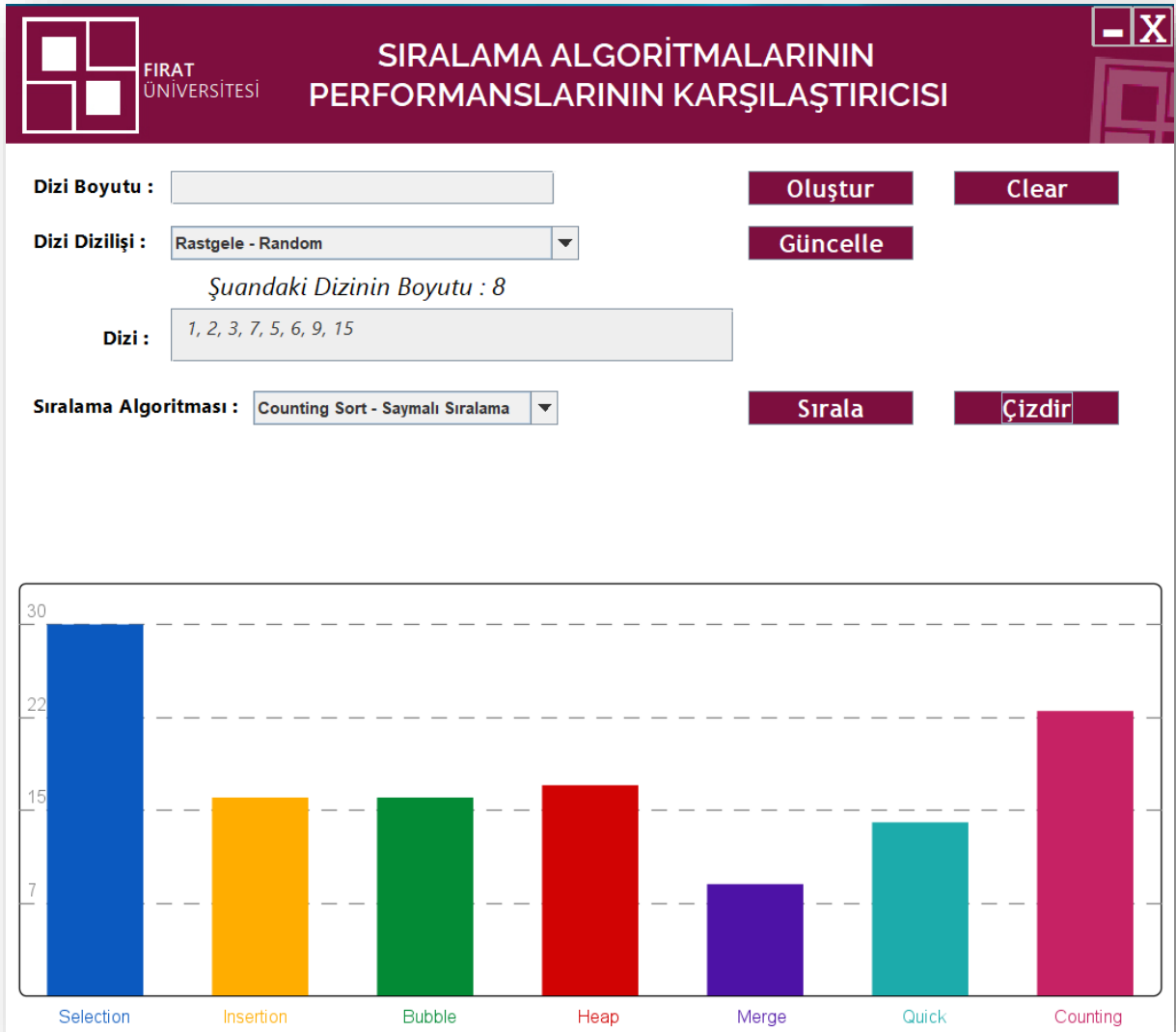



Sıralama Algoritmalarının Ödevi

Arayüz :



