Lampiran 1

MainForm

```
using System;
using System.Drawing;
using System.Windows.Forms;
using System.Collections.Generic;
using System.Linq;
using labirin.classes;

namespace labirin
{
   public partial class mainForm : Form
   {
      private int LebarPutih, LebarHitam;
      private int BorderAtas, BorderBawah, BorderKiri, BorderKanan;
      private Bitmap sourceBmp, finalBmp, nodeOnlyBmp;
      private BitmapProperties bmpProperties;
      private nGraph<Point> myGraph;
      private System.Diagnostics.Stopwatch stopwatch;

      public mainForm()
      {
         Application.DoEvents();
         InitializeComponent();
         stopwatch = new System.Diagnostics.Stopwatch();
      }

      private void Procced()
      {
         Application.DoEvents();
         stopwatch.Reset();
         stopwatch.Start();
         //sourceBmp = new Bitmap(sourceBmp, new System.Drawing.Size(480, 480)); // resize
         Bitmap grayscaleBmp = new Bitmap(sourceBmp.Width, sourceBmp.Height);
         Bitmap binaryBmp = new Bitmap(sourceBmp.Width, sourceBmp.Height);

         finalBmp = new Bitmap(sourceBmp.Width, sourceBmp.Height);
         nodeOnlyBmp = new Bitmap(finalBmp.Width, finalBmp.Height);

         List<Point> meetPoint = new List<Point>();

         #region Konversi ke grayscale & binary
         Application.DoEvents();
         for (int x = 0; x < sourceBmp.Width; x++)
         {
            for (int y = 0; y < sourceBmp.Height; y++)
            {
               Color originalColor = sourceBmp.GetPixel(x, y);

            int intValue = (int)((originalColor.R * .30) + (originalColor.G * .59) + (originalColor.B *
            .11));
               grayscaleBmp.SetPixel(x, y, Color.FromArgb(intValue, intValue, intValue));

               if (intValue < 128)
                  intValue = 0;
               else
                  intValue = 255;
```

```
            binaryBmp.SetPixel(x, y, Color.FromArgb(intValue, intValue, intValue));
            finalBmp.SetPixel(x, y, Color.FromArgb(intValue, intValue, intValue));
        }
    }
    pictureBoxGrayscale.Image = grayscaleBmp;
    pictureBoxBinary.Image = binaryBmp;
    #endregion

    #region Mencari rata ketebalan dinding
    Application.DoEvents();
    bmpProperties = new BitmapProperties(binaryBmp);
    LebarHitam = bmpProperties.BlackWallWidth;
    LebarPutih = bmpProperties.WhiteWallWidth;

    BorderAtas = bmpProperties.UpperBorder;
    BorderBawah = bmpProperties.BottomBorder;
    BorderKiri = bmpProperties.LeftBorder;
    BorderKanan = bmpProperties.RightBorder;

    UpdateStatus(string.Format("Lebar Hitam\t: {0} pixel\n", LebarHitam));
    UpdateStatus(string.Format("Lebar Putih\t: {0} pixel\n\n", LebarPutih));

    UpdateStatus(string.Format("Border Atas\t: {0} pixel\n", BorderAtas));
    UpdateStatus(string.Format("Border Bawah\t: {0} pixel\n", BorderBawah));
    UpdateStatus(string.Format("Border Kiri\t: {0} pixel\n", BorderKiri));
    UpdateStatus(string.Format("Border Kanan\t: {0} pixel\n\n", BorderKanan));
    #endregion

    #region Mencari titik tengah hitam dan putih
    Application.DoEvents();
    List<int> X_Coordinate = new List<int>();
    int stepPoint = bmpProperties.StartPoint.X;
    int halfWhite = (int)Math.Round((decimal)LebarPutih / 2, MidpointRounding.ToEven);
    int halfBlack = (int)Math.Round((decimal)LebarHitam / 2, MidpointRounding.ToEven);
    int step = LebarPutih + LebarHitam;

    while (stepPoint < bmpProperties.StopPoint.X)
    {
        if (stepPoint == bmpProperties.StartPoint.X)
            stepPoint = halfWhite + LebarHitam + bmpProperties.StartPoint.X;
        else
            stepPoint += step;

        if (stepPoint >= bmpProperties.StopPoint.X)
            break;

      //  for (int y = bmpProperties.StartPoint.Y; y < bmpProperties.StopPoint.Y; y++)
        //    finalBmp.SetPixel(stepPoint, y, Color.Red);

        X_Coordinate.Add(stepPoint);
    }

    stepPoint = bmpProperties.StartPoint.Y;

    while (stepPoint < bmpProperties.StopPoint.Y)
    {
        if (stepPoint == bmpProperties.StartPoint.Y)
            stepPoint = halfWhite + LebarHitam + bmpProperties.StartPoint.Y;
        else
            stepPoint += step;
```

```
    if (stepPoint >= bmpProperties.StopPoint.Y)
        break;

    for (int x = bmpProperties.StartPoint.X; x < bmpProperties.StopPoint.X; x++)
    {
      // finalBmp.SetPixel(x, stepPoint, Color.Blue);

        if (X_Coordinate.Contains(x))
            meetPoint.Add(new Point(x, stepPoint));
    }
}
#endregion

#region Mencari titik yang valid dari kumpulan titik yang ditemukan
Application.DoEvents();
int numOfPoint = 1;

int blockX = 0,
    blockY = 0;

myGraph = new nGraph<Point>();
List<Point> validPoint = new List<Point>();
Graphics g = Graphics.FromImage(finalBmp),
         g2 = Graphics.FromImage(nodeOnlyBmp);
Application.DoEvents();
for (int i = 0; i < meetPoint.Count; i++)
{
   if ((meetPoint[i].X + 1 < bmpProperties.StopPoint.X) && (meetPoint[i].Y + 1 <
  bmpProperties.StopPoint.Y))
    {
    if (finalBmp.GetPixel(meetPoint[i].X + 1, meetPoint[i].Y + 1) == Color.FromArgb(255,
  255, 255))
        {
  if (CheckNode(finalBmp, meetPoint[i], halfBlack + halfWhite)) //mengecek node,.,half
  blac + half white
          {
              validPoint.Add(meetPoint[i]);
              myGraph.AddVertex(new nVertex<Point>(meetPoint[i]));

  blockX = (int)Math.Round((decimal)(meetPoint[i].X / step),
  MidpointRounding.ToEven);
  blockY = (int)Math.Round((decimal)(meetPoint[i].Y / step),
  MidpointRounding.ToEven);

              g.FillEllipse(Brushes.BlueViolet, meetPoint[i].X - 5, meetPoint[i].Y - 5, 10, 10);
              g2.FillEllipse(Brushes.BlueViolet, meetPoint[i].X - 5, meetPoint[i].Y - 5, 10,
  10);

              if (blockY > 1)
  UpdateStatus(string.Format("\nX: {0}, Y: {1}\tX: {2}, Y: {3}", meetPoint[i].X,
  meetPoint[i].Y, blockX, blockY));
              else
  UpdateStatus(string.Format("\nX: {0}, Y: {1}\tX: {2}, Y: {3}", meetPoint[i].X,
  meetPoint[i].Y, blockY, blockX));

              numOfPoint++;
          }
        }
    }
```

```
        }

        #endregion

        #region Mencari Gateway
        bool drawOnce;

        //sisi atas
        drawOnce = false;
        Application.DoEvents();
        for (int x = bmpProperties.StartPoint.X; x < bmpProperties.StopPoint.X; x++)
        {
          if ((finalBmp.GetPixel(x, bmpProperties.StartPoint.Y + halfBlack) ==
