Bölüm 5: Açık Artırmalar

Algoritmik Oyun Kuramı

Dört temel tür | Vickrey dürüstlük ispatı | Teklif kırpma | Gelir eşdeğerliği | Rezerv fiyatı | Kazananın laneti | Çift taraflı açık artırma

Sercan KÜLCÜ · Giresun Üniversitesi · Bilgisayar Mühendisliği

© Sercan Külcü. Tüm hakları saklıdır. · Sunumun etkileşimli HTML sürümü: https://sercankulcu.github.io/files/agt/slides/Bolum_05_Acik_Artirmalar.html

TEMEL KAVRAMLAR

Oyunla Başlangıç: Vickrey Açık Artırması

Oyunun kuralı

Herkes teklifini kapalı zarfa yazar; en yüksek teklifi veren nesneyi alır ama kendi teklifini değil, ikinci en yüksek teklifi öder.

Merak sorusu

- Nesne size 70 TL değerindeyse zarfa 70'ten azını mı, 70'i mi, fazlasını mı yazardınız?
- Rakiplerin teklifini bilseydiniz teklifiniz değişir miydi?
- Kazanan kendi teklifini ödeseydi (birinci fiyat), 70 yazmak yine iyi bir fikir olur muydu?

[Vickrey Açık Artırması oyunu →]



Siz nesneyi alırsınız ve 60 TL ödersiniz.

Değeriniz 70 TL ise kazancınız 70 − 60 = 10.

TEMEL KAVRAMLAR

Öğrenme Çıktıları

Türleri çalıştırmak

İngiliz, Hollanda, birinci fiyat ve ikinci fiyat açık artırmalarında kazananı ve ödemeyi bulmak.

Dürüstlüğü ispatlamak

Vickrey açık artırmasında dürüst teklifin zayıf baskın strateji olduğunu durum analiziyle göstermek.

Denge teklifi

Birinci fiyatta $b = (n-1)v/n$ kırpmasını türetmek; gelir eşdeğerliğini (2 teklifçi → 1/3) ve rezerv fiyatının etkisini (5/12) hesaplamak.

Piyasa olguları

Kazananın lanetini açıklamak; çift taraflı açık artırmada eşleştirme ve fiyatı bulmak.

TEMEL KAVRAMLAR

Açık Artırmaların Önemi

- Kaynak tahsisi için verimli mekanizmalar: nesne, onu en çok değerlendirene gitme eğilimindedir.
- Fiyat keşfi ve piyasa şeffaflığı: satıcı, alıcıların değerini bilmeden fiyat oluşturur.
- Mekanizma tasarımının en yaygın uygulaması: tahsis kuralı + ödeme kuralı.

Sanat eserleri

Müzayede evlerinde yükselen fiyatlı İngiliz açık artırması.

Çevrim içi reklam

Her aramada çalışan reklam açık artırmaları.

Telekom lisansları

Spektrum açık artırmaları; milyarlarca liralık çok nesneli tasarımlar.

Kamu ihaleleri, hazine

Kapalı zarf teklifler; borçlanma senedi ihaleleri.

TEMEL KAVRAMLAR

Temel Terimler ve Model

Değer (value) v_i

Teklifçinin nesneye biçtiği gerçek değer; özel bilgidir.

Teklif (bid) b_i

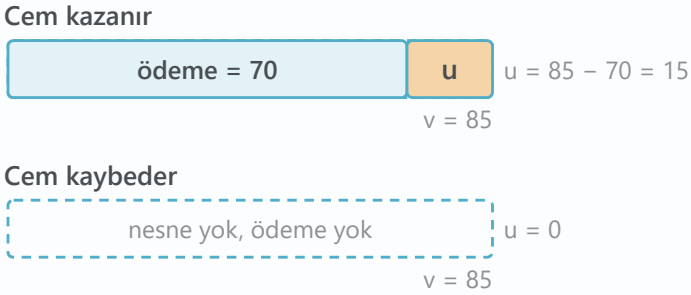
Mekanizmaya bildirilen sayı; değerden farklı olabilir.

Fayda (utility)

Kazanırsa $u_i = v_i - p_i$ (p_i: ödeme), kaybederse 0.

