Bölüm 3: Nash Dengesi

Algoritmik Oyun Kuramı

Biçimsel tanım · En iyi tepki · Baskın strateji eleme · Karışık strateji · Varlık teoremi · Cournot ve Bertrand · Denge seçimi · Korelasyonlu denge · Minimaks · Güvenlik oyunları · PPAD

Sercan KÜLCÜ · Giresun Üniversitesi · Bilgisayar Mühendisliği

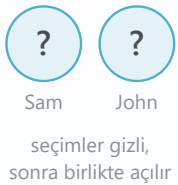
© Sercan Külcü. Tüm hakları saklıdır. · Sunumun etkileşimli HTML sürümü: https://sercankulcu.github.io/files/agt/slides/Bolum_03_Nash_Dengesi.html

Oyunla Başlangıç: Mahkûmun İkilemi

Kural: Sam ve John ayrı odalarda sorgulanır; her biri diğerinin seçimini bilmeden *Sessiz* kalır ya da *İhanet* eder ve tablodaki kadar hapis yatar. Hücrelerde kazanç = −(hapis yılı): büyük sayı iyidir.

Oyun: [Mahkûmun İkilemi oyunu](), iki kişi aynı ekranda ya da bilgisayara karşı, çok turlu.

Sam \ John	Sessiz	İhanet
Sessiz	(−1 , −1)	(−3 , 0)
İhanet	(0 , −3)	(−2 , −2)



Hücrede (Sam, John) kazanç = −(hapis yılı)

Soru 1

Sam'in yerinde olsanız, John'un ne seçeceğini bilmeden sessiz mi kalırdınız, ihanet mi ederdiniz?

Soru 2

Bilgisayar Tit-for-Tat oynarken (ilk tur sessiz, sonra sizin önceki seçiminizi kopyalar) on turda en az kaç yıl birikir?

TEMELLER

Öğrenme Çıktıları

Tanım

Nash dengesini biçimsel olarak tanımlamak ve en iyi tepki (best response) kavramıyla ilişkilendirmek.

Saf denge bulma

Kazanç matrisinde en iyi tepkileri işaretlemek, yinelemeli baskın strateji eleme uygulamak; Cournot ve Bertrand gibi sürekli stratejili oyunlarda dengeyi bulmak.

Karışık denge

Kayıtsızlık ilkesiyle karışık dengeyi elle ve kodla hesaplamak; sıfır toplamlı oyunlarda ve güvenlik oyunlarında minimaks değerini bulmak.

Varlık ve karmaşıklık

Nash'in varlık teoremini, çoklu denge sorununu ve denge hesaplamanın PPAD-tamlığını yorumlamak.

TEMELLER

John Nash Kimdir?

- Amerikalı matematikçi **John Forbes Nash Jr.** (1928–2015).
- 1950'de Princeton'da kısa doktora tezi *Non-Cooperative Games*; aynı yıl *PNAS*'ta varlık ispatını veren iki sayfalık bir not.
- 1994 Nobel Ekonomi Ödülü (Harsanyi ve Selten ile birlikte).
- 2015 Abel Ödülü: kısmi diferansiyel denklemlere katkıları.
- Hayatı *A Beautiful Mind* (2001) filmine konu oldu.



John F. Nash Jr.

Katkısı: von Neumann'ın **iki kişilik sıfır toplamlı** oyunlar için kurduğu çözümü, **n kişilik genel** (sıfır toplamlı olmayan) oyunlara taşıdı.

TEMELLER

Stratejik Davranış ve Uygulama Alanları

- Oyuncular kendi kazançlarını maksimize etmeye çalışır.
- Herkesin kazancı **diğerlerinin seçimlerine de** bağlıdır: “Ben ne yapmalıyım?” sorusu “Onlar ne yapacak?” sorusundan ayrılamaz.

Fiyat rekabeti
İki firma fiyat indirir; ikisi de kâr kaybeder.
`Mahkûmun İkilemi yapısı`

Reklam / silahlanma
Herkes harcar, göreli konum değişmez.
`baskın strateji: harca`

Penaltı atışı
Atıcı ve kaleci tahmin edilemez olmalı.
`karışık strateji, sıfır toplamlı`

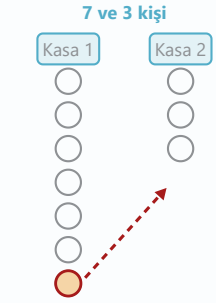
Trafikte şerit seçimi
Herkes hızlı şeride geçerse o şerit yavaşlar.
`tıkanıklık oyunu`

Standart seçimi
Aynı teknolojiyi seçmek herkese kazandırır.
`koordinasyon, çoklu denge`

Kaynak paylaşımı
Ortak bant genişliği, sunucu, yol.
`Anarşinin Bedeli`

TEMELLER

Kimsenin Pişman Olmadığı Durum



sondaki geçer: 7. yerine 4. sıra



geçen 6. sıraya düşer:
kimse geçmez = denge

- Yemekhanede iki kasa var; herkes daha kısa beklemek istiyor ve kuyruk değiştirmek serbest.
- **7 – 3**: uzun kuyruğun sonundaki öğrenci geçerse beklemesi kısalır → bu durum kalıcı değil.
- **5 – 5**: kim geçerse geçsin daha geriye düşer → kimse yerini değiştirmez.
- **Pişmanlık testi:** "Diğerlerinin ne yaptığını görseydim, kendi seçimimi değiştirir miydim?" Herkesin cevabı *hayır* ise durum bir **Nash dengesidir**.

Denge, kendi kendini sürdüren durumdur: herkes diğerlerinin davranışını doğru bekler ve buna göre elinden gelenin en iyisini yapar. "En iyi sonuç" ya da "adil sonuç" anlamına gelmez.

TEMELLER

Nash Dengesi: Biçimsel Tanım

Normal form oyun $G = (N, (S_i), (u_i))$:

- $N = \{1, \ldots, n\}$ oyuncular; S_i oyuncu i'nin strateji kümesi.
- Strateji profili $s = (s_1, \ldots, s_n)$; s_{-i}: i dışındakilerin stratejileri.
- $u_i(s)$: i'nin kazancı (u_i: kazanç fonksiyonu).

Nash dengesi: s^* profili, her oyuncu i ve her $s_i \in S_i$ için

$$u_i(s_i^*, s_{-i}^*) \geq u_i(s_i, s_{-i}^*)$$

Tek taraflı sapma

Diğerleri sabitken hiçbir oyuncu stratejisini değiştirerek kazancını artıramaz.

Kendini doğrulayan anlaşma

Herkes diğerlerinin uyacağına inanıyorsa, kendisi de uyar.

Kararlılık ≠ verimlilik

Denge adalet ya da verimlilik (Pareto optimallik) garantisi vermez.

TEMELLER

Sık Yapılan Hatalar: Nash Dengesi Tanımı

“Denge en iyi sonuçtur.”

Hayır. Mahkûmun İkilemi'nde denge (İ, İ) iki oyuncuya da (S, S)'den kötüdür. Denge kararlılık söyler, verimlilik değil.

“İkisi birlikte değişirse kazanır.”

Tanım yalnızca *tek taraflı* sapmaya bakar. İki oyuncunun birlikte sapması (koalisyon) Nash dengesini bozmaz.

“Bir oyuncu memnunsa yeter.”

Test *her* oyuncu için ayrı yapılır; tek bir oyuncu bile sapmak isterse hücre denge değildir.

“Satır oyuncusu sütunu da seçer.”

Her oyuncu yalnız kendi stratejisini değiştirir: satır oyuncusu aynı sütunda yukarı-aşağı, sütun oyuncusu aynı satırda sağa-sola gider.

“Her oyunda tek bir saf denge var.”

Yazı-Tura'da saf denge yoktur; Geyik Avı'nda iki saf, bir karışık denge vardır.

Ceza mı, kazanç mı?

Hapis yılı gibi maliyetlerde *küçük* sayı iyidir. Kazanca çevirmek için eksi işaret konur: 2 yıl → −2.

En İyi Tepki (Best Response)

- Oyuncu i'nin, diğerlerinin s_{-i} seçimine **en iyi tepkisi**: kazancını en yüksek yapan stratejiler kümesi

$$BR_i(s_{-i}) = \text{argmax}_{s_i \in S_i} u_i(s_i, s_{-i})$$

- Eşdeğer tanım: s* Nash dengesidir ⇔ her i için $s_i^* \in BR_i(s_{-i}^*)$. Denge = **karşılıklı en iyi tepki**.

1. Satır oyuncusu
Her *sütunda*, satır oyuncusunun en büyük kazancını (u_1) işaretle.

2. Sütun oyuncusu
Her *satırda*, sütun oyuncusunun en büyük kazancını (u_2) işaretle.

3. Kesişim
İki kazancı da işaretli hücreler **saf strateji Nash dengeleridir.**

m × n matriste maliyet O(mn): her hücre bir sütun maksimumu ve bir satır maksimumuyla karşılaştırılır.

En İyi Tepkileri İşaretleyerek Dengeyi Bulmak

Sam \ John	Sessiz	İhanet
Sessiz	a (−1 , −1)	b (−3 , 0)
İhanet	c (0 , −3)	d (−2 , −2)

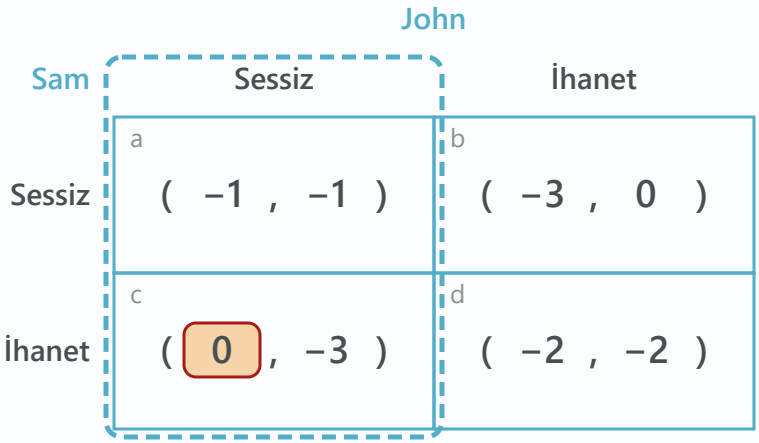
Kazanç = −(hapis yılı)

Sam satır, John sütun oyuncusu

Hücre: (Sam'in, John'un kazancı)

Kazançlar hapis yılının eksi işaretlisidir (−1 = 1 yıl). Her oyuncu kendi sayısını büyütmek ister.

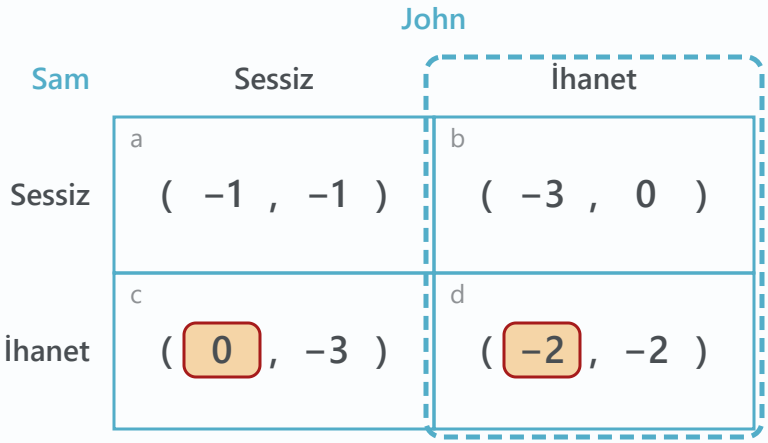
En İyi Tepkileri İşaretleyerek Dengeyi Bulmak



John Sessiz → Sam: İhanet (0 > −1)

John'un **Sessiz** kaldığını varsayalım: Sam bu sütunda yalnızca ilk sayılara bakar. 0 > −1, en iyi tepki İhanet.

En İyi Tepkileri İşaretleyerek Dengeyi Bulmak

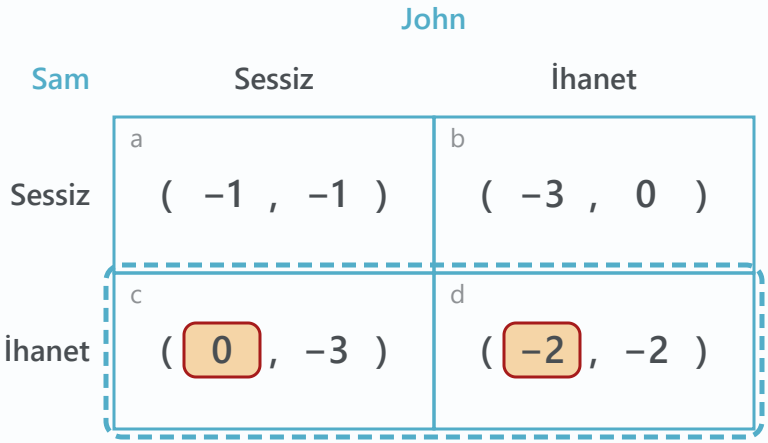


John Sessiz → Sam: İhanet (0 > −1)

John İhanet → Sam: İhanet (−2 > −3)

John **İhanet** ederse: −2 > −3, Sam için en iyi tepki yine İhanet.

En İyi Tepkileri İşaretleyerek Dengeyi Bulmak



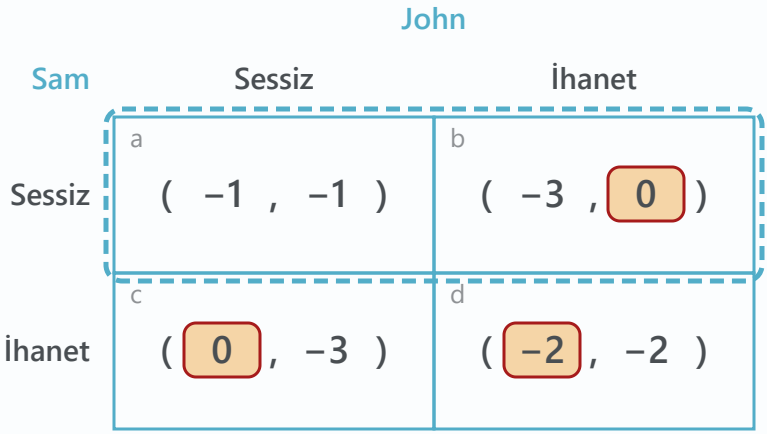
John Sessiz → Sam: İhanet (0 > −1)

John İhanet → Sam: İhanet (−2 > −3)

⇒ Sam için İhanet baskın

İki sütunda da işaret aynı satırda: John ne yaparsa yapsın İhanet daha iyi. Bu bir **baskın strateji**dir.

En İyi Tepkileri İşaretleyerek Dengeyi Bulmak



John Sessiz → Sam: İhanet (0 > −1)

John İhanet → Sam: İhanet (−2 > −3)

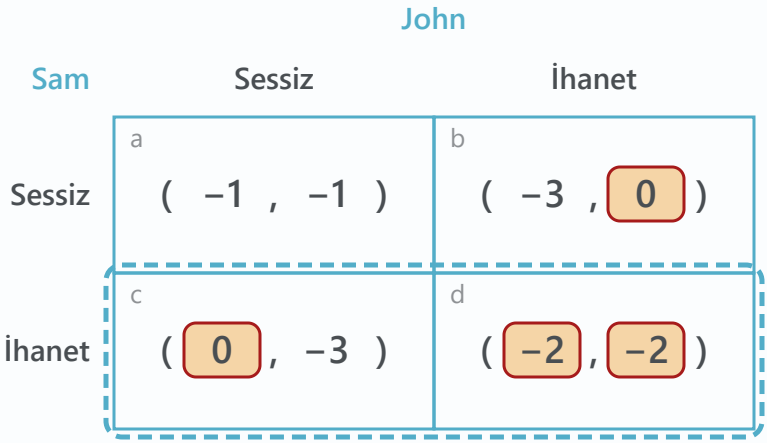
⇒ Sam için İhanet baskın

Sam Sessiz → John: İhanet (0 > −1)

Şimdi John: Sam Sessiz kalırsa John ikinci sayılara bakar: 0 > −1, en iyi tepki İhanet.

EN İYİ TEPKİ

En İyi Tepkileri İşaretleyerek Dengeyi Bulmak



John Sessiz → Sam: İhanet (0 > −1)

John İhanet → Sam: İhanet (−2 > −3)

⇒ Sam için İhanet baskın

Sam Sessiz → John: İhanet (0 > −1)

Sam İhanet → John: İhanet (−2 > −3)

Sam İhanet ederse: −2 > −3, John için de İhanet. Oyun simetrik olduğu için sonuç şaşırtıcı değil.

En İyi Tepkileri İşaretleyerek Dengeyi Bulmak

Sam \ John	Sessiz	İhanet
Sessiz	a: (−1 , −1)	b: (−3 , **0**)
İhanet	c: (**0** , −3)	d: (**−2** , **−2**)

John Sessiz → Sam: İhanet (0 > −1)

John İhanet → Sam: İhanet (−2 > −3)

⇒ Sam için İhanet baskın

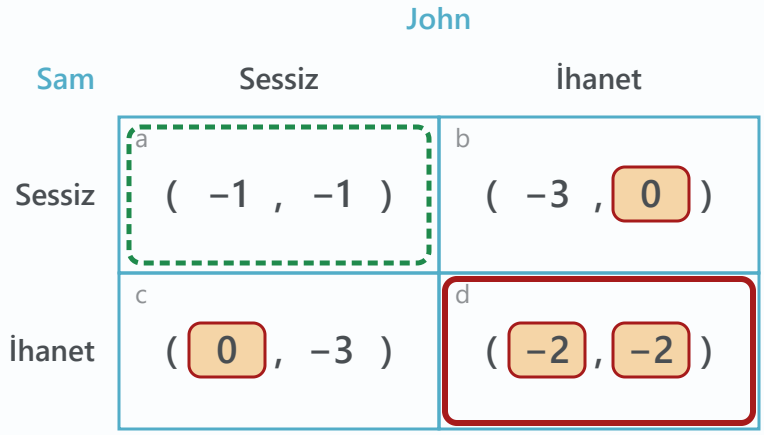
Sam Sessiz → John: İhanet (0 > −1)

Sam İhanet → John: İhanet (−2 > −3)

⇒ John için de İhanet baskın

İki işaretin buluştuğu tek hücre d = (İhanet, İhanet): kimse tek başına sapmak istemez. **Nash dengesi** (−2, −2).

En İyi Tepkileri İşaretleyerek Dengeyi Bulmak



John Sessiz → Sam: İhanet (0 > −1)

John İhanet → Sam: İhanet (−2 > −3)

⇒ Sam için İhanet baskın

Sam Sessiz → John: İhanet (0 > −1)

Sam İhanet → John: İhanet (−2 > −3)

⇒ John için de İhanet baskın

a = (−1, −1) ikisi için de daha iyi!

Oysa a = (Sessiz, Sessiz) ikisine de yalnızca 1'er yıl verirdi. Denge **Pareto optimal değil**: ikilemin özü budur.

Basit Örnek: En İyi Tepkileri İşaretle

Satır oyuncusu \ Sütun oyuncusu	X	Y	Z
A	(4, 3)	(0, 2)	(1, 1)
B	(2, 1)	(3, 3)	(0, 2)
C	(1, 4)	(2, 0)	(3, 2)

Hücrede (u_1, u_2): önce satır, sonra sütun oyuncusunun kazancı. Henüz işaret yok.

Basit Örnek: En İyi Tepkileri İşaretle

Satır oyuncusu \ Sütun oyuncusu	X	Y	Z
A	(**4** , 3)	(0 , 2)	(1 , 1)
B	(2 , 1)	(3 , 3)	(0 , 2)
C	(1 , 4)	(2 , 0)	(3 , 2)

Sütun oyuncusu **X** oynarsa satır oyuncusu u_1 değerlerini karşılaştırır: A:4, B:2, C:1 → en iyisi **A** (4).

Basit Örnek: En İyi Tepkileri İşaretle

Satır oyuncusu \ Sütun oyuncusu	X	Y	Z
A	(**4**, 3)	(0, 2)	(1, 1)
B	(2, 1)	(**3**, 3)	(0, 2)
C	(1, 4)	(2, 0)	(3, 2)

Sütun oyuncusu **Y** oynarsa satır oyuncusu u_1 değerlerini karşılaştırır: A:0, B:3, C:2 → en iyisi **B** (3).

Basit Örnek: En İyi Tepkileri İşaretle

Satır oyuncusu \ Sütun oyuncusu	X	Y	Z
A	(**4**, 3)	(0, 2)	(1, 1)
B	(2, 1)	(**3**, 3)	(0, 2)
C	(1, 4)	(2, 0)	(**3**, 2)

Sütun oyuncusu **Z** oynarsa satır oyuncusu u_1 değerlerini karşılaştırır: A:1, B:0, C:3 → en iyisi **C** (3).

Basit Örnek: En İyi Tepkileri İşaretle

Satır oyuncusu \ Sütun oyuncusu	X	Y	Z
A	(**4**, **3**)	(0, 2)	(1, 1)
B	(2, 1)	(**3**, 3)	(0, 2)
C	(1, 4)	(2, 0)	(**3**, 2)

Satır oyuncusu **A** oynarsa sütun oyuncusu u_2 değerlerini karşılaştırır: X:3, Y:2, Z:1 → en iyisi **X** (3).

Basit Örnek: En İyi Tepkileri İşaretle

Satır oyuncusu \ Sütun oyuncusu	X	Y	Z
A	(**4**, **3**)	(0, 2)	(1, 1)
B	(2, 1)	(**3**, **3**)	(0, 2)
C	(1, 4)	(2, 0)	(**3**, 2)

Satır oyuncusu **B** oynarsa sütun oyuncusu u_2 değerlerini karşılaştırır: X:1, Y:3, Z:2 → en iyisi **Y** (3).

Basit Örnek: En İyi Tepkileri İşaretle

Satır oyuncusu \ Sütun oyuncusu	X	Y	Z
A	(**4**, **3**)	(0, 2)	(1, 1)
B	(2, 1)	(**3**, **3**)	(0, 2)
C	(1, **4**)	(2, 0)	(**3**, 2)

Satır oyuncusu **C** oynarsa sütun oyuncusu u_2 değerlerini karşılaştırır: X:4, Y:0, Z:2 → en iyisi **X** (4).