**** Sadece İstedığınız Algoritmaları Da Seçebilirsiniz.**

**-Kullanıcı Diziyi Oluştururken Dizinin Rastgele,
Küçükten Büyüğe Sıralı Veya
Büyükten Küçüğe Sıralı Olacağını Belirtebilir.**

 **FIRAT
ÜNİVERSİTESİ**

**SIRALAMA ALGORİTMALARININ
PERFORMANSLARININ KARŞILAŞTIRICISI**

Dizi Boyutu :

Oluştur

Clear

Dizi Dizilişi :

Rastgele - Random

Rastgele - Random

Sıralı (Küçükten Büyüğe) - Sorted (Ascending)

Sıralı (Büyükten Küçüğe) - Sorted (Descending)

Güncelle

Dizi :

Sıralama Algoritması :

Counting Sort - Saymalı Sıralama

Sırala

Çizdir

**** Kullanıcı Manuel Olarak Da Diziyi Güncelleyebilir**

JFRAME FORM Kodları :

```
package siralamaalgoritmalar;

import java.awt.Color;
import javax.swing.JFrame;

/*
 * @author Malik
 */

public class siralamaGUI extends javax.swing.JFrame {
    int dizi[] ;
    grafModeli grafModel ;

    public siralamaGUI() {
        initComponents();
        setLocationRelativeTo(null);
        setResizable(false);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        dizi = new int[0];
        grafModel = new grafModeli();
        grafikPanel.add(grafModel);
        grafModel.setVisible(true);
    }

    private void guncceleButtonActionPerformed(java.awt.event.ActionEvent evt) {

        try{

            dizi = diziyeCevir(diziElemanlariTextArea.getText());
            diziElemanlariTextArea.setText(diziYazdir(dizi));
            diziBoyutuLabel.setForeground(Color.black);
            diziBoyutuLabel.setText("Şuandaki Dizinin Boyutu : "+dizi.length);
            // Kullanıcı Manuel Olarak Da Bir Dizi Oluşturabilir.
            diziBoyutTextField.setText("");
            grafModel.istatistikleriSil();

        }catch(Exception e){
            diziBoyutuLabel.setForeground(Color.red);
            diziBoyutuLabel.setText("Girdiğiniz İfadeler Geçersizdir.");
        }
    }
}
```

```

private void clearButtonActionPerformed(java.awt.event.ActionEvent evt) {
    dizi = new int[0];
    diziBoyutTextField.setText("");
    diziElemanlariTextArea.setText("");
    diziBoyutuLabel.setForeground(Color.black);           // Clear : Grafi Ve Diziyi Siliyor.
    diziBoyutuLabel.setText("Şuandaki Dizinin Boyutu : "+dizi.length);
    diziElemanlariTextArea.setText(diziYazdir(dizi));

    grafModel.istatistikleriSil();
}

```

```

private void siralaButtonActionPerformed(java.awt.event.ActionEvent evt) {

    if(dizi.length>0){
        int[] x = dizi.clone();
        int[][] sonuc ;
        int sayac ;
        int[] siralanmisDizi
// ComboBox'un SelectedIndex'e Göre Sıralama Algoritmayı Çalıştırır Ve Onun Karmaşıklığını Kaydeder.

```

```

        switch(siralamaAlgoComboBox.getSelectedIndex()){
            case 0 :
                if(!grafModel.algoModeliVarMi("Selection")){
                    sonuc = selectionSort(x);
                    sayac = sonuc[0][0];
                    siralanmisDizi = sonuc[1];

                    grafModel.barEkle(new algoModeli(sayac, "Selection"));
                }
                break ;

```

```

            case 1 :
                if(!grafModel.algoModeliVarMi("Insertion")){

                    sonuc = insertionSort(x);
                    sayac = sonuc[0][0];
                    siralanmisDizi = sonuc[1];

                    grafModel.barEkle(new algoModeli(sayac, "Insertion"));
                }
                break ;

