Yıldız Teknik Üniversitesi

Ayrık Matematik Proje Ödevi

Öğrenci Bilgileri

Ad: Ekrem

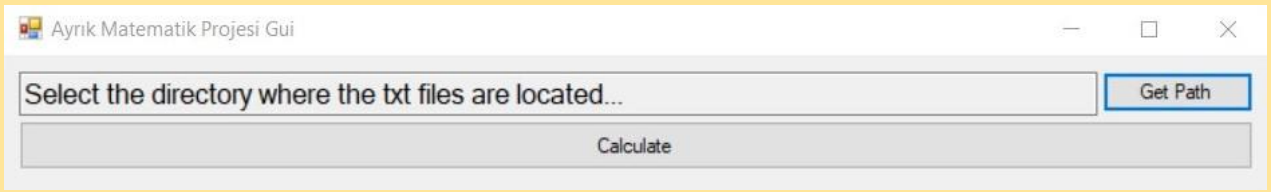
Soyad: Kamaz

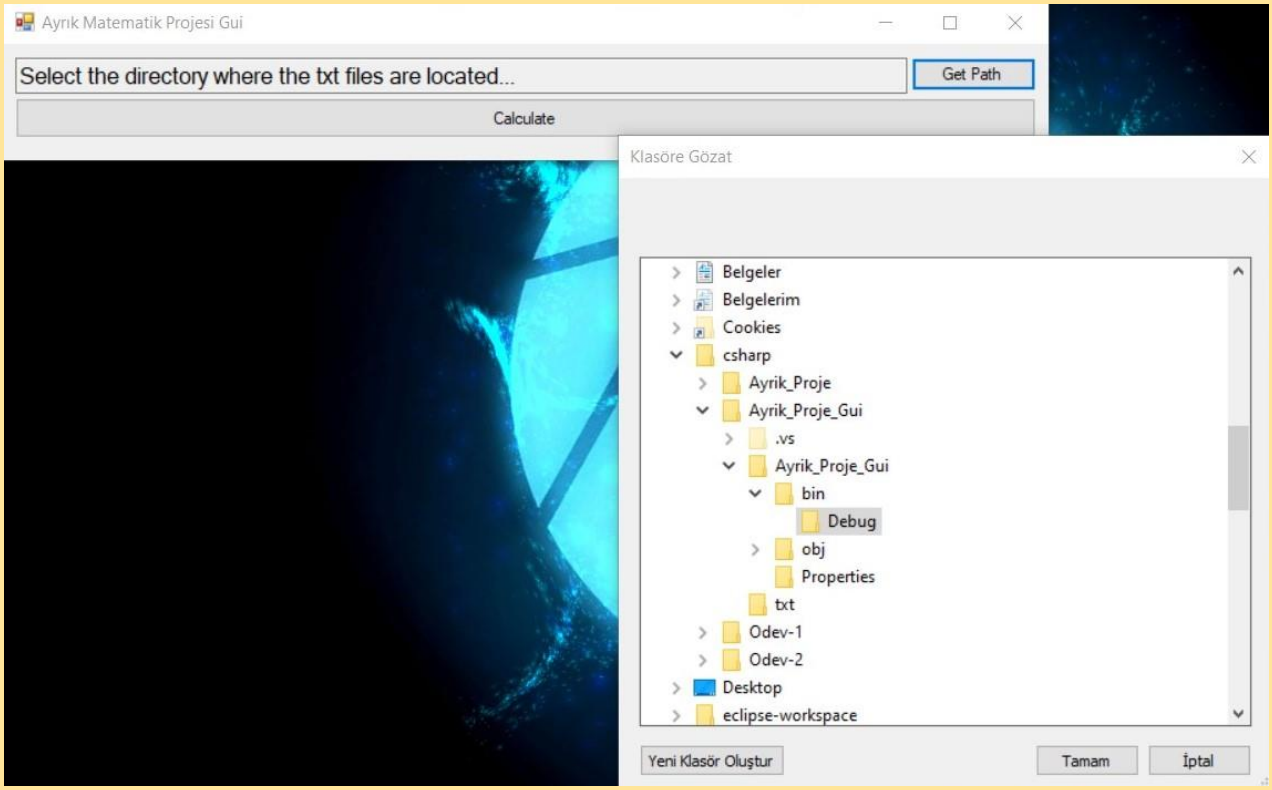
Numara: 18011033

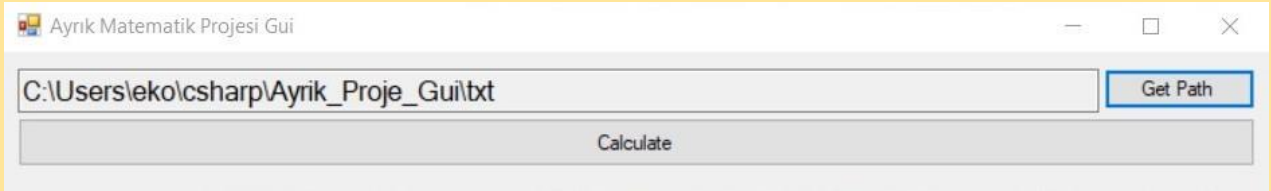
Mail: ekrem.kamaz@yandex.com

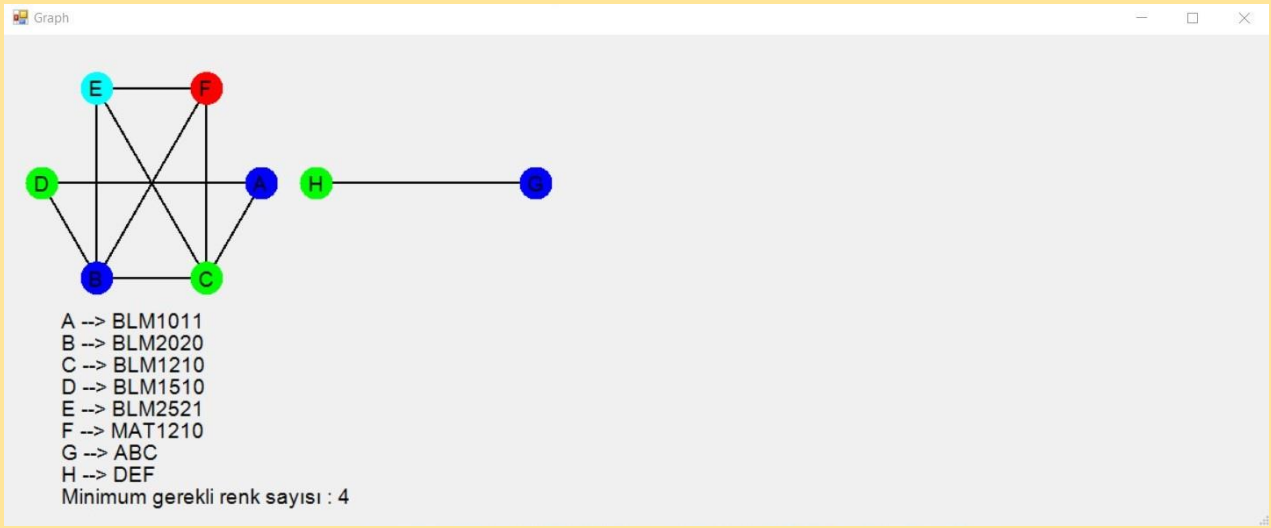
Kullanım

Açılan arayüzde “Get Path” butonu ile klasör tarayıcı arayüzü başlatılır. Varsayılan başlangıç adresi programın çalıştırıldığı dizindir. Buradan txt dosyalarının bulunduğu klasör seçilir ve “Tamam” butonuna basılır. Ardırndan “Calculate” butonuna basılır ve istenen çıktı görsel olarak ekrana yazdırılır.









Algoritma

Ödev için kendi algoritmamı oluşturdum. Bipartite graf tespit etmede kullanılan renklendirme metoduna benzer diyebilirim. Her dersi bir node nesnesi oluşturacak şekilde atadım. Ortak öğrencisi olan dersleri birbirlerinin komşuları olacak şekilde {nodeDegiskeni}.komsular isimli listelere atadım. Sonra bir düğümün renk değerine 0 vererek başladım. Sonra onun komsularını takip ederek "eğer renk değeri atanmamış ise" kontrolü yaparak diğer komşularda olmayan en düşük renk değeri vererek ilerledim. Ekrana çizdirmeden önce de kopuk olan grafları ayrı ayrı generic listeye daha ekledim ki görselliği güzel olsun.

Kod

Bu bölümde FormX.Designer.cs gibi dosyaları açıklamayacağım çünkü arayüzü Visual Studio 2019 IDE'sinin Toolbox'ını kullanarak yaptım.