Bağımsız özel değer (IPV)

Değerler bağımsız çekilir, herkes kendi değerini bilir. Açık artırma kuramının temel modeli.



İkinci fiyat örneği (Cem 85, ikinci teklif 70): yarı doğrusal fayda.

Örneklerde çoğunlukla $v_i \sim U[0, 1]$ (düzgün dağılım) ve risk nötr (risk-neutral) teklifçiler varsayılır.

TEMEL KAVRAMLAR

Değerleme Modelleri

Özel değer

Her teklifçi nesnenin değerini kendi bilgisine göre belirler (ör. kendi kullanacağı tablo).

Ortak değer

Nesnenin gerçek değeri herkes için aynıdır, ancak katılımcıların tahminleri farklıdır (ör. petrol sahası).

Birbirine bağlı değerler

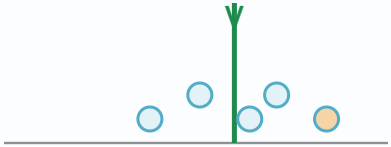
Hem özel bilgi hem de diğer katılımcıların bilgileri değerlemeyi etkiler.

özel değer



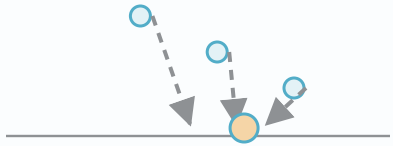
her biri kendi değeri: $v_1 \neq v_2$ ...

ortak değer



tek V, gürültülü tahminler

bağlı değer



v_i = f(kendi + diğer sinyaller)

Ortak değer modeli **kazananın laneti**ne yol açar; kazananın en iyimser tahmini yapan olması beklenen zararı doğurur.

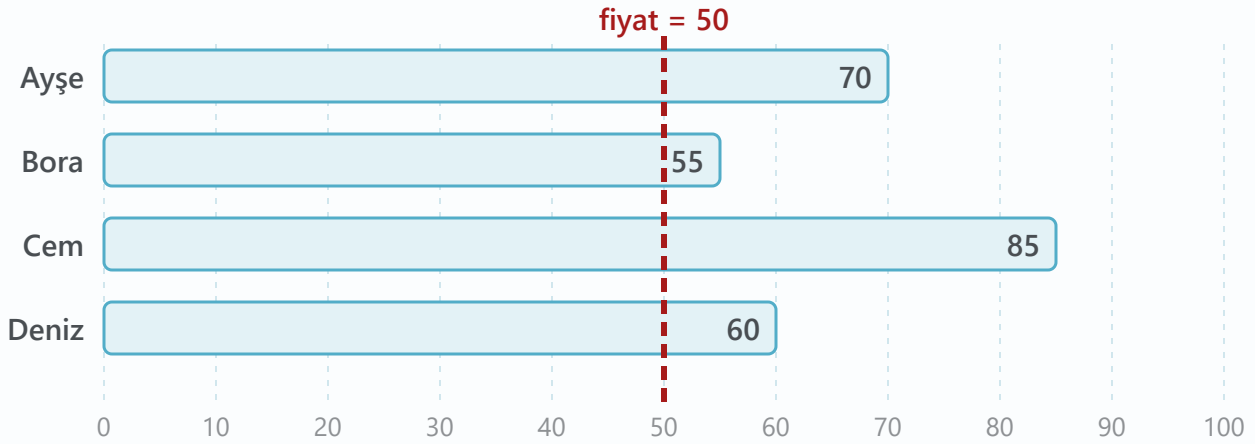
DÖRT TEMEL TÜR

Dört Temel Açık Artırma Türü

Tür	Nasıl işler?	Kazanan	Öder
İngiliz (English)	Fiyat açıkça yükselir; teklifçiler birer birer çekilir.	Son kalan	Son rakibin çekildiği fiyat
Hollanda (Dutch)	Fiyat yüksekten başlar, düşer; ilk "kabul" diyen alır.	İlk kabul eden	O anki fiyat
Birinci fiyat kapalı zarf	Teklifler gizli verilir, bir kez.	En yüksek teklif	Kendi teklifi
İkinci fiyat (Vickrey)	Teklifler gizli verilir, bir kez.	En yüksek teklif	İkinci en yüksek teklif

