Kocaeli Üniversitesi

Bilgisayar Mühendisliği Bölümü

Programlama Laboratuvarı I

Minimum Çevreleyen Çember – B-spline

Rumeysa ÜSTÜN

rmys.ustn2001@gmail.com

Proje Özeti:

Programlama Laboratuvarı I dersinde bizden, dosyadan çekilen koordinatları iki boyutlu düzlemde gösteren, bu noktaları çevreleyen en küçük çemberi çizen ve bu noktalardan veya bu noktaların yakınından geçen eğriyi çizdirmemiz istendi.

Bunlar için bize sunulan arayüz tasarım kütüphanelerinden Graphics.h kütüphanesini kullandım. Graphics.h kütüphanesini ise kullanım kolaylığı ve proje yeterliliği için seçtim.

Bu proje için C programlama dilini ve Dev c++ geliştirme ortamını kullandım.

Noktalardan geçen eğriyi ve minimum çevreleyen çemberi ayrı ayrı iki ekranda gösterdim. Programı çalıştırdığımızda öncelikle minimum çevreleyen çember ortaya çıkıyor ve daha sonra klavyeden girilen herhangi bir tuş ile yeni ekranda noktalardan geçen eğriyi çizdiriyor.

İki boyutlu düzlemde, noktaları yerleştirirken noktaların hangi sayıya denk geldiğini görmek için kesikli çizgileri kullandım.

Minimum çevreleyen çemberin merkezini, camgöbeği renginde ayarladım ve aynı renkte de sağ üst köşede merkez noktasının koordinatları yazdırdım. Aynı şekilde açık kırmızı renginde yarıçapı ayarladım ve aynı renkte de sağ üst köşede yarıçapını yazdırdım.

Sol üst köşede ise eğri veya çember çiziminden hangisinin gösterildiğini yazdırdım.

Çizimleri kalın çizgilerle çizdirdim çünkü koordinat düzleminde ortaya daha çok çıkmasını istedim.

1.GİRİŞ

Bu raporda, verilen noktalar ile minimum çevreleyen çemberi, noktalardan geçen eğri hesaplamaları, çizdirme aşamaları, deneysel sonuçlar, karmaşıklık analizi ve kaba kod bulunmaktadır.

Projemi C dili ile yazdım. Geliştirme ortamım ise Dev C++ idi.

Kullandığım arayüz ise graphics.h kütüphanesi idi. Graphics.h kütüphanesi, kullanım kolaylığı ve yeterlilik bakımından çok işlevsel bir arayüz programıdır.

2.YÖNTEM

2.1 Graphics.h kütüphanesi

Kullandığım Graphics.h kütüphanesinin projede bulunan fonksiyonlar aşağıdaki gibidir.

initwindow(x,y) = Girilen x değeri yatay uzunluğunu, y değeri ise dikey uzunluğunu veren boş bir pencere açmaktadır.

line(x1,y1,x2,y2) = Açılan penceredeki (x1,y1) noktasından (x2,y2) noktasına kadar düz bir çizgi çizmeye yarar. Bu fonksiyonu x ve y eksenlerini çizmek için, sayıların yerini gösteren kısa çizgiler için ve noktaları gösteren kesikli çizgileri çizmek için kullandım.

setcolor(a) = Bu fonksiyon girildiği zaman aşağısında çizilecek/yazılacak her şeyi a değerindeki renk ile yazar/çizer. a değişkeninin alabileceği değerler ve değerlerinin karşılıkları (Tablo-1)'de bulunmaktadır.

settextstyle(a,b,c) = Bu fonksiyon girildiği zaman aşağısındaki yazıların hepsinin değerlerinin değiştirir. a değişkeni yazı tipini, b değişkeni dikey/yatay düzlemde yazdırmasını, c değişkeni ise yazı tipi boyutunu ayarlar. a değerinin alabileceği değerler (Tablo-2)'deki gibidir.

outtextxy(x,y,char_yazı) = Penceredeki (x,y) noktasından başlayarak char_yazıyı yazdırır. char_yazı, char olmalıdır, yoksa hata verir. Bu fonksiyonu koordinat düzlemindeki sayıları yerleştirmek için ve ekrandaki bilgi çıkışları için kullandım.