```

```

            case 2 :
                if(!grafModel.algoModeliVarMi("Bubble")){
                    sonuc = bubbleSort(x);
                    sayac = sonuc[0][0];
                    siralanmisDizi = sonuc[1];

```

```

        grafModel.barEkle(new algoModeli(sayac, "Bubble"));
    }
    break ;

case 3 :
    if(!grafModel.algoModeliVarMi("Heap")){
        sonuc = heapSort(x);
        sayac = sonuc[0][0];
        siralanmisDizi = sonuc[1];

        grafModel.barEkle(new algoModeli(sayac, "Heap"));
    }
    break ;

case 4 :
    if(!grafModel.algoModeliVarMi("Merge")){
        sonuc = mergeSort(x,0 , x.length-1, 0);
        sayac = sonuc[0][0];
        siralanmisDizi = sonuc[1];

        grafModel.barEkle(new algoModeli(sayac, "Merge"));
    }
    break ;

case 5 :
    if(!grafModel.algoModeliVarMi("Quick")){
        sonuc = quickSort(x,0 , x.length-1, 0);
        sayac = sonuc[0][0];
        siralanmisDizi = sonuc[1];

        grafModel.barEkle(new algoModeli(sayac, "Quick"));
    }
    break ;

case 6 :
    if(!grafModel.algoModeliVarMi("Counting")){
        sonuc = countingSort(x);
        sayac = sonuc[0][0];
        siralanmisDizi = sonuc[1];

        grafModel.barEkle(new algoModeli(sayac, "Counting"));
    }
    break ;
}

}else{
    diziBoyutuLabel.setForeground(Color.red);
    diziBoyutuLabel.setText("Sıralanacak Bir Dizi Yok.");
}

```

```

    }

}

private void olusturButtonActionPerformed(java.awt.event.ActionEvent evt) {

    try{
        switch(diziOlusturmaComboBox.getSelectedIndex()){
            case 0 :
                dizi = rastegeleDiziOlustur(Integer.parseInt(diziBoyutTextField.getText().strip()));
                break;

            case 1 :
                dizi =
siraliDiziOlusturKucuktenBuyuge(Integer.parseInt(diziBoyutTextField.getText().strip()));
                // 3 Tane Tip Dizi Olusturulabilir.
                // Rastgele - Küçükten Büyüğe Sıralı - Büyükten Küçüğe Sıralı
                break;

            case 2 :
                dizi = siraliDiziOlusturBuyuktenKucuge(Integer.parseInt(diziBoyutTextField.getText()));
                break;
        }
        diziBoyutuLabel.setForeground(Color.black);
        diziBoyutuLabel.setText("Şuandaki Dizin Boyutu : "+dizi.length);
        diziElemanlariTextArea.setText(diziYazdir(dizi));
        grafModel.istatistikleriSil(); //Yeni Bir Dizi Oluşturunca Eski Kayıtlar Siliniyor.

    }catch(Exception e){
        diziBoyutuLabel.setForeground(Color.red);
        diziBoyutuLabel.setText("Girdiğiniz İfadeler Geçersizdir.");
    }

}

private void closeLabelMouseClicked(java.awt.event.MouseEvent evt) {
    System.exit(0); // Exit
}

private void minimizeLabelMouseClicked(java.awt.event.MouseEvent evt) {
    this.setExtendedState(JFrame.ICONIFIED); //Minimize
}

private void cizdirButtonActionPerformed(java.awt.event.ActionEvent evt) {
    grafModel.repaint(); // Bar Graflar Çizmek İçin
}

```

```

int[] rastegeleDiziOlustur(int boyut){

    if (boyut > 0){
        int yeni[] = new int[boyut];
        for (int i = 0; i < boyut; i++) {
            yeni[i] = (int) (Math.random()*1002);    // [0,1000] Aralıkta Rastgele Bir Dizi Oluşturur.
        }
        return yeni ;
    }
    return new int[0] ;

}