Basit Örnek: En İyi Tepkileri İşaretle



İki kazancı da işaretli hücreler: **(A, X) = (4, 3)** ve **(B, Y) = (3, 3)** → iki saf Nash dengesi. (C, Z) gibi hücrelerde en az bir oyuncu sapmak ister.

Basit Örnek: En İyi Tepkileri İşaretle



Sapma testi (A, X): satır oyuncusu B veya C'ye geçerse 2 ya da 1 alır (< 4); sütun oyuncusu Y veya Z'ye geçerse 2 ya da 1 alır (< 3). Kimse kazanamaz ✓

EN İYİ TEPKİ

Adım Adım: Altını Çizme Yöntemi (Eşitlikli Örnek)

Ayşe \ Bora	Python	Java	C
Python	(4 , 2)	(1 , 1)	(1 , 0)
Java	(0 , 1)	(2 , 4)	(1 , 2)
C	(1 , 0)	(0 , 2)	(1 , 1)

Ayşe ve Bora ortak proje için dil seçiyor; hücrede (Ayşe, Bora). Kural: Ayşe için her **sütunda** ilk sayıların, Bora için her **satırda** ikinci sayıların en büyüğünün altı çizilir.

EN İYİ TEPKİ

Adım Adım: Altını Çizme Yöntemi (Eşitlikli Örnek)

Ayşe \ Bora	Python	Java	C
Python	(4, 2)	(1, 1)	(1, 0)
Java	(0, 1)	(2, 4)	(1, 2)
C	(1, 0)	(0, 2)	(1, 1)

Bora **Python** seçerse Ayşe yalnız ilk sayılara bakar: 4, 0, 1 → en büyük **4** (Python) çizilir.

EN İYİ TEPKİ

Adım Adım: Altını Çizme Yöntemi (Eşitlikli Örnek)

Ayşe \ Bora	Python	Java	C
Python	(4, 2)	(1, 1)	(1, 0)
Java	(0, 1)	(2, 4)	(1, 2)
C	(1, 0)	(0, 2)	(1, 1)

Bora **Java** seçerse: 1, 2, 0 → en büyük **2** (Java).

Adım Adım: Altını Çizme Yöntemi (Eşitlikli Örnek)

Ayşe \ Bora	Python	Java	C
Python	(4, 2)	(1, 1)	(1, 0)
Java	(0, 1)	(2, 4)	(1, 2)
C	(1, 0)	(0, 2)	(1, 1)

Bora **C** seçerse: 1, 1, 1 → üçü **eşit**, üçünün de altı çizilir. Eşitlikte hepsi en iyi tepkidir; yalnız birini işaretlemek sık yapılan hatadır.

Adım Adım: Altını Çizme Yöntemi (Eşitlikli Örnek)

Ayşe \ Bora	Python	Java	C
Python	(4, 2)	(1, 1)	(1, 0)
Java	(0, 1)	(2, 4)	(1, 2)
C	(1, 0)	(0, 2)	(1, 1)

Şimdi Bora: Ayşe **Python** seçerse Bora ikinci sayılara bakar: 2, 1, 0 → **2** (Python).

EN İYİ TEPKİ

Adım Adım: Altını Çizme Yöntemi (Eşitlikli Örnek)

Ayşe \ Bora	Python	Java	C
Python	(4, 2)	(1, 1)	(1, 0)
Java	(0, 1)	(2, 4)	(1, 2)
C	(1, 0)	(0, 2)	(1, 1)

Ayşe **Java** seçerse: 1, 4, 2 → **4** (Java).

EN İYİ TEPKİ

Adım Adım: Altını Çizme Yöntemi (Eşitlikli Örnek)

Ayşe \ Bora	Python	Java	C
Python	(4, 2)	(1, 1)	(1, 0)
Java	(0, 1)	(2, 4)	(1, 2)
C	(1, 0)	(0, 2)	(1, 1)

Ayşe **C** seçerse: 0, 2, 1 → **2** (Java).

EN İYİ TEPKİ

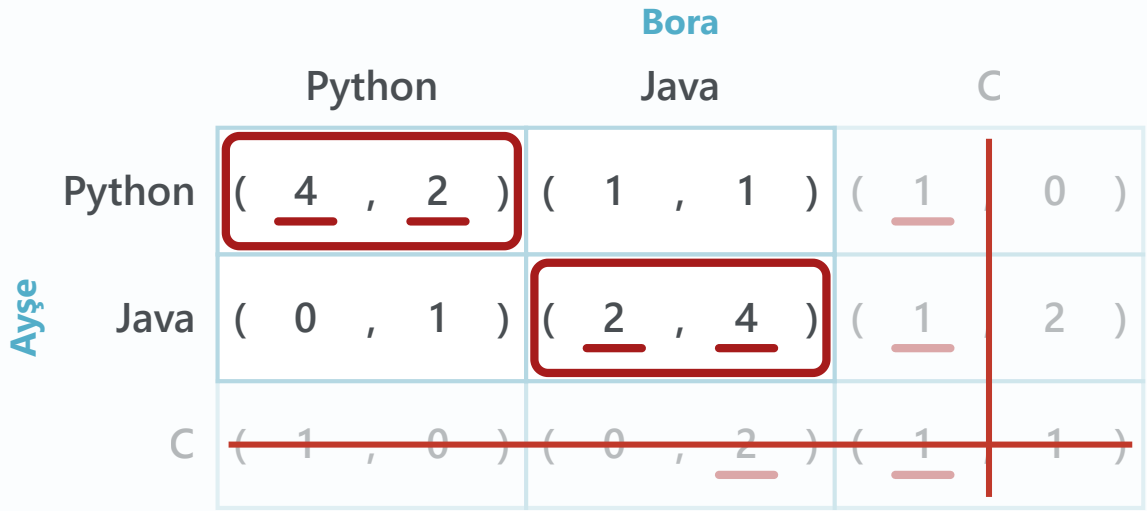
Adım Adım: Altını Çizme Yöntemi (Eşitlikli Örnek)

Ayşe \ Bora	Python	Java	C
Python	(4, 2)	(1, 1)	(1, 0)
Java	(0, 1)	(2, 4)	(1, 2)
C	(1, 0)	(0, 2)	(1, 1)

İki sayının da altı çizili olduğu hücreler: **(Python, Python) = (4, 2)** ve **(Java, Java) = (2, 4)**. (C, C)'de yalnız ilk sayı çizili: Bora Java'ya geçip 1 yerine 2 alır.

EN İYİ TEPKİ

Adım Adım: Altını Çizme Yöntemi (Eşitlikli Örnek)



Kesin eleme burada yetmez: Java, C sütununu siler; sonra Python, C satırını siler. Kalan 2×2'de baskılanan yok, iki denge kalır. Altını çizme ikisini de doğrudan buldu.

EN İYİ TEPKİ

Sık Yapılan Hatalar: En İyi Tepki İşaretleme



Yanlış yön

Satır oyuncusunun en iyisini satır boyunca aramak. O yalnız satırını değiştirir: aynı sütunda bakılır.

Yanlış sayı

Sütun oyuncusu için ilk sayılara bakmak. Sütun oyuncusu ikinci sayıyı büyütmek ister.

Eşitlikte tek işaret

En büyük değer iki kez geçiyorsa ikisi de en iyi tepkidir, ikisi de işaretlenir.

Tek işaret yeter sanmak

Yalnız bir oyuncunun işareti olan hücre denge değildir; *iki* işaret gerekir.

En büyük hücreyi aramak

Denge, sayıları büyük olan hücre değil, karşılıklı en iyi tepki olan hücredir.

Kontrol etmemek

Her aday hücrede sapma testi: oyuncu başka seçeneğe geçince daha çok alıyor mu?

EN İYİ TEPKİ

Üç Oyuncu: Grup Ödevi Oyunu

Üç öğrenci aynı anda **Katıl** ya da **Katılma** der. En az iki kişi katılırsa ödev başarılı olur ve *herkes* 3 puan alır; katılmanın maliyeti 1. Tablo katılan sayısına (k) göre.

Katılan sayısı	Kazançlar	Tek taraflı sapma	Denge mi?
k = 3 (herkes)	3 − 1 = 2'şer	Biri bırakırsa ödev yine başarılı: 3 > 2	Hayır
k = 2	katılanlar 2, katılmayan 3	Katılan bırakırsa ödev batar: 0 < 2. Katılmayan katılırsa 2 < 3	**Evet** (3 profil)
k = 1	katılan −1, ötekiler 0	Katılan bırakırsa 0 > −1	Hayır
k = 0 (kimse)	0'ar	Tek başına katılan −1 alır: −1 < 0	**Evet**

Dört saf denge

Tam iki kişinin katıldığı üç profil ve kimsenin katılmadığı profil. Test n oyuncuda da aynı: her oyuncu için, ötekiler sabitken.

Koordinasyon başarısızlığı

"Kimse katılmaz" dengesi herkese 0 verir; kimse tek başına onu bozamaz. Birlikte sapmak ise Nash tanımının dışındadır.

EN İYİ TEPKİ

Kod İzleme: Saf Nash Dengelerini Bul

	X	Y	Z
A	(4, 3)	(0, 2)	(1, 1)
B	(2, 1)	(3, 3)	(0, 2)
C	(1, 4)	(2, 0)	(3, 2)

U1: satır oyuncusunun, U2: sütun oyuncusunun kazanç matrisi. m = n = 3, **denge = []**.

Adım 1 / 13

```
1 def saf_nash(U1, U2):
▶     m, n, denge = len(U1), len(U1[0]), []
3     for i in range(m):
4         for j in range(n):
5             br1 = max(U1[k][j] for k in range(m))
6             br2 = max(U2[i][l] for l in range(n))
7             if U1[i][j] == br1 and U2[i][j] == br2:
8                 denge.append((i, j))
9     return denge
```

DEĞİŞKENLER

`m = 3` `n = 3` `denge = []`

EN İYİ TEPKİ

Kod İzleme: Saf Nash Dengelerini Bul

	X	Y	Z
A	(4, 3)	(0, 2)	(1, 1)
B	(2, 1)	(3, 3)	(0, 2)
C	(1, 4)	(2, 0)	(3, 2)

Hücre **(A, X)**: U1 = 4 bu sütunun en büyüğü, U2 = 3 bu satırın en büyüğü → koşul **doğru**.

Adım 2 / 13

```
1 def saf_nash(U1, U2):
2     m, n, denge = len(U1), len(U1[0]), []
3     for i in range(m):
4         for j in range(n):
▶             br1 = max(U1[k][j] for k in range(m))
▶             br2 = max(U2[i][l] for l in range(n))
▶             if U1[i][j] == br1 and U2[i][j] == br2:
8                 denge.append((i, j))
9     return denge
```

DEĞİŞKENLER

`i = 0` `j = 0` `U1[i][j] = 4` `br1 = 4` `U2[i][j] = 3` `br2 = 3` `denge = []`

EN İYİ TEPKİ

Kod İzleme: Saf Nash Dengelerini Bul

	X	Y	Z
A	(4, 3)	(0, 2)	(1, 1)
B	(2, 1)	(3, 3)	(0, 2)
C	(1, 4)	(2, 0)	(3, 2)

(0, 0) listeye eklenir: **denge = [(0, 0)]**.

Adım 3 / 13

```
1 def saf_nash(U1, U2):
2     m, n, denge = len(U1), len(U1[0]), []
3     for i in range(m):
4         for j in range(n):
5             br1 = max(U1[k][j] for k in range(m))
6             br2 = max(U2[i][l] for l in range(n))
7             if U1[i][j] == br1 and U2[i][j] == br2:
▶                 denge.append((i, j))
9     return denge
```

DEĞİŞKENLER

`i = 0` `j = 0` `denge = [(0, 0)]`

EN İYİ TEPKİ

Kod İzleme: Saf Nash Dengelerini Bul

	X	Y	Z
A	(4, 3)	(0, 2)	(1, 1)
B	(2, 1)	(3, 3)	(0, 2)
C	(1, 4)	(2, 0)	(3, 2)

Hücre **(A, Y)**: U1 = 0 < 3 ve U2 = 2 < 3 → koşul yanlış, denge değil.

Adım 4 / 13

```
1 def saf_nash(U1, U2):
2     m, n, denge = len(U1), len(U1[0]), []
3     for i in range(m):
4         for j in range(n):
▶             br1 = max(U1[k][j] for k in range(m))
▶             br2 = max(U2[i][l] for l in range(n))
▶             if U1[i][j] == br1 and U2[i][j] == br2:
8                 denge.append((i, j))
9     return denge
```

DEĞİŞKENLER

`i = 0` `j = 1` `U1[i][j] = 0` `br1 = 3` `U2[i][j] = 2` `br2 = 3` `denge = [(0, 0)]`

EN İYİ TEPKİ

Kod İzleme: Saf Nash Dengelerini Bul

	X	Y	Z
A	(4, 3)	(0, 2)	(1, 1)
B	(2, 1)	(3, 3)	(0, 2)
C	(1, 4)	(2, 0)	(3, 2)

Hücre **(A, Z)**: U1 = 1 < 3 ve U2 = 1 < 3 → koşul yanlış, denge değil.

Adım 5 / 13

```
1 def saf_nash(U1, U2):
2     m, n, denge = len(U1), len(U1[0]), []
3     for i in range(m):
4         for j in range(n):
▶             br1 = max(U1[k][j] for k in range(m))
▶             br2 = max(U2[i][l] for l in range(n))
▶             if U1[i][j] == br1 and U2[i][j] == br2:
8                 denge.append((i, j))
9     return denge
```

DEĞİŞKENLER

`i = 0` `j = 2` `U1[i][j] = 1` `br1 = 3` `U2[i][j] = 1` `br2 = 3` `denge = [(0, 0)]`

EN İYİ TEPKİ

Kod İzleme: Saf Nash Dengelerini Bul

	X	Y	Z
A	(4, 3)	(0, 2)	(1, 1)
B	(2, 1)	(3, 3)	(0, 2)
C	(1, 4)	(2, 0)	(3, 2)

Hücre **(B, X)**: U1 = 2 < 4 ve U2 = 1 < 3 → koşul yanlış, denge değil.

Adım 6 / 13

```
1 def saf_nash(U1, U2):
2     m, n, denge = len(U1), len(U1[0]), []
3     for i in range(m):
4         for j in range(n):
▶             br1 = max(U1[k][j] for k in range(m))
▶             br2 = max(U2[i][l] for l in range(n))
▶             if U1[i][j] == br1 and U2[i][j] == br2:
8                 denge.append((i, j))
9     return denge
```

DEĞİŞKENLER

`i = 1` `j = 0` `U1[i][j] = 2` `br1 = 4` `U2[i][j] = 1` `br2 = 3` `denge = [(0, 0)]`

EN İYİ TEPKİ

Kod İzleme: Saf Nash Dengelerini Bul

	X	Y	Z
A	(4, 3)	(0, 2)	(1, 1)
B	(2, 1)	(3, 3)	(0, 2)
C	(1, 4)	(2, 0)	(3, 2)

Hücre **(B, Y)**: U1 = 3 bu sütunun en büyüğü, U2 = 3 bu satırın en büyüğü → koşul **doğru**.

Adım 7 / 13

```
1 def saf_nash(U1, U2):
2     m, n, denge = len(U1), len(U1[0]), []
3     for i in range(m):
4         for j in range(n):
▶             br1 = max(U1[k][j] for k in range(m))
▶             br2 = max(U2[i][l] for l in range(n))
▶             if U1[i][j] == br1 and U2[i][j] == br2:
8                 denge.append((i, j))
9     return denge
```

DEĞİŞKENLER

`i = 1` `j = 1` `U1[i][j] = 3` `br1 = 3` `U2[i][j] = 3` `br2 = 3` `denge = [(0, 0)]`

EN İYİ TEPKİ

Kod İzleme: Saf Nash Dengelerini Bul

	X	Y	Z
A	(4, 3)	(0, 2)	(1, 1)
B	(2, 1)	(3, 3)	(0, 2)
C	(1, 4)	(2, 0)	(3, 2)

(1, 1) listeye eklenir: **denge = [(0, 0), (1, 1)]**.

Adım 8 / 13

```
1 def saf_nash(U1, U2):
2     m, n, denge = len(U1), len(U1[0]), []
3     for i in range(m):
4         for j in range(n):
5             br1 = max(U1[k][j] for k in range(m))
6             br2 = max(U2[i][l] for l in range(n))
7             if U1[i][j] == br1 and U2[i][j] == br2:
▶                 denge.append((i, j))
9     return denge
```

DEĞİŞKENLER

`i = 1` `j = 1`

`denge = [(0, 0), (1, 1)]`

EN İYİ TEPKİ

Kod İzleme: Saf Nash Dengelerini Bul

	X	Y	Z
A	(4, 3)	(0, 2)	(1, 1)
B	(2, 1)	(3, 3)	(0, 2)
C	(1, 4)	(2, 0)	(3, 2)

Hücre **(B, Z)**: U1 = 0 < 3 ve U2 = 2 < 3 → koşul yanlış, denge değil.

Adım 9 / 13

```
1 def saf_nash(U1, U2):
2     m, n, denge = len(U1), len(U1[0]), []
3     for i in range(m):
4         for j in range(n):
▶             br1 = max(U1[k][j] for k in range(m))
▶             br2 = max(U2[i][l] for l in range(n))
▶             if U1[i][j] == br1 and U2[i][j] == br2:
8                 denge.append((i, j))
9     return denge
```

DEĞİŞKENLER

`i = 1` `j = 2` `U1[i][j] = 0` `br1 = 3` `U2[i][j] = 2` `br2 = 3` `denge = [(0, 0), (1, 1)]`

EN İYİ TEPKİ

Kod İzleme: Saf Nash Dengelerini Bul

	X	Y	Z
A	(4, 3)	(0, 2)	(1, 1)
B	(2, 1)	(3, 3)	(0, 2)
C	(1, 4)	(2, 0)	(3, 2)

Hücre **(C, X)**: U1 = 1 < 4 → koşul yanlış, denge değil.

Adım 10 / 13

```
1 def saf_nash(U1, U2):
2     m, n, denge = len(U1), len(U1[0]), []
3     for i in range(m):
4         for j in range(n):
▶             br1 = max(U1[k][j] for k in range(m))
▶             br2 = max(U2[i][l] for l in range(n))
▶             if U1[i][j] == br1 and U2[i][j] == br2:
8                 denge.append((i, j))
9     return denge
```

DEĞİŞKENLER

`i = 2` `j = 0` `U1[i][j] = 1` `br1 = 4` `U2[i][j] = 4` `br2 = 4` `denge = [(0, 0), (1, 1)]`

EN İYİ TEPKİ

Kod İzleme: Saf Nash Dengelerini Bul

	X	Y	Z
A	(4, 3)	(0, 2)	(1, 1)
B	(2, 1)	(3, 3)	(0, 2)
C	(1, 4)	(2, 0)	(3, 2)

Hücre **(C, Y)**: U1 = 2 < 3 ve U2 = 0 < 4 → koşul yanlış, denge değil.

Adım 11 / 13

```
1 def saf_nash(U1, U2):
2     m, n, denge = len(U1), len(U1[0]), []
3     for i in range(m):
4         for j in range(n):
▶             br1 = max(U1[k][j] for k in range(m))
▶             br2 = max(U2[i][l] for l in range(n))
▶             if U1[i][j] == br1 and U2[i][j] == br2:
8                 denge.append((i, j))
9     return denge
```

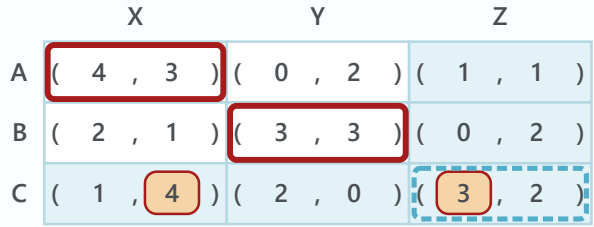
DEĞİŞKENLER

`i = 2` `j = 1` `U1[i][j] = 2` `br1 = 3` `U2[i][j] = 0` `br2 = 4`

`denge = [(0, 0), (1, 1)]`

EN İYİ TEPKİ

Kod İzleme: Saf Nash Dengelerini Bul



Hücre **(C, Z)**: U2 = 2 < 4 → koşul yanlış, denge değil.

Adım 12 / 13

```
1 def saf_nash(U1, U2):
2     m, n, denge = len(U1), len(U1[0]), []
3     for i in range(m):
4         for j in range(n):
▶             br1 = max(U1[k][j] for k in range(m))
▶             br2 = max(U2[i][l] for l in range(n))
▶             if U1[i][j] == br1 and U2[i][j] == br2:
8                 denge.append((i, j))
9     return denge
```

DEĞİŞKENLER

`i = 2` `j = 2` `U1[i][j] = 3`

`br1 = 3` `U2[i][j] = 2` `br2 = 4`

`denge = [(0, 0), (1, 1)]`

EN İYİ TEPKİ

Kod İzleme: Saf Nash Dengelerini Bul

	X	Y	Z
A	(4, 3)	(0, 2)	(1, 1)
B	(2, 1)	(3, 3)	(0, 2)
C	(1, 4)	(2, 0)	(3, 2)

Tüm 9 hücre tarandı: **return [(0, 0), (1, 1)]** → (A, X) ve (B, Y). Toplam iş O(mn·(m+n)); maksimumlar önceden hesaplanırsa O(mn).

