**Minimum Çevreleyen Çember**

**ve B-spline Çizimi**

Feyzanur Yeşildal          Hilal Fatma Gül Nişancı

190201085                            190201041

Bilgisayar Mühendisliği Bölümü

Kocaeli Üniversitesi

[hilalfnisanci@gmail.com](mailto:hilalfnisanci@gmail.com) , [feyzanuryesildal@gmail.com](mailto:feyzanuryesildal@gmail.com)

**Özet:**

Point.txt dosyasında 2 boyutlu düzlemde N tane nokta vardır ve bu noktalar {x,y} değerlerini tutar. Programda, dosyadan bu noktaları okuyarak minimum çevreleyen çember ve N tane noktanın en yakınından geçen eğri çizdirilir. Noktaların ve çemberin koordinat bilgileri iki boyutlu koordinat sisteminde yazdırılır. Çemberin yarıçapı ve merkezi de hesaplanarak ekrana yazılır. Arayüz tasarımı için allegro kütüphanesinden yararlandık.

**Giriş:**

Projenin amacı, dosyadan 2 boyutlu düzlemde N tane noktayı okuyarak minimumu çevreleyen çemberi hesaplamak. Aynı zamanda N tane nokta için, noktaların en yakınından geçen eğriyi hesaplamak ve bunları arayüzü kullanarak ekrana çizdirmek.

Ekrana, noktalar tarafından oluşturulan çemberin merkezi ve yarıçapı da yazdırılmıştır. Ekrana koordinat sistemi ve koordinat sistemine denk gelen sayılar da yazdırılmıştır.

Program c dilinde geliştirilmiştir ve arayüz tasarımı için allegro kütüphanesi kullanılmıştır. Geliştirme ortamı olarak CodeBlocks idesi kullanılmıştır.

**Yöntem:**

Öncelikle N tane nokta bulunduran Points.txt dosyasından dosya okuma  yapılmıştır. Dosyada ki her noktanın x ve y noktası çift boyutlu bir diziye aktarılmıştır ve bu dizi ekrana yazdırılmıştır. Allegro kütüphanesinden yararlanılarak noktalar ekrana çizdirilmiştir ve her noktanın koordinat bilgisi noktanın yanına yazılmıştır. .

Minimum çevreleyen çemberi bulurken N>3 için ve N<3 için ayrı yollar izledik.

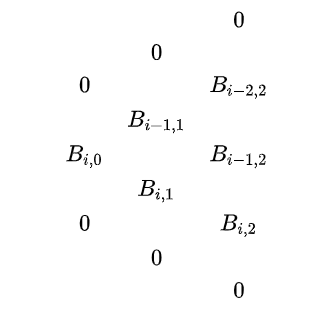
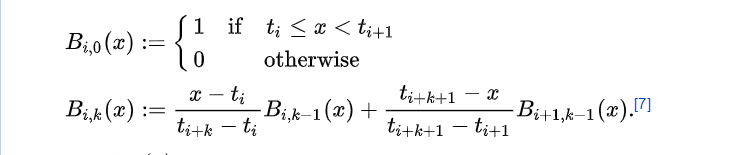
1.durum: N>3 durumunda noktalar arasından en dıştaki noktaları bulduk. Yani x’i en büyük olan nokta ve x’i en küçük olan nokta ile y’si en büyük olan nokta ve y’si en küçük olan nokta bulunmuştur. Bu noktaları baz alarak bir dörtgen mantığı kurduk. Dörtgenin merkezini en dış noktaların koordinat bilgilerinden yararlanarak hesapladık. Dörtgen merkezine en uzak olan 3 tane noktayı uzaklık() fonksiyonu oluşturarak hesapladık ve bu 3 noktanın aralarındaki uzaklığı bulduk. Buradaki amaç en büyük olan mesafenin çap olup olamayacağını kontrol etmektir. En büyük mesafeye sahip olan noktaların dışında kalan noktanın, olası çap uzunluğunun orta noktası ile arasındaki mesafeye göre çemberi ve yarıçapı oluşturduk.