```

```

int[] siraliDiziOlusturKucuktenBuyuge(int boyut){

    if(boyut>0){
        int yeni[] = rastegeleDiziOlustur(boyut);
        yeni = quickSort(yeni, 0, yeni.length-1, 0)[1];    // Küçükten-Büyüğe Sıralı Bir Dizi Oluşturur.
        return yeni ;
    }
    return new int[0];
}

```

```

int[] siraliDiziOlusturBuyuktenKucuge(int boyut){

    if(boyut>0){
        int temp[] = rastegeleDiziOlustur(boyut);
        temp = quickSort(temp, 0, temp.length-1, 0)[1];
        int[] yeni = new int[temp.length];    // Büyükten-Küçüğe Sıralı Bir Dizi Oluşturur.

        for(int i = 0 ; i<temp.length ; i++){
            yeni[temp.length-1-i] = temp[i];
        }
        return yeni ;
    }
    return new int[0];
}

```

```

String diziYazdir(int[] x){
    String sonuc = "";
    for(int i = 0 ; i<x.length ; i++){
        sonuc+=x[i]+" , " ;    //Diziyi String'e Çevirir.
    }
    if(sonuc.length()>=2){
        sonuc = sonuc.substring(0, sonuc.length()-2);
    }
    return sonuc ;
}

```

```
}
```

```
int[] diziyeCevir(String x){  
    String[] sayilar = x.split(",");  
    int[] sonuc = new int[sayilar.length];  
  
    //String'i Diziye Çevirir.  
    for(int i = 0 ; i<sayilar.length ; i++ ){  
        if(Integer.parseInt(sayilar[i].strip())>=0 && 1000>=Integer.parseInt(sayilar[i].strip()))  
            sonuc[i] = Integer.parseInt(sayilar[i].strip());  
    }  
    return sonuc ;  
}
```

```
public static void main(String args[]) {  
    java.awt.EventQueue.invokeLater(new Runnable() {  
        public void run() {  
            new siralamaGUI().setVisible(true);  
        }  
    });  
}
```


Sıralama Algoritmalarının Kodları :

```
int[][] selectionSort(int[] x){

    int i, j, minIndisi, sayac = 0 ;

    for (i = 0 ; i < x.length-1 ; i++) {
        minIndisi= i ;
        for (j = i+1 ; j < x.length ; j++){
            sayac++;           // Karşılaştırma Yapınca Sayaç Artıyor.
            if(x[j] < x[minIndisi]){
                minIndisi = j ;
            }
        }
        if(minIndisi != i){    // Min indis Seçilen indis ile Aynı ise Gereksiz Swap Yapılmaz.
            sayac++;           // Yerdeğiştirme Olunca Sayaç Artıyor.
            int temp = x[minIndisi];
            x[minIndisi] = x[i];
            x[i] = temp;
        }
    }

    int[][]sonuc = {{sayac}, x};

    return sonuc;
}

int[][] insertionSort(int[] x){

    int i, j, temp, sayac = 0 ;

    for( j = 1 ; j<x.length ; j++){
        temp = x[j];
        i = j - 1 ;
        sayac++;           // Karşılaştırma Yapınca Sayaç Artıyor.
        while(i>=0 && x[i] > temp){
            sayac++;           // Yerdeğiştirme Olunca Sayaç Artıyor.
            x[i+1] = x[i];
            i--;
        }
        x[i+1] = temp;
    }
```

```

        sayac++ ;           // Yerdeğiştirme Olunca Sayaç Artıyor.
    }

    int[][]sonuc = {{sayac}, x};

    return sonuc;
}

int[][] bubbleSort(int[] x){
    int sayac = 0 ;

    boolean degistiMi = true ;

    while(degistiMi){           // Bir Gidişte Yerdeğiştirme Gerçekleşmezse Dizi Sıralı Demektir.
        degistiMi = false ;
        for( int i = 0 ; i<x.length-1 ; i++){
            sayac++ ;           // Karşılaştırma Yapınca Sayaç Artıyor.
            if(x[i] > x[i+1]){
                sayac++ ;       // Yerdeğiştirme Olunca Sayaç Artıyor.
                degistiMi = true ;
                int temp = x[i];
                x[i] = x[i+1];
                x[i+1] = temp ;
            }
        }
    }

    int[][]sonuc = {{sayac}, x};

    return sonuc;
}

int[][] heapSort(int[] x){
    int sayac = 0 ;

    Yigin y = new Yigin(x.length);
    for(int i = 0 ; i < x.length ; i++){
        sayac+= y.ekle(new Eleman(x[i])); // Ekleme Yaparken Heap Kaç Adımda Elemanı Yerleştirdi.
    }

    for (int i = x.length-1 ; i >= 0 ; i--) { // Max Heap Olduğu İçin
        int[] sonuc = y.azamiSil();
        x[i] = sonuc[0];
        sayac+=sonuc[1]; // Silme Yaparken Heap Kaç Adımda Elemanı Çıkarttı.
    }

    int[][]sonuc = {{sayac}, x};