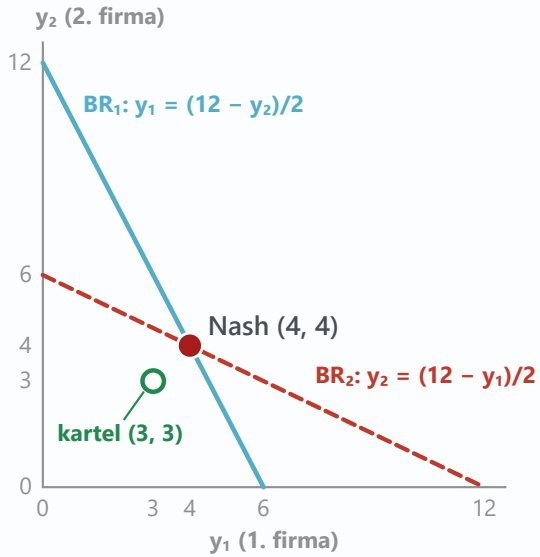
Adım 13 / 13

```
1 def saf_nash(U1, U2):
2     m, n, denge = len(U1), len(U1[0]), []
3     for i in range(m):
4         for j in range(n):
5             br1 = max(U1[k][j] for k in range(m))
6             br2 = max(U2[i][l] for l in range(n))
7             if U1[i][j] == br1 and U2[i][j] == br2:
8                 denge.append((i, j))
▶       return denge
```

DEĞİŞKENLER

`denge = [(0, 0), (1, 1)]`

Sürekli Stratejiler: Cournot Duopolü



- İki firma aynı anda üretim miktarı $y_1, y_2 \geq 0$ seçer (strateji kümesi sonsuz). Fiyat $P = 12 - (y_1 + y_2)$, maliyet 0.
- Kâr $u_1 = y_1 (12 - y_1 - y_2)$. Türev: $12 - 2y_1 - y_2 = 0 \Rightarrow$ **$BR_1(y_2) = \max\{0, (12 - y_2)/2\}$** (miktar negatif olamaz; $y_2 \geq 12$ ise en iyisi üretmemek).
- Nash dengesi = en iyi tepki doğrularının kesişimi: **$y_1 = y_2 = 4$**, fiyat 4, kâr 16'şar.
- Kartel (3'er üretim) ikisine de 18 verir ama dengede değildir: $BR_1(3) = 4{,}5$ ve sapan firma 20,25 alır.

Firmalar sırayla birbirine en iyi tepki verirse üretimler (6, 3), sonra (4,5 ile 3,75), … diye ilerler ve (4, 4)'e yaklaşır. Cournot, 1838.

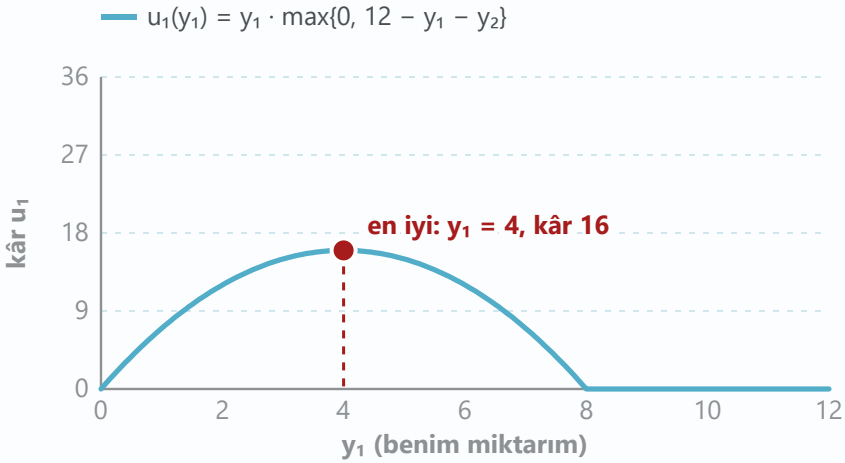
EN İYİ TEPKİ

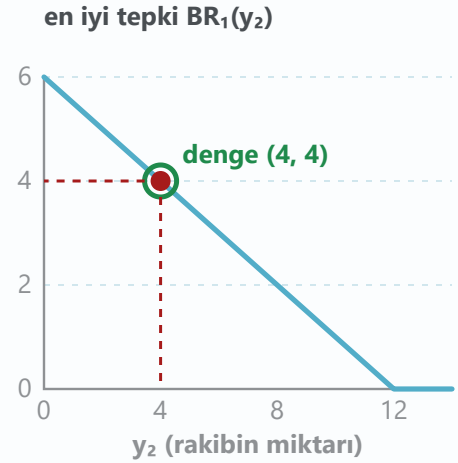
Rakip Ne Kadar Üretirse Ben Ne Kadar Üretmeliyim?

$$u_1(y_1) = y_1 \cdot \max\{0,\ 12 - y_1 - y_2\} \ \Rightarrow\ \mathbf{BR_1(y_2) = \max\{0,\ (12 - y_2)/2\}}$$

y_2 = 4

$BR_1(4) = \max\{0,\ (12 - 4)/2\} = \mathbf{4}$ | fiyat = **4** | en iyi kâr = **16**





Rakip daha çok ürettikçe benim en iyi miktarım azalır; $y_2 \geq 12$ iken üretmemek en iyisidir. İki firmanın birbirine en iyi tepki verdiği tek nokta $\mathbf{y_1 = y_2 = 4}$'tür.

EN İYİ TEPKİ

Bertrand: Miktar Yerine Fiyatla Rekabet

- İki firma aynı malı satar; birim maliyet c = 2 TL, pazar 100 birim. Firmalar **aynı anda fiyat** seçer: ucuz olan pazarın hepsini alır, eşit fiyatta pazar yarı yarıya bölünür.
- (5, 5): her biri (5 − 2) · 50 = 150 kazanır. Biri 4,99'a inerse ≈ 2,99 · 100 ≈ 299: sapma kârlı. c'nin üstündeki *her* ortak fiyatta aynı şey olur.
- c'nin altına inmek zarar ettirir. Tek Nash dengesi **$p_1 = p_2 = c = 2$**, kâr 0.



Rekabet aracı	Denge	Firma kârı
Cournot (miktar), P = 12 − Y, maliyet 0	y = 4'er, fiyat 4	16'şar
Bertrand (fiyat), maliyet c	p = c	0

Sık hata: "İki firma varsa kâr yüksek kalır" demek. Fiyatla rekabette iki firma bile fiyatı maliyete indirir (**Bertrand paradoksu**); kâr ancak ürün farkı, kapasite sınırı ya da tekrarlanan etkileşimle korunur. Sonuç sürekli fiyat içindir; 1 TL'lik adımlarda c'nin hemen üstündeki fiyatlar da denge olabilir.

BASKIN STRATEJİ ELEME

Baskın ve Baskılanan Strateji

Kesin baskılama (strict dominance)

s_i', s_i'yi kesin baskılar: her s_{-i} için $u_i(s_i', s_{-i}) > u_i(s_i, s_{-i})$.

`Rasyonel oyuncu kesin baskılanan stratejiyi asla oynamaz.`

Zayıf baskılama (weak dominance)

Her s_{-i} için ≥ ve en az bir s_{-i} için >.

`Silinirse bazı dengeler kaybolabilir; sonuç eleme sırasına bağlı olabilir.`

- **Baskın strateji**: diğer tüm stratejilerini baskılayan strateji (Mahkûmun İkilemi'nde İhanet). Sahibi rakibini tahmin etmek zorunda değildir; herkesin baskın stratejisi varsa bu profil bir Nash dengesidir (**baskın strateji dengesi**). Her oyunda bulunmaz (TKM).
- **Yinelemeli eleme (IESDS)**: kesin baskılanan stratejileri sil, küçülen oyunda tekrarla. Herkes rasyonel *ve* bunu herkes biliyor (ortak bilgi) varsayımı.
- Kesin baskılamada **sıra önemsizdir**; hiçbir Nash dengesi silinmez. Tek hücre kalırsa o hücre oyunun **tek** Nash dengesidir.
- Test her karşı seçim için ayrı yapılır; "ortalamada daha iyi" yetmez. 2/3 ortalaması oyununda 60'ın üstündeki sayılar 60'tan hiçbir zaman iyi değildir ama bazen eşit kalır; oradaki eleme bu yüzden **zayıf** baskılamadır.

Oyuna Dönüş: Mahkûmun İkilemi

Sam \ John	Sessiz	İhanet
Sessiz	(−1, −1)	(−3, **0**)
İhanet	(**0**, −3)	(**−2**, **−2**)

Kazanç = −yıl: her oyuncunun en iyi tepkisi işaretli

- **Sessiz mi, ihanet mi? Kuramın cevabı:** İhanet baskın strateji: John sessizse 0 > −1, ihanet ederse −2 > −3. Tek Nash dengesi (İhanet, İhanet): (−2, −2), yani 2'şer yıl.
- **Gerçekte:** tek seferlik oyunda insanlar kuramdan çok daha sık işbirliği yapar: 1958–1992 deneylerini birleştiren meta-analize göre katılımcıların kabaca yarısı işbirliği yapar ve iletişim bu oranı belirgin biçimde artırır (Sally, 1995). Güven, adalet ve karşılıklılık da seçimi etkiler.

John Tit-for-Tat oynarken Sam'in 10 turluk planı	Sam'in toplam yılı
Hep Sessiz	10 × 1 = 10
Hep İhanet	0 + 9 × 2 = 18
9 tur Sessiz, son tur İhanet	9 × 1 + 0 = **9** (en az)

Tekrarlanan oyunda ihanet sonraki turlarda cezalandırılır; yalnız son turda ceza kalmaz. Bu “son tur etkisi” geriye doğru çıkarımın konusudur.

Yinelemeli Eleme: 3×3 Örnek

Satır oyuncusu \ Sütun oyuncusu	X	Y	Z
A	(3, 2)	(3, 3)	(1, 4)
B	(3, 2)	(0, 4)	(5, 3)
C	(2, 1)	(2, 0)	(6, 6)

Başlangıç: hiçbir satır diğerini kesin baskılamıyor (ör. A ile B: 3 = 3). Önce sütun oyuncusuna bakalım.

Yinelemeli Eleme: 3×3 Örnek

Satır oyuncusu \ Sütun oyuncusu	X	Y	Z
A	(3, 2)	(3, 3)	(1, 4)
B	(3, 2)	(0, 4)	(5, 3)
C	(2, 1)	(2, 0)	(6, 6)

Kalan satırlarda **Z**, **X**'i kesin baskılar (u_2: 4 > 2, 3 > 2, 6 > 1) → X elenir.

Yinelemeli Eleme: 3×3 Örnek

Satır oyuncusu \ Sütun oyuncusu	X	Y	Z
A	(3 , 2)	(3 , 3)	(1 , 4)
B	(3 , 2)	(0 , 4)	(5 , 3)
C	(2 , 1)	(2 , 0)	(6 , 6)

Kalan sütunlarda **C**, **B**'yi kesin baskılar (u_1: 2 > 0, 6 > 5) → B elenir.

Yinelemeli Eleme: 3×3 Örnek

Satır oyuncusu \ Sütun oyuncusu	X	Y	Z
A	(3, 2)	(3, 3)	(1, 4)
B	(3, 2)	(0, 4)	(5, 3)
C	(2, 1)	(2, 0)	(6, 6)

Kalan satırlarda **Z**, **Y**'yi kesin baskılar (u_2: 4 > 3, 6 > 0) → Y elenir.

Yinelemeli Eleme: 3×3 Örnek

Satır oyuncusu \ Sütun oyuncusu	X	Y	Z
A	(3, 2)	(3, 3)	(1, 4)
B	(3, 2)	(0, 4)	(5, 3)
C	(2, 1)	(2, 0)	(6, 6)

Kalan sütunlarda **C**, **A**'yı kesin baskılar (u_1: 6 > 1) → A elenir.

Yinelemeli Eleme: 3×3 Örnek

Satır oyuncusu \ Sütun oyuncusu	X	Y	Z
A	(3, 2)	(3, 3)	(1, 4)
B	(3, 2)	(0, 4)	(5, 3)
C	(2, 1)	(2, 0)	**(6, 6)**

Tek hücre kaldı: **(C, Z) = (6, 6)**. Bu, oyunun tek Nash dengesidir (en iyi tepki işaretlemesi de aynı sonucu verir).

Adım Adım: Bir Eleme Ötekinin Yolunu Açar

Satır oyuncusu	Sütun oyuncusu: Sol	Merkez	Sağ
Üst	(1, 5)	(5, 6)	(6, 2)
Orta	(3, 5)	(6, 0)	(2, 2)
Alt	(2, 2)	(0, 2)	(1, 5)

Hiçbir oyuncunun baskın stratejisi yok; başta tek bir kesin baskılama var. Satırlar yalnız **ilk** sayılarla, sütunlar yalnız **ikinci** sayılarla karşılaştırılır.

Adım Adım: Bir Eleme Ötekinin Yolunu Açar

Satır oyuncusu	Sütun oyuncusu: Sol	Sütun oyuncusu: Merkez	Sütun oyuncusu: Sağ
Üst	(1, 5)	(5, 6)	(6, 2)
Orta	(3, 5)	(6, 0)	(2, 2)
Alt	(2, 2)	(0, 2)	(1, 5)

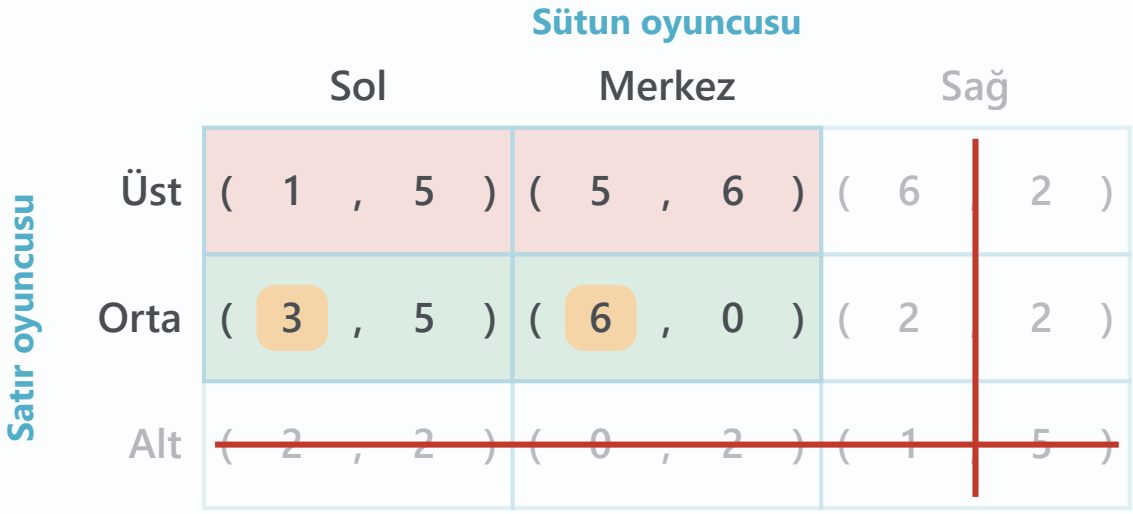
Orta, Alt'ı her sütunda yener: 3 > 2, 6 > 0, 2 > 1 → **Alt silinir**.

Adım Adım: Bir Eleme Ötekinin Yolunu Açar

Satır oyuncusu \ Sütun oyuncusu	Sol	Merkez	Sağ
Üst	(1, **5**)	(5, 6)	(6, 2)
Orta	(3, **5**)	(6, 0)	(2, 2)
~~Alt~~	~~(2, 2)~~	~~(0, 2)~~	~~(1, 5)~~

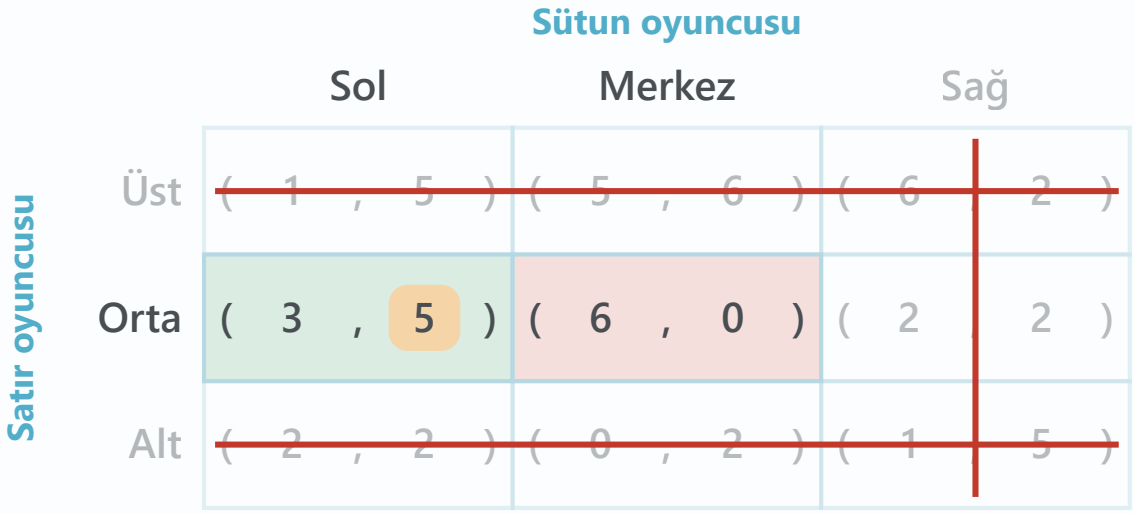
Kalan satırlarda Sol, Sağ'ı yener: 5 > 2 (Üst), 5 > 2 (Orta) → **Sağ silinir**. Alt dururken olmazdı: orada Sağ 5 > Sol 2.

Adım Adım: Bir Eleme Ötekinin Yolunu Açar



Kalan sütunlarda Orta, Üst'ü yener: 3 > 1, 6 > 5 → **Üst silinir**. Sağ dururken Üst orada 6 > 2 ile kurtuluyordu.

Adım Adım: Bir Eleme Ötekinin Yolunu Açar



Yalnız Orta satırı kaldı: Sol 5, Merkez 0 → **Merkez silinir**.

Adım Adım: Bir Eleme Ötekinin Yolunu Açar



Kalan **(Orta, Sol) = (3, 5)** tek Nash dengesidir. (Üst, Merkez) = (5, 6) ikisine de daha iyi ama denge değil: satır oyuncusu Orta'ya geçip 6 alır.

Kod İzleme: Yinelemeli Baskın Strateji Eleme

	X	Y	Z
A	(3 , 2)	(3 , 3)	(1 , 4)
B	(3 , 2)	(0 , 4)	(5 , 3)
C	(2 , 1)	(2 , 0)	(6 , 6)

Kalan satır ve sütun listeleri: **R = [0, 1, 2]** (A, B, C), **S = [0, 1, 2]** (X, Y, Z).

Adım 1 / 11

```
1  def ele(U1, U2):
▶      R, S = [0, 1, 2], [0, 1, 2]   # kalanlar
3      while True:
4          x = [r for r in R for k in R if
5               all(U1[k][c] > U1[r][c] for c in S)]
6          y = [c for c in S for d in S if
7               all(U2[r][d] > U2[r][c] for r in R)]
8          if x: R.remove(x[0])      # satır elenir
9          elif y: S.remove(y[0])    # sütun elenir
10         else: return R, S
```

DEĞİŞKENLER

`R = [0, 1, 2]` `S = [0, 1, 2]`

Kod İzleme: Yinelemeli Baskın Strateji Eleme

	X	Y	Z
A	(3 , 2)	(3 , 3)	(1 , 4)
B	(3 , 2)	(0 , 4)	(5 , 3)
C	(2 , 1)	(2 , 0)	(6 , 6)

Hiçbir satır baskılanmıyor (x = []). Sütun X, Z tarafından baskılanıyor → **y = [0]**.

Adım 2 / 11

```
1  def ele(U1, U2):
2      R, S = [0, 1, 2], [0, 1, 2]   # kalanlar
3      while True:
▶          x = [r for r in R for k in R if
▶               all(U1[k][c] > U1[r][c] for c in S)]
▶          y = [c for c in S for d in S if
▶               all(U2[r][d] > U2[r][c] for r in R)]
8          if x: R.remove(x[0])      # satır elenir
9          elif y: S.remove(y[0])    # sütun elenir
10         else: return R, S
```

DEĞİŞKENLER

`R = [0, 1, 2]` `S = [0, 1, 2]` `x = []` `y = [0]`

Kod İzleme: Yinelemeli Baskın Strateji Eleme

	X	Y	Z
A	(3 , 2)	(3 , 3)	(1 , 4)
B	(3 , 2)	(0 , 4)	(5 , 3)
C	(2 , 1)	(2 , 0)	(6 , 6)

S.remove(0): X sütunu silinir; sıradaki turda satırlar yeniden denetlenir.

Adım 3 / 11

```
1  def ele(U1, U2):
2      R, S = [0, 1, 2], [0, 1, 2]   # kalanlar
3      while True:
4          x = [r for r in R for k in R if
5               all(U1[k][c] > U1[r][c] for c in S)]
6          y = [c for c in S for d in S if
7               all(U2[r][d] > U2[r][c] for r in R)]
8          if x: R.remove(x[0])      # satır elenir
▶          elif y: S.remove(y[0])    # sütun elenir
10         else: return R, S
```

DEĞİŞKENLER

R = [0, 1, 2] S = [1, 2]

x = [] y = [0]

Kod İzleme: Yinelemeli Baskın Strateji Eleme

	X	Y	Z
A	(3 , 2)	(3 , 3)	(1 , 4)
B	(3 , 2)	(0 , 4)	(5 , 3)
C	(2 , 1)	(2 , 0)	(6 , 6)

Satır B, C tarafından baskılanıyor → **x = [1]**. Sütunlara da bakılır: y = [].

Adım 4 / 11

```
1  def ele(U1, U2):
2      R, S = [0, 1, 2], [0, 1, 2]   # kalanlar
3      while True:
▶          x = [r for r in R for k in R if
▶               all(U1[k][c] > U1[r][c] for c in S)]
▶          y = [c for c in S for d in S if
▶               all(U2[r][d] > U2[r][c] for r in R)]
8          if x: R.remove(x[0])      # satır elenir
9          elif y: S.remove(y[0])    # sütun elenir
10         else: return R, S
```

DEĞİŞKENLER

R = [0, 1, 2] S = [1, 2]

x = [1] y = []

Kod İzleme: Yinelemeli Baskın Strateji Eleme

	X	Y	Z
A	(3 , 2)	(3 , 3)	(1 , 4)
B	(3 , 2)	(0 , 4)	(5 , 3)
C	(2 , 1)	(2 , 0)	(6 , 6)

R.remove(1): B satırı silinir, döngü küçülen oyunla yeniden başlar.

Adım 5 / 11

```
1  def ele(U1, U2):
2      R, S = [0, 1, 2], [0, 1, 2]   # kalanlar
3      while True:
4          x = [r for r in R for k in R if
5               all(U1[k][c] > U1[r][c] for c in S)]
6          y = [c for c in S for d in S if
7               all(U2[r][d] > U2[r][c] for r in R)]
▶          if x: R.remove(x[0])      # satır elenir
9          elif y: S.remove(y[0])    # sütun elenir
10         else: return R, S
```

DEĞİŞKENLER

`R = [0, 2]` `S = [1, 2]` `x = [1]` `y = []`

Kod İzleme: Yinelemeli Baskın Strateji Eleme

	X	Y	Z
A	(3 , 2)	(3 , 3)	(1 , 4)
B	(3 , 2)	(0 , 4)	(5 , 3)
C	(2 , 1)	(2 , 0)	(6 , 6)

Hiçbir satır baskılanmıyor (x = []). Sütun Y, Z tarafından baskılanıyor → **y = [1]**.

