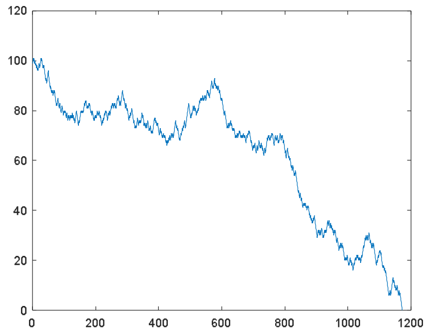
Ders 11:

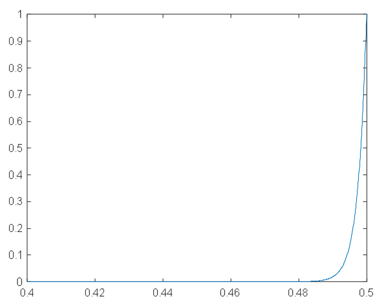
- Kumarbaz. Her oyunda p olasılıkla 1 TL kazanıyor, 1-p olasılıkla 1 TL kaybediyor. Elinde N TL'si var. Ya M+N TL'yi kazanıp sevinecek ya da 0 TL ile eve dönecek. Kumarbazın sevinme olasılığı?
  - Önce tek bir oyunu gerçekleyelim.




```
m=100;
n=100;
h=m+n;
p=0.473; % 1 oyunda kazanma ihtimali
t=0;
while (n<h) && (n>0) % oyun sürüyor
    if rand()>(1-p) % kazandı
        n=n+1;
    else
        n=n-1;
    end
    t=t+1;
    R(t)=n;
end
çizdir(R);
```

  - Şimdi bu işlemi çok sayıda tekrarlayıp kazanma sayısını bulalım.

```
m=10;
en=100;
h=m+en;
p=0.473; % 1 oyunda kazanma ihtimali
K=10000; % denemesayisi;
kaz=0;
for i=1:K
    n=en;
    while (n<h) && (n>0)
        if rand()>(1-p) % kazansin
            n=n+1;
        else
            n=n-1;
        end
    end
    if n==h
        kaz=kaz+1;
    end
end
yazdır(kaz/K)
```



  - Sonuç= 0.339 (p=0.473, n=100, m=10)
  - Sonuç= 0.341 (p=0.473, n=1000, m=10)
  - Sonuç= 0.115 (p=0.473, n=100, m=20)
  - Sonuç= 0.114 (p=0.473, n=200, m=20)
  - Sonuç= 0.115 (p=0.473, n=1000, m=20)
  - Sonuç= 0.000018 (p=0.473, n=100, m=100)
  - Sonuç= 0.000020186 (p=0.473, n=1000000, m=100)
  - İşin gerçeği sonuç (kazanma olasılığı) n ile ilgili değil. $sonuç \leq (\frac{p}{1-p})^m$
  - Ana sonuç: kasa her zaman kazanır, oynamayın ☺

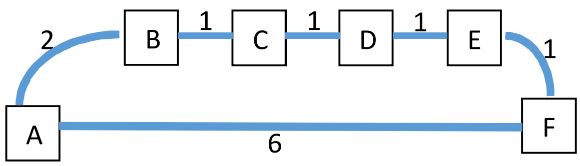
- Matris çarpımı:
  - 2 matrisin çarpımı A(n,m), B(m,p), C(n,p)=A*B

```
for i=1:n
    for j=1:p
        sum=0;
        for k=1:m
            sum=sum+a(i,k)*b(k,j);
        end
        c(i,j)=sum;
    end
end
```

  - Karmaşıklığı O(n*p*m)
  - 2 kare matris olsaydı $O(n^3)$, daha iyi algoritmalar var. $O(n^{2.XX})$
- Elimizde çarpılacak 3 matris olsun. A(10,30), B(30,5), C(5,60)
  - (AB)C mi? A(BC) mi? Sonuçlar aynı ama işlem karmaşıklığı?
  - (AB)C yi yapalım.
    - A*B'nin karmaşıklığı= 10*30*5=1500
    - Çıkanla C'yi çarpalım = 10*5*60=3000
    - O halde (AB)C'nin karmaşıklığı=4500
  - A(BC) yi yapalım.
    - B*C'nin karmaşıklığı= 30*5*60=9000
    - A ile çıkanı çarpalım = 10*30*60=18000
    - O halde A(BC)'nin karmaşıklığı=27000

- External Merge Sort: Sıralanacak dizi RAM'e sığmazsa nasıl sıralarız? Diyelim ki HD'de 9 GB data var. 1 GB RAM var.
  - Önce datayı RAM'e sığacak sayıda parçaya böl.
  - Her bölümü HD'den RAM'e çekip sıralayıp (mergesort ya da quicksort ile) HD'ye yaz.
  - Her bölgenin %10'luk ilk kısımlarını RAM'e çek. 100 MB * 9 = 900 MB geriye kalan 100 MB buffer.
  - 9 diziyi 9 yollu birleştirerek buffer'ı doldur. Buffer dolunca HD'e yaz. 9 bölgeden biri boşalınca boşalanla alakalı sonraki %10'luğu HD'den al.

- Gezgin satıcı problemi
  - Tüm şehirlerden en az 1 kez geçmek şartıyla en kısa rota nedir
  - N şehir için optimum çözümü bulmanın karmaşıklığı n! ☹
- Şu N sayı içinde toplamı K olan bir alt küme var mı?
  - optimum çözümü bulmanın karmaşıklığı n! ☹
- Sezgisel (heuristic) algoritmalar: çözüm uzayı çok büyük olduğunda bunu sınırlayan kural, varsayım vb.
- Gezgin satıcı için en yaygın sezgisel algoritma. Bir şehirden başla ve en yakınına git.
  - Karmaşıklığı: $N-1+N-2+N-3+…+1 \approx N^2 << n!$ süper ☺
  - Ama optimum çözümü garantilemez ☹
  - Örneğin



- Sezgisel algoritma B'den başlasın. B C D E F A rotasını bulur. Rota uzunluğu 10
- B'den başlayan daha iyi bir çözüm: B A B C D E F. Rota uzunluğu 8
- Optimal çözümlerden biri: A B C D E F. Rota uzunluğu: 6
- Bazı sezgisel yaklaşımlar bazı problem türlerinde, bazı kısıtlar altında optimal çözümü garantiler. Ama çoğunlukla böyle değildir.
- Optimumu bulmanın şart olmadığı durumlarda çok fazla beklemek yerine optimal olmayan ama hızlı bulunan çözümler tercih edilir. Gezgin satıcıya sen 5 yıl bekle sana optimal rotayı vereceğim diyemeyiz ☺