```

```

    return sonuc ;
}

private int merge(int[] x, int solIndis, int ortaIndis, int sagIndis) {
    int i, j, k, sayac = 0 ;

    int n1 = ortaIndis - solIndis + 1 ;
    int n2 = sagIndis - ortaIndis ;

    int[] sol = new int[n1+1]; // Sonsuz Eklemek İçin
    int[] sag = new int[n2+1]; // Sonsuz Eklemek İçin

    for( i = 0 ; i < n1 ; i++){
        sol[i] = x[solIndis+i];
    }
    for( i = 0 ; i < n2 ; i++){
        sag[i] = x[ortaIndis+i+1];
    }

    sol[n1] = Integer.MAX_VALUE;
    sag[n2] = Integer.MAX_VALUE;

    i = 0 ;
    j = 0 ;

    sayac++; // Diziyi 2'ye Bölünce Sayaç Artıyor.
    for (k = solIndis ; k<=sagIndis ; k++){
        sayac++; // Karşılaştırma Yapınca Sayaç Artıyor.
        if(sol[i]<=sag[j]){
            x[k] = sol[i];
            i++;
        }else{
            x[k] = sag[j];
            j++;
        }
    }

    return sayac ;
}

int[][] mergeSort(int[] x, int bas, int son, int sayac){

    int pivot = 0 ;
    if(bas<son){
        pivot = (bas+son)/2 ;
        mergeSort(x, bas, pivot, sayac);
        mergeSort(x, pivot+1, son, sayac);
        sayac+= merge(x, bas, pivot, son);
    }
}

```

```

    }

    int[][]sonuc = {{sayac}, x};

    return sonuc;
}

private int[] parcala(int[] x, int bas, int son){
    int sayac = 0 ;

    int pivot = x[son];
    int i = bas - 1 ;

    for(int j = bas ; j < son ; j++){
        sayac++; // Karşılaştırma Yapınca Sayaç Artıyor.
        if(x[j] <= pivot){
            sayac++; // Yerdeğiştirme Olunca Sayaç Artıyor.
            i++;
            int temp = x[i];
            x[i] = x[j];
            x[j] = temp ;
        }
    }

    int temp = x[i+1];
    x[i+1] = x[son] ;
    x[son] = temp ;

    int[] sonuc = {i+1, sayac};

    return sonuc ;
}

int[][] quickSort(int[] x, int bas, int son,int sayac){

    if(bas<son){
        int[] temp = parcala(x, bas, son);
        int pivot = temp[0];
        sayac+=temp[1];
        quickSort(x, bas, pivot-1, sayac);
        quickSort(x, pivot+1, son, sayac);
    }

    int[][]sonuc = {{sayac}, x};

    return sonuc;
}