          Color.FromArgb(255, 255, 255)) && (!drawOnce))
           {
          if (CheckNodeHelper(finalBmp, new Point(x, bmpProperties.StartPoint.Y), new Point(x,
          bmpProperties.StartPoint.Y + LebarHitam)))
              {
                        validPoint.Add(new Point(x + halfWhite, bmpProperties.StartPoint.Y +
          halfBlack));
          myGraph.AddVertex(new nVertex<Point>(new Point(x + halfWhite,
          bmpProperties.StartPoint.Y + halfBlack)));

          g.FillEllipse(Brushes.BlueViolet, (x + halfWhite) - 5, (bmpProperties.StartPoint.Y +
          halfBlack) - 5, 10, 10);
          g2.FillEllipse(Brushes.BlueViolet, (x + halfWhite) - 5, (bmpProperties.StartPoint.Y +
          halfBlack) - 5, 10, 10);
                  drawOnce = true;
              }
           }
if (finalBmp.GetPixel(x, bmpProperties.StartPoint.Y + halfBlack) == Color.FromArgb(0, 0, 0))
              drawOnce = false;
        }

        //sisi bawah
        drawOnce = false;
        Application.DoEvents();
        for (int x = bmpProperties.StartPoint.X; x < bmpProperties.StopPoint.X; x++)
        {
          if ((finalBmp.GetPixel(x, bmpProperties.StopPoint.Y - halfBlack) ==
          Color.FromArgb(255, 255, 255)) && (!drawOnce))
           {
           if (CheckNodeHelper(finalBmp, new Point(x, bmpProperties.StopPoint.Y -
          LebarHitam), new Point(x, bmpProperties.StopPoint.Y)))
              {
                 validPoint.Add(new Point(x + halfWhite, bmpProperties.StopPoint.Y - halfBlack));
          myGraph.AddVertex(new nVertex<Point>(new Point(x + halfWhite,
          bmpProperties.StopPoint.Y - halfBlack)));

          g.FillEllipse(Brushes.BlueViolet, (x + halfWhite) - 5, (bmpProperties.StopPoint.Y -
          halfBlack) - 5, 10, 10);
          g2.FillEllipse(Brushes.BlueViolet, (x + halfWhite) - 5, (bmpProperties.StopPoint.Y -
          halfBlack) - 5, 10, 10);
                  drawOnce = true;
              }
           }
if (finalBmp.GetPixel(x, bmpProperties.StopPoint.Y - halfBlack) == Color.FromArgb(0, 0, 0))
              drawOnce = false;
        }
```

```
//sisi kiri
drawOnce = false;
Application.DoEvents();
for (int y = bmpProperties.StartPoint.Y; y < bmpProperties.StopPoint.Y; y++)
{
 if ((finalBmp.GetPixel(bmpProperties.StartPoint.X + halfBlack, y) ==
 Color.FromArgb(255, 255, 255)) && (!drawOnce))
  {
 if (CheckNodeHelper(finalBmp, new Point(bmpProperties.StartPoint.X, y), new
 Point(bmpProperties.StartPoint.X + LebarHitam, y)))
     {
                validPoint.Add(new Point(bmpProperties.StartPoint.X + halfBlack, y +
 halfWhite));
 myGraph.AddVertex(new nVertex<Point>(new Point(bmpProperties.StartPoint.X +
 halfBlack, y + halfWhite)));

 g.FillEllipse(Brushes.BlueViolet, (bmpProperties.StartPoint.X + halfBlack) - 5, (y +
 halfWhite) - 5, 10, 10);
 g2.FillEllipse(Brushes.BlueViolet, (bmpProperties.StartPoint.X + halfBlack) - 5, (y +
 halfWhite) - 5, 10, 10);
        drawOnce = true;
     }
 }
          if (finalBmp.GetPixel(bmpProperties.StartPoint.X + halfBlack, y) ==
 Color.FromArgb(0, 0, 0))
     drawOnce = false;
}

//sisi kanan
drawOnce = false;
Application.DoEvents();
for (int y = bmpProperties.StartPoint.Y; y < bmpProperties.StopPoint.Y; y++)
{
 if ((finalBmp.GetPixel(bmpProperties.StopPoint.X - halfBlack, y) ==
 Color.FromArgb(255, 255, 255)) && (!drawOnce))
  {
 if (CheckNodeHelper(finalBmp, new Point(bmpProperties.StopPoint.X - LebarHitam, y),
 new Point(bmpProperties.StopPoint.X, y)))
     {
        validPoint.Add(new Point(bmpProperties.StopPoint.X - halfBlack, y + halfWhite));
 myGraph.AddVertex(new nVertex<Point>(new Point(bmpProperties.StopPoint.X -
 halfBlack, y + halfWhite)));

 g.FillEllipse(Brushes.BlueViolet, (bmpProperties.StopPoint.X - halfBlack) - 5, (y +
 halfWhite) - 5, 10, 10);
 g2.FillEllipse(Brushes.BlueViolet, (bmpProperties.StopPoint.X - halfBlack) - 5, (y +
 halfWhite) - 5, 10, 10);
        drawOnce = true;
     }
 }
if (finalBmp.GetPixel(bmpProperties.StopPoint.X - halfBlack, y) == Color.FromArgb(0, 0, 0))
        drawOnce = false;
}
#endregion

#region Menghubungkan antar titik
Application.DoEvents();
var groupsX = from xpt in validPoint
          orderby xpt.X
          group xpt by xpt.Y into xgrp
```

```
            select new { xCoordinate = xgrp.Key, xMember = xgrp };

UpdateStatus("\n\n~ Koordinat Y ~");
Application.DoEvents();
foreach (var group in groupsX)
{
 UpdateStatus(string.Format("\n{0} jumlah point untuk koordinat Y di {1}.\n",
 group.xMember.Count(), group.xCoordinate));
  for (int i = 0; i < group.xMember.Count() - 1; i++)
  {
 if (IsConnected(finalBmp, group.xMember.ElementAt(i), group.xMember.ElementAt(i +
 1), 'x'))
     {
 UpdateStatus(string.Format("-> Koordinat {0} dan {1} terhubung. Jarak antara koordinat
 {2} blok.\n",
            group.xMember.ElementAt(i),
            group.xMember.ElementAt(i + 1),
 (group.xMember.ElementAt(i + 1).X - group.xMember.ElementAt(i).X) / LebarPutih));

          myGraph.AddEdge(new nVertex<Point>(group.xMember.ElementAt(i)),
            new nVertex<Point>(group.xMember.ElementAt(i + 1)),
            group.xMember.ElementAt(i + 1).X - group.xMember.ElementAt(i).X);

 g.DrawLine(new Pen(Color.BlueViolet), group.xMember.ElementAt(i),
 group.xMember.ElementAt(i + 1));
 g2.DrawLine(new Pen(Color.BlueViolet), group.xMember.ElementAt(i),
 group.xMember.ElementAt(i + 1));

 g.DrawString(((group.xMember.ElementAt(i + 1).X - group.xMember.ElementAt(i).X) /
 LebarPutih).ToString(),
            new Font("Arial", 10),
            Brushes.Red,
 (int)Math.Round((decimal)(group.xMember.ElementAt(i + 1).X -
 group.xMember.ElementAt(i).X) / 2, MidpointRounding.ToEven) +
 group.xMember.ElementAt(i).X - 5,
            group.xMember.ElementAt(i).Y);

 g2.DrawString(((group.xMember.ElementAt(i + 1).X - group.xMember.ElementAt(i).X)
 / LebarPutih).ToString(),
            new Font("Arial", 10),
            Brushes.Red,
 (int)Math.Round((decimal)(group.xMember.ElementAt(i + 1).X -
 group.xMember.ElementAt(i).X) / 2, MidpointRounding.ToEven) +
 group.xMember.ElementAt(i).X - 5,
            group.xMember.ElementAt(i).Y);
     }
  }
}
Application.DoEvents();
var groupsY = from ypt in validPoint
          orderby ypt.Y
          group ypt by ypt.X into ygrp
          select new { yCoordinate = ygrp.Key, yMember = ygrp };

UpdateStatus("\n~ Koordinat X ~");
foreach (var group in groupsY)
{
 UpdateStatus(string.Format("\n{0} jumlah point untuk koordinat X di {1}.\n",