Adım 6 / 11

```
1  def ele(U1, U2):
2      R, S = [0, 1, 2], [0, 1, 2]   # kalanlar
3      while True:
▶          x = [r for r in R for k in R if
▶               all(U1[k][c] > U1[r][c] for c in S)]
▶          y = [c for c in S for d in S if
▶               all(U2[r][d] > U2[r][c] for r in R)]
8          if x: R.remove(x[0])      # satır elenir
9          elif y: S.remove(y[0])    # sütun elenir
10         else: return R, S
```

DEĞİŞKENLER

R = [0, 2] | S = [1, 2] | x = [] | y = [1]

Kod İzleme: Yinelemeli Baskın Strateji Eleme

	X	Y	Z
A	(3 , 2)	(3 , 3)	(1 , 4)
B	(3 , 2)	(0 , 4)	(5 , 3)
C	(2 , 1)	(2 , 0)	(6 , 6)

S.remove(1): Y sütunu silinir; sıradaki turda satırlar yeniden denetlenir.

Adım 7 / 11

```
1  def ele(U1, U2):
2      R, S = [0, 1, 2], [0, 1, 2]   # kalanlar
3      while True:
4          x = [r for r in R for k in R if
5               all(U1[k][c] > U1[r][c] for c in S)]
6          y = [c for c in S for d in S if
7               all(U2[r][d] > U2[r][c] for r in R)]
8          if x: R.remove(x[0])      # satır elenir
▶          elif y: S.remove(y[0])    # sütun elenir
10         else: return R, S
```

DEĞİŞKENLER

R = [0, 2] S = [2] x = [] y = [1]

Kod İzleme: Yinelemeli Baskın Strateji Eleme

	X	Y	Z
A	(3 , 2)	(3 , 3)	(1 , 4)
B	(3 , 2)	(0 , 4)	(5 , 3)
C	(2 , 1)	(2 , 0)	(6 , 6)

Satır A, C tarafından baskılanıyor → **x = [0]**. Sütunlara da bakılır: y = [].

Adım 8 / 11

```
1  def ele(U1, U2):
2      R, S = [0, 1, 2], [0, 1, 2]   # kalanlar
3      while True:
▶          x = [r for r in R for k in R if
▶               all(U1[k][c] > U1[r][c] for c in S)]
▶          y = [c for c in S for d in S if
▶               all(U2[r][d] > U2[r][c] for r in R)]
8          if x: R.remove(x[0])      # satır elenir
9          elif y: S.remove(y[0])    # sütun elenir
10         else: return R, S
```

DEĞİŞKENLER

`R = [0, 2]` `S = [2]` `x = [0]` `y = []`

Kod İzleme: Yinelemeli Baskın Strateji Eleme

	X	Y	Z
A	(3 , 2)	(3 , 3)	(1 , 4)
B	(3 , 2)	(0 , 4)	(5 , 3)
C	(2 , 1)	(2 , 0)	(6 , 6)

R.remove(0): A satırı silinir, döngü küçülen oyunla yeniden başlar.

Adım 9 / 11

```
1  def ele(U1, U2):
2      R, S = [0, 1, 2], [0, 1, 2]   # kalanlar
3      while True:
4          x = [r for r in R for k in R if
5               all(U1[k][c] > U1[r][c] for c in S)]
6          y = [c for c in S for d in S if
7               all(U2[r][d] > U2[r][c] for r in R)]
▶          if x: R.remove(x[0])      # satır elenir
9          elif y: S.remove(y[0])    # sütun elenir
10         else: return R, S
```

DEĞİŞKENLER

`R = [2]` `S = [2]` `x = [0]` `y = []`

Kod İzleme: Yinelemeli Baskın Strateji Eleme

	X	Y	Z
A	(3 , 2)	(3 , 3)	(1 , 4)
B	(3 , 2)	(0 , 4)	(5 , 3)
C	(2 , 1)	(2 , 0)	(6 , 6)

Ne satır ne sütun baskılanıyor: **x = y = []**.

Adım 10 / 11

```
1  def ele(U1, U2):
2      R, S = [0, 1, 2], [0, 1, 2]   # kalanlar
3      while True:
▶          x = [r for r in R for k in R if
▶               all(U1[k][c] > U1[r][c] for c in S)]
▶          y = [c for c in S for d in S if
▶               all(U2[r][d] > U2[r][c] for r in R)]
8          if x: R.remove(x[0])      # satır elenir
9          elif y: S.remove(y[0])    # sütun elenir
10         else: return R, S
```

DEĞİŞKENLER

R = [2] | S = [2] | x = [] | y = []

Kod İzleme: Yinelemeli Baskın Strateji Eleme

	X	Y	Z
A	(3 , 2)	(3 , 3)	(1 , 4)
B	(3 , 2)	(0 , 4)	(5 , 3)
C	(2 , 1)	(2 , 0)	(6 , 6)

return [2], [2] → yalnızca (C, Z) kaldı. 4 elemede sırayla X, B, Y, A silindi.

Adım 11 / 11

```
1 def ele(U1, U2):
2     R, S = [0, 1, 2], [0, 1, 2]   # kalanlar
3     while True:
4         x = [r for r in R for k in R if
5              all(U1[k][c] > U1[r][c] for c in S)]
6         y = [c for c in S for d in S if
7              all(U2[r][d] > U2[r][c] for r in R)]
8         if x: R.remove(x[0])      # satır elenir
9         elif y: S.remove(y[0])    # sütun elenir
▶         else: return R, S
```

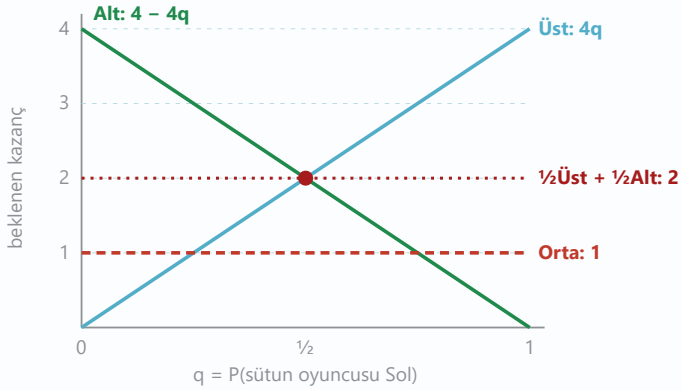
DEĞİŞKENLER

R = [2] S = [2] x = [] y = []

Karışık Stratejiyle Baskılama

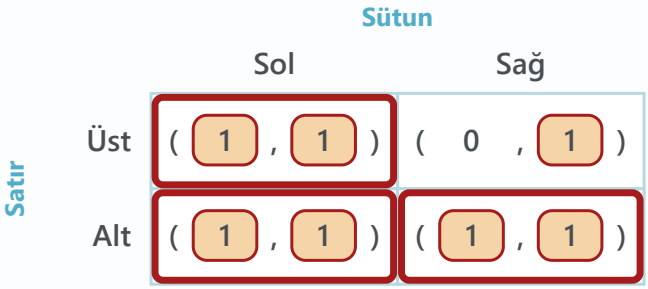
u_1	Sol	Sağ
Üst	4	0
Orta	1	1
Alt	0	4

- Orta'yı hiçbir saf satır baskılamaz: Sağ'da Üst'ten (1 > 0), Sol'da Alt'tan (1 > 0) iyidir.
- Üst ile Alt'ı ½–½ karıştıran satır oyuncusu her sütunda 2 alır: 2 > 1 ve 2 > 1 → Orta, **karışık stratejiyle kesin baskılanır** ve silinebilir.
- Grafikte: Orta'nın doğrusu (1) her q için Üst ile Alt'ın üst zarfının (en az 2) altında: Orta hiçbir inanca karşı en iyi tepki değildir.



Sık hata: "Onu yenen tek bir saf strateji yok, o hâlde silinemez." Karışımlar da karşılaştırmaya girer; eleme yaparken önce saf, sonra ½–½ gibi basit karışımlar denenir.

Zayıf Baskılamanın Tuzağı



- Satır için Alt, Üst'ü **zayıf** baskılar (1 = 1, 1 > 0). Sütun oyuncusu her durumda kayıtsızdır.
- Saf Nash dengeleri: **(Üst, Sol), (Alt, Sol), (Alt, Sağ)** — üç tane.
- Üst silinirse (Üst, Sol) dengesi kaybolur; eleme sırası farklı sonuçlar bırakabilir.
- Ders: kesin baskılama güvenli; zayıf baskılama **denge kaybettirebilir**.

Kesin eleme her oyunda tek hücreye inmez: Cinsiyetler Savaşı'nda hiçbir strateji baskılanmaz. O zaman en iyi tepki işaretlemesine ve karışık stratejilere başvururuz.

Sık Yapılan Hatalar: Baskın Strateji Eleme

Yanlış sayılar
İki satırı sütun oyuncusunun kazançlarıyla karşılaştırmak. Satır ↔ ilk sayı, sütun ↔ ikinci sayı.

"Çoğunlukla büyük" yeter sanmak
Kesin baskılama için kalan *her* sütunda büyük olmalı; tek bir eşitlik ya da küçüklük elemeyi durdurur.

Silinenleri saymak
Silinmiş sütun artık karşılaştırmaya girmez. Bir eleme zincirinde bir strateji (ör. Üst) ancak başka bir strateji (Sağ) silindikten sonra elenebilir.

Zayıfı kesin sanmak
Eşitlik varsa baskılama zayıftır; silmek denge kaybettirebilir ve sonuç sıraya bağlı olabilir.

Baskın ≠ baskılanan
Baskın strateji diğer *hepsini* yener; baskılanan strateji en az *birine* yenilir. Baskın strateji olmadan da eleme yapılır.

Kalanların hepsini denge sanmak
Eleme birden çok hücre bırakırsa (ör. dil seçimi oyununda 2×2 kalır) dengeler kalan küçük oyunda altını çizerek bulunur.

KARIŞIK STRATEJİ

Karışık Strateji ve Kayıtsızlık İlkesi

- **Karışık strateji**: oyuncunun her saf stratejisine verdiği olasılıklar (ör. Taş 1/3, Kağıt 1/3, Makas 1/3). Kazanç, beklenen kazançtır. 2×2 oyunlarda p = σ_1(ilk satır), q = σ_2(ilk sütun).
- Saf dengesi olmayan oyunlarda (Yazı-Tura, TKM, penaltı, denetim, blöf) oyuncular **tahmin edilemez** olmak ister. TKM dengesi (1/3, 1/3, 1/3): her eylem 0 getirir. TKM'de 1/3–1/3–1/3 dışındaki her karışımı rakip sömürebilir.
- **Destek (support)**: pozitif olasılık verilen saf stratejiler.

Kayıtsızlık ilkesi: Dengede her oyuncu, desteğindeki tüm saf stratejilerden *aynı* beklenen kazancı alır (destek dışındakiler daha fazla vermez). Aksi hâlde daha iyi olana tüm olasılığı kaydırırdı.

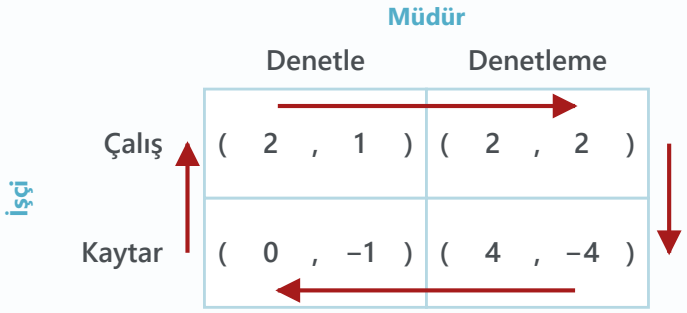
Şaşırtıcı sonuç

Satır oyuncusunun karışımı, *sütun oyuncusunun* kazançlarından hesaplanır (ve tersi).

Yorum

Olasılıklar “zar atmak” ya da büyük bir nüfustaki davranış oranları (ör. penaltıların yönü) olarak okunabilir.

Karışık Denge Hesabı: Denetim Oyunu



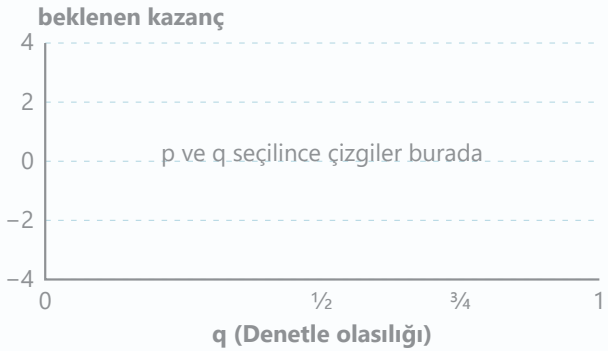
Denetim oyunu (inspection game)

Satır: işçi, sütun: müdür. Ücret 4, çaba maliyeti 2, üretim 6, denetim maliyeti 1. Kaytaran yakalanırsa ücret alamaz. Örnek: Çalış–Denetle hücresinde işçi 4 − 2 = 2, müdür 6 − 4 − 1 = 1 alır.

Oklar: her hücreden bir oyuncu kârlı sapıyor → en iyi tepkiler döngü oluşturur, **saf Nash dengesi yok**.

Karışık Denge Hesabı: Denetim Oyunu

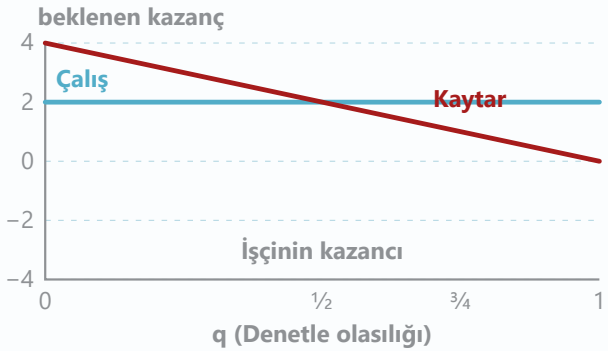
	Denetle	Denetleme
Çalış	(2 , 1)	(2 , 2)
Kaytar	(0 , −1)	(4 , −4)



1. İşçi **p** olasılıkla Çalış, müdür **q** olasılıkla Denetle oynar.

Karışık Denge Hesabı: Denetim Oyunu

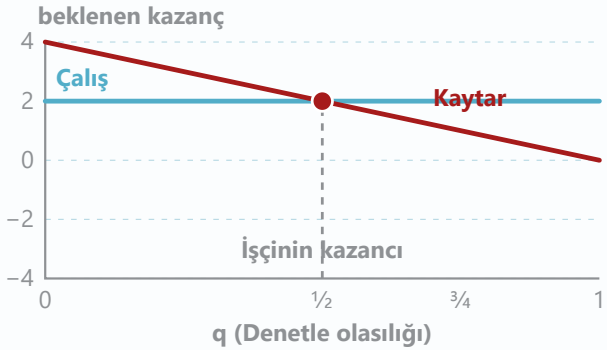
	Denetle	Denetleme
Çalış	(2 , 1)	(2 , 2)
Kaytar	(0 , −1)	(4 , −4)



1. İşçi **p** olasılıkla Çalış, müdür **q** olasılıkla Denetle oynar.
2. İşçi: E[Çalış] = 2q + 2(1−q) = **2**; E[Kaytar] = 0·q + 4(1−q) = **4 − 4q**

Karışık Denge Hesabı: Denetim Oyunu

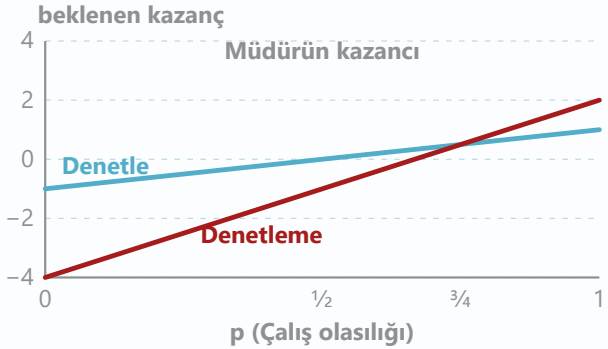
	Denetle	Denetleme
Çalış	(2 , 1)	(2 , 2)
Kaytar	(0 , −1)	(4 , −4)



1. İşçi **p** olasılıkla Çalış, müdür **q** olasılıkla Denetle oynar.
2. İşçi: E[Çalış] = 2q + 2(1−q) = **2**; E[Kaytar] = 0·q + 4(1−q) = **4 − 4q**
3. İşçi kayıtsız: 2 = 4 − 4q ⇒ **q* = 1/2**

Karışık Denge Hesabı: Denetim Oyunu

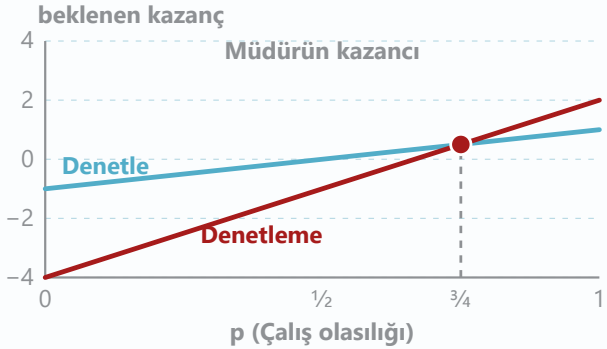
	Denetle	Denetleme
Çalış	(2 , 1)	(2 , 2)
Kaytar	(0 , −1)	(4 , −4)



1. İşçi **p** olasılıkla Çalış, müdür **q** olasılıkla Denetle oynar.
2. İşçi: E[Çalış] = 2q + 2(1−q) = **2**; E[Kaytar] = 0·q + 4(1−q) = **4 − 4q**
3. İşçi kayıtsız: 2 = 4 − 4q ⇒ **q* = 1/2**
4. Müdür: E[Denetle] = 1·p − 1·(1−p) = **2p − 1**; E[Denetleme] = 2p − 4(1−p) = **6p − 4**

Karışık Denge Hesabı: Denetim Oyunu

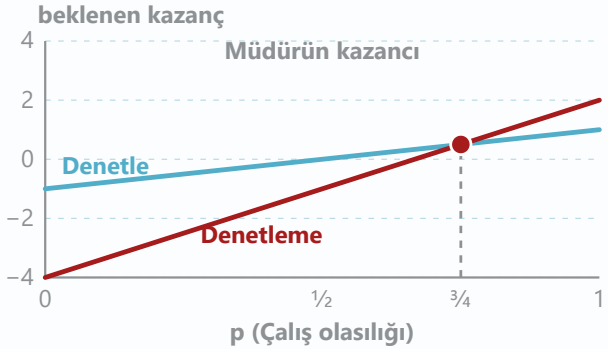
	Denetle	Denetleme
Çalış	(2 , 1)	(2 , 2)
Kaytar	(0 , −1)	(4 , −4)



1. İşçi **p** olasılıkla Çalış, müdür **q** olasılıkla Denetle oynar.
2. İşçi: E[Çalış] = 2q + 2(1−q) = **2**; E[Kaytar] = 0·q + 4(1−q) = **4 − 4q**
3. İşçi kayıtsız: 2 = 4 − 4q ⇒ **q* = 1/2**
4. Müdür: E[Denetle] = 1·p − 1·(1−p) = **2p − 1**; E[Denetleme] = 2p − 4(1−p) = **6p − 4**
5. Müdür kayıtsız: 2p − 1 = 6p − 4 ⇒ **p* = 3/4**

Karışık Denge Hesabı: Denetim Oyunu

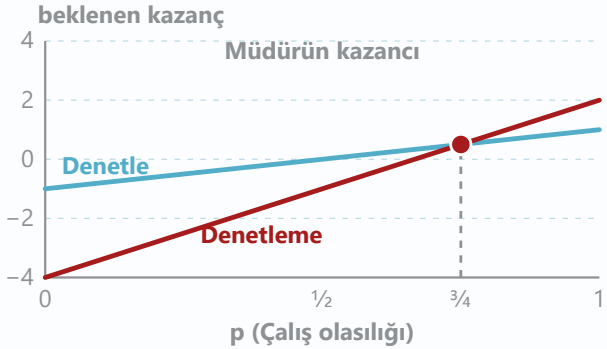
	Denetle	Denetleme
Çalış	(2 , 1)	(2 , 2)
Kaytar	(0 , −1)	(4 , −4)



1. İşçi **p** olasılıkla Çalış, müdür **q** olasılıkla Denetle oynar.
2. İşçi: E[Çalış] = 2q + 2(1−q) = **2**; E[Kaytar] = 0·q + 4(1−q) = **4 − 4q**
3. İşçi kayıtsız: 2 = 4 − 4q ⇒ **q* = 1/2**
4. Müdür: E[Denetle] = 1·p − 1·(1−p) = **2p − 1**; E[Denetleme] = 2p − 4(1−p) = **6p − 4**
5. Müdür kayıtsız: 2p − 1 = 6p − 4 ⇒ **p* = 3/4**
6. Denge: (p*, q*) = (3/4, 1/2). Beklenen kazanç: işçi 2, müdür 1/2.

Karışık Denge Hesabı: Denetim Oyunu

	Denetle	Denetleme
Çalış	(2 , 1)	(2 , 2)
Kaytar	(0 , −1)	(4 , −4)



1. İşçi **p** olasılıkla Çalış, müdür **q** olasılıkla Denetle oynar.
2. İşçi: E[Çalış] = 2q + 2(1−q) = **2**; E[Kaytar] = 0·q + 4(1−q) = **4 − 4q**
3. İşçi kayıtsız: 2 = 4 − 4q ⇒ **q* = 1/2**
4. Müdür: E[Denetle] = 1·p − 1·(1−p) = **2p − 1**; E[Denetleme] = 2p − 4(1−p) = **6p − 4**
5. Müdür kayıtsız: 2p − 1 = 6p − 4 ⇒ **p* = 3/4**
6. Denge: (p*, q*) = (3/4, 1/2). Beklenen kazanç: işçi 2, müdür 1/2.
7. Ceza sertleşirse (yakalanan kaytarıcı −2): q* = 1/3 olur, p* = ¾ **değişmez**!

KARIŞIK STRATEJİ

Ceza Değişince Kimin Olasılığı Değişir?