Public Form1://Constructorında sadece `InitializeComponent()` fonksiyonunu çağırır.

```
private void button1_Click(object sender, EventArgs e)
```

Bu bölümde "`FileBrowserDialog`" nesnesi oluşturup çağırdım. Seçilen adresi de "`textbox1`" in içinde yazdırdım.

```
private void button2_Click(object sender, EventArgs e)
```

Bu bölümde "`Form2`" nesnesi oluşturup çağırdım. İçindeki "`String path`" değişkenine "`textBox1`" de yazılı değeri gönderdim.

Public Form2:// Constructorında en önemli fonksiyon olan Calculate() isimli fonksiyonu çağırır.

```
public void Calculate()
```

Satır 28'de Dersler.txt den alınan her satır verisi "`String[] dersler`" dizisine atanır.

Satır 29'daki döngü ile "`String[] dersler`" dizisinde gezilirek her ders ismi için "`dersListesi`" isimli generic listeye "`Node`" nesnesi oluşturulup eklenir.

Satır 33'de "`Node ders`" değişkeni yardımı ile "`dersListesi`" generic listesinde gezilir.

Satır 35'de dersin ismi ile kayıtlı olan text dosyasındaki öğrenci numarası verileri "`String[] ogrenciler`" dizisine atanır.

Satır 36'da "`ogrenciler`" isimli dizide gezilerek her veri nesnenin "`alanOgrenciler`" isimli generic listesine eklenir.

Satır 41'de "`komsulariEsle()`" fonksiyonu çağrılır.

Satır 42'de "`markAll()`" fonksiyonu çağırılır ve dönüş değeri "`renkSayisi`" değişkenine atanır.

```
static int markAll(List<Node> dugumler)
```

Satır 47'de içerisine gönderilen "`dugumler`" generic dizisinde gezilir.

Satır 50'de "`item`" nesnesinin "mark" değeri -1 mi (yani işaretlenmemiş mi) kontrol edilir. Her elemanı gezme sebebi ise "`mark()`" fonksiyonu kopukluk olan grafları komşuları takip ederek işaretlediği ve böylece her elemana ulaşamadığı için. Ve her elemanın "mark" değerini kontrol ederek en büyük değeri "`renkSayisi`" değişkenine atamak için.

Satır 53'de her elemanı da ayrı bir generic liste olan "`graflar`" isimli generic diziye eleman oluşturup eklenir. Bu dizinin var olma sebebi ise tek parça olmayan, kopuklukları olan grafları ayrı ayrı

dizilere ekleyerek ekrana çizdirme aşamasında kolayca yanyana ayrı graflar olarak çizebilmek. "tmp" isimli değişken yardımı ile güncel olarak kaçıncı parçada bulunulduğu verisi tutulur.

Satır 54'de "mark()" recursive fonksiyonu çağırılır ve işaretleme işlemi kopukluk olana kadar veya sonlanana kadar çalışmak üzere başlatılır. İçerisine gönderilen "graflar[tmp]" nin anlamı ise şudur. "tmp" bulunulan parçanın sayısını tutar. Parçalanma olduğu zaman ise "graflar" generic dizisine yeni eleman oluşturulup eklenir ve "tmp" 1 arttırılır. Burada "graflar[tmp]" ile güncel eklenecek graf listesine geçiş yapmış oluyoruz.

Satır 56'da bulunan "short if" yardımı ile en büyük "mark" değeri "renkSayisi" değişkenine atanır.

Satır 58'de "renkSayisi" (0'dan başladığı için) bir fazlası return edilir.

```
static void mark(Node dugum, int marker,List<Node> graf)
```

Satır 62'de "mark" değeri -1 olup olmadığı kontrol ediliyor. "markAll()" fonksiyonundayken kontrol etmiştik neden tekrar ediyoruz diye düşünebilirsiniz fakat satır 70'de fonksiyonumuz recursive çalıştığından işaretli olanlar için fonksiyondan çıkış sağlamış oluyoruz.

Satır 64'de "checkKomsular()" fonksiyonu false verene kadar "marker" değerimiz 1 arttırılır.

Satır 68'de güncel "marker" değeri, bulunulan "dugum" nesnesinin mark değerine atanır.

Satır 69'da parçalı graf çizimini kolaylaştırmak amacı ile kullandığımız "graflar" generic dizisinin, bulunulan güncel elemanına (elemanın kendisi de bir generic dizi) işaretlenen nesneyi ekliyoruz.