fillellipse(x,y,r1,r2) = Merkezi (x,y) noktası olan ve yarıçapları r1, r2 olan içi dolu bir elips çizer. Bu fonksiyonu noktaları çizmek için kullandım. Çünkü nokta koymaya yarayan fonksiyon yetersiz kaldı.

circle(x,y,r) = Merkezi (x,y) noktası olan ve yarıçapı r olan bir çember çizmeye yarar. Bu fonksiyonu minimum çevreleyen çemberi çizmek için kullandım.

setlinestyle(a,b,c) = Bu fonksiyon girildiği zaman aşağısındaki çizilenlerin hepsinin ayarını değiştirir. a değişkeni çizgi tipini (kesikli, düz…) belirler. b değişkeni ise sadece a değişkeni 4 parametresini aldığında çalışır. Kesikli çizgilerin uzunluğunun 16 bite kadar çizmesine yarar. c değişkeni ise kalınlığı sağlar. c, 1 ise ince, 3 ise kalın çizer. a değişkeninin alabileceği değerler (Tablo-3)'teki gibidir.

getch() = Klavyeden bir tuşa basılana kadar var olan ekranı gösterir. Tuşa basıldığında diğer ekrana geçer. Bu fonksiyonu çember ve eğriyi farklı koordinatlarda çizmek ve bu koordinatların arasındaki geçişi sağlamak amacıyla kullandım.

clearviewport() = Bu fonksiyon ise ekrandaki her şeyi temizler. Diğer koordinata geçiş için kullandım.

closegraph() = Pencereyi kapatır.

İsim	Değer	İsim	Değer
SOLID_LINE	0	DASHED_LINE	3
DOTTED_LINE	1	USERBIT_LINE	4
CENTER_LINE	2		

(Tablo-3)

Renk	Değer	Renk	Değer
Siyah	0	Koyu Gri	8
Mavi	1	Açık Mavi	9
Yeşil	2	Açık Yeşil	10
Camgöbeği	3	Açık camgöbeği	11
Kırmızı	4	Açık Kırmızı	12
Eflatun	5	Mor	13
Kahverengi	6	Sarı	14
Açık Gri	7	Beyaz	15

(Tablo-1)

İsim	Değer	İsim	Değer
DEFAULT_FONT	0	SIMPLEX_FONT	6
TRIPLEX_FONT	1	TRIPLEX_SCR_FONT	7
SMALL_FONT	2	COMPLEX_FONT	8
SANS_SERIF_FONT	3	EUROPEAN_FONT	9
GOTHIC_FONT	4	BOLD_FONT	10
SCRIPT_FONT	5		

(Tablo-2)

2.2 Dosyadan Veri Çekme

Dosyadan veri çekmek için öncelikle "dosya.txt" adlı dosyayı okuttum. Sonrasında dosyadaki her şeyi bir char dizisine attım. Böylelikle char dizisinde artık noktaları alabilecek duruma geldim.

Noktaları alırken for ile sürekli olarak rakam veya "-" işaretinin olup olmadığının kontrolünü yaptım. Eğer rakamsa birden fazla basamağı var mı diye, eğer "-" işareti ise rakamın basamaklarını almak için for'un içinde bir for daha yaptım. Aldığım sayıları bir char dizisine atadım. Bütün basamakları aldığım zaman char dizisini integer bir değere döndürüp noktalar matrisine atadım.

Noktalar matrisinin tam olarak ifadesi şu şekildedir:

Noktalar[indis][0/1] → indis denilen sayı kaçıncı nokta olduğunu verir. 0/1 olması ise x veya y olduğunu gösterir. 0 ise x olduğunu, 1 ise y olduğunu ifade ediyor. Örneğin:

Noktalar[6][1] →6. İndisteki noktanın y'si

2.3 Kullanılan B-spline yöntemi

Benim kullandığım b-spline yöntemi, kuadratik b-spline yöntemidir. Bu yöntemin belli formülleri vardır.[1] Ben bu formülleri ise koda çevirme işlemini yaptım.

