// -----END LOGIC TO FORM MATRIX------

SetConsoleTextAttribute(p, FOREGROUND_INTENSITY |FOREGROUND_INTENSITY);

  // Adjecancy matrix without repeating colors
  // Define S={Vertices of Pn and Cn receiving different colors} and


  // V-S ={Vertices of Pn and Cn receiving repeated colors}
  // Assign different colors to the vertices in S. If vi and vj belongs to V-S
  // and vi,vj are adjacent then we assign repeated colors 1,2 (say) to vi and vj
  // respectively.Also vk belongs to V-S and vk is not adjacent to any vi belongs
  // to V-S then assign either color 1 or color 2 to vk.If vi belongs to V-S then
  // remove the corresponding rows and columns of vi from the adjacency matrix.

cout <<"The Sub Matrix after removing the repeating colors are" << "\n"<< "\n";
for (i =0; i<n; i++)
{
for (j =0; j<n; j++)
{
if (j == 0)
{
jj =0;
}
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";
} system("pause"); cout <<
"\n";
//cout << d;
cout << "\n";
if (n %4== 1)
{
cout << "Number of (2X2) sub Matrices are "<< d-1<< " "<< "\n";
cout << "Number of (3X3) sub Matrices are "<< "1" << " "<< "\n" << "\n";
}
else
{
cout << "Number of (2X2) sub Matrices are "<< d<< " "<< "\n";
cout << "Number of (3X3) sub Matrices are "<< "0" << " "<< "\n" << "\n";
}
if (n %4!= 1)
{
```

```
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";
}
}
else
{
for (i =0; i<ii; i++)
{
for (j =0; j<ii; j++)
{
if (cc[i][j] == 1|| j== i|| j>= ii +3|| i>= ii -3&& j>= ii -3)
{
SetConsoleTextAttribute(p, FOREGROUND_RED |FOREGROUND_INTENSITY);
cout << cc[i][j] << " ";
}
else
```

```
{

SetConsoleTextAttribute(p, FOREGROUND_INTENSITY |FOREGROUND_INTENSITY);
cout << cc[i][j] << " ";
}
}
cout << "\n";
}
}
cout << "\n";
SetConsoleTextAttribute(p, FOREGROUND_INTENSITY |FOREGROUND_INTENSITY);
//system("pause");
cout << "Total Dominator Chromatic Number is" << "\n" << "2+[(2 *number of (2X2) matrices)+(3 *
  number of (3X3) matrices)]
cout << "\n";
if (n %4== 1)
{
cout << "\n";
cout << "TOTAL DOMINATOR CHROMATIC NUMBER IS "<< 2+(2 *d-2) +3<< "\n";
}
else
{
cout << "\n";
cout << "TOTAL DOMINATOR CHROMATIC NUMBER IS "<< 2+(2 *d) << "\n";
}
system("pause");
// free ary and mat
for (int i=0; i<N; ++i)
delete[] mat[i];
```

```
delete[] mat;

delete[] cc[i]; delete[] cc; delete[]
a[i]; delete[] a; delete[] bb[i];
delete[] bb; delete[] b[i]; delete[] b;
return main();
}
            } // ------- end of option 1 and 2 if (opt == 3)
{
return 0;
}
}
cout << "please enter the above values " << "\n";
system("pause");
return main();        // restart program cout << "\n";
system("pause");
return 0;
}
```

RUNTIME TEST

```
please choose the below options

1. Path
2. Cycle
3. Exit
1
Enter the Value of Pn
11

The Adjacency Matrix for P11
with the repeating colors highlighted in red

0 1 0 0 0 0 0 0 0 0 0
1 0 1 0 0 0 0 0 0 0 0
0 1 0 1 0 0 0 0 0 0 0
0 0 1 0 1 0 0 0 0 0 0
0 0 0 1 0 1 0 0 0 0 0
0 0 0 0 1 0 1 0 0 0 0
0 0 0 0 0 1 0 1 0 0 0
0 0 0 0 0 0 1 0 1 0 0
0 0 0 0 0 0 0 1 0 1 0
0 0 0 0 0 0 0 0 1 0 1
0 0 0 0 0 0 0 0 0 1 0

The Sub Matrix after removing the repeating colors are

0 1 0 0 0 0
1 0 0 0 0 0
0 0 0 1 0 0
0 0 1 0 0 0
0 0 0 0 0 1
0 0 0 0 1 0
Press any key to continue . . .