Satır 70'de bulunulan nesnenin her komşusunu gezerek aynı "graf" generic listesini göndererek "mark()" fonksiyonunu çağırıyoruz.

```
static bool checkKomsular(Node dugum, int marker)
```

İçine gönderilen "dugum" isimli nesnenin tüm komşularına sırayla bakılır ve herhangi biri "marker" değişkenine eşitse "true" return edilir. Eğer "marker" komşularda bulunamamış ise "false" return edilir.

```
static void komsulariEsle(List<Node> dersListesi)
```

"dersListesi" generic dizisinde bulunan bütün nesnelerde gezilerek her nesne ikili olarak kıyaslanır. Eğer "searchStudentConflict()" fonksiyonu birbirleri için "true" döndürürse birbirlerinin "komsular" isimli generic dizilerine eklenir.

```
static bool searchStudentConflict(Node ders1, Node ders2)
```

Her ders nesnesinin "ogrenciler" generic dizilerinde gezer ve eşleşme olan dersler için "true" return eder.

```
private void Form2_Paint(object sender, PaintEventArgs e)
```

Satır 130'da her graf parçasında gezilir ve o graf parçasında bulunan düğümlere koordinat değerleri atanır. Koordinat değerleri nesnenin içerisindeki "coordX" ve "coordY" değişkenlerinde tutulur. Koordinat adresleri verilirken her düğüm bir çemberin eksenindeymiş gibi yerleştirilir. Bunun sebebi her hangi 2 düğümün üst üste çakışmamasıdır. Bunu sağlayan en akıllıca yöntem çember ekseninde yerleştirmektir. Bunu yaparken de çemberin merkezinin koordinatları "konumMerkeziX" ve "konumMerkeziY" değişkenlerinde tutulur. Bu merkez adresleri trigonometri yardımı ile düğümlerin koordinatlarının hesaplanmasını sağlar. Eğer grafta parçalanma var ise yeni parçayı sağ tarafa ayrı graf olarak çizmek için "konumMerkeziY" 250 arttırılarak yeni grafdaki düğümlerin yeni çember merkezini baz alması sağlanır.

Satır 139'da her graf parçasında gezilir ve o graf parçasında bulunan düğümlerin koordinat değerlerinden komşularının koordinat değerlerine doğru çizgi çizilir.

Satır 149'da her graf parçasında gezilir ve o graf parçasında bulunan düğümlerin koordinatlarında, içerisinde belirtilen renkte bir daire oluşturulur. Bu işi yaparken fonksiyonu biraz karmaşıklaştırdım (Satır 153) çünkü renk sayısını bir listeden çekmek değil de RGB renk değerlerini otomatik oluşturmak istiyordum. Bunu yaparken epey zorlandım çünkü değerlerin hiçbir zaman 0,0,0 (siyah) ve 255,255,255 (beyaz) olmaması, ve nesnelerde tutulan "`mark`" değerinin RGB renk kodu karşılığı her aynı "`mark`" değeri ile özdeş her farklı "`mark`" değerinden de farklı olmalıydı. Aklıma gelen ilk yöntem "`mark`" değerini binary değerine çevirip her basamağın 255 ile çarpımını RGB değerlerine dağıtarak kullanmak oldu. Bu oldukça işlevliydi fakat sadece maksimum 6 farklı renge kadar kullanılabilirdi. Çünkü üç haneli sayıdan 8 ihtimalimiz vardı, siyah ve beyaz renklerini iptal ederek bunu da 6 ya düşürmüş olduk. Kalan ihtimaller:

{(0,0,1), (0,1,0), (0,1,1), (1,0,0), (0,0,1), (1,1,0)}

Binary kodda böyle bir durum olduğu için onun yerine 3 tabanında sayıya dönüştürerek her sayının kesirli karşılığını 255 ile çarparak kullanmayı denedim. Böylece program binary deki ile aynı mantık çerçevesinde 25 renge kadar destek veriyordu. (Desteklenen renk sayısı = n^3 – 2 , n: sayı tabanı, 2: siyah ve beyaz) Fakat renkler binary dekine göre daha cansız ve daha yakın tonlardaydı. Ayırt etme konusunda engel olmuyordu fakat ben ikisini birden kullandım ki fazla renk gerekmedikçe güzel görünen yöntemi kullansın. Bunu da bir "`short if`" fonksiyonu kullanarak yaptım.

Satır 155'de ise her düğümün, oluşturulduğu sırada constructor'u tarafından atanmasını sağladığım "`nickName`" değerini dairenin üzerine yazdırdım.

Satır 158'de her düğümün "`nickName`" değerlerinin karşılığında bulunan "`name`" değerlerinin açıklaması yazdırılır.

Satır 163'de de en son kullanılan renk sayısı bilgisi yazdırılır.