Kuadratik b-spline interpolasyonu, verilen noktalardan geçen eğrinin denklemini oluşturur. Ben de bu denklemi bulup x'i verilen noktaların en küçük x'inden başlatıp sürekli olarak 0.0001 arttırdım. Ve her x için bir nokta koydurdum. Böylelikle noktalar birleşerek bir eğri meydana getirdiler.

Öncelikle bulmamız gereken denklemin yapısı şu şekildedir:

$$S_i(x) = a_i + b_i(x-x_i) + c_i(x-x_i)^2 \quad \textbf{(0)}$$

Bu denklemdeki a_i, b_i, c_i değerlerini bulmak için kaynakçada belirttiğim makaleden yararlandım. Sonuç olarak karşıma şu denklemler çıktı:

$a_i = y_i$ **(1)**

$x_{i+1} - x_i = h_i \rightarrow a_i + b_i h_i + c_i h_i^2 = a_{i+1}$ **(2)**

$b_i + 2c_i h_i = b_{i+1}$ **(3)**

$c_0 = 0$ **(4)**

İlk olarak h_i değişkenlerini buluruz. Sonra (4) denklemini (2) denkleminin i=0 durumu için yerine koyarız ve b_0'ı buluruz. Daha sonrasında (3) denkleminde i=0 durumunda bilinenleri yerine koyduğumuzda b_1'i elde ederiz. Böylelikle tekrar (2) denklemine döneriz ve bu sefer i=1 için c_1'i buluruz. Tekrar (3), sonrasında (2) derken bütün bilinmeyenleri bir döngüde buluruz.

Bütün bilinmeyenleri bulduktan sonra (0) denkleminde yerine koyarız. Artık x yerine bir değer koyduğumuzda karşımıza bir değer çıkar. Bu denklem bize girilen iki nokta arasındaki eğrinin denklemini verir. Bu denklemi, verilen iki nokta arasındaki küçük x'ten başlayarak x'i sürekli 0.0001 arttırarak devamlı y değerlerini buldurursak ve bu noktaları sürekli olarak grafikte yerine koyarsak karşımıza bir spline çıkıyor. Bu spline'a kuadratik b-spline denir.

Bu hesaplamalar, grafiğin ilk aralığını lineer çizmesine sebep oluyordu. Bu nedenle ben 0. indise en küçük x'in 0.001 kadar azını atadım ve bu problemi böylece çözmüş oldum.

2.4 Minimum Çevreleyen çember çizimi

Bu çemberi çizmek için ilk düşündüğüm şey en uzak iki noktanın çap olması idi. Fakat yazdığım zaman bazı noktalarda hata aldım.

Bu nedenle araştırmaya koyuldum. Birçok internet sitesinden sonra bir videoya[2] denk geldim ve bu videodan öğrendiklerimle üzerine birkaç araştırmamdan sonra nihai sonuca vardım.

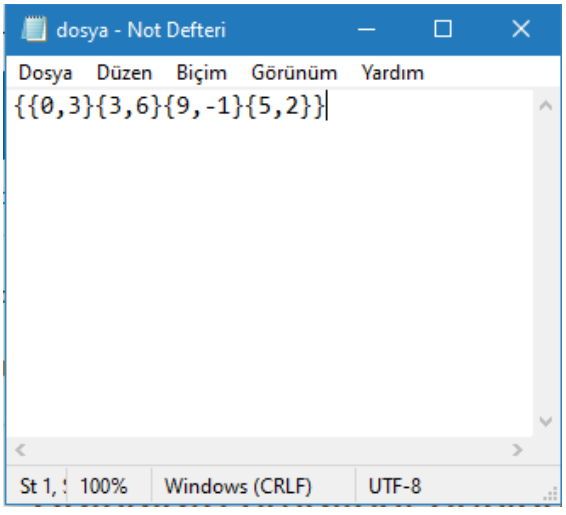
Çemberi çizmenin mantığı kısaca şöyledir. İlk başta verilen x ve y koordinatlarının ortalaması alınarak buna merkez denir. Bunun yapılmasının sebebi çemberin içinde bir nokta seçmek, çünkü videoda da dediği gibi uzakta nokta seçmek hesabı zorlaştırır.