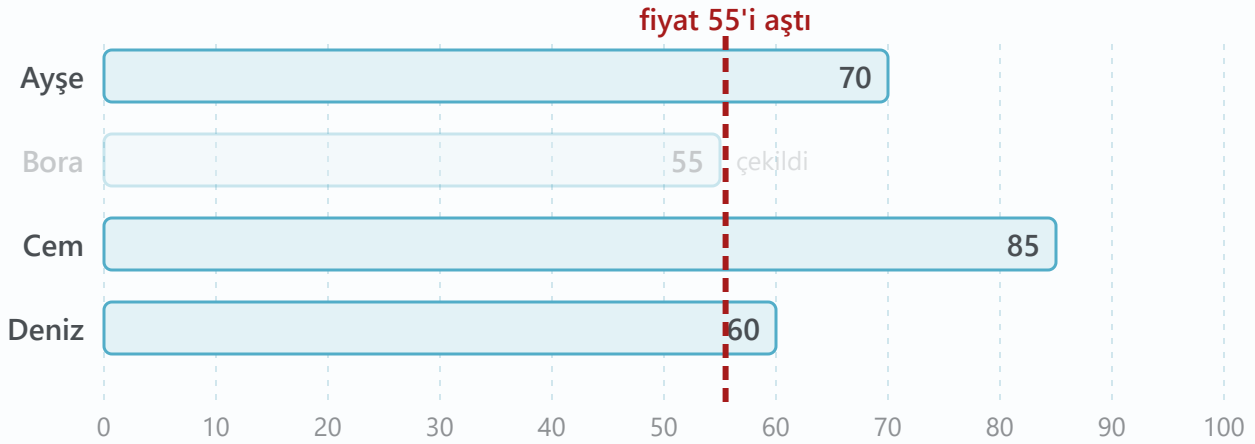
Dört kuralı aynı dört kişiyle deneyelim. Her kişinin ödemeye razı olduğu en yüksek tutar (sınırı): Ayşe 70, Bora 55, Cem 85, Deniz 60. Şimdilik herkes sınırını teklif ediyor.

Adım Adım: İngiliz Açık Artırması (Yükselen Fiyat)



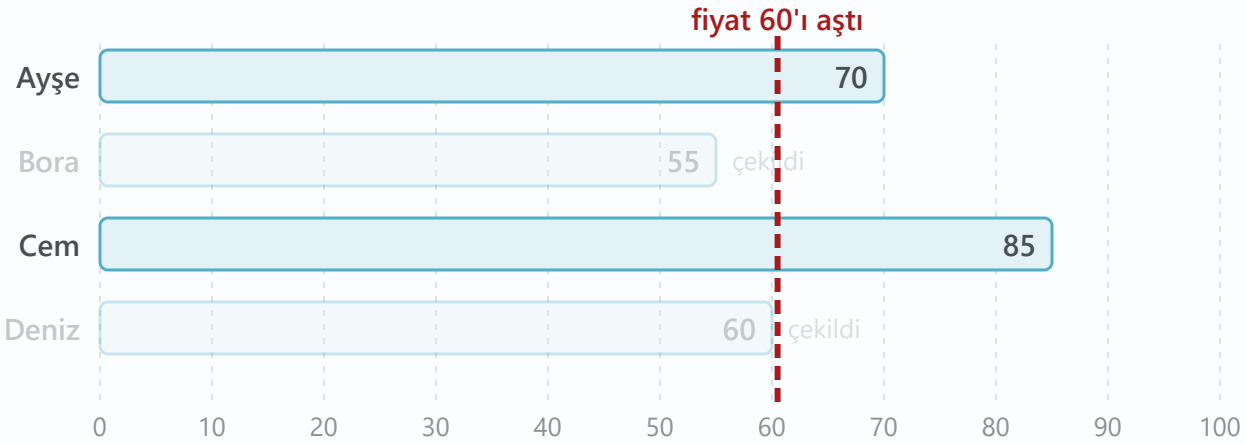
Fiyat 40'tan başlayıp yükselir. Fiyat = 50: herkesin sınırı 50'nin üstünde, **dört kişi de içeride**.