2.durum: N<3 için aslında 3 durum var. N=1, N=2, N=3 durumları.  N=1 durumu bir çember belirtmediği için ekrana sadece noktanın kendisi çizdirilmiştir. N=2 durumu bir çember için gereken en az nokta sayısıdır. Bu iki nokta çemberin çapını belirtir ve aralarındaki mesafenin yarısı yarıçaptır. Bu iki noktadan geçen bir doğru düşünürsek bu doğrunun orta noktası bize merkez bilgilerini vermektedir (Ox = (X0 + X1)/2, Oy = (Y0 + Y1)/2). N=3 durumu için noktaların bir üçgen oluşturduğunu düşündük ve bu üçgenin çevrel çemberini hesapladık. Fakat üçgen geniş açılı ise minimum çember oluşturmayacaktır. Bu yüzden bu 3 nokta arasındaki mesafelere bakarak en büyük olanı çap yaptık ve yarıçap hesapladık. Eğer üçgen geniş açılı değilse çevrel çember merkezini kartezyen koordinat formülünden yararlanarak hesapladık. Bu durum için yarıçapı Heron formülünden yararlanarak bulduk.

Bulduğumuz merkez ve yarıçap bilgilerini kullanarak ve allegro kütüphanesinde bulunan çember fonksiyonu ile ekrana minimum çevreleyen çemberi ve merkezini çizdirdik.

B-spline eğrisini hesaplamak için öncelikle noktaları, x bilgilerine bakarak küçükten büyüğe doğru sıraladık. Eğer x koordinatları aynı olan noktalar varsa bu noktaların y koordinatlarına bakarak küçükten büyüğe doğru sıraladık. Böylelikle eğriyi oluşturmak için kullanacağımız kontrol noktalarını elde etmiş olduk.

B-spline eğrisini eğrilerin birleşimi şeklinde değil de küçük noktaların birleşimleri şeklinde oluşturduk. Bunun için bir döngü içerisinde t değişkeni tanımladık ve t değişkenini 0.0005 arttırarak spline çizdirmeye çalıştık. Her nokta için bir x ve y bilgisi olmalıydı bunun için x ve y hesaplayan iki fonksiyon oluşturduk ve döngü içerisinde her seferinde bu fonksiyonları çağırarak x-y bilgilerini geri döndürdük.  Bu fonksiyonlar içindeki x-y bilgilerini b-spline eğrisinin tanımında yer alan özyinelemeli formülden hesapladık. Formüldeki değerleri kullanarak denklemlerimizi oluşturduk ve her 4 kontrol noktası ile bu denklemler arasında çarpım yapıp x ve y bilgilerini elde ettik. Matematiksel formüller aşağıda verildiği gibidir.



**Karşılaştığımız sorunlar ve çözümler:**

İlk karşılaştığımız sorun grafik için kullanacağımız kütüphanelerle ilgili oldu. Türkçe kaynak yetersizliğinden dolayı çok fazla araştırma yapmamız gerekti. Kütüphaneyi CodeBlocks’a eklerken birçok hata ile karşılaştık. İlk başta Graphics.h ve OpenGL kütüphanelerini kullanmaya çalıştık fakat aldığımız hatalar sonucu Allegro kütüphanesini kurduk.

Allegro kullanırken koordinat sistemi pencerenin sol üst köşesindeydi. Bunu çözmek için pencere boyutunun yarısını kullanarak orijini pencerenin ortasına taşıdık ve (-20,20) olacak şekilde düzenledik. Aynı şekilde girdiğimiz noktalarda bu aralıkta olduğu için kütüphanenin koordinat sistemine göre noktaların koordinat bilgilerini değiştirerek çizdirdik.

Sonrasında minimum çevreleyen çember çiziminde hata aldık. Algoritmamızda kontrol etmediğimiz durumlar için çemberi doğru çizememiştik. Nokta sayısına bağlı olarak algoritmayı tekrardan düzenlediğimizde sorun ortadan kalktı.

B-spline çiziminde eğriyi nasıl oluşturmamız gerektiğini tam anlayamamıştık. Daha sonra eğrinin tanımından yola çıkarak denklemleri düzenledik ve koda düzenlenmiş halini yazdık. Fakat yine de eğri için oluşturduğumuz algoritma tam olarak doğru çalışmıyor.