 group.yMember.Count(), group.yCoordinate));
  for (int i = 0; i < group.yMember.Count() - 1; i++)
```

```
            {
   if (IsConnected(finalBmp, group.yMember.ElementAt(i), group.yMember.ElementAt(i + 1),
'y'))
                {
UpdateStatus(string.Format("-> Koordinat {0} dan {1} terhubung. Jarak antara koordinat {2}
blok.\n",
                    group.yMember.ElementAt(i),
                    group.yMember.ElementAt(i + 1),
(group.yMember.ElementAt(i + 1).Y - group.yMember.ElementAt(i).Y) / LebarPutih));

                myGraph.AddEdge(new nVertex<Point>(group.yMember.ElementAt(i)),
                    new nVertex<Point>(group.yMember.ElementAt(i + 1)),
                    group.yMember.ElementAt(i + 1).Y - group.yMember.ElementAt(i).Y);

g.DrawLine(new Pen(Color.BlueViolet), group.yMember.ElementAt(i),
group.yMember.ElementAt(i + 1));
g2.DrawLine(new Pen(Color.BlueViolet), group.yMember.ElementAt(i),
group.yMember.ElementAt(i + 1));

g.DrawString(((group.yMember.ElementAt(i + 1).Y - group.yMember.ElementAt(i).Y) /
LebarPutih).ToString(),
                    new Font("Arial", 10),
                    Brushes.Red,
                    group.yMember.ElementAt(i).X,
(int)Math.Round((decimal)(group.yMember.ElementAt(i + 1).Y -
group.yMember.ElementAt(i).Y) / 2, MidpointRounding.ToEven) +
group.yMember.ElementAt(i).Y - 5);

g2.DrawString(((group.yMember.ElementAt(i + 1).Y - group.yMember.ElementAt(i).Y) /
LebarPutih).ToString(),
                    new Font("Arial", 10),
                    Brushes.Red,
                    group.yMember.ElementAt(i).X,
(int)Math.Round((decimal)(group.yMember.ElementAt(i + 1).Y -
group.yMember.ElementAt(i).Y) / 2, MidpointRounding.ToEven) +
group.yMember.ElementAt(i).Y - 5);
                }
            }
        }

        #endregion
        Application.DoEvents();
        pb_result.Image = finalBmp;
        pb_result.Invalidate();
        UpdateStatus("\nJumlah Node: " + myGraph.Count());
        stopwatch.Stop();

UpdateStatus("\nRunning Time: " + ((double)stopwatch.ElapsedMilliseconds /
1000).ToString("F2", System.Globalization.CultureInfo.CreateSpecificCulture("ID-id")));
    }

    private bool CheckNode(Bitmap bitmap, Point point, int step)
    {
        Application.DoEvents();
        bool isNode = false;
        bool left, top, right, bottom;

        top = CheckNodeHelper(bitmap, new Point(point.X, point.Y - step), point);
        bottom = CheckNodeHelper(bitmap, point, new Point(point.X, point.Y + step));
        left = CheckNodeHelper(bitmap, new Point(point.X - step, point.Y), point);
```

```
    right = CheckNodeHelper(bitmap, point, new Point(point.X + step, point.Y));

    if (left && top)
        isNode = true;
    if (top && right)
        isNode = true;
    if (right && bottom)
        isNode = true;
    if (bottom && left)
        isNode = true;
    if (left && top && right && bottom)
        isNode = true;
    if (left && !top && !right && !bottom)
        isNode = true;
    if (!left && top && !right && !bottom)
        isNode = true;
    if (!left && !top && right && !bottom)
        isNode = true;
    if (!left && !top && !right && bottom)
        isNode = true;

    return isNode;
}

private bool CheckNodeHelper(Bitmap bitmap, Point point1, Point point2)
{
    Application.DoEvents();
    bool noWall = true;
    if (point1.X == point2.X)
    {
        for (int y = point1.Y; y < point2.Y; y++)
        {
            if (bitmap.GetPixel(point1.X, y) == Color.FromArgb(0, 0, 0))
            {
                noWall = false;
                break;
            }
        }
    }
    else
    {
        Application.DoEvents();
        for (int x = point1.X; x < point2.X; x++)
        {
            if (bitmap.GetPixel(x, point1.Y) == Color.FromArgb(0, 0, 0))
            {
                noWall = false;
                break;
            }
        }
    }
    return noWall;
}

private bool IsConnected(Bitmap bitmap, Point point1, Point point2, char XorY)
{
    bool connected = true;
    Application.DoEvents();
    if (XorY == 'x')
    {
```

```
            for (int x = point1.X; x <= point2.X; x++)
            {
                if (bitmap.GetPixel(x, point1.Y) == Color.FromArgb(0, 0, 0))
                {
                    connected = false;
                    break;
                }
            }
        }
        else
        {
            Application.DoEvents();
            for (int y = point1.Y; y <= point2.Y; y++)
            {
                if (bitmap.GetPixel(point1.X, y) == Color.FromArgb(0, 0, 0))
                {
                    connected = false;
                    break;
                }
            }
        }
        return connected;
    }

    private Bitmap DrawBackground(int width, int height)
    {
        Bitmap result = new Bitmap(width, height);
        Application.DoEvents();
        for (int x = 0; x < width; x++)
            for (int y = 0; y < height; y++)
                result.SetPixel(x, y, Color.FromArgb(0, 0, 0));
        return result;
    }

    private void DrawFromZero(ref Bitmap bitmap, bool draw_wall)
    {
        int halfPutih = (int)Math.Round((decimal)LebarPutih / 2, MidpointRounding.ToEven);
        int halfHitam = (int)Math.Round((decimal)LebarHitam / 2, MidpointRounding.ToEven);
        Graphics graphic = Graphics.FromImage(bitmap);
        Application.DoEvents();
        if (draw_wall)
        {
            for (int x = 0; x < myGraph.Count(); x++)
            {
                for (int y = 0; y < myGraph.Count(); y++)
                {
                    if (x == y)
                        continue;

                    if (myGraph.Cost(x, y) >= 0)
                    {

                        if (myGraph.GetVertex(x).Value.Y == myGraph.GetVertex(y).Value.Y)
                        {
                            graphic.DrawLine(new Pen(Color.FromArgb(255, 255, 255), LebarPutih),
                                myGraph.GetVertex(x).Value.X - halfPutih,
myGraph.GetVertex(x).Value.Y,
                                myGraph.GetVertex(y).Value.X + halfPutih,
myGraph.GetVertex(y).Value.Y);
                        }
```

```
                    else
                    {
                        graphic.DrawLine(new Pen(Color.FromArgb(255, 255, 255), LebarPutih),
                            myGraph.GetVertex(x).Value.X, myGraph.GetVertex(x).Value.Y -
halfHitam,
