TAVLAMA BENZETİMİ # (Simulated Annealing)

→ Fonksiyonlar .m dosyasında kayıt edilir.
→ fonksiyonda çıktılar solda, sağda değişkenler.
→ Dosya adı ile = 'den sonraki yazın adı aynı olmalıdır
→ Dosyanın sağında * işareti varsa yapılan işlemlerin kayıt edilmediği manasına gelir.

⇒ min toplam (x_i^2), [-5,5], i=1,...,d minimize edelim

→ unifrnd (-5,5,[1,6]) → -5,5 arası 1'den 6'ya kadar 6 değer üretir
↳ üniform, sürekli değerler üretiyoruz.

- Programı çalıştırmak için function'dan sonrasını kopyalayıp çalıştır.
  - Fonk. daki değerlendirek fonk. girilebilir yada ayrıca değişkene değer atanabilir.
  - workspace'de değişkenlerin kayıt altına alınması daha mantıklıdır.
  - Her ; koymadığım satır için, sonuç varsa onu ekrana yazdırır.
    ; varsa sonucu göstermez. workspace'e atar
  - Fonksiyon içinde herhangi bir veriyi workspace'de kayıtlı olmasını istiyorsak sol tarafa onu da yazmalıyız.

→ cozum = x_1 x_2 x_3 x_4 x_5 x_6 , c2 = cozum.^2

$c2 = x_1^2\ x_2^2\ x_3^2\ x_4^2\ x_5^2\ x_6^2$

Sum(c2) // c2 tüm elemanlarını toplar.

```
function [cozum obj] = (as, us, d)
        cozum = unifrnd(as, us, [1 d]);
        obj = sum(cozum.^2);
end
```

- veriler.mat → var olan değişkenleri tutar. workspace boşalınca verileri görebilirsin.
- Komşuluk; komşuluk = %5 ise = $\frac{(\text{üstsınır} - \text{altsınır}) * 5}{100}$
→ Amaç fonk karşılaştırıyoruz ve hangisi daha iyiyse onu alıyoruz. Bu süreçte komşuluk değerlerine bakarak işlem yapıyoruz.

```
function [cozum obj degisim_miktari komsu obj_komsu] = simann(as, us, d, delta)
        cozum = unifrnd(as, us, [1 d]);
        obj = sum(cozum.^2);
        degisim_miktari = unifrnd(-(us-as)*delta/2, (us-as)*delta/2, [1 d]);
        komsu = cozum + degisim_miktari;
        obj_komsu = sum(komsu.^2);
end
```

SA-1

- Sonuç her karşılaştırdığımız zaman farklı bir şey geliyorsa, rastsal değişkenlerle işlem yapılıyor gibi düşünebiliriz.
- Eğer yeni elde edilen amaç fonk. daha iyi ise; yeni amaç fonk. o seçilir.
  Eğer daha kötü ise kabul olasılığı hesaplanır.

$$p_a = \exp(-de/T) \rightarrow (e^{-\Delta e/T})$$

(de: değişim miktarı, T = sıcaklık)

Ardından 0-1 arasında rastsal sayı oluşturulur ve bu sayı p_a ile karşılaştırılır.
Eğer rs < pa ise çözüm = komşu ve obj = obj - komşu olur

- Bitiş sıcaklığına kadar iterasyonlara devam ediyoruz.
- İterasyonlardaki en küçük değeri almayı hedefliyoruz. Bundan dolayı elde edilen değer ile önceki değerleri karşılaştırıyoruz. Bu sayede sadece amacımız olan değerleri saklıyoruz.
- Parametrelerle oynayarak en iyi değerleri elde etmeye çalışıyoruz.

Soğutma katsayısı (sk) = 0.90 → 0.99 yaptık. Baya iyi sonuçlar elde edildi.
komşuluk büyüklüğü
başlangıç sıcaklığı
son sıcaklık
→ En önemli parametreler

#Simulated Annuling ile Gezgin Satıcı Problemi

- 10 tane şehirin birbirine olan uzaklığı üzerinden gidelim.
- 10x10'luk simetrik bir matris elde ederiz.
- Bir şehirden çıkıp, başlangıç şehrine en kısa yoldan dönmeye çalışalım.
- Yukarıdaki örneği bunun için kullanacağız. Algoritma aynı, çözüm sayısı sınırlı, amaç fonk. değişti.

→ distances → Girdi değerimiz.

```
randperm(): rastsal sayıları sıralar.
Size() // matrisin boyutlarını verir.
sehirSayisi = Size(distances,1); // satır sayısının değerini tutar.
cozum = randperm(sehirSayisi); // Başlangıç çözümü. Distances'daki değerlere göre çıkarımı toplayacağız.
obj = 0; // Toplama değişkeni olduğu için 0'dan başlattık.
for i=1 : sehirSayisi-1
    sehir1 = cozum(i);
    sehir2 = cozum(i+1);
    obj = obj + distances(sehir1, sehir2);
end
sehir1 = cozum(end);
sehir2 = cozum(1);      obj = obj + distances(sehir1, sehir2);
```

- Komşu yapısı değişiyor. Swap yöntemi ile rastsal olarak komşulara gidebiliriz.
  2 şehir seçip onları yerlerini değiştirebiliriz ve şekilde bir komşuluk yaratıp değerimizi tekrar hesaplayabiliriz.
- unidrnd(10) // 10 değer arasından 1 değeri bize rastsal olarak üretir.
  ↳ 2 kere bu komutu çalıştırarak swap edeceğimiz 2 şehri bulabiliriz.

unidrnd(10, [1 2]); // 1 satır 2 sutunlu satır vektörü yaratıyoruz.
(↳ 2 rastsal sayı üretir. Ama aynı sayılarda gelebilir. (1/10)
↳ Tekrar etmeyen sayılar gelmelidir.

- randperm(10) // Birbirini tekrar etmeyen 1-10 arası sayıları sıralar.

[7 9] 10 2 3 1 4 6 5 8
↳ Bu 2'si hiç bir zaman aynı olmayacak ilk 2'sini alırsak bizim isimiz!
görecektir.

```
deg = randperm(10);
deg = deg(1:2);
```
} ilk 2 elemanı almış oluruz.
Bu 2 eleman aslında bizim 2 şehrimiz, bunların yerlerini değiştireceğiz.

```
komsu = cozum
dummy = komsu(deg(1));
komsu(deg(1)) = komsu(deg(2));
komsu(deg(2)) = dummy;
```
} Swap işlemi
Komşuyo bittik.
Bundan sonra komşunun amaç fonksiyon değeri hesaplanmalıdır.

```
obj_komsu = 0;
for i=1:sehirsayisi-1
    sehir1 = komsu(i);
    sehir2 = komsu(i+1);
    obj_komsu = obj_komsu + distances(sehir1, sehir2);
end
sehir1 = komsu(end);
sehir2 = komsu(i);
obj_komsu = obj_komsu + distance(sehir1, sehir2);
```
// Komşunun Amaç Fonksiyonu değeri //

:
İyileşme var mı?
:
Varsa devam et
:

Bu kısım bir önceki ile aynı!!

```
T = 10000; // Başlangıç sıcaklığı
Tend = 0.1; // Son sıcaklık
sk = 0.9; // soğutma katsayısı
```
Bu değerlerde 111 iterasyon sürdü.

SA-3