İşçi kayıtsız: 2 = c·q + 4(1 − q) ⇒ **q* = 2/(4 − c)** · Müdür kayıtsız: 2p − 1 = 6p − 4 ⇒ **p* = 3/4**

c = 0

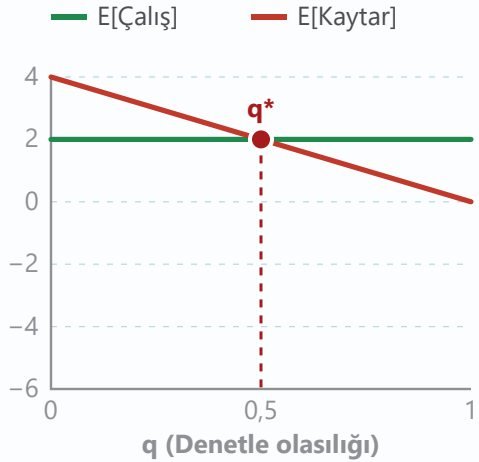
q* = 2/(4 − c) = **1/2 = 0,5** | p* = **3/4**, c'den bağımsız | işçinin denge kazancı = **2**

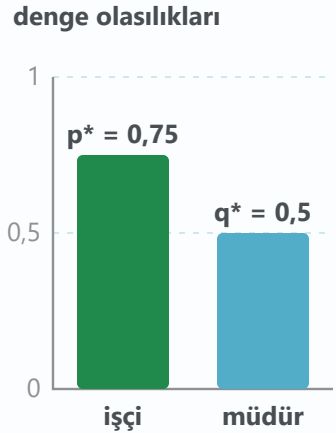
Müdür

işçi	Denetle	Denetleme
Çalış	(2, 1)	(2, 2)
Kaytar	(0, −1)	(4, −4)

hücre: (işçi, müdür)

c: yakalanan kaytarıcının kazancı





İşçinin kendi kazancı c değişse de işçinin karışımı p* = 3/4 olarak kalır; değişen, müdürün denetleme olasılığı q*'dır. Her oyuncunun denge olasılığını **rakibinin** kazançları belirler.

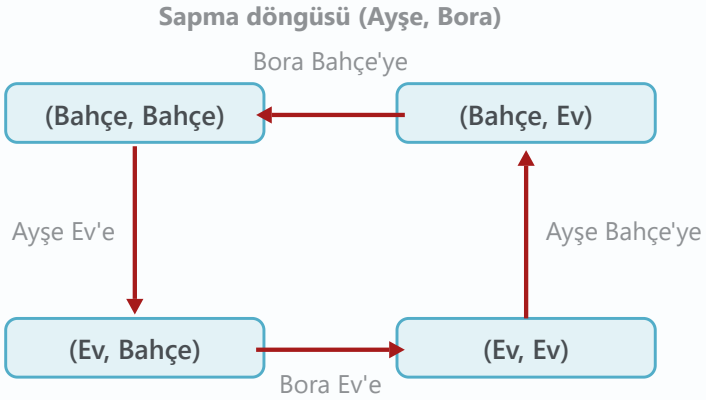
Adım Adım: Kayıtsızlık Yöntemiyle Karışık Denge

Ayşe (saklanır) \ Bora (arar)	Bahçe	Ev
Bahçe	(0 , 3)	(2 , 0)
Ev	(1 , 0)	(0 , 1)

Ayşe saklanır (satır), Bora arar (sütun); hücrede keyif puanı (Ayşe, Bora). Bora bahçede bulursa 3, evde bulursa 1 alır; Ayşe bahçede bulunmazsa 2, evde bulunmazsa 1.

Adım Adım: Kayıtsızlık Yöntemiyle Karışık Denge

Ayşe (saklanır) \ Bora (arar)	Bahçe	Ev
Bahçe	(0 , 3)	(2 , 0)
Ev	(1 , 0)	(0 , 1)



Altını çizince hiçbir hücrede iki çizgi yok: en iyi tepkiler bir **döngü** kurar (oklar). Saf Nash dengesi yok → karışık denge aranır.

Adım Adım: Kayıtsızlık Yöntemiyle Karışık Denge

Ayşe (saklanır) \ Bora (arar)	Bahçe	Ev
Bahçe	(0 , 3)	(2 , 0)
Ev	(1 , 0)	(0 , 1)

p = P(Ayşe Bahçe), q = P(Bora Bahçe)

Değişkenler: Ayşe Bahçe'yi **p**, Bora Bahçe'yi **q** olasılıkla seçer. Ayşe karıştırıyorsa iki seçeneği ona **eşit** kazanç vermeli; bu eşitlik q'yu verir.

Adım Adım: Kayıtsızlık Yöntemiyle Karışık Denge

Ayşe (saklanır) \ Bora (arar)	Bahçe	Ev
Bahçe	(0 , 3)	(2 , 0)
Ev	(1 , 0)	(0 , 1)

p = P(Ayşe Bahçe), q = P(Bora Bahçe)

Ayşe: E[Bahçe] = 0·q + 2·(1 − q) = 2 − 2q

Ayşe Bahçe'de: Bora q ile bahçeye bakar (0), 1 − q ile eve bakar (2). Beklenen kazanç **2 − 2q**.

Adım Adım: Kayıtsızlık Yöntemiyle Karışık Denge

Ayşe (saklanır) \ Bora (arar)	Bahçe	Ev
Bahçe	(0 , 3)	(2 , 0)
Ev	(1 , 0)	(0 , 1)

p = P(Ayşe Bahçe), q = P(Bora Bahçe)

Ayşe: E[Bahçe] = 0·q + 2·(1 − q) = 2 − 2q

Ayşe: E[Ev] = 1·q + 0·(1 − q) = q

Ayşe Ev'de: Bora bahçeye bakarsa (q) 1, eve bakarsa 0. Beklenen kazanç **q**.

Adım Adım: Kayıtsızlık Yöntemiyle Karışık Denge

Ayşe (saklanır) \ Bora (arar)	Bahçe	Ev
Bahçe	(0 , 3)	(2 , 0)
Ev	(1 , 0)	(0 , 1)

p = P(Ayşe Bahçe), q = P(Bora Bahçe)

Ayşe: E[Bahçe] = 0·q + 2·(1 − q) = 2 − 2q

Ayşe: E[Ev] = 1·q + 0·(1 − q) = q

2 − 2q = q ⇒ 2 = 3q ⇒ q* = 2/3

Eşitle ve çöz: 2 − 2q = q ⇒ **q* = 2/3**. Bora bahçeye 2/3 olasılıkla bakmalı ki Ayşe kayıtsız kalsın.

Adım Adım: Kayıtsızlık Yöntemiyle Karışık Denge

Ayşe (saklanır) \ Bora (arar)	Bahçe	Ev
Bahçe	(0 , 3)	(2 , 0)
Ev	(1 , 0)	(0 , 1)

p = P(Ayşe Bahçe), q = P(Bora Bahçe)

Ayşe: E[Bahçe] = 0·q + 2·(1 − q) = 2 − 2q

Ayşe: E[Ev] = 1·q + 0·(1 − q) = q

2 − 2q = q ⇒ 2 = 3q ⇒ q* = 2/3

Bora: E[Bahçe] = 3·p + 0·(1 − p) = 3p

Bora: E[Ev] = 0·p + 1·(1 − p) = 1 − p

Aynısı Bora için: Bora'nın iki seçeneği, Ayşe'nin p'sine göre: E[Bahçe] = **3p**, E[Ev] = **1 − p**.

Adım Adım: Kayıtsızlık Yöntemiyle Karışık Denge

Ayşe (saklanır) \ Bora (arar)	Bahçe	Ev
Bahçe	(0 , 3)	(2 , 0)
Ev	(1 , 0)	(0 , 1)

p = P(Ayşe Bahçe), q = P(Bora Bahçe)

Ayşe: E[Bahçe] = 0·q + 2·(1 − q) = 2 − 2q

Ayşe: E[Ev] = 1·q + 0·(1 − q) = q

2 − 2q = q ⇒ 2 = 3q ⇒ q* = 2/3

Bora: E[Bahçe] = 3·p + 0·(1 − p) = 3p

Bora: E[Ev] = 0·p + 1·(1 − p) = 1 − p

3p = 1 − p ⇒ 4p = 1 ⇒ p* = 1/4

3p = 1 − p ⇒ **p* = 1/4**. Dikkat: p*, Ayşe'nin değil **Bora'nın** sayılarından çıktı.

Adım Adım: Kayıtsızlık Yöntemiyle Karışık Denge

Ayşe (saklanır) \ Bora (arar)	Bahçe	Ev
Bahçe	(0 , 3)	(2 , 0)
Ev	(1 , 0)	(0 , 1)

p = P(Ayşe Bahçe), q = P(Bora Bahçe)

Ayşe: E[Bahçe] = 0·q + 2·(1 − q) = 2 − 2q

Ayşe: E[Ev] = 1·q + 0·(1 − q) = q

2 − 2q = q ⇒ 2 = 3q ⇒ q* = 2/3

Bora: E[Bahçe] = 3·p + 0·(1 − p) = 3p

Bora: E[Ev] = 0·p + 1·(1 − p) = 1 − p

3p = 1 − p ⇒ 4p = 1 ⇒ p* = 1/4

Kazanç: Ayşe 2/3, Bora 3/4

Kontrol: 0 < 1/4 < 1 ve 0 < 2/3 < 1 ✓. Kazançlar: Ayşe 2 − 2·(2/3) = **2/3**, Bora 3·(1/4) = **3/4**. Bahçeyi seven Ayşe orada yalnız %25 saklanır.

KARIŞIK STRATEJİ

Kod İzleme: `karisik_2x2(A, B)`

satır \ sütun oyuncusu	Opera	Futbol
Opera	(3 , 2)	(0 , 0)
Futbol	(0 , 0)	(2 , 3)

Cinsiyetler Savaşı: A satır, B sütun oyuncusunun kazançları (0 = Opera, 1 = Futbol).

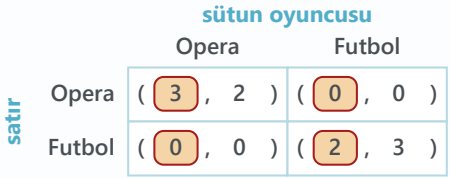
Adım 1 / 8

```
1  from fractions import Fraction as F
▶  def karisik_2x2(A, B):
3      da = A[0][0] - A[0][1] - A[1][0] + A[1][1]
4      if da == 0: return None      # denklem yok
5      q = F(A[1][1] - A[0][1], da)  # satır kayıtsız
6      db = B[0][0] - B[0][1] - B[1][0] + B[1][1]
7      if db == 0: return None
8      p = F(B[1][1] - B[1][0], db)  # sütun kayıtsız
9      if not (0 < p < 1 and 0 < q < 1):
10         return None               # iç denge yok
11     u1 = A[0][0] * q + A[0][1] * (1 - q)
12     return p, q, u1
```

DEĞİŞKENLER

—

Kod İzleme: `karisik_2x2(A, B)`



Satır oyuncusunun kazanç farklarından payda: da = 3 − 0 − 0 + 2 = **5**.

Adım 2 / 8

```
1  from fractions import Fraction as F
2  def karisik_2x2(A, B):
▶      da = A[0][0] - A[0][1] - A[1][0] + A[1][1]
4      if da == 0: return None      # denklem yok
5      q = F(A[1][1] - A[0][1], da)  # satır kayıtsız
6      db = B[0][0] - B[0][1] - B[1][0] + B[1][1]
7      if db == 0: return None
8      p = F(B[1][1] - B[1][0], db)  # sütun kayıtsız
9      if not (0 < p < 1 and 0 < q < 1):
10         return None               # iç denge yok
11     u1 = A[0][0] * q + A[0][1] * (1 - q)
12     return p, q, u1
```

DEĞİŞKENLER

`da = 5`

Kod İzleme: `karisik_2x2(A, B)`



da ≠ 0, denetim geçilir. u_1(Opera, q) = 3q, u_1(Futbol, q) = 2(1 − q) eşitlenir ⇒ q = (2 − 0)/5 = **2/5**: sütun oyuncusu Opera'yı 2/5 olasılıkla oynar.

Adım 3 / 8

```
1  from fractions import Fraction as F
2  def karisik_2x2(A, B):
3      da = A[0][0] - A[0][1] - A[1][0] + A[1][1]
▶      if da == 0: return None     # denklem yok
▶      q = F(A[1][1] - A[0][1], da)  # satır kayıtsız
6      db = B[0][0] - B[0][1] - B[1][0] + B[1][1]
7      if db == 0: return None
8      p = F(B[1][1] - B[1][0], db)  # sütun kayıtsız
9      if not (0 < p < 1 and 0 < q < 1):
10         return None               # iç denge yok
11     u1 = A[0][0] * q + A[0][1] * (1 - q)
12     return p, q, u1
```

DEĞİŞKENLER

`da = 5` `q = 2/5`

Kod İzleme: `karisik_2x2(A, B)`

		sütun oyuncusu: Opera	sütun oyuncusu: Futbol
satır	Opera	(3 , **2**)	(0 , **0**)
satır	Futbol	(0 , **0**)	(2 , **3**)
sütun olasılığı		2/5	3/5

Şimdi sütun oyuncusunun kazançları: db = 2 − 0 − 0 + 3 = **5**.

Adım 4 / 8

```
1  from fractions import Fraction as F
2  def karisik_2x2(A, B):
3      da = A[0][0] - A[0][1] - A[1][0] + A[1][1]
4      if da == 0: return None      # denklem yok
5      q = F(A[1][1] - A[0][1], da)  # satır kayıtsız
▶      db = B[0][0] - B[0][1] - B[1][0] + B[1][1]
7      if db == 0: return None
8      p = F(B[1][1] - B[1][0], db)  # sütun kayıtsız
9      if not (0 < p < 1 and 0 < q < 1):
10         return None               # iç denge yok
11     u1 = A[0][0] * q + A[0][1] * (1 - q)
12     return p, q, u1
```

DEĞİŞKENLER

`da = 5` `q = 2/5` `db = 5`

Kod İzleme: `karisik_2x2(A, B)`

		sütun oyuncusu: Opera	sütun oyuncusu: Futbol	satır olasılığı
satır	Opera	(3 , **2**)	(0 , **0**)	**3/5**
satır	Futbol	(0 , **0**)	(2 , **3**)	**2/5**
sütun olasılığı		2/5	3/5	

db ≠ 0. $u_2(p, \text{Opera}) = 2p$, $u_2(p, \text{Futbol}) = 3(1 - p)$ eşitlenir ⇒ p = **3/5**: satır oyuncusu Opera'yı 3/5 olasılıkla oynar.

Adım 5 / 8

```
1  from fractions import Fraction as F
2  def karisik_2x2(A, B):
3      da = A[0][0] - A[0][1] - A[1][0] + A[1][1]
4      if da == 0: return None      # denklem yok
5      q = F(A[1][1] - A[0][1], da)  # satır kayıtsız
6      db = B[0][0] - B[0][1] - B[1][0] + B[1][1]
▶      if db == 0: return None
▶      p = F(B[1][1] - B[1][0], db)  # sütun kayıtsız
9      if not (0 < p < 1 and 0 < q < 1):
10         return None               # iç denge yok
11     u1 = A[0][0] * q + A[0][1] * (1 - q)
12     return p, q, u1
```

DEĞİŞKENLER

`da = 5` `q = 2/5` `db = 5` `p = 3/5`

Kod İzleme: `karisik_2x2(A, B)`

		sütun oyuncusu: Opera	sütun oyuncusu: Futbol	satır olasılığı
satır	Opera	(3 , 2)	(0 , 0)	3/5
satır	Futbol	(0 , 0)	(2 , 3)	2/5
sütun olasılığı		2/5	3/5	

0 < p, q < 1 doğru: iki olasılık da gerçek bir karışım ⇒ iç karışık denge var, return None atlanır.


```
1  from fractions import Fraction as F
2  def karisik_2x2(A, B):
3      da = A[0][0] - A[0][1] - A[1][0] + A[1][1]
4      if da == 0: return None      # denklem yok
5      q = F(A[1][1] - A[0][1], da)  # satır kayıtsız
6      db = B[0][0] - B[0][1] - B[1][0] + B[1][1]
7      if db == 0: return None
8      p = F(B[1][1] - B[1][0], db)  # sütun kayıtsız
▶      if not (0 < p < 1 and 0 < q < 1):
10         return None               # iç denge yok
11     u1 = A[0][0] * q + A[0][1] * (1 - q)
12     return p, q, u1
```

DEĞİŞKENLER

`da = 5` `q = 2/5` `db = 5` `p = 3/5`

Kod İzleme: `karisik_2x2(A, B)`

		sütun oyuncusu: Opera	sütun oyuncusu: Futbol	satır olasılığı
satır	Opera	(**3** , 2)	(**0** , 0)	3/5
	Futbol	(0 , 0)	(2 , 3)	2/5
	sütun olasılığı	2/5	3/5	

Beklenen kazanç $u_1 = 3\cdot(2/5) + 0\cdot(3/5) = \mathbf{6/5}$; $u_2 = 2\cdot(3/5) = 6/5$. İki saf dengenin (2 ya da 3) ikisinden de kötü.

Adım 7 / 8

```
1  from fractions import Fraction as F
2  def karisik_2x2(A, B):
3      da = A[0][0] - A[0][1] - A[1][0] + A[1][1]
4      if da == 0: return None      # denklem yok
5      q = F(A[1][1] - A[0][1], da)  # satır kayıtsız
6      db = B[0][0] - B[0][1] - B[1][0] + B[1][1]
7      if db == 0: return None
8      p = F(B[1][1] - B[1][0], db)  # sütun kayıtsız
9      if not (0 < p < 1 and 0 < q < 1):
10         return None               # iç denge yok
▶      u1 = A[0][0] * q + A[0][1] * (1 - q)
12     return p, q, u1
```

DEĞİŞKENLER

`da = 5` `q = 2/5` `db = 5` `p = 3/5` `u1 = 6/5`

Kod İzleme: `karisik_2x2(A, B)`

		sütun oyuncusu: Opera	sütun oyuncusu: Futbol	satır olasılığı
satır	Opera	(3 , 2)	(0 , 0)	3/5
satır	Futbol	(0 , 0)	(2 , 3)	2/5
sütun olasılığı		2/5	3/5	

Sonuç (3/5, 2/5, 6/5). Bedava Para'da q = 0 ⇒ None. Mahkûmun İkilemi'nde da = 0 ⇒ ilk denetim None döndürür.

Adım 8 / 8

```
1  from fractions import Fraction as F
2  def karisik_2x2(A, B):
3      da = A[0][0] - A[0][1] - A[1][0] + A[1][1]
4      if da == 0: return None      # denklem yok
5      q = F(A[1][1] - A[0][1], da)  # satır kayıtsız
6      db = B[0][0] - B[0][1] - B[1][0] + B[1][1]
7      if db == 0: return None
8      p = F(B[1][1] - B[1][0], db)  # sütun kayıtsız
9      if not (0 < p < 1 and 0 < q < 1):
10         return None               # iç denge yok
11     u1 = A[0][0] * q + A[0][1] * (1 - q)
▶      return p, q, u1
```

DEĞİŞKENLER

`da = 5` `q = 2/5` `db = 5` `p = 3/5` `u1 = 6/5`

Altı Klasik Oyunun Karışık Dengeleri

Oyun	Saf NE	Karışık NE (p*, q*)	(u_1, u_2)	Karışık dengede ne olur?
Yazı-Tura Eşleştirme	0	1/2, 1/2	(0, 0)	Dört sonuç %25'er; tek denge budur, kimse sömürülemez
Tavuk (kavşak)	2	9/10, 9/10	(−1/10, −1/10)	Çarpışma $(1/10)^2$ = %1; çarpışma −k olsaydı kırma (k − 1)/k
Cinsiyetler Savaşı	2	3/5, 2/5	(6/5, 6/5)	Buluşamama 0,36 + 0,16 = %52
Geyik Avı (geyik 4, tavşan 3)	2	3/4, 3/4	(3, 3)	Ayrı düşme 3/8 = %37,5; birlikte geyik 9/16 ≈ %56
Saf Koordinasyon	2	1/3, 1/3	(10/3, 10/3)	Farklı karar 4/9 ≈ %44; parti ne kadar cazipse o kadar az seçilir
Bedava Para	2	yok (yalnız uç: 0, 0)	—	q > 0 ise Yazı kesin daha iyi; kayıtsızlık yalnız q = 0'da

p*, q*: ilk stratejinin (Yazı, Kır, Opera, Geyik, Parti, Yazı) oynanma olasılığı. **Tavuk:** (Kır, Kır) = (0, 0), (Kır, Düz) = (−1, 1), (Düz, Düz) = (−10, −10). **Saf Koordinasyon:** (Parti, Parti) = (10, 10), (Ev, Ev) = (5, 5). **Bedava Para:** yalnız (Yazı, Yazı) = (1, 1), öteki hücreler 0.

Sık Yapılan Hatalar: Karışık Denge

p'yi kendi sayılarından bulmak

Satır oyuncusunun p'si *sütun oyuncusunu* kayıtsız bırakır; bu yüzden sütun oyuncusunun kazançlarından hesaplanır.

(1 − q)'yu unutmak

E = 0·q + 2·(1 − q) = 2 − 2q'dur; "2q" değil. Her terim kendi olasılığıyla çarpılır.

Sonucu sınamamak

p ya da q, 0 ile 1 arasında çıkmazsa iç karışık denge yoktur (genelde bir strateji baskılanmıştır).

Saf dengeleri atlamak

Karışık hesap saf dengelerin yerini tutmaz; önce altını çiz, sonra karışık dengeyi ara.

"Kayıtsızsa her p olur"

Oyuncu kendi seçenekleri arasında kayıtsızdır, ama dengede rakibini kayıtsız bırakan tek p'yi oynar.