                            myGraph.GetVertex(y).Value.X, myGraph.GetVertex(y).Value.Y +
halfHitam);
                    }
                }
            }
        }
        Application.DoEvents();
        for (int x = 0; x < myGraph.Count(); x++)
        {
            for (int y = 0; y < myGraph.Count(); y++)
            {
                if (x == y)
                    continue;

                if (myGraph.Cost(x, y) >= 0)
                {

                    if (myGraph.GetVertex(x).Value.Y == myGraph.GetVertex(y).Value.Y)
                    {
                        graphic.DrawLine(new Pen(Color.DarkBlue, 1),
                            myGraph.GetVertex(x).Value.X, myGraph.GetVertex(x).Value.Y,
                            myGraph.GetVertex(y).Value.X, myGraph.GetVertex(y).Value.Y);

graphic.DrawString(((myGraph.GetVertex(y).Value.X - myGraph.GetVertex(x).Value.X) /
LebarPutih).ToString(),
                            new Font("Arial", 10),
                            Brushes.Tomato,
(int)Math.Round((decimal)(myGraph.GetVertex(y).Value.X - myGraph.GetVertex(x).Value.X) /
2, MidpointRounding.ToEven) + myGraph.GetVertex(x).Value.X - 5,
                            myGraph.GetVertex(x).Value.Y);
                    }
                    else
                    {
                        graphic.DrawLine(new Pen(Color.DarkBlue, 1),
                            myGraph.GetVertex(x).Value.X, myGraph.GetVertex(x).Value.Y,
                            myGraph.GetVertex(y).Value.X, myGraph.GetVertex(y).Value.Y);

graphic.DrawString(((myGraph.GetVertex(y).Value.Y - myGraph.GetVertex(x).Value.Y) /
LebarPutih).ToString(),
                            new Font("Arial", 10),
                            Brushes.Tomato,
                            myGraph.GetVertex(x).Value.X,
(int)Math.Round((decimal)(myGraph.GetVertex(y).Value.Y - myGraph.GetVertex(x).Value.Y) /
2, MidpointRounding.ToEven) + myGraph.GetVertex(x).Value.Y - 5);
                    }

graphic.FillEllipse(Brushes.Red, myGraph.GetVertex(x).Value.X - 5,
myGraph.GetVertex(x).Value.Y - 5, 10, 10);
graphic.FillEllipse(Brushes.Red, myGraph.GetVertex(y).Value.X - 5,
myGraph.GetVertex(y).Value.Y - 5, 10, 10);
                }
            }
        }
    }
```

```
        private delegate void stringDelegate(string s);

        private void UpdateStatus(string text)
        {
            Application.DoEvents();
            if (richTextBoxNotes.InvokeRequired)
            {
                stringDelegate sd = new stringDelegate(UpdateStatus);
                this.Invoke(sd, new object[] { text });
                return;
            }
            else
                richTextBoxNotes.Text += string.Format("{0}", text);
        }

        private void openToolStripMenuItem_Click(object sender, EventArgs e)
        {
            Application.DoEvents();
            if (myGraph != null)
                myGraph = null;

            DialogResult dr = openImageDialog.ShowDialog();
            if (dr == DialogResult.OK)
            {
                sourceBmp = new Bitmap(openImageDialog.FileName);
                pictureBoxAsli.Image = sourceBmp;

                saveToolStripMenuItem.Enabled = true;
                buttonProceed.Enabled = true;

                System.IO.FileInfo info = new System.IO.FileInfo(openImageDialog.FileName);
toolStripStatusLabel.Text = string.Format("{0} | {1} x {2} | {3} KB.", info.FullName,
sourceBmp.Width, sourceBmp.Height, info.Length / 1000);
            }
        }

        private void saveToolStripMenuItem_Click(object sender, EventArgs e)
        {
            Application.DoEvents();
            if (saveFileDialog.ShowDialog() == DialogResult.OK)
            {
                List<string> lines = new List<string>();
                lines.Add(string.Format("BMP_LEBAR={0}", sourceBmp.Width));
                lines.Add(string.Format("BMP_TINGGI={0}", sourceBmp.Height));
                lines.Add(string.Format("LINE_HITAM={0}", LebarHitam));
                lines.Add(string.Format("LINE_PUTIH={0}", LebarPutih));
                for (int i = 0; i < myGraph.Count(); i++)
lines.Add(string.Format("VERTEX[{0}]=X_{1},Y_{2}", i,
myGraph.GetVertex(i).Value.X.ToString(), myGraph.GetVertex(i).Value.Y.ToString()));
                for (int x = 0; x < myGraph.Count(); x++)
                    for (int y = 0; y < myGraph.Count(); y++)
                        if (myGraph.Cost(x, y) >= 0)
                            lines.Add(string.Format("EDGE={0}_{1}_{2}", x, y, myGraph.Cost(x, y)));
 using (System.IO.StreamWriter file = new
System.IO.StreamWriter(@saveFileDialog.FileName))
                foreach (string line in lines)
                    file.WriteLine(line);
            }
        }
```

```
        private void loadToolStripMenuItem_Click(object sender, EventArgs e)
        {
            Application.DoEvents();
            if (openFileDialog.ShowDialog() == DialogResult.OK)
            {
                stopwatch.Start();
                string fileCache = string.Empty;

using (System.IO.StreamReader sr = new System.IO.StreamReader(openFileDialog.FileName))
                    fileCache = sr.ReadToEnd();

                int width = int.MinValue;
                int height = int.MinValue;
                myGraph = new nGraph<Point>();

string[] lines = fileCache.Split(new char[] { '\n', '\r' }, StringSplitOptions.RemoveEmptyEntries);

                foreach (string line in lines)
                {
                    if (line.ToLower().Contains("bmp"))
                    {
                        string value = line.Split(new char[] { '=' },
StringSplitOptions.RemoveEmptyEntries)[1];
                        {
                            if (line.ToLower().Contains("lebar"))
                                width = int.Parse(value);
                            else if (line.ToLower().Contains("tinggi"))
                                height = int.Parse(value);
                        }
                    }
                    if (line.ToLower().Contains("line"))
                    {
                        string value = line.Split(new char[] { '=' },
StringSplitOptions.RemoveEmptyEntries)[1];
                        {
                            if (line.ToLower().Contains("hitam"))
                                LebarHitam = int.Parse(value);
                            else if (line.ToLower().Contains("putih"))
                                LebarPutih = int.Parse(value);
                        }
