East West University
Wednesday, August 8, 2018

Project Name: Draw Graph

Project Member: (Sec - 4)

1. Arpa Datta(2017-2-60-064)
2. Sabbir Ahmed(2017-2-60-063)
3. Sayed Atique Newaz(2017-2-60-067)
4. Sadat Ahmed(2017-2-60-062)
5. Anika Tahsin(2017-2-60-117)

Background

This project is for draw some graphs using programming language and rules of discrete mathematics. There will be 5 kind of graph you can draw using our program. By using this program one can easily identify and takes note about the graph. It will help you to count edges and the direction of those edges.

Objectives

- **Wheel Graph:** User given node will show the edge number and edges in this graph.
- **Cycle Graph:** Calculate the edges and show them.
- **Complete Graph:** Will show the edges and edge to edge direction.
- **Complete Bipartite Graph:** It will show the edges between two independent nodes.
- **Cube Graph:** Edge number and Vertices number will calculate and show the edge to edge connection.

Scope

By using OOP this program will be done. There will be some base classes. Base class will be designed as the graph name and their options. In those base classes we will calculate the edge numbers and edge to edge connections. We provide some option to user if they want to draw multiple graphs at on time. User can independently choose what they want to draw. There is some limitation in graph that must be fulfilled to draw. By using this program one can easily calculate the edges and edge to edge connection only has to give the node number as input.

Submitted To

Md. Ashraful Islam

Lecturer

For Wheel Graph:

1. Wheel Graph Code:

```
for(int i=1;i<vertice;i++)
        {
          for(int j=i;j<=i;j++)
          {
            x1=j+1;
            x2=j-1;
            x3=j+2;
            if( x1<1 || x1>vertice )
            {
              x1=vertice;
            }
            else if( x2<1 || x2>vertice )
            {
              x2=vertice;
            }
            else if( x3 > vertice )
            {
              x3=vertice-1;
              if( x3==i )
              {
                x3=1;
              }
            }
```

2. Wheel Graph Pseudocode:

- Checking the insert node is equal or bigger than 4 then checking the edges for wheel graph.
- Calculating edge to edge connection using the above logic implementation.
- The node will not be lower than one and bigger than the inserting node.
- If this happen than the if condition will work.
- After doing each work the edge to edge connection will appear.
- At last the Thank You note will be shown for using this program.

P.T.O

For Cycle Graph:

1. Cycle Graph Code:

```
for(int i=1;i<=vertice;i++)
        {
          for(int j=i;j<=i;j++)
          {
             x1=j+1;
             x2=j-1;
             if( x1<1 || x1>vertice )
             {
                x1=vertice;
                if( x1==i )
                {
                   x1=1;
                }
             }
             else if( x2<1 || x2>vertice )
             {
                x2=vertice;
                if( x2==i )
                {
                   x2=1;
                }
             }
```

2. Cycle Graph Pseudocode:
    - Checking the input vertices number if it is greater than 2 then the action will began
    - Calculating edge to edge connection using the above logic implementation.
    - The node will not be lower than one and bigger than the inserting node.
    - If this happen than the if condition will work.
    - After doing each work the edge to edge connection will appear.
    - At last the Thank You note will be shown for using this program.

P.T.O

For Complete Graph:

1. Complete Graph Code

```
for(int i=1;i<=vertice;i++)
        {
          for(int j=i;j<=i;j++)
          {
            x1=j+1;
            if( x1>vertice)
              {
                x1=1;
              }
           cout << "\n"<<i<<" "<<x1<<endl;

          }
        }
```

2. Complete Graph Pseudocode:

- Checking the input vertices number if it is greater than 1 then the action will begin.
- Calculating edge to edge connection using the above logic implementation.
- The node will not be lower than one and bigger than the inserting node.
- If this happen than the if condition will work.
- After doing each work the edge to edge connection will appear.
- At last the Thank You note will be shown for using this program.

For Complete Bipartite Graph:

1. Complete Bipartite Graph Code

```
for(int i=1;i<=v1;i++)
        {
          for(int j=0;j<v2;j++)
          {
            var=65+j;
           cout << "\n"<<i<<" "<<(char)var<<endl;

          }
        }
```

2. Complete Bipartite Pseudocode:

- For Complete Bipartite Graph there must be up to 10 nodes.
- Calculating edge to edge connection using the above logic implementation.
- The node will not be lower than one and bigger than the inserting node.
- If this happen than the if condition will work.
- After doing each work the edge to edge connection will appear.
- At last the Thank You note will be shown for using this program.

For Cube Graph:

1. Cube Graph Code

```
if(v1<=2)
            {
               for (int i=1;i<=vertice;i++)
               {
                  for(int j=i;j<=i;j++)
                  {
                     x1=j+1;
                     if(x1>vertice)
                     {
                        x1=1;
                     }
                     cout << "\n"<<i<<" "<<x1<<endl;
                  }

               }
            }

            else if(v1==3)
            {
               for (int i=1;i<=4;i++)
               {
                  for(int j=i;j<=i;j++)
                  {
                     x1=j+1;
                     if(x1>4)
                     {
                        x1=1;
                     }
                     cout << "\n"<<i<<" "<<x1<<endl;
                  }

               }
               for (int i=1;i<=4;i++)
               {
                  for(int j=i;j<=i;j++)
                  {

                     if(var2>68)
                     {
                        var2='A';
                     }
                     cout << "\n"<<(char)var1<<" "<<(char)var2<<endl;
                     var1++;
                     var2++;
                  }

               }
               var1=65;
```

P.T.O

```
            for(int i=0; i<4;i++)
            {
               for(int j=i;j<=i;j++)
               {
                  cout << "\n"<<i+1<<" "<<(char)var1<<endl;
                  var1++;
               }
            }
         }
```

2. Cube Graph Pseudocode

    - For Cube Graph there must be up to 3 nodes.
    - Calculating edge to edge connection using the above logic implementation.
    - If this happen than the if condition will work.
    - After doing each work the edge to edge connection will appear.
    - At last the Thank You note will be shown for using this program.

Discussion:

Errors in the measurement of material flexibility may have resulted from our testing method. In all but one case, we tested strength and flexibility on the same sample of the materials. Both tests involved applying stress to the samples. In this section we can tell ensure you that most of this program will work perfectly in order to your Input. You can easily understand what is and what will be done with this program.

Limitation:

This program will not work after the while execution. That’s one of the limit of this program. On Wheel, Cycle and Complete graph you can’t show multiple value at a time that’s the another limitation of this program. On the other hand, on Complete Bipartite Graph the program limitation is only for up to 10 input if the input is bigger than 10 then the program will show an invalid output text. And the last Graph is Cube here is the biggest limitation stays, only up to 3 input is available. If the input limit cross 3 then the program will not execute. This are the limitations of this program.