Bu noktayla ilk indisteki nokta ile arasındaki uzaklığı buluyor ve buna yarıçap diyor. Daha sonra bu merkezi diğer noktalar ile uzaklığını hesaplıyor. Eğer yarıçap denilen değişken diğer noktalarla uzaklığından küçükse karşılaştırılan diğer uzaklık yarıçap oluyor. Bununla birlikte ortalama ise ilgili nokta ile çıkarılıp sabit bir sayıyla çarpılıp kendisiyle toplanıyor. Yani böylelikle merkez noktasını sürekli olarak değiştirmiş ve çember gittikçe küçülmüş oluyor.

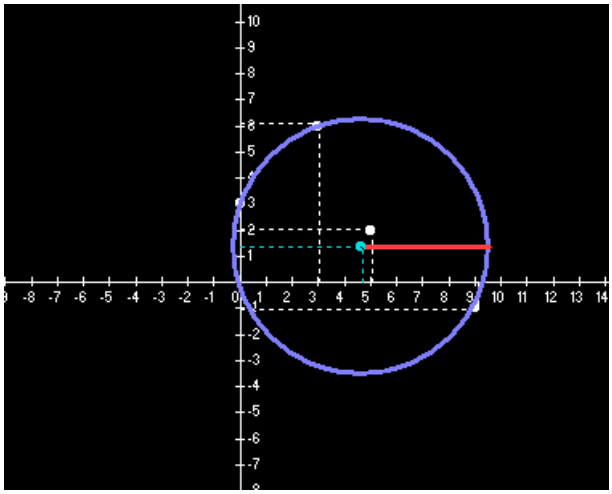
Bunu yaklaşık 20 000 defa yapıyor ki en doğru sonuca ulaşabilsin. En sonunda ise bize doğru merkezi ve yarıçapı veriyor.

3. DENEYSEL SONUÇLAR

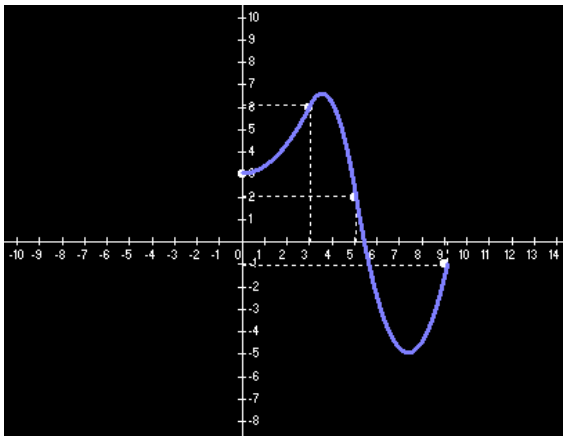
Noktaları girme:



Minimum çevreleyen çember:



Noktaların üzerinden geçen eğri:



Verilen çıktı:

```
Dosyanin icindekiler: {{0,3}{3,6}{9,-1}{5,2}}

Merkez noktasi: ( 4.654, 1.346 )
Yaricapi: 4.939
--------------------------------
Process exited after 152.3 seconds with return value 0
Press any key to continue . . .
```

4. SONUÇ

Bu proje sayesinde matris ve dizileri daha rahat kullanmayı öğrendim. Ayrıca interpolasyonun ne olduğunu, nasıl hesaplandığını, minimum çevreleyen çemberin nasıl hesaplandığını, var olan formülleri nasıl koda çevirebilceğini keşfettim. Kaba kod, yalancı kod yazmayı öğrendim. Karmaşıklık analizi yapmayı öğrendim. Bana gerçekten çok şey kattı, teşekkürler.

5.KAYNAKÇA

[1] Spline İnterpolasyonu Makale → https://acikders.ankara.edu.tr/mod/resource/view.php?id=2286

[2] Çevreleyen Çember Video→ https://www.youtube.com/watch?v=WGOSX6d17TU&feature=youtu.be

[3] Kaba Kod Yazımı → https://www.mustafayemural.com/c-my000012/

https://www.elektrikport.com/teknik-kutuphane/yalanci-kod-nedir/14925#ad-image-0

Readme Nasıl Yazılır → https://tr.wikipedia.org/wiki/README

Karmaşıklık analizi → https://www.slideshare.net/DenizKILIN/yzm-2116-blm-2-algoritma-analizi