```

```

int[][] countingSort(int[] x){

    int sayac = 0 ;

    int sonuc[] = new int[x.length+1] ;

    int max = x[0];
    for (int i = 0; i < x.length; i++) {
        if(x[i]>max)
            max = x[i];
    }

    int[] yardimci = new int[max+1];

    for(int i = 0 ; i < x.length ; i++){
        yardimci[x[i]]++;
        sayac++;
    }

    for (int i = 1; i < yardimci.length ; i++) {
        yardimci[i] += yardimci[i - 1];
        sayac++;
    }

    for (int i = x.length - 1; i >= 0; i--) {
        sonuc[yardimci[x[i]] - 1] = x[i];
        yardimci[x[i]]--;
    }

    for(int i = 0; i < x.length; i++) {
        x[i] = sonuc[i];
    }

    int[][] finalSonuc = {{sayac}, x};
    return finalSonuc;
}

```

Heap Sort İçin Heap Yapısı :

```
class Eleman {
    int icerik;

    Eleman(int icerik) {
        this.icerik = icerik;
    }
}

class Yigin {    // Heap Sort Algoritması Gerçekleştirmek İçin Heap Yapısı.

    Eleman[] dizi;
    int elemanSayisi;

    public Yigin(int boyut) {
        dizi = new Eleman[boyut];
        this.elemanSayisi = 0;
    }

    boolean bosMu() {
        return elemanSayisi == 0;
    }

    int yukariCik(int no) {
        int sayac = 0 ;

        int ata = (no - 1) / 2;
        while (ata >= 0 && dizi[ata].icerik < dizi[no].icerik) {
            sayac++ ;                // Yukarıya Çıkınca Sayaç Artıyor.
            Eleman tmp = dizi[ata];
            dizi[ata] = dizi[no];
            dizi[no] = tmp;
            no = ata;
            ata = (no - 1) / 2;
        }

        return sayac;
    }

    int ekle(Eleman yeni) {
        int sayac = 0 ;

        elemanSayisi++;
        dizi[elemanSayisi - 1] = yeni;
        sayac = yukariCik(elemanSayisi - 1);
    }
}
```

```

    return sayac ;
}

int asagiIn(int no) {

    int sayac = 0 ;

    int altSol = 2 * no + 1;
    int altSag = 2 * no + 2;
    while ((altSol < elemanSayisi && dizi[no].icerik < dizi[altSol].icerik) || (altSag < elemanSayisi
&& dizi[no].icerik < dizi[altSag].icerik)) {
        sayac++; // Aşağıya Doğru İnince Sayaç Artıyor.
        if (altSag >= elemanSayisi || dizi[altSol].icerik > dizi[altSag].icerik) {
            Eleman tmp = dizi[no];
            dizi[no] = dizi[altSol];
            dizi[altSol] = tmp;
            no = altSol;
        } else {
            Eleman tmp = dizi[no];
            dizi[no] = dizi[altSag];
            dizi[altSag] = tmp;
            no = altSag;
        }
        altSol = 2 * no + 1;
        altSag = 2 * no + 2;
    }
    return sayac ;
}

int[] azamiSil() {
    int sayac = 0 ;
    if (!bosMu()) {
        Eleman tmp = dizi[0];
        dizi[0] = dizi[elemanSayisi - 1];
        elemanSayisi--;
        sayac = asagiIn(0);

        int[] sonuc = {tmp.icerik, sayac};
        return sonuc;
    } else {
        return null;
    }
}
}