Adım Adım: İngiliz Açık Artırması (Yükselen Fiyat)



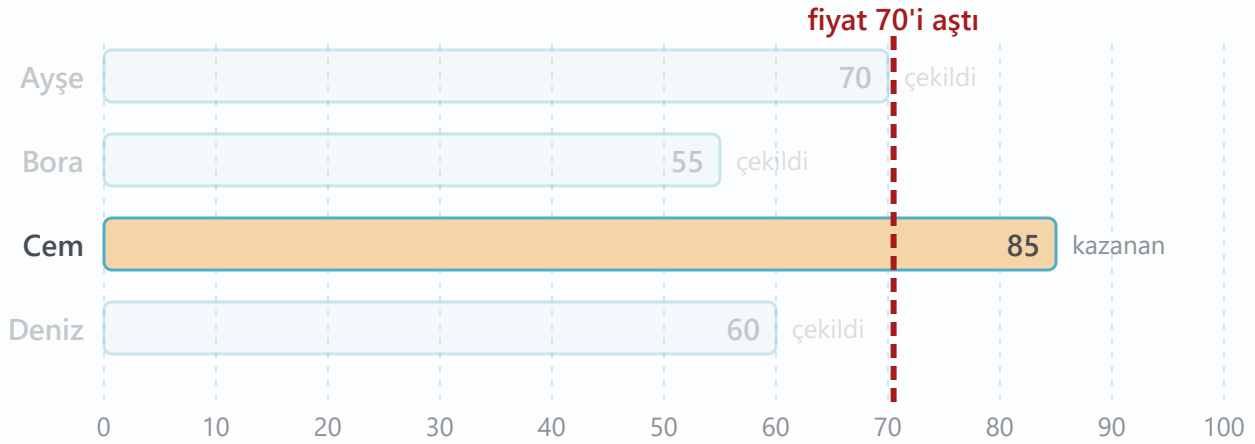
Fiyat **55**'i geçince Bora çekilir (sınırı 55). Kalanlar: Ayşe, Cem, Deniz.

Adım Adım: İngiliz Açık Artırması (Yükselen Fiyat)



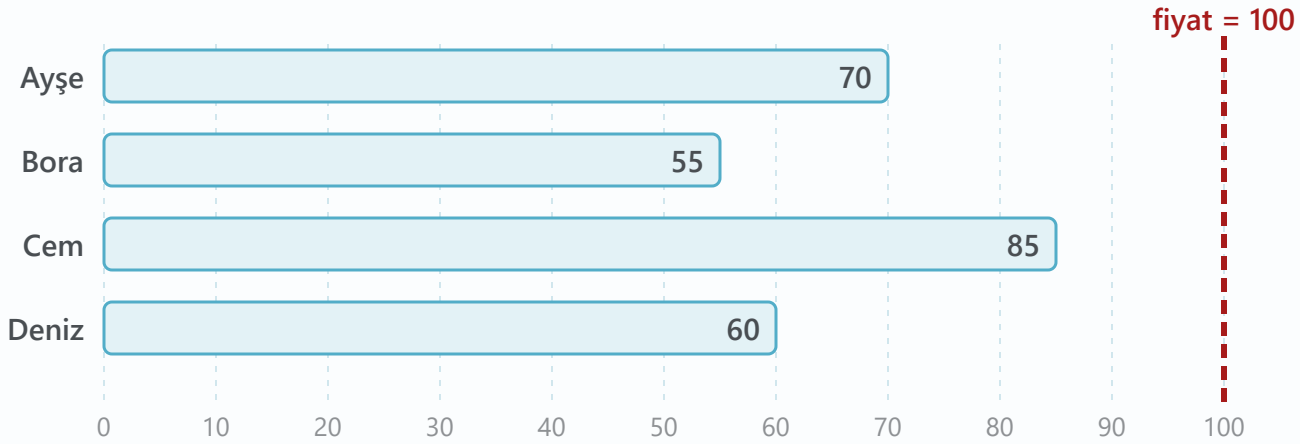
Fiyat **60**'ı geçince Deniz çekilir (sınırı 60). Kalanlar: Ayşe, Cem.