                    }
                    if (line.ToLower().Contains("vertex"))
                    {
                        string value = line.Split(new char[] { '=' },
StringSplitOptions.RemoveEmptyEntries)[1];
string[] coodinate = value.Split(new char[] { ',' }, StringSplitOptions.RemoveEmptyEntries);
string x_coodinate = coodinate[0].Split(new char[] { '_' },
StringSplitOptions.RemoveEmptyEntries)[1];
string y_coodinate = coodinate[1].Split(new char[] { '_' },
StringSplitOptions.RemoveEmptyEntries)[1];

myGraph.AddVertex(new nVertex<Point>(new Point(int.Parse(x_coodinate),
int.Parse(y_coodinate))));
                    }
                    if (line.ToLower().Contains("edge"))
                    {
                        string value = line.Split(new char[] { '=' },
StringSplitOptions.RemoveEmptyEntries)[1];
```

```
                string[] edge = value.Split(new char[] { '_' },
StringSplitOptions.RemoveEmptyEntries);
                string index1 = edge[0];
                string index2 = edge[1];
                string edgeCost = edge[2];

myGraph.AddEdge(myGraph.GetVertex(int.Parse(index1)),
myGraph.GetVertex(int.Parse(index2)), int.Parse(edgeCost));
            }
        }
        finalBmp = DrawBackground(width, height);
        nodeOnlyBmp = new Bitmap(width, height);
        DrawFromZero(ref finalBmp, true);
        DrawFromZero(ref nodeOnlyBmp, false);
        if (labirinLineToolStripMenuItem.Checked)
            pb_result.Image = finalBmp;
        else
            pb_result.Image = nodeOnlyBmp;
        stopwatch.Stop();
 UpdateStatus("Running Time: " + ((double)stopwatch.ElapsedMilliseconds /
1000).ToString("F2", System.Globalization.CultureInfo.CreateSpecificCulture("ID-id")));
    }
}

private void lineOnlyToolStripMenuItem_Click(object sender, EventArgs e)
{
    Application.DoEvents();
    if (!lineOnlyToolStripMenuItem.Checked)
        lineOnlyToolStripMenuItem.Checked = true;
    labirinLineToolStripMenuItem.Checked = false;
    pb_result.Image = nodeOnlyBmp;
}

private void labirinLineToolStripMenuItem_Click(object sender, EventArgs e)
{
    Application.DoEvents();
    if (!labirinLineToolStripMenuItem.Checked)
        labirinLineToolStripMenuItem.Checked = true;
    lineOnlyToolStripMenuItem.Checked = false;
    pb_result.Image = finalBmp;
}

private void buttonProceed_Click(object sender, EventArgs e)
{
    Application.DoEvents();
    Cursor = Cursors.WaitCursor;
    if (richTextBoxNotes.Text.Length > 0)
        richTextBoxNotes.Clear();
    System.Diagnostics.Stopwatch stopwatch = new System.Diagnostics.Stopwatch();
    stopwatch.Start();
    Procced();
    stopwatch.Stop();
    buttonProceed.Enabled = false;
    Cursor = Cursors.Arrow;
}

private void actualSizeToolStripMenuItem_Click(object sender, EventArgs e)
{
    Application.DoEvents();
    if (!actualSizeToolStripMenuItem.Checked)
```

```
        {
            actualSizeToolStripMenuItem.Checked = true;
            normalToolStripMenuItem.Checked = false;
            pb_result.SizeMode = PictureBoxSizeMode.AutoSize;
            pb_result.Dock = DockStyle.None;
        }
    }

    private void normalToolStripMenuItem_Click(object sender, EventArgs e)
    {
        Application.DoEvents();
        if (!normalToolStripMenuItem.Checked)
        {
            normalToolStripMenuItem.Checked = true;
            actualSizeToolStripMenuItem.Checked = false;
            pb_result.SizeMode = PictureBoxSizeMode.Zoom;
            pb_result.Dock = DockStyle.Fill;
        }
    }
}

public class BitmapProperties
{
    private Bitmap _bitmap;
    private const int NUM_OF_SAMPLE = 101;
    private int _upperBorder, _rightBorder, _bottomBorder, _leftBorder;
    private int _whiteWall, _blackWall;


    public BitmapProperties(Bitmap bitmap)
    {
        Application.DoEvents();
        _bitmap = bitmap;

        CalculateBorder();
        CalculateWall();
    }

    private void CalculateBorder()
    {
        Application.DoEvents();
        int[] upperSample, bottomSample, rightSample, leftSample;

        leftSample = new int[_bitmap.Height];
        rightSample = new int[_bitmap.Height];
        upperSample = new int[_bitmap.Width];
        bottomSample = new int[_bitmap.Width];

        int beforeWallSample, afterWallSample;

        int xSpliter = _bitmap.Width / NUM_OF_SAMPLE,
            ySpliter = _bitmap.Height / NUM_OF_SAMPLE;

        int xPosition = 0,
            yPosition = 0;

        while (xPosition < _bitmap.Width)
        {
            beforeWallSample = afterWallSample = 0;
```

```
    for (int y1 = 0; y1 < _bitmap.Height; y1++)
    {
      if (_bitmap.GetPixel(xPosition, y1) == Color.FromArgb(255, 255, 255))
        beforeWallSample++;
      else
        break;
    }

    for (int y2 = _bitmap.Height - 1; y2 > 0; y2--)
    {
      if (_bitmap.GetPixel(xPosition, y2) == Color.FromArgb(255, 255, 255))
        afterWallSample++;
      else
        break;
    }

    upperSample[xPosition] = beforeWallSample;
    bottomSample[xPosition] = afterWallSample;

    xPosition += xSpliter;
  }

  while (yPosition < _bitmap.Height)
  {
    beforeWallSample = 0;
    afterWallSample = 0;

    for (int x1 = 0; x1 < _bitmap.Width; x1++)
    {
      if (_bitmap.GetPixel(x1, yPosition) == Color.FromArgb(255, 255, 255))
        beforeWallSample++;
      else
        break;
    }

    for (int x2 = _bitmap.Width - 1; x2 > 0; x2--)
    {
      if (_bitmap.GetPixel(x2, yPosition) == Color.FromArgb(255, 255, 255))
        afterWallSample++;
      else
        break;
    }

    leftSample[yPosition] = beforeWallSample;
    rightSample[yPosition] = afterWallSample;

    yPosition += ySpliter;
  }

  var upperResult = from us in upperSample
                    where us != 0
                    group us by us into gus
                    let count_gus = gus.Count()
                    orderby count_gus descending
                    select new { Value = gus.Key, Count = count_gus };

  var bottomResult = from bs in bottomSample
                     where bs != 0
                     group bs by bs into gbs
                     let count_gbs = gbs.Count()
```

```
                    orderby count_gbs descending
                    select new { Value = gbs.Key, Count = count_gbs };

    if ((upperResult.Count() > 0) && (upperResult.ToArray().Count() > 1))
        if (upperResult.ToList()[0].Count > 10)
            _upperBorder = upperResult.ToArray()[0].Value;
