Research Paper

Volume : 4 | Issue : 5 | May 2015 • ISSN No 2277 - 8179

A Generic C++ Program for Finding Incident and Adjacent Matrices in Zero Divisor Graphs

Engineering

KEYWORDS : C++ programming, Algebraic approach, Commutative ring, Zero divisor graph.

Karan Pathak Second Year B.Tech School of Computer Science Engineering VIT University, Vellore

Dr.J. Ravi Sankar Assistant Professor (Senior) School of Advanced Sciences VIT University, Vellore

ABSTRACT

Zero divisor graphs are a family of frequently encountered optimization and enumeration graphs. Although they are specific each for itself, theory provides a mathematical framework for treatment of graph structure and finding the matrices for corresponding zero divisor graphs, in a general way. In this paper we describe a generic C++ library based on algebraic approach for finding the adjacent and incident matrices for a zero divisor graph in single program. The classes and functions in the C++ program are very compact and written with the intention to be extensively combined. To use this program, we can find all the possible adjacent and incident matrices of zero divisor graphs up to large value.

1. Introduction

Let R be a commutative ring and let Z(R) be its set of zero-divisors. The zero-divisor graph of a ring is the graph (simple) whose vertex set is the set of non-zero zero-divisors, and an edge is drawn between two distinct vertices if their product is zero. Throughout this paper, we consider the commutative ring by R and zero divisor graph Γ(R) by $\Gamma(Z_n)$. The idea of a zero-divisor graph of a commutative ring was introduced by I.Beck in [1],where he was mainly interested in colourings. The zero divisor graph is very useful to find the algebraic structures and properties of rings.

Finding matrices in zero divisor graphs, in a broad sense, can be seen as a general name for various kinds of problems originating from different fields. They are usually connected to areas of algebra and graph theory where determination and evaluation of matrices in zero divisor graphs is a commonly encountered problem.

Our aim is to present a general framework for dealing with all adjacent and incident matrices of zero divisor graphs simultaneously, enabling in that way application of general algorithms. As it will be shown in the next section, with the use of algebraic approach to finding matrices, we shall accomplish specified goals.

2. Algebraic Approach of Matrices in Zero Divisor Graphs

Algebraic approach can seem very attractive from the theoretical point of view as it enables abstraction of matrix specific details and treats the core of the adjacent and incident matrices of Γ(Zn), in a unifying way. Although there are many interesting theoretical issues concerning algebraic approach, in this paper we shall point out usefulness of practical application of such method to finding adjacent and incident matrices of Γ(Zn). Algebraic approach will be used as a basis for design and implementation of a small, but very flexible and applicable generic C library of classes and functions for dealing with matrices.Use of such library will provide programming environment for elegant treatment of finding matrices in Γ(Zn)in a general way. We use algebraic approach developed of Γ(Zn) in [3, 4, 5], which treats given suitable algebraic structure of Γ(Zn).

3. Implementation of Algorithms to Finding the Adjacent and Incident Matrices of Zero Divisor Graphs

The main purpose of the C++ library that is tobe described is to provide a generic programmingframework for finding the adjacent and incident matrices of Γ(Zn) ingeneral way. Ideas of algebraic approach presentedin the previous section will be used asa guide for design and implementation of theC++ library [2]. Algebraic approach consists of two basic steps: first one is interpretation of a given graph with asuitably chosen algebraic structure and second one is solvingobtained system of equations by a generally applicablealgorithm. In this section our goal isto show how to effectively and elegantly representthese algebraic structures byusing some of syntactic advantages of the C++ programming language.In that manner, we shall start descriptionof our C++ library with classes whichrepresent respective operations.

In this section, we give the algorithm to finding the adjacent and incident matrices for given zero divisor graph using C++. Our program consists of four steps: The first one is, to check the given number is prime or composite. Second is, if the number is non-prime, then find all the vertices and edges of Γ(Zn). Third is, to find the incident matrix of given Γ(Zn) and fourth is to find the adjacent matrix of given Γ(Zn).

```
#include <stdio.h>

int main()

{

    int var1,var2,var3,i,j,n;

    printf("enter number>>");

    scanf("%d",&n);

int zero_divisor_0[n-

1],count_0,group_0[n][2],index_of_group_0=0;

// CHECKING FOR PRIME//
```

Here we check the given number is prime number or not. If the number is prime, we cannot get any zero divisor graphs. The zero divisor graphs are exits in only composite numbers.

```
var3=1;

for(i=2;i<=n/2;i++)

{ if(n%i==0)

{ var3=2;break; }

}

if(var3==1)

 {        printf("ZERO DIVISOR = { Ø }");

return 0;
```

```
}

// CREATING ZERO DIVISORS USING RELATIVE PRIME CONCEPT//
```

Two numbers are said to be relatively prime numbers, then the common divisor of that two numbers is one. For example, 5 and 6 are relatively prime numbers and 4 and 6 are non-relatively prime numbers, because the common divisor of 4 and 6 is 2.

```
count_0=0;

for(i=2;i<n;i++)

{ for(j=i;j>=2;j--)

    { if((n%j==0) && (i%j==0))

    {  zero_divisor_0[count_0]=i;count_0++;

    goto l;

        }

    }


    l:

continue;

}

//PRINTING ZERO DIVISORS (VERTICES) OF ZERO DIVISOR GRAPH//
```