Grafik kullanımı → http://kadirselen.blogspot.com/2014/01/grafik-kullanm.html

http://www.erdiyurdakul.com/graphics-h-kutuphanesinin-fonksiyonlari/

https://www.geeksforgeeks.org/setcolor-function-c/

https://www.geeksforgeeks.org/setlinestyle-function-c/

https://fatihalparslann.wordpress.com/2017/02/02/c-dilinde-graphics-h-kutuphanesi-1-bolum/

Grafik yükleme kurma → https://drive.google.com/file/d/1DdLhqOUfz95W8nG3weN_KUcYQn_TUSL7/view

https://www.youtube.com/watch?v=H5bjXTz-HHo&feature=emb_logo

Çevreleyen Çember Kaynak → https://www.nayuki.io/res/smallest-enclosing-circle/computational-geometry-lecture-6.pdf

https://www.geeksforgeeks.org/minimum-enclosing-circle-set-1/#:~:text=A%20minimum%20enclosing%20circle

%20is,circle%20or%20on%20its%20boundaries.&text=Explanation%3A,inside%20or%20on%20the%20circle.

https://www.cs.mcgill.ca/~cs507/projects/1998/jacob/problem.html

KARMAŞIKLIK ANALİZİ

Karmaşıklık analizi yapmak için döngüleri tek tek inceleyeceğim çünkü diğer kısımlar sabit olarak alındığı için bir katkısı olmuyor.

1. **Döngü:** Bu döngü dosya içeriği kadar döner. Dosya içeriğine N dersek karmaşıklık O(N) olur

```
while(!feof(dosya))//Dosyanın sonuna g
{
    dosya_icerigi[i]=fgetc(dosya);  //
    i++;
}
```

2. **Döngü:** Burada iç içe döngü bulunmaktadır. Dıştaki döngü n defa döner. İçerideki ilk döngü de n defa döner. İçerideki ikinci döngü de n defa döner. İç içe olduğu zaman çarparız. Alt alta ise en büyüğünü alırız. İkisi de aynı olduğu için n olarak alıyoruz ve dışarıdaki döngü ile çarpıyoruz. Böylelikle karmaşıklığımız $O(N^2)$ oluyor.

```
for(int j=0,n=strlen(dosya_icerigi); j<n ; j++) //Dosy
{
    bayrak=0;
    sayi=0;

    if(isdigit(dosya_icerigi[j]) || dosya_icerigi[j]==
    {
        nokta[sayi]=dosya_icerigi[j]; //Rakamı noktan
        bayrak=1;
        for(int b=j+1; b<n; b++) //Eğer birden fazla b
        {
            sayi++;
            if(!isdigit(dosya_icerigi[b]))
            {
                j++;
                break;
            }
            nokta[sayi] = dosya_icerigi[b];
            j++;
        }
    }
    if(bayrak==1)
    {
        noktalar[a/2][a%2]=atoi(nokta); //Basit bir m
        a++;
    }
    for(int r=0; r<n; r++) //Nokta dizisini daha sonra
        nokta[r]=NULL;
}
```

3. **Döngü:** Burada iç içe döngü bulunmaktadır. Dışarıdaki döngü 20000 defa döner içerideki döngü de bir N sayısı kadar döner. İkisini çarptığımız zaman 20000N olur fakat karmaşıklıkta katsayıların önemi olmadığı için direkt olarak N alacağız. Böylece karmaşıklık O(N) oluyor.

```
for (int r = 0; r < 20000; r++)
{
    yaricap = uzaklik(X_ortalama - noktalar[0][0], Y_ortalama - noktalar[0][1]);
    indis=0;
    for (int j = 1; j < (a/2); j++)
    {
        uzak2 = uzaklik(X_ortalama - noktalar[j][0], Y_ortalama - noktalar[j][1]);
        if (yaricap < uzak2)
        {
            indis = j;
            yaricap = uzak2;
        }
    }
    X_ortalama = X_ortalama + (noktalar[indis][0] - X_ortalama)*sbt;
    Y_ortalama = Y_ortalama + (noktalar[indis][1] - Y_ortalama)*sbt;
    sbt = sbt * 0.999;
}
```