    if ((bottomResult.Count() > 0) && (bottomResult.ToArray().Count() > 1))
        if (bottomResult.ToList()[0].Count > 10)
            _bottomBorder = bottomResult.ToArray()[0].Value;

    var leftResult = from ls in leftSample
                     where ls != 0
                     group ls by ls into gls
                     let count_gls = gls.Count()
                     orderby count_gls descending
                     select new { Value = gls.Key, Count = count_gls };

    var rightResult = from rs in rightSample
                      where rs != 0
                      group rs by rs into grs
                      let count_grs = grs.Count()
                      orderby count_grs descending
                      select new { Value = grs.Key, Count = count_grs };

    if ((leftResult.Count() > 0) && (leftResult.ToArray().Count() > 1))
        if (leftResult.ToList()[0].Count > 10)
            _leftBorder = leftResult.ToArray()[0].Value;
    if ((rightResult.Count() > 0) && (rightResult.ToArray().Count() > 1))
        if (rightResult.ToList()[0].Count > 10)
            _rightBorder = rightResult.ToArray()[0].Value;
}

private void CalculateWall()
{
    Application.DoEvents();
    int xStart = _bitmap.Width - (_bitmap.Width - LeftBorder), //titik awal spliter wall
        xEnd   = _bitmap.Width - RightBorder;

    int yStart = _bitmap.Height - (_bitmap.Height - UpperBorder),
        yEnd   = _bitmap.Height - BottomBorder;

    int xSpliter = (xEnd - xStart) / NUM_OF_SAMPLE, //lebar antar spliter (titik sample)
        ySpliter = (yEnd - yStart) / NUM_OF_SAMPLE;

    int xPosition = xStart,
        yPosition = yStart;

    int whiteSample, blackSample;

    int topWhiteValue = 0, topBlackValue = 0,
        sideWhiteValue = 0, sideBlackValue = 0;

    int[] topBlackWall, sideBlackWall,
          topWhiteWall, sideWhiteWall;

    topBlackWall  = new int[xEnd];
    topWhiteWall  = new int[xEnd];
    sideBlackWall = new int[yEnd];
    sideWhiteWall = new int[yEnd];
```

```
        bool passingWhite;

        while (xPosition < topBlackWall.Length)
        {
            passingWhite = false;
            whiteSample = blackSample = 0;

            for (int y = yStart; y < yEnd; y++)
            {
                if ((_bitmap.GetPixel(xPosition, y) == Color.FromArgb(0, 0, 0)) &&
(!passingWhite))
                    blackSample++;
                else if (_bitmap.GetPixel(xPosition, y) == Color.FromArgb(255, 255, 255))
                {
                    whiteSample++;

                    if (!passingWhite)
                        passingWhite = true;
                }
                else if ((_bitmap.GetPixel(xPosition, y) == Color.FromArgb(0, 0, 0)) &&
(passingWhite))
                    break;
            }

            topBlackWall[xPosition] = blackSample;
            topWhiteWall[xPosition] = whiteSample;

            xPosition += xSpliter;
        }

        while (yPosition < sideBlackWall.Length)
        {
            passingWhite = false;
            whiteSample = 0;
            blackSample = 0;

            for (int x = xStart; x < xEnd; x++)
            {
                if ((_bitmap.GetPixel(x, yPosition) == Color.FromArgb(0, 0, 0)) &&
(!passingWhite))
                    blackSample++;
                else if (_bitmap.GetPixel(x, yPosition) == Color.FromArgb(255, 255, 255))
                {
                    whiteSample++;

                    if (!passingWhite)
                        passingWhite = true;
                }
                else if ((_bitmap.GetPixel(x, yPosition) == Color.FromArgb(0, 0, 0)) &&
(passingWhite))
                    break;
            }

            sideBlackWall[yPosition] = blackSample;
            sideWhiteWall[yPosition] = whiteSample;
            yPosition += ySpliter;
        }

        var topBlackResult = from tbw in topBlackWall
                             where tbw != 0
```

```
                    group tbw by tbw into gtbw
                    let count_gtbw = gtbw.Count()
                    orderby count_gtbw descending
                    select new { Value = gtbw.Key, Count = count_gtbw };

            var topWhiteResult = from tww in topWhiteWall
                    where tww != 0
                    group tww by tww into gtww
                    let count_gtww = gtww.Count()
                    orderby count_gtww descending
                    select new { Value = gtww.Key, Count = count_gtww };

            topWhiteValue = topWhiteResult.ToArray()[0].Value;
            topBlackValue = topBlackResult.ToArray()[0].Value;

            var sideBlackResult = from sbw in sideBlackWall
                    where sbw != 0
                    group sbw by sbw into gsbw
                    let count_gsbw = gsbw.Count()
                    orderby count_gsbw descending
                    select new { Value = gsbw.Key, Count = count_gsbw };

            var sideWhiteResult = from sww in sideWhiteWall
                    where sww != 0
                    group sww by sww into gsww
                    let count_gsww = gsww.Count()
                    orderby count_gsww descending
                    select new { Value = gsww.Key, Count = count_gsww };

            sideBlackValue = sideBlackResult.ToArray()[0].Value;
            sideWhiteValue = sideWhiteResult.ToArray()[0].Value;

_whiteWall = (int)Math.Round((decimal)(topWhiteValue + sideWhiteValue) / 2,
MidpointRounding.ToEven);
_blackWall = (int)Math.Round((decimal)(topBlackValue + sideBlackValue) / 2,
MidpointRounding.ToEven);
        }

        public int WhiteWallWidth
        {
            get { return _whiteWall; }
        }

        public int BlackWallWidth
        {
            get { return _blackWall; }
        }

        public int UpperBorder
        {
            get { return _upperBorder; }
        }

        public int BottomBorder
        {
            get { return _bottomBorder; }
        }

        public int LeftBorder
        {
```

```
        get { return _leftBorder; }
      }

      public int RightBorder
      {
        get { return _rightBorder; }
      }

      public Point StartPoint
      {
        get { return new Point(_bitmap.Width - (_bitmap.Width - LeftBorder), _bitmap.Height -
(_bitmap.Height - UpperBorder)); }
      }

      public Point StopPoint
      {
        get { return new Point(_bitmap.Width - RightBorder, _bitmap.Height - BottomBorder); }
      }
    }
}
```