Adım Adım: İngiliz Açık Artırması (Yükselen Fiyat)



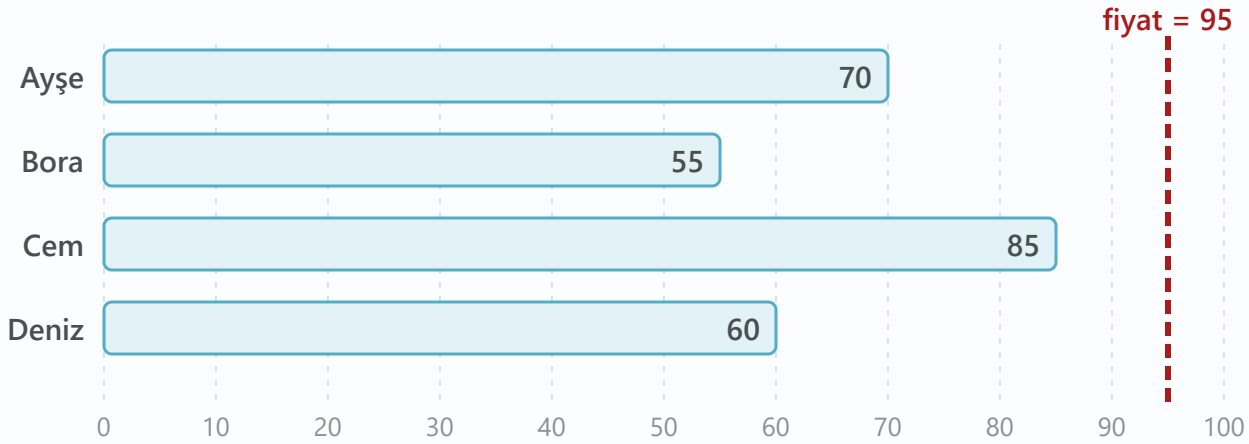
Fiyat 70'i geçince son rakip Ayşe çekilir → **Cem kazanır, ≈ 70 öder** (artım küçüldükçe tam 70): ikinci en yüksek değer.

Adım Adım: Hollanda Açık Artırması (Düşen Fiyat)



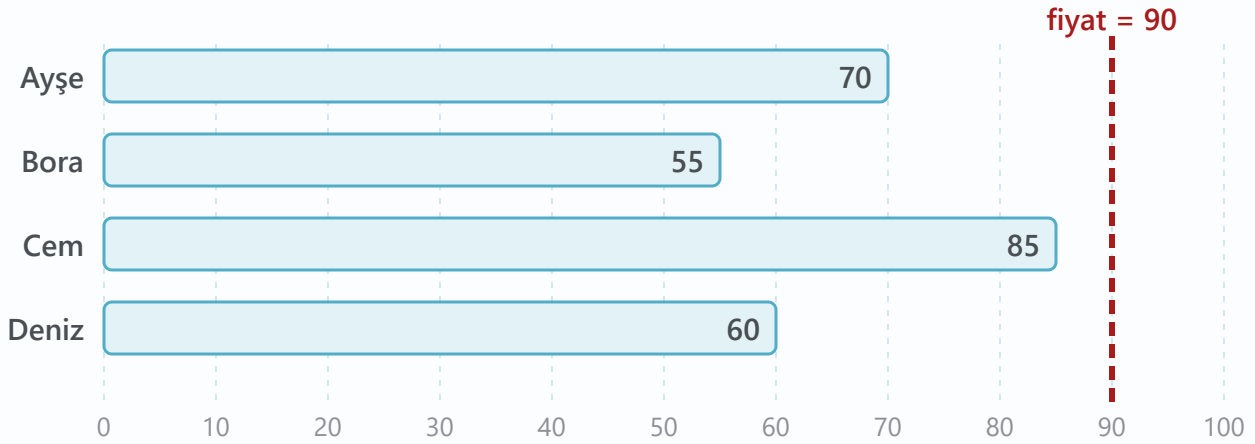
Fiyat = 100: bu fiyat herkesin sınırının üstünde; **kimse kabul etmez**, fiyat 5 düşer.