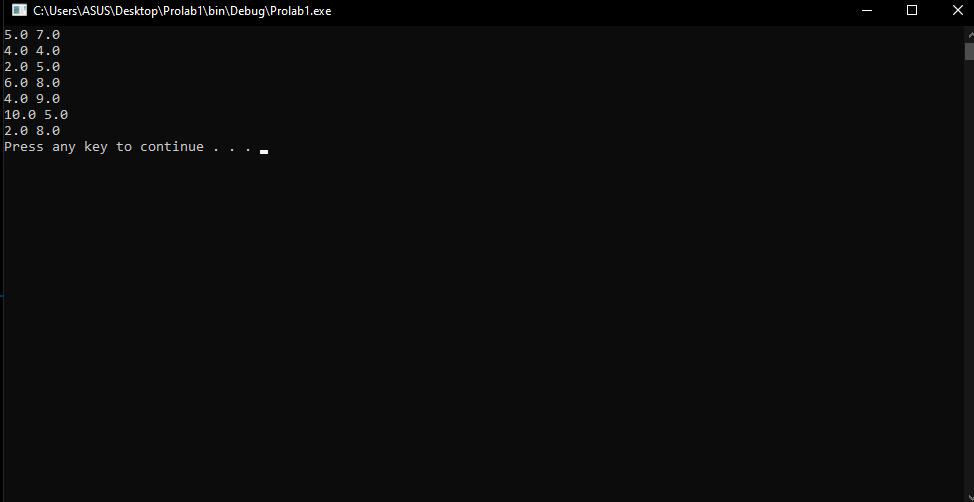
**Kullandığımız Kütüphaneler**

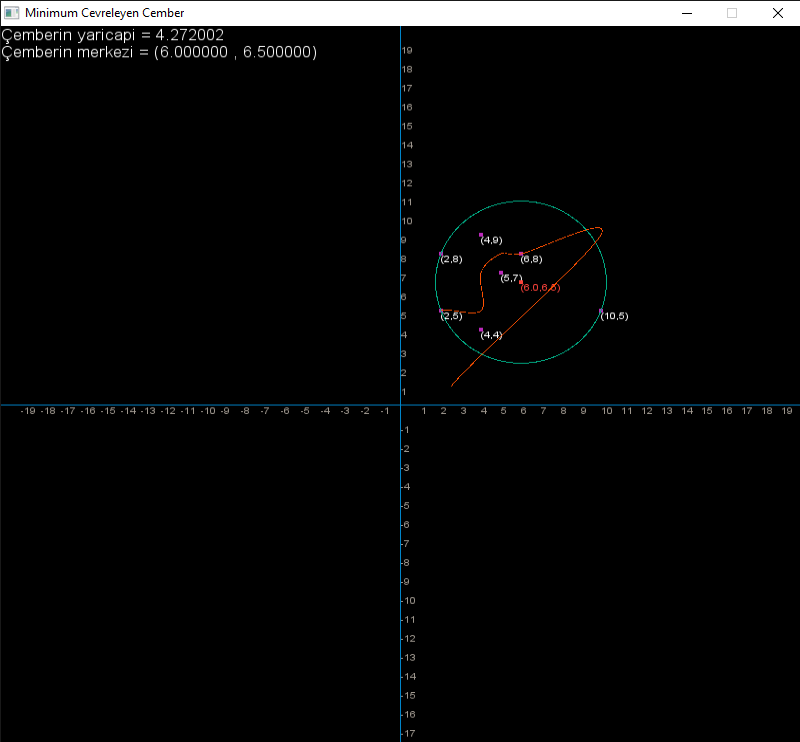
* **stdio.h :** C diline temel giriş ve çıkış yeteneklerini sağlar.
* **stdlib.h :** İşlemlerde çeşitli performans için kullanılır. Dönüşümler, rastgele numaralar , bellek ayırma, proses kontrolü, çevre, arama, sinyalizasyon ve sıralama içindir.
* **math.h :** Yaygın matematik fonksiyonlarını içerir.
* **allegro5/allegro.h :** Allegro kütüphanesinin içindeki fonksiyonları kullanmamızı sağlar.
* **allegro5/allegro\_native\_dialog.h :** Açılan pencere ile bilgisayarın iletişimini sağlar.
* **allegro5/allegro\_primitives.h :** İlkel eklenti işlevlerinin kullanılmasını sağlar.
* **allegro5/allegro\_font.h :** Yazı tipi eklentilerini kullanmamızı sağlar.
* **allegro5/allegro\_ttf.h :** Yazı tiplerini kullanmamızı sağlar.