Lampiran 2

Class Graph

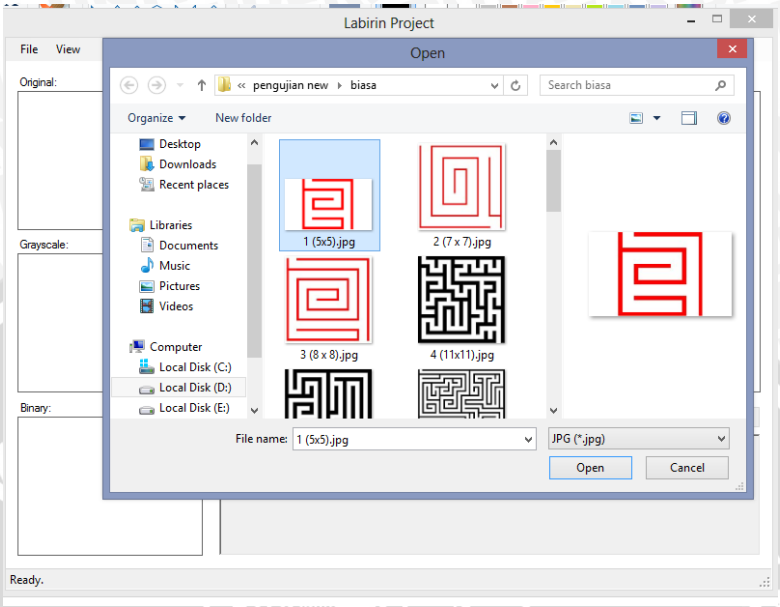
```
using System;
using System.Collections.Generic;

namespace labirin.classes
{
   public class nVertex<T>
   {
      public nVertex(T value)
      {
         Value = value;
         Visited = false;
      }
      public bool Visited { get; set; }
      public T Value;
   }

   public class nGraph<T>
   {
      private int count;
      private int[,] adjMatriks;
      private nVertex<T>[] vertices;

      public nGraph()
         : this(9999)
      {
      }

      public nGraph(int numOfNode)
      {
         count = 0;
         adjMatriks = new int[numOfNode, numOfNode];
         vertices = new nVertex<T>[numOfNode];
         for (int i = 0; i < numOfNode; i++)
            for (int j = 0; j < numOfNode; j++)
               adjMatriks[i, j] = -1;
      }

      public void AddVertex(nVertex<T> value)
      {
         if (count < vertices.Length)
         {
            vertices[count] = value;
            count++;
         }
         else
            throw new ApplicationException("List penuh!");
      }

      public void AddEdge(nVertex<T> vertex1, nVertex<T> vertex2, int cost)
      {
         int vIndex1 = GetIndex(vertex1),
            vIndex2 = GetIndex(vertex2);
         if (vIndex1 >= 0)
            if (vIndex2 >= 0)
               adjMatriks[vIndex1, vIndex2] = cost;
            else
```

```
                throw new ApplicationException("Vertex 2 tidak ada dalam list!");
            else
                throw new ApplicationException("Vertex 1 tidak ada dalam list!");
        }

        public bool IsConnected(nVertex<T> vertex1, nVertex<T> vertex2, out int cost)
        {
            int vIndex1 = GetIndex(vertex1),
                vIndex2 = GetIndex(vertex2);
            if (vIndex1 >= 0)
                if (vIndex2 >= 0)
                {
                    cost = adjMatriks[GetIndex(vertex1), GetIndex(vertex2)];
                    if (cost >= 0)
                        return true;
                    else
                        return false;
                }
                else
                    throw new ApplicationException("Vertex 2 tidak ada dalam list!");
            else
                throw new ApplicationException("Vertex 1 tidak ada dalam list!");
        }

        public int GetIndex(nVertex<T> value)
        {
            int retValue = -1;
            for (int i = 0; i < count; i++)
            {
                if (vertices[i].Value.Equals(value.Value))
                {
                    retValue = i;
                    break;
                }
            }
            return retValue;
        }

        public int Cost(int indexOfVertex1, int indexOfVertex2)
        {
            return adjMatriks[indexOfVertex1, indexOfVertex2];
        }

        public int Count()
        {
            return this.count;
        }

        public nVertex<T> GetVertex(int index)
        {
            return vertices[index];
        }
    }
}
```

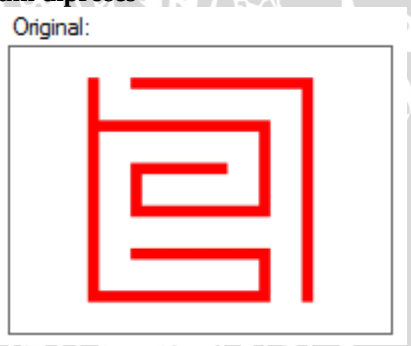
Lampiran 3

Simulasi Proses Pengolahan Citra Labirin Ke Dalam *Edge* dan *Vertex*

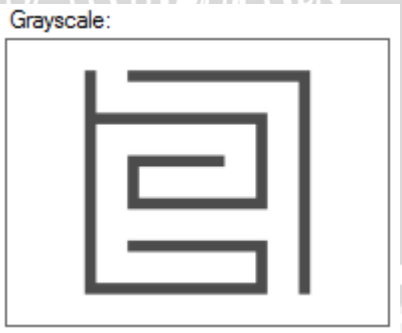
- ***Open Image (JPG, JPEG, BMP, PNG)* atau hasil dari *scanner***