4. **Döngü:** Burada iki döngü de sabit değerdedir. Bu nedenle bu döngüleri hesaba katmadım.

```
for(float i=227.5; i<=1072.5; i=i+16.25)
{
    sprintf(sayi1,"%d",c);
    line(i,354.5,i,360.5);//x ekseninin kısa çizgilerini çizer.
    outtextxy(i-6,361.5,sayi1);//x ekseninin sayılarını yerleştirir.
    c++;
}

for(float i=32.5; i<=682.5; i=i+16.25)
{

    if(!b==0)
    {
        sprintf(sayi1,"%d",b);
        outtextxy(654,i-6,sayi1);//y ekseninin sayılarını yerleştirir.
    }

    line(647,i,653,i);//y ekseninin kısa çizgilerini çizer.
    b--;
}
```

5. **Döngü:** Buradaki döngü n defa dönmüştür. Bu nedenle karmaşıklığı O(N) buluruz.

```
for(int i=0; i<a/2; i++)
{
    setlinestyle(1,3,1);
    if(noktalar[i][0]!=0)
        line(650+(noktalar[i][0]*16.5),357.5,650+(noktalar[i][0]*16.5),357.5-(noktalar[i][1])*16.5);//Nokt
    if(noktalar[i][1]!=0)
        line(650,357.5-(noktalar[i][1])*16.5,650+(noktalar[i][0]*16.5),357.5-(noktalar[i][1])*16.5);//Nokt
    setlinestyle(0,0,3);
    fillellipse((20+(noktalar[i][0]/2.0))*32.5,(11-(noktalar[i][1]/2.0))*32.5,2,2);//Noktaları gösterir.
}
```

6. **Döngü:** Burada iç içe 2 döngü bulunmaktadır. İçerideki de dışarıdaki de N defa dönmüştür. Yani karmaşıklık $O(N^2)$ olarak buluruz

```
for(int e=0; e<a/2; e++)
{
    for(int o=e+1; o<a/2; o++)
    {
        //Burada e. indexteki noktanın x'ini dizinin e. indexten büyük

        if(noktalar[e][0]>noktalar[o][0])
        {
            //Noktanın x'lerinin yerini değiştirir.
            temp=noktalar[e][0];
            noktalar[e][0]=noktalar[o][0];
            noktalar[o][0]=temp;
            //Noktanın y'lerinin yerini değiştirir.
            temp=noktalar[e][1];
            noktalar[e][1]=noktalar[o][1];
            noktalar[o][1]=temp;
        }
        else if(noktalar[e][0]==noktalar[o][0]) //x'ler eşitse y'lere
        {
            if(noktalar[e][1]>noktalar[o][1])
            {
                //Noktanın x'lerinin yerini değiştirir.
                temp=noktalar[e][0];
                noktalar[e][0]=noktalar[o][0];
                noktalar[o][0]=temp;
                //Noktanın y'lerinin yerini değiştirir.
                temp=noktalar[e][1];
                noktalar[e][1]=noktalar[o][1];
                noktalar[o][1]=temp;
            }
        }

    }
}
```

7. **Döngü:** İç içe döngü vardır. Bu nedenle karmaşıklık $O(N^2)$ olur.

```
for(int nt=0; nt<(a/2)-1; nt++)
{
    for(float nt2=noktalar[nt][0]; nt2<noktalar[nt+1][0]; nt2+=0.0001)
    {
        hi[nt]=noktalar[nt+1][0]-noktalar[nt][0];
        bi[nt]=bi[nt-1]+2.0*ci[nt-1]*hi[nt-1];
        ci[nt]=(noktalar[nt+1][1]-noktalar[nt][1]-bi[nt]*hi[nt])/pow(hi[nt],2);
        s=noktalar[nt][1]+bi[nt]*(nt2+0.001-noktalar[nt][0])+ci[nt]*pow((nt2+0.001-noktalar[nt][0]),2);
        line(650+(nt2)*16.5,357.5-s*16.5,650+(nt2+0.001)*16.5,357.5-s*16.5); //Burada eğrinin x'ine sürekli
    }
}
```

Sonuç olarak bu döngüler alt alta sıralandığı için içlerinden en büyük olanını alırız. Ve böylece zaman karmaşıklığımız $O(N^2)$ olur.