**Kullandığımız Fonksiyonlar**

* b\_splinex()
* b\_spliney()
* koordinat\_sistemi()
* uzaklık()
* yaricap\_hesapla()
* karteyzen\_merkezx()
* kartezyen\_merkezy()
* cizim\_koordinatx()
* cizim\_koordinaty()

**Deneysel Sonuçlar:**





**Sonuç:**

Bizden istenen minimum çevreleyen çember için yaptığımız araştırmalar sonucunda, dosyadan okunan N tane nokta ile oluşturduğumuz algoritma içinde çemberi hesaplamak. Çember algoritmasında N’in üçten büyük koşulu için x ve y değerlerinin maksimum-minimum değerleri hesaplandı. Hesaplanan bu noktalar bir dörtgenin kenarlarında düşünülerek, hayali dörtgenin merkez koordinatları hesaplandı ve bu dörtgenin merkezine en uzak olan noktalardan yararlanılarak çember çizme algoritmasını tamamlamış olduk.

B-spline için dosyadan okunan N tane değere en yakın eğriyi çizdirmemiz gerekiyordu. Matematiksel olarak önce b-spline eğrisini bir fonksiyon içinde hesaplayıp N değerlerini göndererek noktalarımızın en yakınından geçen eğriyi hesaplamış olduk.

Daha sonra da allegro kütüphanesini kullanarak çemberimizi, noktalarımızı ve b-spline fonksiyonunu renklendirdik ve çizdirdik. Ayrıca ekrandaki 2 boyutlu koordinat sistemini de renklendirerek koordinat sistemindeki sayıları yazdırdık.

**Big-O Karmaşıklık Analizi:**

satır 199 -> n

satır 213 -> n+2n

satır 227 -> n+n+n+n-1=4n-1

satır 246 -> 3n

satır 266 -> 7n-6

satır 284 -> 7n-6

satır 315 -> 2n

satır 323 -> n + n\*n + .. =n^2

satır 514 -> n + n\*n+... = n^2

satır 531 -> n^2

satır 544 -> n + 3n=4n

* Bu algoritma için gerekli zaman n^2 ile doğru orantılıdır.

Big-O karmaşıklık analizi yapılırken bağıntıdaki en büyük artışı yöneten terim alınır ve diğerleri göz ardı edilir. Bu durumda algoritmamızdaki döngülere bakmamız gerekecektir. Buna göre algoritmamızın toplam karmaşıklığı T(n) = O(n^2) olarak bulunur.

**KABA KOD**

BAŞLA

DOSYA \*dosya

CHAR c

DOSYA f=open(DOSYA\_YOLU,”r”)==NULL

SAYISAL n=1

EĞER dosya=fopen(DOSYA\_YOLU,”r”)

YAZ “dosya acilamadi\n”

ÇIKIŞ 1

EĞER BİTTİ

DÖNGÜ ((c=fgetc(dosya))!=EOF

YAZ “%c”,c

EĞER (c==’\n’)

ARTTIR n++

DÖNGÜ BİTTİ

FLOAT noktalar[n][2]

SAYISAL i,j

 DOSYAYI AÇ dosya=fopen(DOSYA\_YOLU,”r”)

DÖNGÜ i=0;i<n;i++

DÖNGÜ j=0;j<2;j++

DOSYA YAZDIR fscanf(dosya,”%f”,&noktalr[i][j])

DÖNGÜ BİTTİ

DÖNGÜ BİTTİ

DOSYAYI KAPAT fclose(dosya)

FLOAT x,y

DÖNGÜ i=0; i<n; i++

EŞİTLE x=noktalar[i][0];

EŞİTLE y=noktalar[i][1];

EĞER n!=1

ÇİZDİR

EĞER BİTTİ

DÖNGÜ BİTTİ

FLOAT noktax[n], noktax[n]

DÖNGÜ i=0; i<n; i++

EŞİTLE noktax[i]=noktalar[i][0];

EŞİTLE noktay[i]=noktalar[i][1];

DÖNGÜ BİTTİ

FLOAT xmax[2],ymax[2],xmin[2],ymin[2];

EŞİTLE xmax[0]=noktax[0];

EŞİTLE ymax[0]=noktay[0];

DÖNGÜ i=1; i<n; i++

EĞER noktax[i]>xmax[0]

EŞİTLE xmax[0]=noktax[i];