Number of (2x2) sub Matrices are 3
Number of (3x3) sub Matrices are 0

0 1 0 0 0 0
1 0 0 0 0 0
0 0 0 1 0 0
0 0 1 0 0 0
0 0 0 0 0 1
0 0 0 0 1 0

Total Dominator Chromatic Number is
 2 + [(2 * number of (2x2) matrices) + (3 * number of (3x3) matrices)]

TOTAL DOMINATOR CHROMATIC NUMBER IS 8
```

RUNTIME TEST

```
please choose the below options

1. Path
2. Cycle
3. Exit
2
Enter the Value of Cn
13

The Adjacency Matrix for C13
with the repeating colors highlighted in red

0 1 0 0 0 0 0 0 0 0 0 0 1
1 0 1 0 0 0 0 0 0 0 0 0 0
0 1 0 1 0 0 0 0 0 0 0 0 0
0 0 1 0 1 0 0 0 0 0 0 0 0
0 0 0 1 0 1 0 0 0 0 0 0 0
0 0 0 0 1 0 1 0 0 0 0 0 0
0 0 0 0 0 1 0 1 0 0 0 0 0
0 0 0 0 0 0 1 0 1 0 0 0 0
0 0 0 0 0 0 0 1 0 1 0 0 0
0 0 0 0 0 0 0 0 1 0 1 0 0
0 0 0 0 0 0 0 0 0 1 0 1 0
0 0 0 0 0 0 0 0 0 0 1 0 1
1 0 0 0 0 0 0 0 0 0 0 1 0

The Sub Matrix after removing the repeating colors are

0 1 0 0 0 0 0
1 0 0 0 0 0 0
0 0 0 1 0 0 0
0 0 1 0 0 0 0
0 0 0 0 0 1 0
0 0 0 0 1 0 1
0 0 0 0 0 1 0
Press any key to continue . . .


Number of (2x2) sub Matrices are 2
Number of (3x3) sub Matrices are 1

0 1 0 0 0 0 0
1 0 0 0 0 0 0
0 0 0 1 0 0 0
0 0 1 0 0 0 0
0 0 0 0 0 1 0
0 0 0 0 1 0 1
0 0 0 0 0 1 0

Total Dominator Chromatic Number is
 2 + [(2 * number of (2x2) matrices) + (3 * number of (3x3) matrices)]

TOTAL DOMINATOR CHROMATIC NUMBER IS 9
```

REFERENCES

1. Harrary F , Graph theory,Addition-Wesley Reading Mass.1969.
2. Vijayalekshmi.A, Total dominator colorings in paths,International Journal of Mathematical Combinatorics, 2012; 2: 89-95

3. Vijayalekshmi.A, Virgin Alangara Sheeba.J, Total dominator chromatic number of Paths, Cycles and Ladder graphs, International Journal of Con temporary Mathematical Sciences, Vol 13,2018,no. 5,199-204
4. Vijayalekshmi.A, Total dominator colorings in cycles, International Journal of Mathematical Combinatorics, 2012; 4: 92-97
5. Jinnah M.I and Vijayalekshmi A, Total dominator colorings in graphs,Ph.D Thesis, University of Kerala.2010
6. Terasa W.Haynes,Stephen T.Hedetniemi,Peter J.slater,Domination in Graphs,Marcel Dekker,New York,1998
7. Terasa W.Haynes,Stephen T.Hedetniemi,Peter J.slater,Domination in Graphs - Advanced Topics, Marceel Dekker,NewYork,1998.