Adım Adım: Hollanda Açık Artırması (Düşen Fiyat)



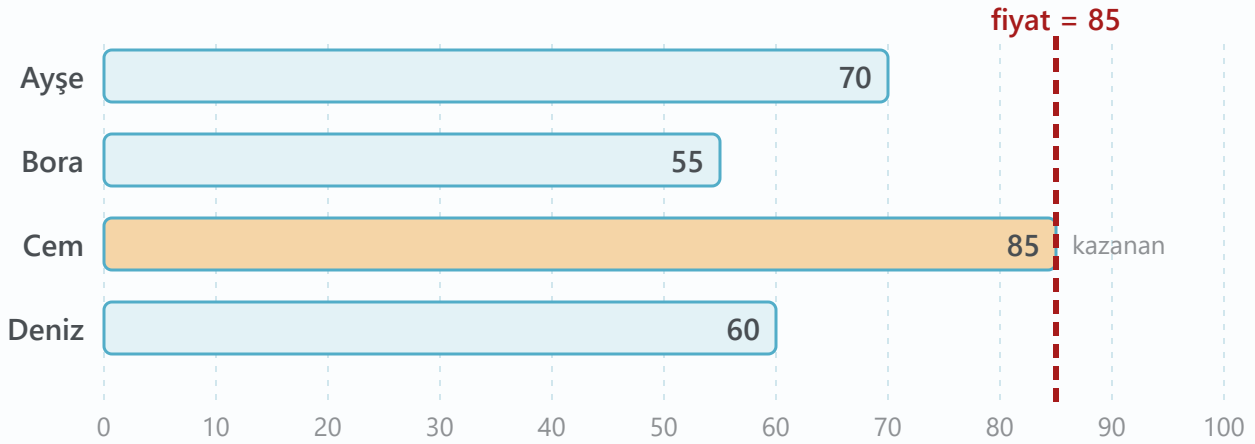
Fiyat = 95: hâlâ kimse kabul etmiyor (en yüksek sınır 85).

Adım Adım: Hollanda Açık Artırması (Düşen Fiyat)



Fiyat = 90: hâlâ kimse kabul etmiyor (en yüksek sınır 85).

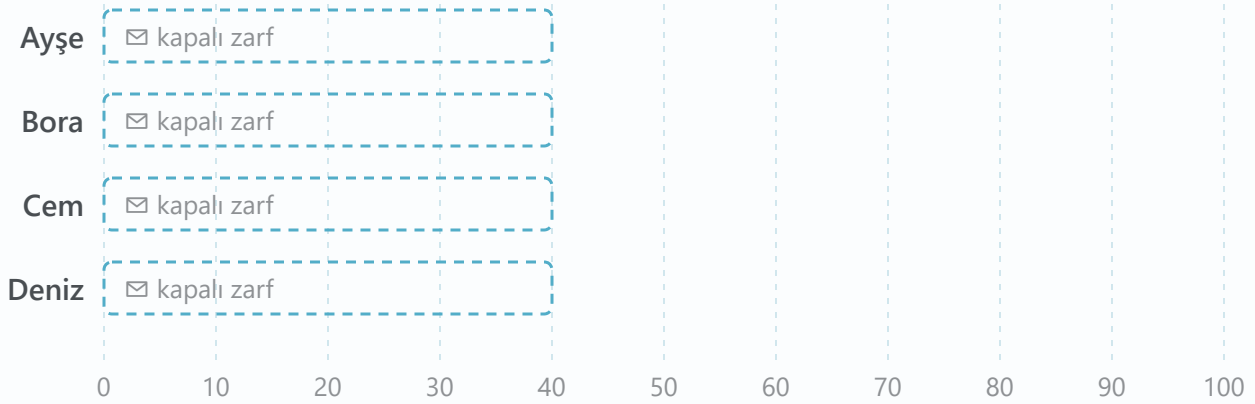
Adım Adım: Hollanda Açık Artırması (Düşen Fiyat)



Fiyat = 85: fiyat Cem'in sınırına indi → **Cem “kabul” der, 85 öder**. Diğerleri hiç konuşamadan açık artırma biter.