```

Bar Graf Sınıfı :

```
import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.RenderingHints;
import javax.swing.JPanel;

/**
 *
 * @author Malik
 */
public class grafModeli extends JPanel{
    algoModeli[] graflar ;
    int enBuyukKarmasiklik;
    int modelSayisi;

    public grafModeli(){
        super();
        setSize(850, 550);
        setLocation(0, 0);
        setBackground(new Color(255, 255, 255));
        graflar = new algoModeli[7];
        modelSayisi = 0 ;
    }

    public boolean doluMu(){
        return modelSayisi == graflar.length;
    }

    public void barEkle(algoModeli model){
        if(!doluMu()){
            if(enBuyukKarmasiklik<model.karmasiklik)
                enBuyukKarmasiklik = model.karmasiklik;

            graflar[modelSayisi] = model ;
            modelSayisi++;
        }
    }

    protected boolean algoModeliVarMi(String algoAdi){

        for(int i = 0 ; i < modelSayisi ; i++){
            if(graflar[i].algoAdi.equals(algoAdi))
                // Sıralama Algoritması Önceden Kullanılmış Olup Olmadığına Bakmak İçin.
```



```

        return true;
    }
    return false ;
}

public void istatistikleriSil(){
    graflar = new algoModeli[7];
    modelSayisi = 0 ;
    enBuyukKarmasiklik = -1 ;
    repaint();
}

@Override
public void paint(Graphics g){

    super.paintComponent(g);
    Graphics2D g2d = (Graphics2D) g ;
    g2d.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
RenderingHints.VALUE_ANTIALIAS_ON);
    g2d.setFont(new Font("Purisa", Font.PLAIN, 13));
    borderCiz(g2d);
    istatistikleriCiz(g2d);

}

public void borderCiz(Graphics2D g){
    g.setColor(Color.black);
    g.drawRoundRect(10, 20, 830, 300, 10, 10);
}

public void olcekleriCiz(Graphics2D g){

    g.setColor(Color.gray);
    for (int i = 10; i < 840; i+=20) {
        g.drawLine(i, 50, i+10, 50);
        g.drawLine(i, 118, i+10, 118);
        g.drawLine(i, 185, i+10, 185);
        g.drawLine(i, 253, i+10, 253);
    }

    g.drawString(Integer.toString(enBuyukKarmasiklik), 15, 45);
    g.drawString(Integer.toString((int) (enBuyukKarmasiklik*( double ) 3/4 )), 15, 113);
    g.drawString(Integer.toString((int) (enBuyukKarmasiklik*( double ) 1/2 )), 15, 180);
    g.drawString(Integer.toString((int) (enBuyukKarmasiklik*( double ) 1/4 )), 15, 248);

}

```

// Diziye Ve Grafi Sil.

// Tüm Grafiği Çizmek.

// Çerçeve Çizmek.

// Ölçekleri Çizmek.

```

public void istatistikleriCiz(Graphics2D g){
    if(modelSayisi<=0 || modelSayisi>7){
        return ;
    }

    Color[] renkler = {new Color(11, 89, 191), new Color(255, 173, 0), new Color(3, 138, 53), new
    Color(209, 3, 3),
        new Color(77, 18, 166), new Color(28, 171, 171), new Color(198, 34, 100)};
    // Graph Bar Renkleri.

    olcekleriCiz(g);

    double yukseklikOrani = (double) 270/enBuyukKarmasiklik;
    // Graph Bar Uzunlukları Ayarlamak İçin.
    int baslangicX = 390 - 60*(modelSayisi-1);

    for(int i = 0 ; i< modelSayisi ; i++){
        g.setColor(renkler[i]);
        g.fillRect(baslangicX+120*i, 320- (int)(graflar[i].karmasiklik*yukseklikOrani), 70,
(int)(graflar[i].karmasiklik*yukseklikOrani));
        g.drawString(graflar[i].algoAdi, baslangicX+120*i + ( 80 - 7*graflar[i].algoAdi.length() )/2 ,
340);
    }
    g.setColor(Color.black);

}

}

class algoModeli{
    int karmasiklik ;
    String algoAdi ; //Bir Graph Bar Yapısı
    boolean cizildiMi ;

    public algoModeli(int karmasiklik, String algoAdi) {
        this.karmasiklik = karmasiklik;
        this.algoAdi = algoAdi;
        cizildiMi = false ;
    }

}

```