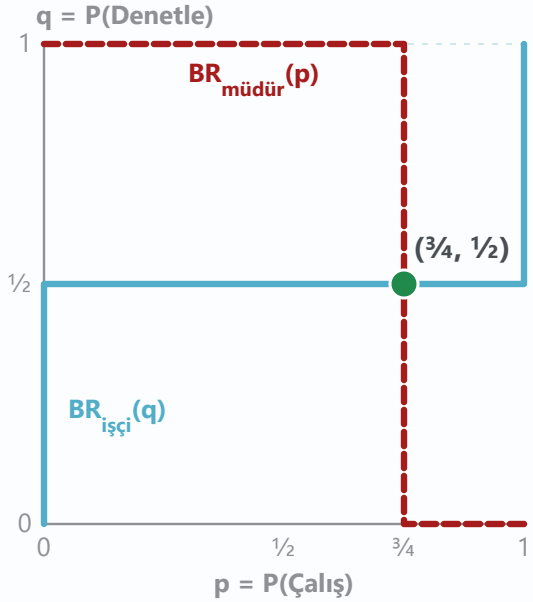
Eşitliği yanlış kurmak

Eşitlenen, *aynı oyuncunun* iki seçeneğinin beklenen kazancıdır; iki oyuncunun kazançları eşitlenmez.

Nash'in Varlık Teoremi (1950)

Teorem. Sonlu sayıda oyuncu ve sonlu strateji kümesi olan **her** oyunun en az bir (karışık strateji) Nash dengesi vardır.

- İspat fikri: "herkes herkese en iyi tepki veriyor" koşulunu sağlayan bir karışık profil her zaman vardır; bunu sabit nokta teoremleri (**Kakutani**, Brouwer) garanti eder. Bu profil Nash dengesidir.
- Garanti **karışık** stratejiler içindir; saf denge olmayabilir (Yazı-Tura).
- Sonsuz strateji kümelerinde denge olmayabilir (ör. "en büyük sayıyı söyleyen kazanır").



Denetim oyunu: iki en iyi tepki eğrisinin kesişimi = denge

Adım Adım: Kanal Paylaşımında Üç Denge

Cihaz 1 \ Cihaz 2	Gönder	Bekle
Gönder	(−2 , −2)	(3 , 1)
Bekle	(1 , 3)	(0 , 0)

İki cihaz aynı kanala paket göndermek ister. İkisi gönderirse çarpışır (−2'şer); biri gönderirse o 3, bekleyen 1 alır; ikisi de beklerse kanal boşa gider (0). Tavuk oyunu yapısı.

Adım Adım: Kanal Paylaşımında Üç Denge

Cihaz 1 \ Cihaz 2	Gönder	Bekle
Gönder	(−2, −2)	(3, 1)
Bekle	(1, 3)	(0, 0)

Cihaz 1 (ilk sayılar): karşısı Gönder derse $-2 < 1$ → **Bekle**; karşısı Bekle derse $3 > 0$ → **Gönder**. Karşısının tersini yapmak ister.

Adım Adım: Kanal Paylaşımında Üç Denge

Cihaz 1 \ Cihaz 2	Gönder	Bekle
Gönder	(−2, −2)	(3, 1)
Bekle	(1, 3)	(0, 0)

Cihaz 2 için aynısı (ikinci sayılar). İki çizgili hücreler: **(Gönder, Bekle) = (3, 1)** ve **(Bekle, Gönder) = (1, 3)**: iki saf denge.

Adım Adım: Kanal Paylaşımında Üç Denge

Cihaz 1 \ Cihaz 2	Gönder	Bekle
Gönder	(−2, −2)	(3, 1)
Bekle	(1, 3)	(0, 0)

q = P(Cihaz 2 Gönder)

u_1(Gönder, q) = −2·q + 3·(1 − q) = 3 − 5q

Üçüncü denge karışık: q = P(Cihaz 2 Gönder). Cihaz 1'in Gönder'den beklentisi −2q + 3(1 − q) = **3 − 5q**.

Adım Adım: Kanal Paylaşımında Üç Denge

Cihaz 1 \ Cihaz 2	Gönder	Bekle
Gönder	(−2, −2)	(3, 1)
Bekle	(1, 3)	(0, 0)

q = P(Cihaz 2 Gönder)

$u_1(\text{Gönder}, q) = -2 \cdot q + 3 \cdot (1 - q) = 3 - 5q$

$u_1(\text{Bekle}, q) = 1 \cdot q + 0 \cdot (1 - q) = q$

Bekle'den beklentisi: karşısı gönderirse 1, beklerse 0 → **q**.

Adım Adım: Kanal Paylaşımında Üç Denge

Cihaz 1 \ Cihaz 2	Gönder	Bekle
Gönder	(−2, −2)	(3, 1)
Bekle	(1, 3)	(0, 0)

q = P(Cihaz 2 Gönder)

$u_1(\text{Gönder}, q) = -2 \cdot q + 3 \cdot (1 - q) = 3 - 5q$

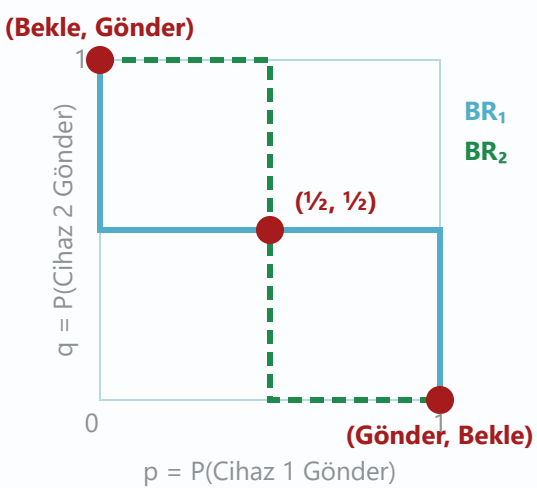
$u_1(\text{Bekle}, q) = 1 \cdot q + 0 \cdot (1 - q) = q$

$3 - 5q = q \Rightarrow q^* = 1/2$; simetri: $p^* = 1/2$

Kayıtsızlık: $3 - 5q = q \Rightarrow$ **$q^* = 1/2$**. Oyun simetrik olduğu için Cihaz 2'nin kayıtsızlığı da **$p^* = 1/2$** verir.

Adım Adım: Kanal Paylaşımında Üç Denge

Cihaz 1 \ Cihaz 2	Gönder	Bekle
Gönder	(−2, −2)	(3, 1)
Bekle	(1, 3)	(0, 0)



En iyi tepki eğrileri: BR_1 (düz) ve BR_2 (kesikli) **üç noktada** kesişir: iki köşe saf dengeler, orta nokta karışık denge.

Adım Adım: Kanal Paylaşımında Üç Denge

Cihaz 1 \ Cihaz 2	Gönder	Bekle
Gönder	(−2 , −2)	(3 , 1)
Bekle	(1 , 3)	(0 , 0)

Karışık dengede (½, ½) sonuçlar:

çarpışma (G, G): ¼ → −2

tek gönderen (başarı): ½

boş kanal (B, B): ¼ → 0

beklenen kazanç: ½ (saf dengede 3 ya da 1)

Karışık denge adil ama verimsiz: zamanın ¼'ünde çarpışma, ¼'ünde boş kanal; her cihaz ortalama ½ alır. Ağ protokollerindeki rastgele bekleme (backoff) bu rastgeleleştirmenin uygulamadaki karşılığıdır.

Kaç Denge Olur? Tek Sayı Kuralı

Oyun	Saf denge	Karışık denge	Toplam
Mahkûmun İkilemi	1	0	1
Saklambaç, Denetim, Yazı-Tura	0	1	1
Kanal paylaşımı, Tavuk, Cinsiyetler Savaşı, Geyik Avı, Saf Koordinasyon	2	1	3
Dil seçimi (3×3)	2	1 (Ayşe Python'u 3/4, Bora Python'u 1/5 olasılıkla seçer)	3
Zayıf Baskılamanın Tuzağı (eşitlikli)	3	sonsuz	—
Bedava Para (eşitlikli)	2	0	2 (çift)

Tek sayılılık (Wilson, 1971)
Kazançlarında özel eşitlikler olmayan (dejenere olmayan) oyunlarda denge sayısı sonludur ve *tek sayıdır* (1, 3, 5, …).

Pratik kural
İki saf denge buldunuz mu? Büyük olasılıkla bir karışık denge daha vardır; kayıtsızlıkla onu da arayın. Eşitlikli oyunlarda kural bozulabilir.

Çoklu Denge: Geyik Avı

Avcı 1 \ Avcı 2	Geyik	Tavşan
Geyik	(4 , 4)	(0 , 3)
Tavşan	(3 , 0)	(3 , 3)

Geyik Avı: geyik ancak birlikte avlanır (4'er); tavşan, ortak ne yaparsa yapsın tek başına 3 garanti eder.

Adım 1 / 5

Oyuncu adları: satırda Avcı 1, sütunda Avcı 2; hücrede (Avcı 1, Avcı 2).

Çoklu Denge: Geyik Avı

Avcı 1 \ Avcı 2	Geyik	Tavşan
Geyik	(4, 4)	(0, 3)
Tavşan	(3, 0)	(3, 3)

En iyi tepkiler iki hücrede çakışır: **(Geyik, Geyik)** ve **(Tavşan, Tavşan)**. Hangisi oynanacak?

Oyuncu adları: satırda Avcı 1, sütunda Avcı 2; hücrede (Avcı 1, Avcı 2).

Çoklu Denge: Geyik Avı

Avcı 1 \ Avcı 2	Geyik	Tavşan
Geyik	(4, 4)	(0, 3)
Tavşan	(3, 0)	(3, 3)

Üçüncü denge karışık: karşı taraf Geyik'i q ile seçiyorsa 4q = 3 ⇒ **q = ¾**. Geyik ancak ortağın ≥ %75 olasılıkla geleceğine inanılırsa en iyi tepkidir.

Oyuncu adları: satırda Avcı 1, sütunda Avcı 2; hücrede (Avcı 1, Avcı 2).

Çoklu Denge: Geyik Avı

Avcı 1 \ Avcı 2	Geyik	Tavşan
Geyik	**(4, 4)**	(0, 3)
Tavşan	(3, 0)	**(3, 3)**

Pareto baskınlık: (Geyik, Geyik) iki avcıya da daha çok verir (4 > 3) → verimlilik ölçütü onu seçer.

Oyuncu adları: satırda Avcı 1, sütunda Avcı 2; hücrede (Avcı 1, Avcı 2).

Çoklu Denge: Geyik Avı

Avcı 1 \ Avcı 2	Geyik	Tavşan
Geyik	(4, 4)	(0, 3)
Tavşan	(3, 0)	(3, 3)

Risk baskınlık (Harsanyi–Selten): sapma kayıpları çarpımı (G,G): (4−3)·(4−3) = 1; (T,T): (3−0)·(3−0) = 9 → **(Tavşan, Tavşan)** daha "güvenli".

Oyuncu adları: satırda Avcı 1, sütunda Avcı 2; hücrede (Avcı 1, Avcı 2).

Denge Seçimi: Hangisi Oynanır?

Pareto baskınlık

Herkesi daha iyi yapan dengeyi seç. Geyik Avı'nda (G, G). Güven gerektirir.

Risk baskınlık

Karşı tarafın ne yapacağından emin değilsen kayıp riski düşük olanı seç. Geyik Avı'nda (T, T).

Odak noktası (Schelling, 1960)

Kültür, alışkanlık ya da belirginlik bir dengeyi öne çıkarır: "İstanbul'da buluşacaksak saat 12'de Taksim".

Korelasyonlu denge (correlated equilibrium)

Aumann (1974): ortak bir sinyal (trafik ışığı) oyuncuları dengelerden birine yönlendirir; kavşaktaki Tavuk oyununu çözer.

Öğrenme ve evrim

Tekrarlanan oyunda oyuncular deneyerek öğrenir; hangi dengeye varılacağı başlangıçtaki davranışa bağlıdır.

Adalet sorunu

Cinsiyetler Savaşı'nda saf dengelerin biri bir oyuncuyu, diğeri öbürünü kayırır; karışık denge ise ikisine de yalnızca 6/5 verir.

Varsayımı Değiştir: Ya Bir Oyuncu Hata Yaparsa?

Mahkûmun İkilemi (eksi hapis yılı)

	Sessiz	**İhanet**
Sessiz	(−1, −1)	(−3, 0)
İhanet	(0, −3)	(−2, −2)

Geyik Avı

	Geyik	**Tavşan**
Geyik	(4, 4)	(0, 3)
Tavşan	(3, 0)	(3, 3)

Rakip ε olasılıkla niyet etmediği hamleyi yaparsa **en iyi tepki değişir mi?**

Varsayımı Değiştir: Ya Bir Oyuncu Hata Yaparsa?

Mahkûmun İkilemi (eksi hapis yılı)

	Sessiz	**İhanet**
Sessiz	(−1, −1)	(−3, 0)
İhanet	(0, −3)	(−2, −2)

Geyik Avı

	Geyik	**Tavşan**
Geyik	(4, 4)	(0, 3)
Tavşan	(3, 0)	(3, 3)

Rakip ε olasılıkla niyet etmediği hamleyi yaparsa **en iyi tepki değişir mi?**

Mahkûmun İkilemi: değişmez

Rakip p olasılıkla Sessiz kalsın.
Sessiz: $-1 \cdot p - 3 \cdot (1 - p) = 2p - 3$
İhanet: $0 \cdot p - 2 \cdot (1 - p) = 2p - 2$
İhanet her p için 1 yıl daha iyi.

Varsayımı Değiştir: Ya Bir Oyuncu Hata Yaparsa?

Mahkûmun İkilemi (eksi hapis yılı)

	Sessiz	**İhanet**
Sessiz	(−1, −1)	(−3, 0)
İhanet	(0, −3)	(−2, −2)

Geyik Avı

	Geyik	**Tavşan**
Geyik	(4, 4)	(0, 3)
Tavşan	(3, 0)	(3, 3)

Rakip ε olasılıkla niyet etmediği hamleyi yaparsa **en iyi tepki değişir mi?**

Mahkûmun İkilemi: değişmez

Rakip p olasılıkla Sessiz kalsın.
Sessiz: $-1 \cdot p - 3 \cdot (1 - p) = 2p - 3$
İhanet: $0 \cdot p - 2 \cdot (1 - p) = 2p - 2$
İhanet her p için 1 yıl daha iyi.

Geyik Avı: hata payına bağlı

Rakip q olasılıkla Tavşan'a kaçsın.
Geyik: $4 \cdot (1 - q)$ · Tavşan: her durumda 3
Geyik ancak $4(1 - q) \geq 3$, yani **$q \leq 1/4$** iken en iyi tepki. q daha büyükse güvenli seçim risk baskın Tavşan'dır.

Baskın strateji dengesi hatalara dayanıklıdır, koordinasyon dengeleri değildir.

Korelasyonlu Denge (Aumann, 1974): Kavşakta Trafik Işığı

Sürücü 1 \ Sürücü 2	Kır	Düz git
Kır	(0, 0)	(−1, 1)
Düz git	(1, −1)	(−10, −10)

Saf dengeler: (Kır, Düz), (Düz, Kır)

Karışık denge: kırma 9/10

çarpışma 1/100, kazanç −1/10

Işıksız kavşağa iki sürücü aynı anda gelir: yol verir (**Kır**) ya da geçer (**Düz git**). Bu bir Tavuk oyunudur: iki saf denge birer sürücüyü kayırır; koordinasyon yoksa simetrik karışık dengede her sürücü 9/10 olasılıkla kırar, %1 çarpışma olur ve beklenen kazanç −1/10'dur.

Korelasyonlu Denge (Aumann, 1974): Kavşakta Trafik Işığı

Sürücü 1 \ Sürücü 2	Kır	Düz git
Kır	(0, 0) μ = 1/3	(−1, 1) μ = 1/3
Düz git	(1, −1) μ = 1/3	(−10, −10) μ = 0

Işık 3 hücreden birini 1/3 ile seçer, her sürücüye YALNIZ kendi önerisini söyler.

Bir **aracı** (trafik ışığı) ortak olasılık dağılımı μ'den bir hücre çeker ve her sürücüye yalnız kendi önerisini fısıldar.

Korelasyonlu Denge (Aumann, 1974): Kavşakta Trafik Işığı

Sürücü 1 \ Sürücü 2	Kır	Düz git
Kır	(0, 0) **μ = 1/3**	(−1, 1) **μ = 1/3**
Düz git	(1, −1) **μ = 1/3**	(−10, −10) **μ = 0**

S1'e "Kır" dendi ⇒ S2: Kır ½, Düz ½

Uy (Kır): 0·½ + (−1)·½ = −½

Sap (Düz): 1·½ + (−10)·½ = −4,5

−½ > −4,5 ⇒ uymak iyi

Öneri "Kır" ise Bayes kuralıyla rakip ½–½ Kır/Düz; uymak (−½) sapmaktan (−4,5) iyidir.

Korelasyonlu Denge (Aumann, 1974): Kavşakta Trafik Işığı

Sürücü 1 \ Sürücü 2	Kır	Düz git
Kır	(0, 0) **μ = 1/3**	(−1, 1) **μ = 1/3**
Düz git	(1, −1) **μ = 1/3**	(−10, −10) **μ = 0**

S1'e "Düz git" dendi ⇒ S2 kesin Kır

Uy (Düz): 1 Sap (Kır): 0

1 > 0 ⇒ uymak iyi

Sürücü 2 için de simetrik.

Öneri "Düz git" ise rakibe kesin "Kır" denmiştir; uymak (1) sapmaktan (0) iyidir. Hiçbir öneriden sapmak kârlı değil ⇒ **korelasyonlu denge**.

Korelasyonlu Denge (Aumann, 1974): Kavşakta Trafik Işığı

Sürücü 1 \ Sürücü 2	Kır	Düz git
Kır	(0, 0) μ = 1/3	(−1, 1) μ = 1/3
Düz git	(1, −1) μ = 1/3	(−10, −10) μ = 0



Beklenen kazanç 0'ar (karışık dengedeki −1/10'dan iyi), çarpışma hiç yok. Her Nash dengesi bir korelasyonlu dengedir; tersi doğru değil. Korelasyonlu denge doğrusal programla polinom zamanda bulunur; ışık sürelerini yoğunluğa göre ayarlamak ise ayrı bir tasarım kararıdır.

Sıfır Toplamlı Oyun: Maksimin ve Minimaks

Satır (maks.) \ Sütun (min.)	X	Y	Z
A	4	3	5
B	2	1	6
C	5	2	0

Sıfır toplamlı: hücrede yalnızca satır oyuncusunun kazancı var; sütun oyuncusu aynı miktarı **kaybeder** ($u_2 = -u_1$).

Sıfır Toplamlı Oyun: Maksimin ve Minimaks

Satır (maks.) \ Sütun (min.)	X	Y	Z	satır min.
A	4	**3**	5	3
B	2	**1**	6	1
C	5	2	**0**	0

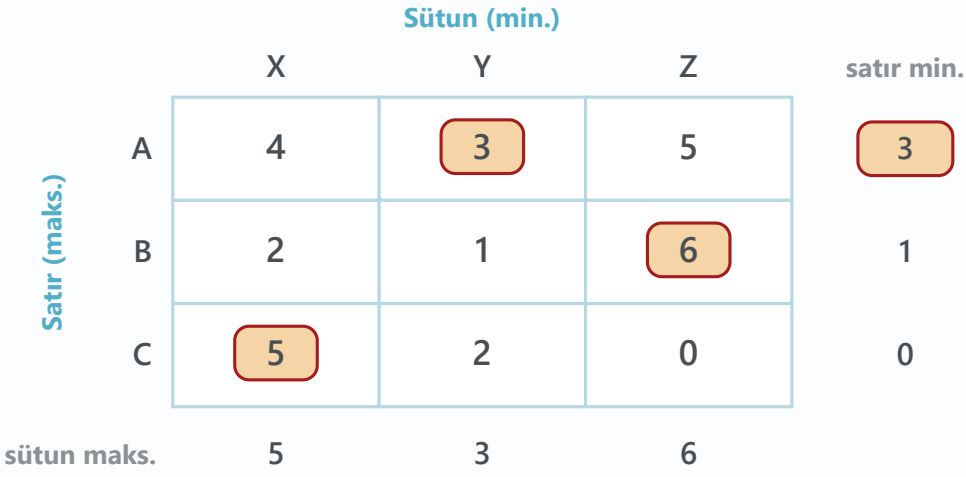
Satır oyuncusu kötümser düşünür: her satırda rakibin ona bırakacağı **en kötü** değer: 3, 1, 0.

Sıfır Toplamlı Oyun: Maksimin ve Minimaks

Satır (maks.) \ Sütun (min.)	X	Y	Z	satır min.
A	4	3	5	3
B	2	1	6	1
C	5	2	0	0

Maksimin = max(3, 1, 0) = **3** (satır A). Satır oyuncusu en az 3'ü garanti eder.

Sıfır Toplamlı Oyun: Maksimin ve Minimaks



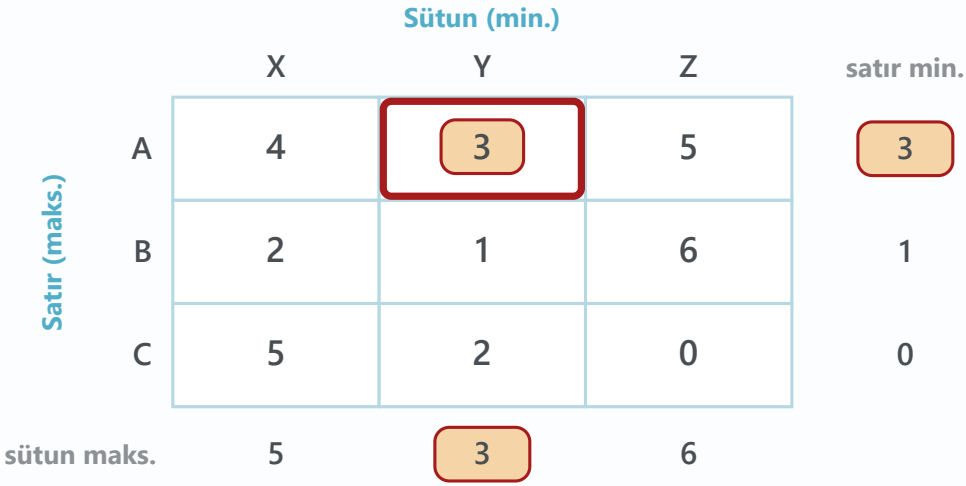
Sütun oyuncusu da kötümser: her sütunda satır oyuncusunun alabileceği **en yüksek** değer: 5, 3, 6.

Sıfır Toplamlı Oyun: Maksimin ve Minimaks



Minimaks = min(5, 3, 6) = **3** (sütun Y). Sütun oyuncusu en fazla 3 kaybetmeyi garanti eder.