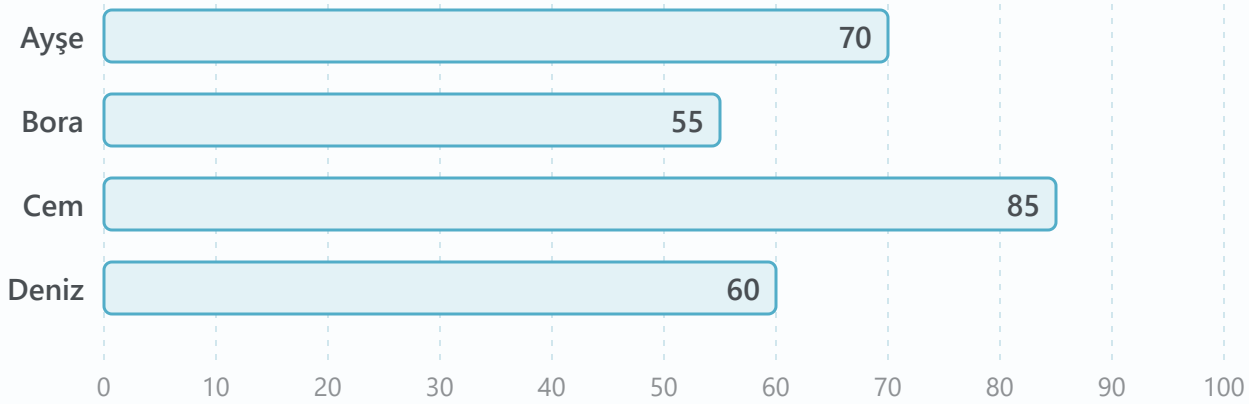
DÖRT TEMEL TÜR

Adım Adım: Birinci ve İkinci Fiyat Kapalı Zarf



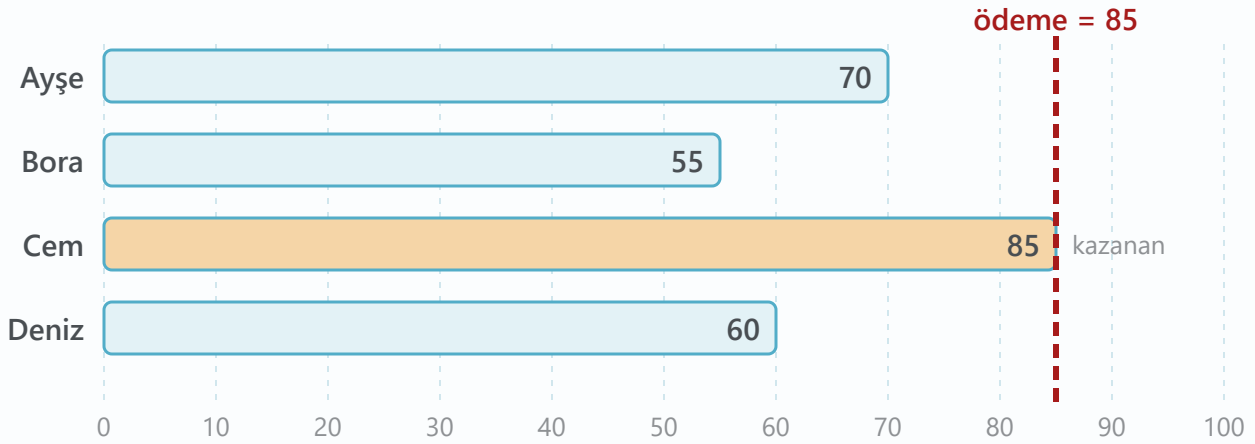
Kapalı zarf: herkes teklifini bir kez, gizlice verir. Kimse diğerlerinin teklifini görmez.