- ***Image original* sebelum diproses**

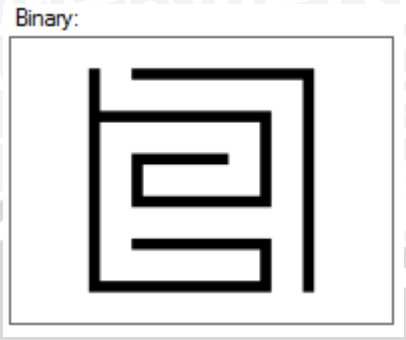


- **Proses *Grayscaling* (*Grayscale* = 0.30R + 0.559G + 0.11B)**

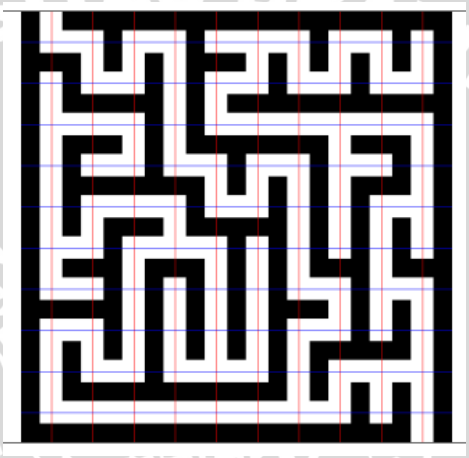


- **Proses Threshold**

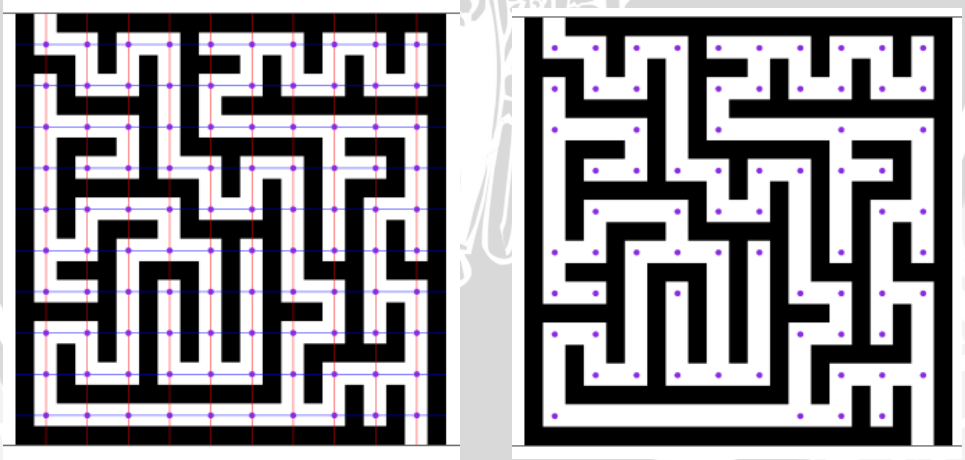
Jika *pixel* yang diseleksi bernilai kurang dari 128 maka nilai *pixel* tersebut akan diubah kedalam *pixel* 0 (hitam), begitu juga sebaliknya jika nilai *pixel* tersebut lebih dari 128 maka akan diubah kedalam *pixel* 255 (putih)



- **Pencarian *meetpoint***

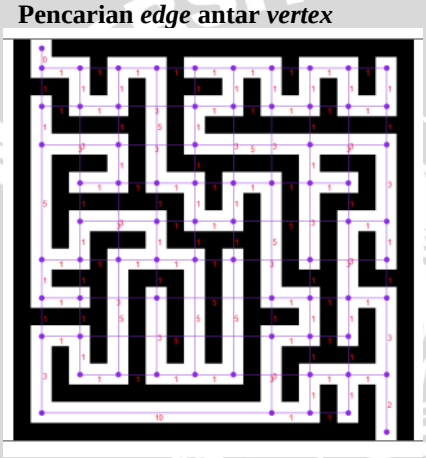


- **Pencarian *Node* yang *Valid***

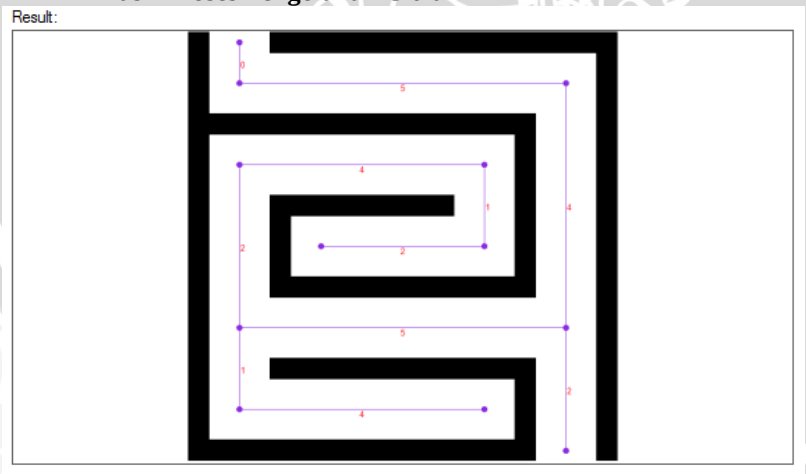


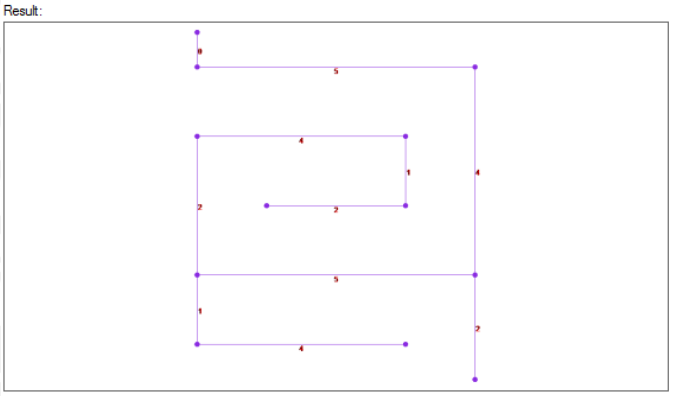
- **Pencarian *Gateway***

- **Pencarian *edge* antar *vertex***

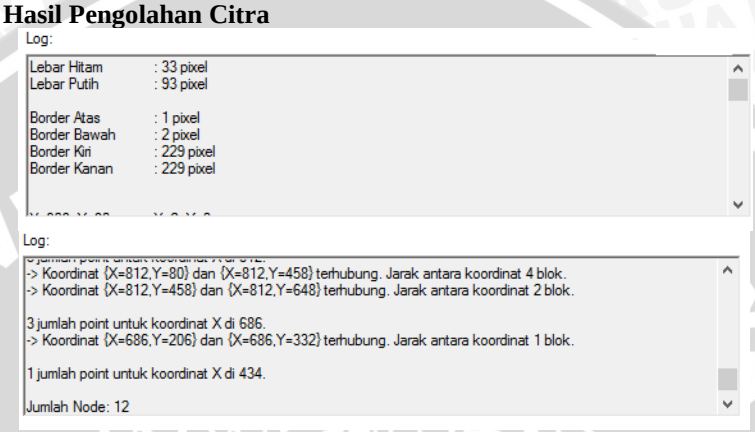


- **Hasil Proses Pengolahan Citra**

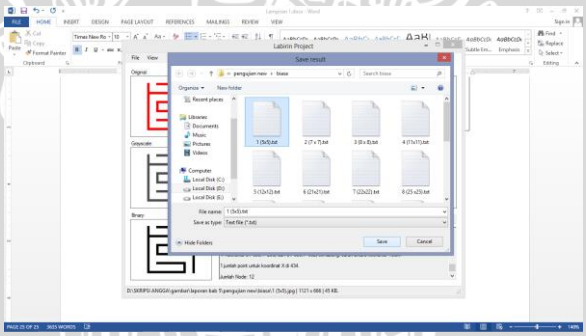




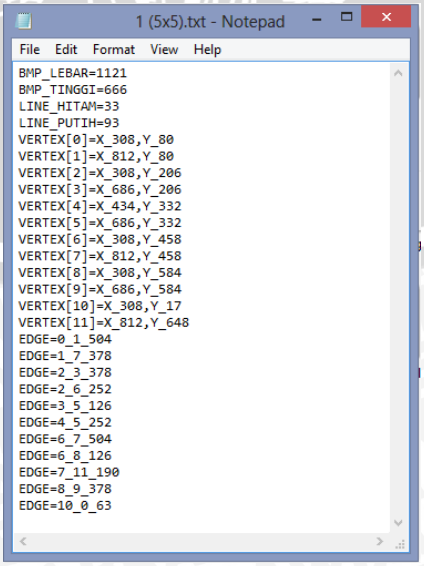
- ***Log* Hasil Pengolahan Citra**



- **Proses *Save* Ke Dalam *Plaintext***



- **Hasil Pengolahan**



- **Proses *Load* Hasil Pengolahan**