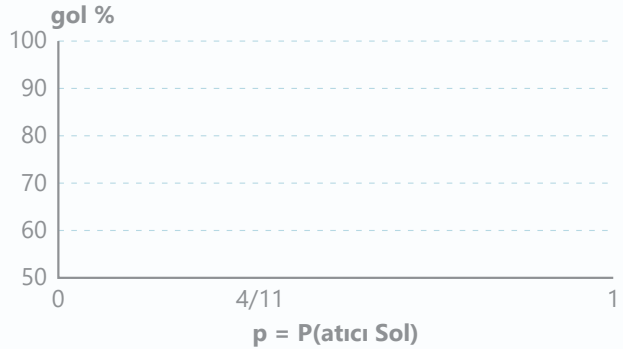
Sıfır Toplamlı Oyun: Maksimin ve Minimaks



Maksimin = minimaks = 3 ⇒ **eyer noktası** (A, Y): satırının en küçüğü, sütununun en büyüğü. Saf Nash dengesidir; **oyunun değeri v = 3**.

Eyer Noktası Yoksa: Penaltı Atışı

Atıcı \ Kaleci	Sol	Sağ	min
Sol	60	95	60
Sağ	90	70	70
maks	90	95	

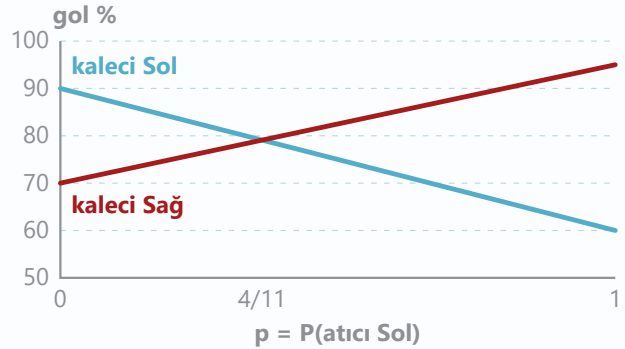


1. Satır min. 60, 70 → maksimin 70; sütun maks. 90, 95 → minimaks 90. 70 < 90: **eyer noktası yok**.

Örnek değerler (gerçek veride benzer yapı: Palacios-Huerta, 2003). Sol/Sağ atıcıya göredir.

Eyer Noktası Yoksa: Penaltı Atışı

Atıcı \ Kaleci	Sol	Sağ
Sol	60	95
Sağ	90	70

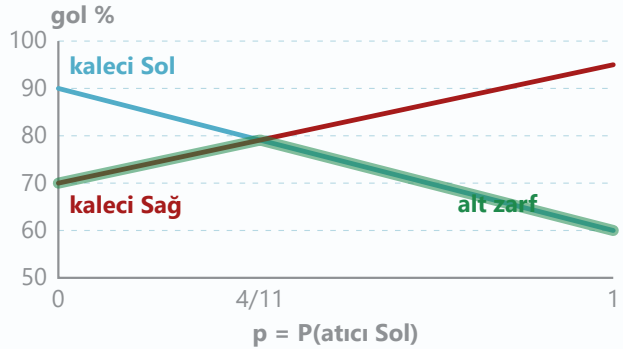


1. Satır min. 60, 70 → maksimin 70; sütun maks. 90, 95 → minimaks 90. 70 < 90: **eyer noktası yok**.
2. Atıcı p ile Sol: kaleci Sol ⇒ 60p + 90(1−p) = **90 − 30p**; kaleci Sağ ⇒ 95p + 70(1−p) = **70 + 25p**.

Örnek değerler (gerçek veride benzer yapı: Palacios-Huerta, 2003). Sol/Sağ atıcıya göredir.

Eyer Noktası Yoksa: Penaltı Atışı

Atıcı \ Kaleci	Sol	Sağ
Sol	60	95
Sağ	90	70

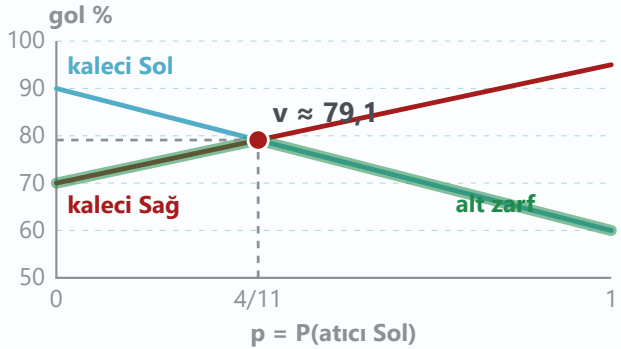


1. Satır min. 60, 70 → maksimin 70; sütun maks. 90, 95 → minimaks 90. 70 < 90: **eyer noktası yok**.
2. Atıcı p ile Sol: kaleci Sol ⇒ 60p + 90(1−p) = **90 − 30p**; kaleci Sağ ⇒ 95p + 70(1−p) = **70 + 25p**.
3. Kaleci gol yüzdesini en aza indirir: atıcının garantisi iki doğrunun **alt zarfı**.

Örnek değerler (gerçek veride benzer yapı: Palacios-Huerta, 2003). Sol/Sağ atıcıya göredir.

Eyer Noktası Yoksa: Penaltı Atışı

Atıcı \ Kaleci	Sol	Sağ
Sol	60	95
Sağ	90	70

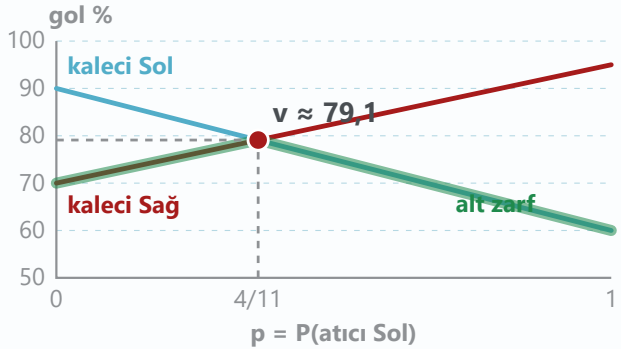


1. Satır min. 60, 70 → maksimin 70; sütun maks. 90, 95 → minimaks 90. 70 < 90: **eyer noktası yok**.
2. Atıcı p ile Sol: kaleci Sol ⇒ 60p + 90(1−p) = **90 − 30p**; kaleci Sağ ⇒ 95p + 70(1−p) = **70 + 25p**.
3. Kaleci gol yüzdesini en aza indirir: atıcının garantisi iki doğrunun **alt zarfı**.
4. Alt zarfın tepesi: 90 − 30p = 70 + 25p ⇒ **p* = 4/11 ≈ 0,36**, **v = 870/11 ≈ %79,1**.

Örnek değerler (gerçek veride benzer yapı: Palacios-Huerta, 2003). Sol/Sağ atıcıya göredir.

Eyer Noktası Yoksa: Penaltı Atışı

Atıcı \ Kaleci	Sol	Sağ
Sol	60	95
Sağ	90	70

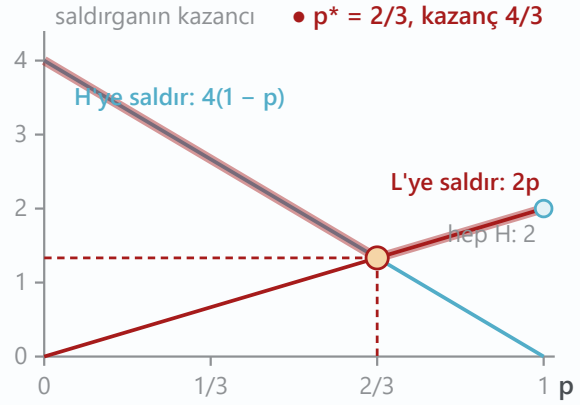


1. Satır min. 60, 70 → maksimin 70; sütun maks. 90, 95 → minimaks 90. 70 < 90: **eyer noktası yok**.
2. Atıcı p ile Sol: kaleci Sol ⇒ 60p + 90(1−p) = **90 − 30p**; kaleci Sağ ⇒ 95p + 70(1−p) = **70 + 25p**.
3. Kaleci gol yüzdesini en aza indirir: atıcının garantisi iki doğrunun **alt zarfı**.
4. Alt zarfın tepesi: 90 − 30p = 70 + 25p ⇒ **p* = 4/11 ≈ 0,36**, **v = 870/11 ≈ %79,1**.
5. Kaleci q ile Sol: 60q + 95(1−q) = 90q + 70(1−q) ⇒ **q* = 5/11**; kalecinin garantisi de %79,1 ⇒ maks-min = min-maks.

Örnek değerler (gerçek veride benzer yapı: Palacios-Huerta, 2003). Sol/Sağ atıcıya göredir.

Güvenlik Oyunları (Security Games): Rastgele Devriye

- Sınırlı kaynakla çok sayıda hedef korunur: havalimanı, liman, orman devriyesi. Savunucu n hedeften k'sini korur: C(n, k) saf strateji; n = 20, k = 5 için **15.504**.
- Saldırgan devriye düzenini **gözlemleyebilir**: sabit düzen atlatılır, bu yüzden savunucu rastgeleleştirir. Los Angeles Havalimanı'nda ARMOR sistemi (2007) devriyeleri bu modelle planladı (Tambe, 2011).



Tek bekçi, iki hedef

Havalimanı H değeri 4, liman L değeri 2. Bekçi H'yi p, L'yi 1 − p olasılıkla korur; korunmayan hedefe saldıran değerini kazanır, korunana saldıran 0.

Eşitle: 4(1 − p) = 2p ⇒ **p = 2/3**; saldırganın kazancı **4/3**. Bekçi hep H'yi korusa saldırgan L'den 2 alırdı.

von Neumann Minimaks Teoremi (1928)

Her sonlu iki kişilik sıfır toplamlı oyunda $\max_x \min_y x^T A y = \min_y \max_x x^T A y = v$

Denge = minimaks stratejileri

(x*, y*) Nash dengesi ⇔ x* maksimin, y* minimaks stratejisi. Tüm dengelerde değer aynı: v.

Değiştirilebilirlik

İki denge (x, y) ve (x′, y′) ise (x, y′) de dengedir: koordinasyon sorunu yok.

Doğrusal programlama

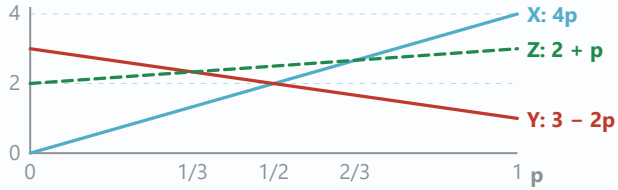
Satır oyuncusu, her sütuna karşı en az v alacağı x karışımını ve en büyük v'yi arar; bu bir doğrusal programdır ve bilgisayar hızlı çözer.

Öğrenme bağlantısı

Oyuncular geçmişte iyi giden stratejilerin ağırlığını artırarak öğrenirse ortalama stratejileri minimaksa yaklaşır.

Nash teoremi, minimaks teoreminin sıfır toplamlı olmayan n kişilik oyunlara genellemesi olarak görülebilir; ama genel oyunda değer tek değildir ve LP ile çözülemez.

Kod İzleme: `maksimin(A)` — 2 × n Sıfır Toplamlı



aday p

g(p)

A = [[4, 1, 3], [0, 3, 2]]: satır oyuncusu p ile 1. satırı oynarsa her sütun bir doğru verir. Saf maksimin 1, minimaks 3: eyer noktası yok.

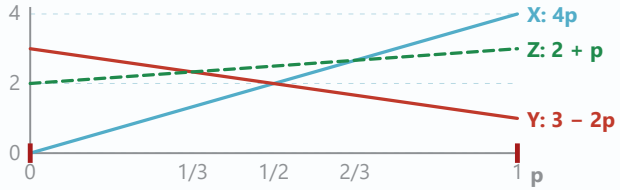
Adım 1 / 13

```
1  from fractions import Fraction as F
▶ def maksimin(A):                  # A: 2 x n
3      n, aday = len(A[0]), [F(0), F(1)]
4      for j in range(n):
5          for k in range(j + 1, n):
6              d = A[0][j] - A[1][j] - A[0][k] + A[1][k]
7              if d != 0:
8                  aday.append(F(A[1][k] - A[1][j], d))
9      g = lambda p: min(p*a+(1-p)*b for a, b in zip(*A))
10     p = max((c for c in aday if 0 <= c <= 1), key=g)
11     return p, g(p)
```

DEĞİŞKENLER

—

Kod İzleme: `maksimin(A)` — 2 × n Sıfır Toplamlı



aday p	0	1
g(p)	·	·

n = 3 sütun. Adaylar önce uç noktalar: p = 0 ve p = 1.

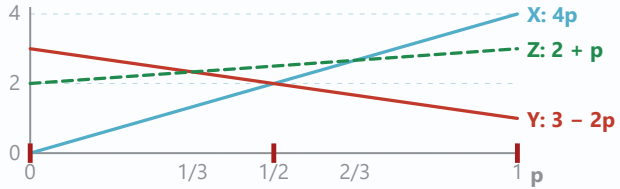
Adım 2 / 13

```
1  from fractions import Fraction as F
2  def maksimin(A):                    # A: 2 x n
▶      n, aday = len(A[0]), [F(0), F(1)]
4      for j in range(n):
5          for k in range(j + 1, n):
6              d = A[0][j] - A[1][j] - A[0][k] + A[1][k]
7              if d != 0:
8                  aday.append(F(A[1][k] - A[1][j], d))
9      g = lambda p: min(p*a+(1-p)*b for a, b in zip(*A))
10     p = max((c for c in aday if 0 <= c <= 1), key=g)
11     return p, g(p)
```

DEĞİŞKENLER

`n = 3` `aday = [0, 1]`

Kod İzleme: `maksimin(A)` — 2 × n Sıfır Toplamlı



aday p	0	1	1/2
g(p)	·	·	·

Sütun X ile Y: d = 6 ≠ 0 ⇒ doğrular p = **1/2**'de kesişir, aday listesine eklenir.

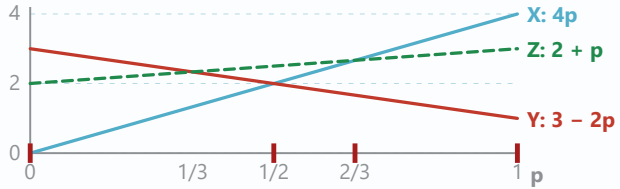
Adım 3 / 13

```
1  from fractions import Fraction as F
2  def maksimin(A):                    # A: 2 x n
3      n, aday = len(A[0]), [F(0), F(1)]
4      for j in range(n):
5          for k in range(j + 1, n):
▶              d = A[0][j] - A[1][j] - A[0][k] + A[1][k]
▶              if d != 0:
▶                  aday.append(F(A[1][k] - A[1][j], d))
9      g = lambda p: min(p*a+(1-p)*b for a, b in zip(*A))
10     p = max((c for c in aday if 0 <= c <= 1), key=g)
11     return p, g(p)
```

DEĞİŞKENLER

`n = 3` `aday = [0, 1, 1/2]` `j = 0` `k = 1` `d = 6`

Kod İzleme: `maksimin(A)` — 2 × n Sıfır Toplamlı



aday p	0	1	1/2	2/3
g(p)	·	·	·	·

Sütun X ile Z: d = 3 ≠ 0 ⇒ doğrular p = **2/3**'te kesişir, aday listesine eklenir.

Adım 4 / 13

```
1  from fractions import Fraction as F
2  def maksimin(A):                    # A: 2 x n
3      n, aday = len(A[0]), [F(0), F(1)]
4      for j in range(n):
5          for k in range(j + 1, n):
▶              d = A[0][j] - A[1][j] - A[0][k] + A[1][k]
▶              if d != 0:
▶                  aday.append(F(A[1][k] - A[1][j], d))
9      g = lambda p: min(p*a+(1-p)*b for a, b in zip(*A))
10     p = max((c for c in aday if 0 <= c <= 1), key=g)
11     return p, g(p)
```

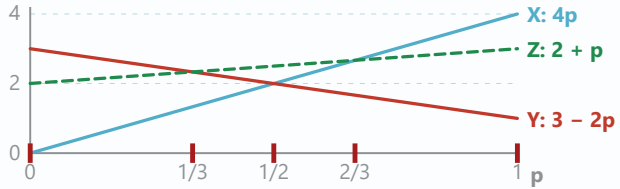
DEĞİŞKENLER

`n = 3`

`aday = [0, 1, 1/2, 2/3]`

`j = 0` `k = 2` `d = 3`

Kod İzleme: `maksimin(A)` — 2 × n Sıfır Toplamlı



aday p	0	1	1/2	2/3	1/3
g(p)	·	·	·	·	·

Sütun Y ile Z: d = −3 ≠ 0 ⇒ doğrular p = **1/3**'te kesişir, aday listesine eklenir.

Adım 5 / 13

```
1  from fractions import Fraction as F
2  def maksimin(A):                    # A: 2 x n
3      n, aday = len(A[0]), [F(0), F(1)]
4      for j in range(n):
5          for k in range(j + 1, n):
▶              d = A[0][j] - A[1][j] - A[0][k] + A[1][k]
▶              if d != 0:
▶                  aday.append(F(A[1][k] - A[1][j], d))
9      g = lambda p: min(p*a+(1-p)*b for a, b in zip(*A))
10     p = max((c for c in aday if 0 <= c <= 1), key=g)
11     return p, g(p)
```

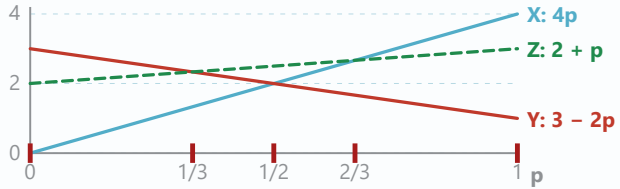
DEĞİŞKENLER

`n = 3`

`aday = [0, 1, 1/2, 2/3, 1/3]`

`j = 1` `k = 2` `d = -3`

Kod İzleme: `maksimin(A)` — 2 × n Sıfır Toplamlı



aday p	0	1	1/2	2/3	1/3
g(p)	·	·	·	·	·

g(p) = $\min_j$ [p·A[0][j] + (1 − p)·A[1][j]]: sütun oyuncusu her p'ye en kötü sütunla yanıt verir (alt zarf).

Adım 6 / 13

```
1  from fractions import Fraction as F
2  def maksimin(A):                    # A: 2 x n
3      n, aday = len(A[0]), [F(0), F(1)]
4      for j in range(n):
5          for k in range(j + 1, n):
6              d = A[0][j] - A[1][j] - A[0][k] + A[1][k]
7              if d != 0:
8                  aday.append(F(A[1][k] - A[1][j], d))
▶      g = lambda p: min(p*a+(1-p)*b for a, b in zip(*A))
10     p = max((c for c in aday if 0 <= c <= 1), key=g)
11     return p, g(p)
```

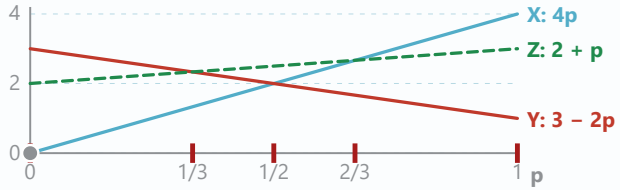
DEĞİŞKENLER

`n = 3`

`aday = [0, 1, 1/2, 2/3, 1/3]`

`j = 2` `k = 2` `d = −3`

Kod İzleme: `maksimin(A)` — 2 × n Sıfır Toplamlı



aday p	0	1	1/2	2/3	1/3
g(p)	0	·	·	·	·

max, adayı dener: g(0) = 0. Şimdiye kadarki en iyi.

Adım 7 / 13

```
1  from fractions import Fraction as F
2  def maksimin(A):                    # A: 2 x n
3      n, aday = len(A[0]), [F(0), F(1)]
4      for j in range(n):
5          for k in range(j + 1, n):
6              d = A[0][j] - A[1][j] - A[0][k] + A[1][k]
7              if d != 0:
8                  aday.append(F(A[1][k] - A[1][j], d))
9      g = lambda p: min(p*a+(1-p)*b for a, b in zip(*A))
▶      p = max((c for c in aday if 0 <= c <= 1), key=g)
11     return p, g(p)
```

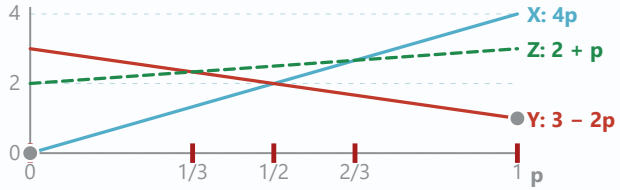
DEĞİŞKENLER

`n = 3`

`aday = [0, 1, 1/2, 2/3, 1/3]`

`j = 2` `k = 2` `d = -3`

Kod İzleme: `maksimin(A)` — 2 × n Sıfır Toplamlı



aday p	0	1	1/2	2/3	1/3
g(p)	0	1	·	·	·

max, adayı dener: g(1) = 1. Şimdiye kadarki en iyi.

Adım 8 / 13

```
1  from fractions import Fraction as F
2  def maksimin(A):                    # A: 2 x n
3      n, aday = len(A[0]), [F(0), F(1)]
4      for j in range(n):
5          for k in range(j + 1, n):
6              d = A[0][j] - A[1][j] - A[0][k] + A[1][k]
7              if d != 0:
8                  aday.append(F(A[1][k] - A[1][j], d))
9      g = lambda p: min(p*a+(1-p)*b for a, b in zip(*A))
▶      p = max((c for c in aday if 0 <= c <= 1), key=g)
11     return p, g(p)
```

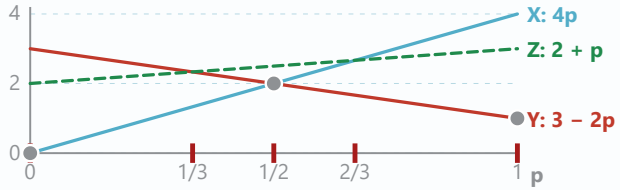
DEĞİŞKENLER

`n = 3`

`aday = [0, 1, 1/2, 2/3, 1/3]`

`j = 2` `k = 2` `d = −3`

Kod İzleme: `maksimin(A)` — 2 × n Sıfır Toplamlı



aday p	0	1	1/2	2/3	1/3
g(p)	0	1	2	·	·

max, adayı dener: g(1/2) = 2. Şimdiye kadarki en iyi.