EŞİTLE xmax[0]=noktax[i];

EĞER noktay[i]>ymax[1]

EĞER ymax[1]=noktay[i];

EĞER ymax[0]=noktax[i];

EĞER BİTTİ

DÖNGÜ BİTTİ

// 349

//363

FLOAT w = xmax[0]-xmin[0], h = ymax[1]-ymin[1];

EĞER (ymax[0]==xmax[0] || ymin[0]==xmin[0]) && (ymax[0]==xmin[0] || ymin[0] || xmax[0])

EŞİTLE yaricap = uzaklik(ymin[0],ymin[1],ymax[0],ymax[1]);

EŞİTLE yaricap=yaricap/2;

EŞİTLE merkezx = w/2+xmin[0];

EŞİTLİ merkezy = h/2+ymin[1];

EĞER BİTTİ

FLOAT t[n]

DÖNGÜ i=0; i<n; i++

EŞİTLE t[i] = uzaklik(w/2+xmin[0], h/2+ymin[0], noktax[i],noktay[i]);

// 382

//406

FLOAT m1,m2,m3

EŞİTLE m1=uzaklik(noktax[n-1],noktay[n-1],noktax[n-2],noktay[n-2])

EŞİTLE m2=uzaklik(noktax[n-1],noktay[n-1],noktax[n-3],noktay[n-3])

EŞİTLE m3=uzaklik(noktax[n-2],noktay[n-2],noktax[n-3],noktay[n-3])

//412

//508

EĞER n>3

EĞER m1>m2 || m1>m3

EŞİTLE gecicix = (noktax[n-1]+noktax[n-2])/2;

EŞİTLE geciciy = (noktay[n-1]+noktay[n-2])/2

EŞİTLE d = uzaklik(noktax[n-3],noktay[n-3],gecicix,geciciy)

EĞER d <= m1/2

EŞİTLE yaricap = m1/2;

EŞİTLE merkezx = gecicix;

EŞİTLE merkezy = geciciy;

EĞER

EŞİTLE merkezx = karteyzen\_merkezx(noktax[n-1],noktay[n-1],noktax[n-2],noktay[n-2],noktax[n-3],noktay[n-3]);

EŞİTLE merkezy =kartezyen\_merkezy(noktax[n-1],noktay[n-1],noktax[n-2],noktay[n-2],noktax[n-3],noktay[n-3]);

EŞİTLE yaricap = yaricap\_hesapla(noktax[n-1],noktay[n-1],noktax[n-2],noktay[n-2],noktax[n-3],noktay[n-3]);

EĞER BİTTİ

EĞER BİTTİ

EĞER m2>m1 || m2>m3

EŞİTLE gecicix = (noktax[n-1]+noktax[n-3])/2;

EŞİTLE geciciy = (noktay[n-1]+noktay[n-3])/2;

EŞİTLE d = uzaklik(noktax[n-2],noktay[n-2],gecicix,geciciy)

EĞER d <= m2/2

EŞİTLE yaricap = m2/2;

EŞİTLE merkezx = gecicix;

EŞİTLE merkezy = geciciy;

EĞER

EŞİTLE merkezx = karteyzen\_merkezx(noktax[n-1],noktay[n-1],noktax[n-2],noktay[n-2],noktax[n-3],noktay[n-3])

EŞİTLE merkezy = kartezyen\_merkezy(noktax[n-1],noktay[n-1],noktax[n-2],noktay[n-2],noktax[n-3],noktay[n-3])

EŞİTLE yaricap = yaricap\_hesapla(noktax[n-1],noktay[n-1],noktax[n-2],noktay[n-2],noktax[n-3],noktay[n-3])

EĞER BİTTİ

EĞER

EŞİTLE gecicix = (noktax[n-2]+noktax[n-3])/2

EŞİTLE geciciy = (noktay[n-2]+noktay[n-3])/2

EŞİTLE  d = uzaklik(noktax[n-1],noktay[n-1],gecicix,geciciy);

EĞER d <= m3/2

EŞİTLE yaricap = m3/2

EŞİTLE merkezx = gecicix

EŞİTLE merkezy = geciciy

EĞER

EŞİTLE merkezx =karteyzen\_merkezx(noktax[n-1],noktay[n-1],noktax[n-2],noktay[n-2],noktax[n-3],noktay[n-3]);