If a and b are two non-zero elements of a ring Znsuch that a.b = 0, then 'a' and 'b' are the zero divisors of commutativering Zn. In particular, 'a' is a left zero divisor and 'b' is a rightzero divisor. Let Γ(Zn) be a zero divisor graph with n = 15. Using above algorithms, we get 3, 5, 6, 9, 10, 12 are non-relatively prime numbers of 15. Clearly, these points are called the zero divisor of Γ(Zn). In general, the zero divisor Z(R) is called the vertex set of Γ(Zn). That is, the numbers 3, 5, 6, 9, 10 and 12 are vertices of Γ(Zn), where n=15.

```
printf("ZERO DIVISOR OF Z%d={",n);

for(i=0;i<count_0;i++)

{printf(" %d ",zero_divisor_0[i]);}

printf("}");

printf("\nno. of zero divisor are:%d\n",count_0);

//  CALCULATING EDGES OF ZERO DIVISOR GRAPH//
```

Let x and y in Z(R), then the vertices x and y are adjacent if and only if xy = 0. For example in Γ(Z15), the vertices '5' and '9' are adjacent vertices, because multiples of these number is divisible by '15' and the remainder is zero. Hence, there exist an edge connect between these two vertices. But the vertices '3' and '6' are non-adjacent vertices in Γ(Z15), because multiples of this number is not divisible by '15'. In this manner, we can find all the edges of zero divisor graphs.

```
for(index_of_group_0=0,i=0;i<count_0;i++)

for(j=i+1;j<count_0;j++)

if((zero_divisor_0[i]*zero_divisor_0[j])%n==0)                    {
{group_0[index_of_group_0][0]=zero_divisor_0[i];

group_0[index_of_group_0][1]=zero_divisor_0[j];

index_of_group_0++;

}

//SHOWING EDGES OF Γ(Zn) //
```

After finding the edges of Γ(Zn), then print (x, y), where the vertices 'x' and 'y' are adjacent vertices.

```
printf("EDGES ARE:\n");

for(i=0;i<index_of_group_0;i++)

{  printf("e%d={%d,%d}\n",(i+1),group_0[i][0],group_0[i][1]);

 }

int adjacent_matrix_0[count_0+1][count_0+1];

//ADJACENCY MATRIX//
```

After finding the edges, then construct the adjacent matrix of zero divisor graph.

```
for(i=0;i<=count_0;i++)

{ for(j=0;j<=count_0;j++)

    adjacent_matrix_0[i][j]=0;

}

for(i=0;i<count_0;i++)

{ adjacent_matrix_0[i+1][0]=zero_divisor_0[i];

 adjacent_matrix_0[0][i+1]=zero_divisor_0[i];

 }

//WRITING  EDGES IN ADJACENCY MATRIX (var1=row || var2=column)//

for(i=0;i<index_of_group_0;i++)

{for(j=0;j<count_0+1;j++)

  { //FOR ROW

    if(adjacent_matrix_0[0][j]==group_0[i][0])  var1=j;

    ///FOR COLUMN

    if(adjacent_matrix_0[j][0]==group_0[i][1])          var2=j;

  }

adjacent_matrix_0[var1][var2]=1;adjacent_matrix_0[var2][var1]=1;

}
```

Volume : 4 | Issue : 5 | May 2015 • ISSN No 2277 - 8179

```
//DELETING LOOPS

for(i=0;i<count_0;i++)

adjacent_matrix_0[i][i]=0;

printf("ADJACENCY MATRIX:\n");

for(i=0;i<count_0+1;i++)

{ for(j=0;j<count_0+1;j++)

printf(" %d ",adjacent_matrix_0[i][j]);

printf("\n");

}

//INCIDENT MATRIX

printf("INCIDENT MATRIX:\n");

printf(" 0 ");

for(i=0;i<index_of_group_0;i++)

{printf("e%d ",i+1);

}

printf("\n");

for(i=0;i<count_0;i++)

{printf(" %d ",zero_divisor_0[i]);

for(j=0;j<index_of_group_0;j++)

if(zero_divisor_0[i]==group_0[j][0]  ||  zero_
divisor_0[i]==group_0[j][1])

printf(" 1 ");

else

printf(" 0 ");

printf("\n");

}

return 0;

}
```

4. Conclusions

In this paper we have described a generic C++ library of classes and functions for finding adjacent and incident matrices in zero divisor graphs, in a general way. The established programming framework is essentially based on using algebraic approach of zero divisor graphs to finding matrices. Efficiency of the implemented algorithms is generally inferior to specialized algorithms used for solving particular matrices in zero divisor graph due to differences in their time complexities. Nevertheless, the intention of described library is to be used as a support for fast prototyping, testing and experimentation. Moreover, since the fundamental data structure behind all algorithms is matrices, possibility for easy and direct parallelization should also not be disregarded.

REFERENCE

[1] I.Beck, Colouring of Commutative Rings, J. Algebra, 116,(1988),208-226. | [2] B. Stroustrup., The C++ Programming Language,Third Edition, Addison-Wesley, 1998. | [3] J.RaviSankar and S.Meena, Connected Domination number of a commutative ring, International Journal of Mathematical Research,5,(2012), No -1, 5-11. | [4] J. Ravi Sankar,S.Sangeetha, R.Vasanthakumari and S.Meena, Crossing Number of a Zero Divisor Graph, International Journal of Algebra,6, (2012), No -32,1499 – 1505. | [5] J.RaviSankar and S.Meena,On Weak Domination in a Zero Divisor Graph, International Journal of Applied Mathematics, 26,(2013),No – 1, 83 – 91. |