Adım 9 / 13

```
1  from fractions import Fraction as F
2  def maksimin(A):                    # A: 2 x n
3      n, aday = len(A[0]), [F(0), F(1)]
4      for j in range(n):
5          for k in range(j + 1, n):
6              d = A[0][j] - A[1][j] - A[0][k] + A[1][k]
7              if d != 0:
8                  aday.append(F(A[1][k] - A[1][j], d))
9      g = lambda p: min(p*a+(1-p)*b for a, b in zip(*A))
▶      p = max((c for c in aday if 0 <= c <= 1), key=g)
11     return p, g(p)
```

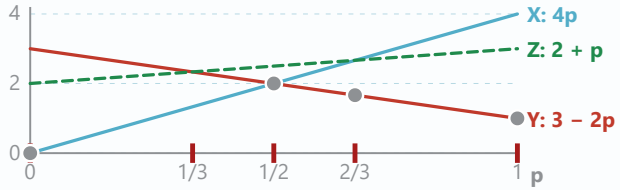
DEĞİŞKENLER

`n = 3`

`aday = [0, 1, 1/2, 2/3, 1/3]`

`j = 2` `k = 2` `d = -3`

Kod İzleme: `maksimin(A)` — 2 × n Sıfır Toplamlı



aday p	0	1	1/2	2/3	1/3
g(p)	0	1	2	5/3	·

max, adayı dener: g(2/3) = 5/3.

Adım 10 / 13

```
1  from fractions import Fraction as F
2  def maksimin(A):                    # A: 2 x n
3      n, aday = len(A[0]), [F(0), F(1)]
4      for j in range(n):
5          for k in range(j + 1, n):
6              d = A[0][j] - A[1][j] - A[0][k] + A[1][k]
7              if d != 0:
8                  aday.append(F(A[1][k] - A[1][j], d))
9      g = lambda p: min(p*a+(1-p)*b for a, b in zip(*A))
▶      p = max((c for c in aday if 0 <= c <= 1), key=g)
11     return p, g(p)
```

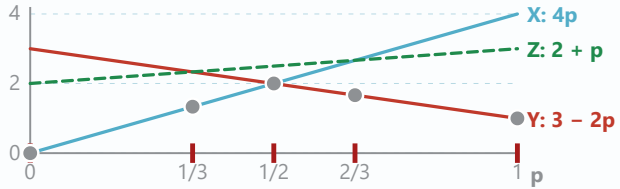
DEĞİŞKENLER

`n = 3`

`aday = [0, 1, 1/2, 2/3, 1/3]`

`j = 2` `k = 2` `d = -3`

Kod İzleme: `maksimin(A)` — 2 × n Sıfır Toplamlı



aday p	0	1	1/2	2/3	1/3
g(p)	0	1	2	5/3	4/3

max, adayı dener: g(1/3) = 4/3.

Adım 11 / 13

```
1  from fractions import Fraction as F
2  def maksimin(A):                    # A: 2 x n
3      n, aday = len(A[0]), [F(0), F(1)]
4      for j in range(n):
5          for k in range(j + 1, n):
6              d = A[0][j] - A[1][j] - A[0][k] + A[1][k]
7              if d != 0:
8                  aday.append(F(A[1][k] - A[1][j], d))
9      g = lambda p: min(p*a+(1-p)*b for a, b in zip(*A))
▶      p = max((c for c in aday if 0 <= c <= 1), key=g)
11     return p, g(p)
```

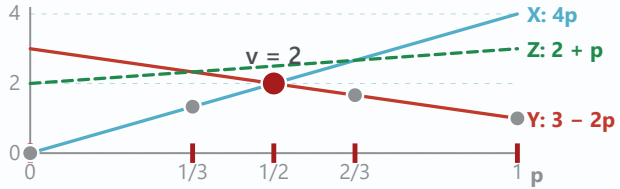
DEĞİŞKENLER

`n = 3`

`aday = [0, 1, 1/2, 2/3, 1/3]`

`j = 2` `k = 2` `d = −3`

Kod İzleme: `maksimin(A)` — 2 × n Sıfır Toplamlı



aday p	0	1	1/2	2/3	1/3
g(p)	0	1	2	5/3	4/3

En büyük garanti p = 1/2'de: **v = 2**. Z sütununun doğrusu zarfın üstünde kalır: sütun oyuncusu Z'yi hiç oynamaz.

Adım 12 / 13

```
1  from fractions import Fraction as F
2  def maksimin(A):                    # A: 2 x n
3      n, aday = len(A[0]), [F(0), F(1)]
4      for j in range(n):
5          for k in range(j + 1, n):
6              d = A[0][j] - A[1][j] - A[0][k] + A[1][k]
7              if d != 0:
8                  aday.append(F(A[1][k] - A[1][j], d))
9      g = lambda p: min(p*a+(1-p)*b for a, b in zip(*A))
▶      p = max((c for c in aday if 0 <= c <= 1), key=g)
11     return p, g(p)
```

DEĞİŞKENLER

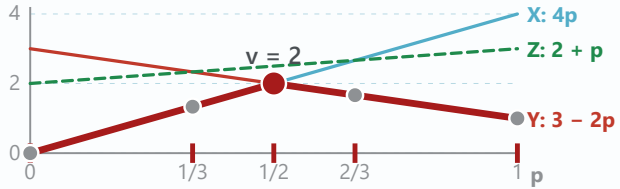
`n = 3`

`aday = [0, 1, 1/2, 2/3, 1/3]`

`j = 2` `k = 2` `d = −3`

`p = 1/2`

Kod İzleme: `maksimin(A)` — 2 × n Sıfır Toplamlı



aday p	0	1	1/2	2/3	1/3
g(p)	0	1	2	5/3	4/3

Dönüş (1/2, 2). Minimaks tarafı: sütun oyuncusu X'i 1/3, Y'yi 2/3 ile oynarsa satır en fazla 2 alır ⇒ max-min = min-max = 2.

Adım 13 / 13

```
1  from fractions import Fraction as F
2  def maksimin(A):                    # A: 2 x n
3      n, aday = len(A[0]), [F(0), F(1)]
4      for j in range(n):
5          for k in range(j + 1, n):
6              d = A[0][j] - A[1][j] - A[0][k] + A[1][k]
7              if d != 0:
8                  aday.append(F(A[1][k] - A[1][j], d))
9      g = lambda p: min(p*a+(1-p)*b for a, b in zip(*A))
10     p = max((c for c in aday if 0 <= c <= 1), key=g)
▶      return p, g(p)
```

DEĞİŞKENLER

`n = 3`

`aday = [0, 1, 1/2, 2/3, 1/3]`

`j = 2` `k = 2` `d = −3`

`p = 1/2` `dönüş = (1/2, 2)`

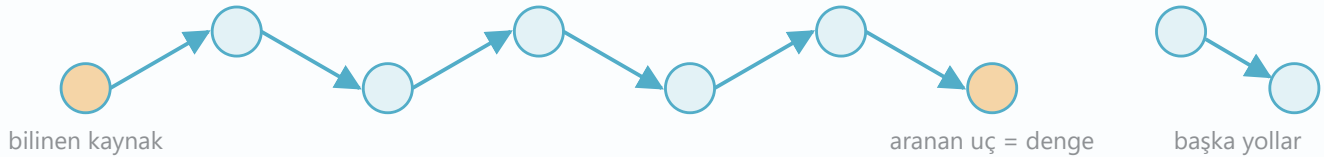
Dengeyi Hesaplamak Ne Kadar Zor?

Problem	Karmaşıklık	Yöntem / kaynak
Saf Nash dengesi (matris verili)	Polinom	En iyi tepki işaretleme, O(mn)
İki kişilik sıfır toplamlı, karışık	Polinom	Doğrusal programlama (minimaks)
İki kişilik genel oyun, karışık	**PPAD-tam**	Chen–Deng (2006)
Üç ve daha çok oyuncu, tam denge	**FIXP-tam**	Etessami–Yannakakis (2007)
Üç ve daha çok oyuncu, ε-yaklaşık denge	**PPAD-tam**	Daskalakis–Goldberg–Papadimitriou (2006)
En yüksek refahlı denge / tek denge mi?	NP-zor	Gilboa–Zemel (1989); Conitzer–Sandholm (2008)

- **Destek sayımı** ve Lemke–Howson pratikte çalışır ama en kötü durumda üsteldir.
- Üç oyunculu oyunlarda denge **irrasyonel** olabilir; bu yüzden ε-yaklaşık denge (sapmanın en çok ε kazandırdığı profil) aranır. İki kişilik oyunlarda dengeler rasyoneldir.
- İleri okuma: son satır "özel bir dengeyi" aramanın zorluğudur. Sabit ε (ör. 0,1) için iki kişilik oyunlarda yarı-polinom algoritma vardır (Lipton–Markakis–Mehta, 2003); çok oyunculu oyunlarda PPAD-zor kalır (Rubinstein, 2015).

PPAD: Var Olduğu Bilinen Çözümü Bulmak

- Nash dengesi **her zaman vardır** → “denge var mı?” sorusu önemsiz; NP-tamlık bu tür (çözümü garanti) problemler için uygun değildir.
- **PPAD** (Papadimitriou, 1994): çözümün var olduğu kesin bilinen, ama bulmak için çok uzun bir yolu adım adım izlemek gerekebilen problemlerin sınıfı.



- Benzetme: bir yolun başından yürüyüp sonunu aramak; yolun sonu kesin vardır, ama yol üstel uzunlukta olabilir.
- Nash dengesi bulmak bu sınıftadır. Sonuç: genel oyunlarda denge verimli bulunamayabilir → **oyuncular dengeyi “bulur” mu?**

Pratikte: Yaklaşık Denge ve Hesaplama Araçları

- **ε-Nash dengesi**: kazançlar [0, 1] aralığına ölçeklenmişken hiçbir oyuncu tek taraflı sapmayla ε'dan fazla kazanamaz. Gerçek zamanlı sistemler (reklam açık artırmaları, piyasalar) çoğu zaman bununla yetinir.
- Tam dengeyi bulmak zor olabilir, ama "neredeyse kimsenin sapmak istemediği" bir profil çoğu zaman hızlıca bulunur; az sayıda stratejiyi karıştıran basit profiller bile iyi yaklaşıklık verir.
- **İleri okuma:** iki en iyi tepki ve bir yarı yarıya karıştırmayla kurulan ½-yaklaşık denge (Daskalakis–Mehta–Papadimitriou, 2006); küçük destekli ε-dengeler (Lipton–Markakis–Mehta, 2003).

```python
import nashpy as nash
A = [[2, 2], [0, 4]]    # işçi: Çalış / Kaytar
B = [[1, 2], [-1, -4]]  # müdür: Denetle / Denetleme
oyun = nash.Game(A, B)
for x, y in oyun.support_enumeration():
    print(x, y)  # [0.75 0.25] [0.5 0.5]
```

Denetim oyunu Nashpy ile: destek sayımı p* = 3/4, q* = 1/2 dengesini verir. Gambit de aynı işi (destek sayımı, Lemke–Howson, LP) arayüzle yapar.

Başka denge kavramları da vardır; ör. **korelasyonlu denge** ve sıralı oyunlarda **alt oyun mükemmel denge**.

ÖZET

Tartışma: Denge, Adalet ve Rasyonellik

Denge adil mi?

Olmak zorunda değildir. Denge yalnızca karşılıklı en iyi tepkidir: Cinsiyetler Savaşı'nda kazançlar eşit değildir, Mahkûmun İkilemi'nde herkes için daha iyi bir sonuç vardır. *Adalet ≠ denge*.

İnsanlar rasyonel mi?

Gerçek kararlar bilgi eksikliği, zaman baskısı, duygular ve sosyal normlardan etkilenir (sınırlı rasyonellik, Herbert Simon). Model, stratejik bağımlılığı incelemek için açık bir karşılaştırma noktasıdır.

Etik sınırlar

Aldatma, manipülasyon, adil olmayan avantaj… "Kazanmak için her yol mubah mı?" Stratejik davranışın sınırlarını kurallar ve kurumlar çizer — bu da mekanizma tasarımının konusudur.

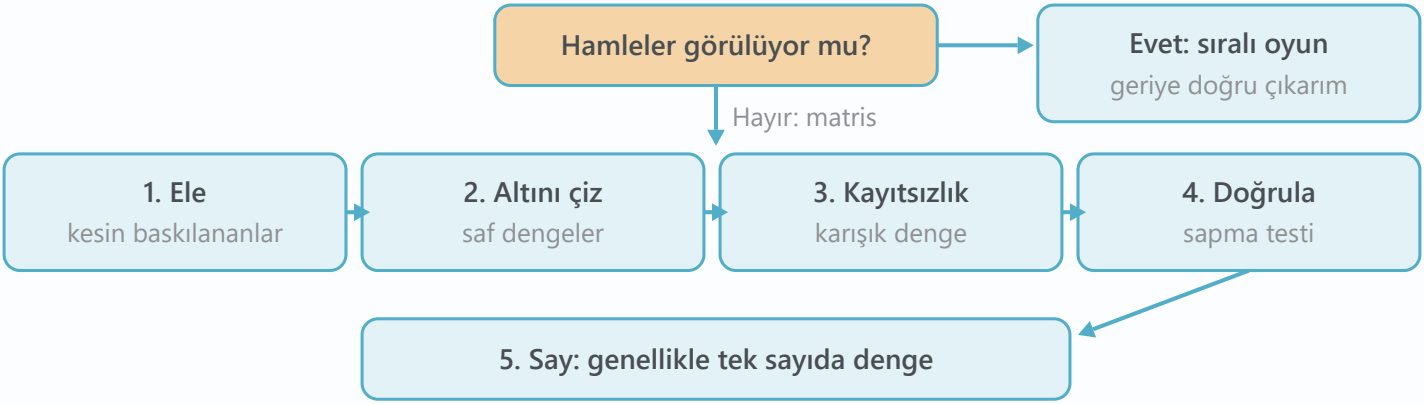
ÖZET

Özet

Kavram	Ne söyler?	Nasıl bulunur?
Nash dengesi	Kimse tek başına sapmak istemez	Karşılıklı en iyi tepki
Baskın strateji eleme	Rasyonel oyuncu baskılananı oynamaz	Kesin baskılananları yinelemeli sil
Karışık denge	Destekteki stratejiler eşit kazandırır	Kayıtsızlık denklemleri
Varlık (Nash 1950)	Sonlu oyunda karışık denge hep var	Kakutani / Brouwer sabit noktası
Çoklu denge	Kuram tek sonucu seçmeyebilir	Pareto / risk baskınlık, odak noktası
Korelasyonlu denge	Aracının önerisine uymak herkesin çıkarına	Uyma kısıtları + doğrusal programlama
Minimaks	Sıfır toplamlıda maks-min = min-maks = v	Eyer noktası ya da LP
Karmaşıklık	Genel oyunda denge PPAD-tam	Lemke–Howson, destek sayımı

ÖZET

Kontrol Listesi: Bir Oyunu Adım Adım Çözmek



Önce okuma

Kim satır, kim sütun? Hücrede ilk sayı kimin? Sayılar kazanç mı, maliyet mi (yıl, dakika)?

Sonra yazma

Her dengeyi profil olarak yazın: (Orta, Sol) = (3, 5); karışıkta her iki oyuncunun olasılığını ve beklenen kazancı verin.

ÖZET

Kendini Sına (1/2)

Oyun A

	Sol	Orta	Sağ
Üst	(3 , 1)	(1 , 2)	(2 , 0)
Alt	(0 , 2)	(2 , 3)	(1 , 1)

Oyun B	Sol	Sağ
Üst	(2, 1)	(0, 0)
Alt	(0, 0)	(1, 2)

1. Oyun A'da sütun oyuncusunun baskın stratejisi var mı?
2. Oyun A'nın saf Nash dengelerini altını çizerek bulun.
3. Oyun B'nin kaç dengesi vardır? Karışık dengede p = P(Üst) ve q = P(Sol) nedir?
4. Oyun B'de karışık dengedeki beklenen kazançlar saf dengelerinkinden iyi mi?

▶ **Cevapları göster**

ÖZET

Kendini Sına (2/2)

Doğru mu, yanlış mı?

1. Kesin baskılanan bir strateji bir Nash dengesinde oynanabilir.
2. Karışık dengede satır oyuncusunun olasılığı kendi kazançlarından hesaplanır.
3. Alt oyun mükemmel denge (oyun ağacında geriye doğru çıkarımla bulunan denge) her zaman bir Nash dengesidir.
4. Üç oyunculu oyunda iki oyuncunun birlikte sapması da denenir.

Sıralı oyun

5. Oyuncu 1 Sol ya da Sağ der. Sağ → oyun (1, 1) ile biter. Sol → Oyuncu 2, a → (3, 0) ya da b → (0, 2) seçer. Geriye doğru çıkarımla dengeyi ve sonucu bulun.
6. Oyuncu 2 “Sol derse a seçerim” sözü verse, Oyuncu 1 buna güvenmeli mi?

▶ **Cevapları göster**

Örnek Problemler

Oyun 1	L	M	R
U	(2, 1)	(1, 2)	(0, 0)
D	(3, 0)	(0, 1)	(1, 3)

Oyun 2	L	R
U	(4, 1)	(0, 3)
D	(1, 2)	(2, 0)

1. Oyun 1'in saf Nash dengelerini en iyi tepki işaretlemesiyle bulun.
2. Oyun 2'nin karışık dengesini kayıtsızlık ilkesiyle bulun.
3. Sıfır toplamlı $A = [[3, -1, 2], [1, 0, 4], [-2, -3, 5]]$: eyer noktası var mı? Değer?
4. "Her Nash dengesi Pareto optimaldir." Doğru mu?
5. Varlık teoremi neden karışık stratejiler için ifade edilir?
6. Cournot, $P = 15 - Y$, maliyet 0: Nash dengesindeki miktarlar? Üretimi eşit bölen kartelde her firma kaç üretir, kartel neden dengede değildir?

▶ **Cevapları göster**

ÖZET

Alıştırmalar (1/2)

Oyun 1	Sol	Sağ
Üst	(4, 3)	(1, 2)
Alt	(3, 1)	(2, 4)

Oyun 2	0	1	2
0	(2, 5)	(1, 3)	(3, 1)
1	(5, 2)	(3, 3)	(1, 2)
2	(0, 0)	(5, 5)	(0, 0)

1. Oyun 1'de baskın strateji var mı? Saf Nash dengelerini en iyi tepki işaretlemesiyle bulun.
2. Oyun 2'de (satır i, sütun j) kesin baskılanan stratejileri yinelemeli olarak silin. Ne kalır? Saf Nash dengesiyle karşılaştırın.
3. TKM'de rakip yarı yarıya Taş ve Kağıt oynuyor (hiç Makas yok). En iyi tepkiniz ve beklenen kazancınız nedir?

▶ **Cevapları göster**

Alıştırmalar (2/2)

4. Sıfır toplamlı A = [[3, 1, 4], [2, 0, 1], [5, 2, 3]] ve B = [[0, 2], [3, 1]] matrislerinde maksimin ve minimaksı bulun. Hangisinde eyer noktası var?
5. (X, X) = (4, 1), (Y, Y) = (1, 4), diğer hücreler (0, 0). Karışık dengede p* = P(satır X), q* = P(sütun X) ve buluşamama olasılığı nedir?
6. Tavuk'ta çarpışma −10 yerine −20 olsaydı karışık dengede kırma ve çarpışma olasılıkları ne olurdu?
7. Geyik Avı'nda (geyik 4, tavşan 3) tavşan 3 yerine 2 kazandırsaydı karışık dengede geyik olasılığı ne olurdu? Güven kolaylaşır mı?

▶ **Cevapları göster**

ÖZET

Etkileşimli Oyunlar ve Laboratuvarlar

Matris ve Nash Laboratuvarı

Herhangi bir matriste en iyi tepkiler ve dengeler.

Nashpy ve Gambit

Dengeyi bilgisayarla bulma: destek sayımı, Lemke–Howson.

Albay Blotto

Birlik dağıtma: saf dengesi olmayan sıfır toplamlı oyun.

Mahkûmun İkilemi

Baskın strateji ve denge simülatörü.

Taş-Kağıt-Makas

Karışık denge: 1/3–1/3–1/3.

Kaynaklar

- J. F. Nash, “Equilibrium points in n-person games”, *PNAS* 36(1), 1950; “Non-cooperative games”, *Annals of Mathematics* 54(2), 1951.
- J. von Neumann, “Zur Theorie der Gesellschaftsspiele”, *Mathematische Annalen* 100, 1928.
- M. J. Osborne, *An Introduction to Game Theory*, Oxford University Press, 2004 (Bölüm 2–4).
- N. Nisan, T. Roughgarden, É. Tardos, V. Vazirani (ed.), *Algorithmic Game Theory*, Cambridge, 2007 (Bölüm 1–3).
- T. Roughgarden, *Twenty Lectures on Algorithmic Game Theory*, Cambridge, 2016 (Ders 13, 18 ve 20).
- C. Daskalakis, P. Goldberg, C. Papadimitriou, “The complexity of computing a Nash equilibrium”, *SIAM J. Computing* 39(1), 2009.
- K. Etessami, M. Yannakakis, “On the complexity of Nash equilibria and other fixed points”, *SIAM J. Computing* 39(6), 2010 · R. J. Lipton, E. Markakis, A. Mehta, “Playing large games using simple strategies”, *ACM EC*, 2003 · A. Rubinstein, “Inapproximability of Nash equilibrium”, *STOC*, 2015.

Kaynaklar (2/2)

- J. C. Harsanyi, R. Selten, *A General Theory of Equilibrium Selection in Games*, MIT Press, 1988.
- A. R. Karlin, Y. Peres, *Game Theory, Alive*, AMS, 2017 (sıfır toplamlı ve genel oyunlar, Nash teoremi).
- D. Easley, J. Kleinberg, *Networks, Crowds, and Markets*, Cambridge, 2010 (Bölüm 6: Oyunlar).
- M. Tambe, *Security and Game Theory*, Cambridge University Press, 2011 (güvenlik oyunları, ARMOR).
- D. Sally, "Conversation and cooperation in social dilemmas: a meta-analysis of experiments from 1958 to 1992", *Rationality and Society* 7(1), 1995 · R. Wilson, "Computing equilibria of n-person games", *SIAM J. Applied Mathematics* 21, 1971.
- Ders sayfaları: Roughgarden CS364A · Shah CSC304 · Araçlar: Gambit, Nashpy · Video: Game Theory 101

Son

Bölüm 3: Nash Dengesi

Sonraki bölüm: Bölüm 4: Mekanizma Tasarımı

Çalışma soruları · Terim sözlüğü