EŞİTLE merkezy = kartezyen\_merkezy(noktax[n-1],noktay[n-1],noktax[n-2],noktay[n-2],noktax[n-3],noktay[n-3])

EŞİTLE yaricap = yaricap\_hesapla(noktax[n-1],noktay[n-1],noktax[n-2],noktay[n-2],noktax[n-3],noktay[n-3]);

EĞER BİTTİ

EĞER BİTTİ

//567

//575

FLOAT temp

DÖNGÜ int i=0; i<n-1; i++

DÖNGÜ int j=i+1; j<n; j++

EĞER noktax[i]>noktax[j]

EŞİTLE temp = noktax[i];

EŞİTLE noktax[i] = noktax[j];

EŞİTLE noktax[j] = temp;EŞİTLE

EŞİTLE temp=noktay[i]

EŞİTLE  noktay[i] = noktay[j]

EŞİTLE noktay[j] = temp

EĞER BİTTİ

DÖNGÜ BİTTİ

DÖNGÜ BİTTİ

DÖNGÜ int i=0; i<n-1; i++

DÖNGÜ int j=i+1; j<n; j++

EĞER noktax[i] == noktax[j] || noktay[i]>noktay[j]

EŞİTLE temp = noktay[i]

EŞİTLE noktay[i] = noktay[j]

EŞİTLE noktay[j] = temp

EĞER BİTTİ

DÖNGÜ BİTTİ

DÖNGÜ BİTTİ

DÖNGÜ t=0; t <n; t += 0.005

EĞER n>2

EŞİTLE float px = b\_splinex(n,noktax,t)

EŞİTLE float py = b\_spliney(n,noktay,t)

ÇİZDİR

DÖNDÜR

BİTİR

**SÖZDE KOD**

Başla

1-Arayüz işlemlerini yap ve arayüz penceresini oluştur

2-Koordinat sistemi fonksiyonunu çizdir

3- Dosyayı aç

4- Nokta sayısını bulmak için dosya sonuna gelene kadar döngüyü çalıştır

5- Dosyayı okuma modunda aç

6- Döngü içinde noktaları matrise aktar

7- Dosyayı kapat

8- Döngüye gir dosyadaki noktaları koordinat sisteminde çizdir

9- Merkez ve yarıçapı tanımla

10- Maks-x min-x ve maks-y min-y olan noktaları döngü içinde bul

11- Eğer çember üstünde iki nokta olursa yarıçap merkezi hesapla

12- Eğer nokta sayısı bire eşitse merkezi hesapla ve çizdir

13- Eğer nokta sayısı ikiye eşitse merkez ve yarıçapı hesapla ve çizdir

14- Eğer nokta sayısı üçe eşitse Kartezyen koordinatlarını hesapla ve merkez ile yarıçapı bulup çizdir

15- Eğer nokta sayısı üçten fazlaysa noktaların uzaklıklarına bakıp Kartezyen formülünden merkez ve yarıçapı hesapla, çemberi çizdir

16- Noktaları x koordinatlarına göre döngü içinde sırala

17- Eğer x koordinatları eşit olan noktalar varsa döngü içinde y koordinatlarına göre sırala

18- B-spline için döngüyü çalıştır.

19- Eğer  nokta sayısı ikiden büyükse b-spline çizdir.

20- Eğer nokta sayısı ikiye eşitse iki nokta arası eğri çizdir.

21- Sistemi durdur

BİTİR

**KAYNAKÇA**

<https://www.allegro.cc/manual/5/index.html>

<https://en.wikipedia.org/wiki/Smallest-circle_problem>

<https://en.wikipedia.org/wiki/B-spline>

<https://en.wikipedia.org/wiki/De_Boor's_algorithm>

<http://cagataykiziltan.net/programin-calisma-hizi-ve-algoritma-verimliligi/zaman-karmasikligi-ve-buyuk-o-notasyonu-time-complexity-and-big-o-notation/>

<https://pages.mtu.edu/~shene/COURSES/cs3621/NOTES/spline/de-Boor.html>

<http://web.mit.edu/hyperbook/Patrikalakis-Maekawa-Cho/node17.html>

<https://tr.wikipedia.org/wiki/%C3%87evrel_%C3%A7ember>

<https://www.codeproject.com/Articles/1165267/Coding-Challenge-Smallest-Circle-Problem>