Adım Adım: Birinci ve İkinci Fiyat Kapalı Zarf



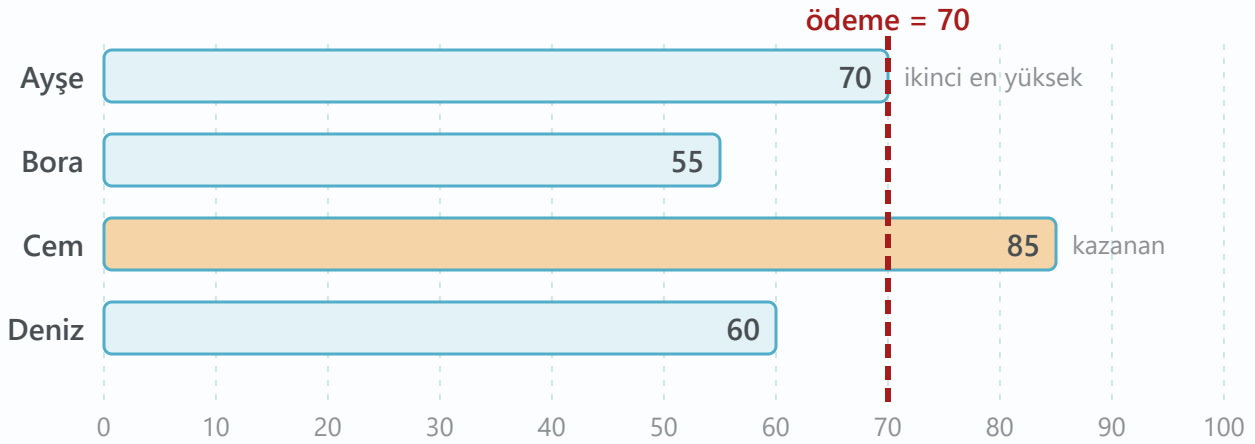
Zarflar açılır: 70, 55, 85, 60. Kazanan her iki kuralda da en yüksek teklif: **Cem**.

Adım Adım: Birinci ve İkinci Fiyat Kapalı Zarf



Birinci fiyat: Cem kendi teklifini öder → **85**. Değeri 85 ise faydası 0.

Adım Adım: Birinci ve İkinci Fiyat Kapalı Zarf



İkinci fiyat (Vickrey): Cem ikinci en yüksek teklifi öder → **70**. Faydası 85 − 70 = 15.

DÖRT TEMEL TÜR

Aynı Teklifler, Dört Sonuç

Tür	Kazanan	Ödeme	Kimin teklifi belirliyor?
İngiliz	Cem	≈ 70	Son çekilen rakip (Ayşe)
İkinci fiyat	Cem	70	İkinci en yüksek teklif (Ayşe)
Hollanda	Cem	85	Kazananın kendisi
Birinci fiyat	Cem	85	Kazananın kendisi

İngiliz ≈ ikinci fiyat

Özel değerlerde değerine kadar kalmak baskındır; ikisi de ikinci en yüksek değeri ödetir.

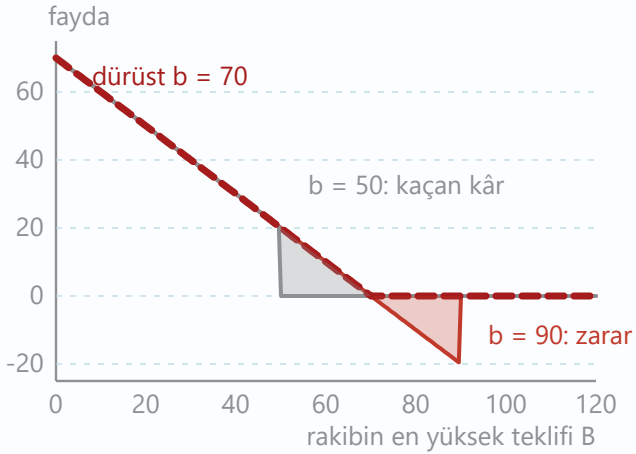
Hollanda ≡ birinci fiyat

Stratejik olarak özdeştir: "hangi fiyatta kabul ederim?" sorusu gizli teklif vermekle aynıdır.

Dikkat: bu tabloda herkes sınırını teklif etti. Birinci fiyatta rasyonel teklifçi **teklifini kırpar**, bu yüzden gelirler aslında eşitlenir (gelir eşdeğerliği).

Vickrey Açık Artırmasının Stratejik Özelliği

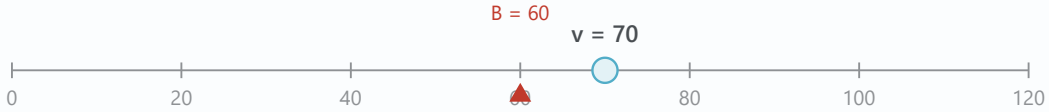
- En yüksek teklifi veren kazanır ve ikinci en yüksek teklif kadar ödeme yapar.
- Bağımsız özel değerli tek nesne modelinde gerçek değeri teklif etmek **zayıf baskın stratejidir** (Vickrey, 1961).
- Fikir: teklifin, *ne ödeyeceğini* değil yalnızca *kazanıp kazanmayacağını* belirler.



v = 70. Kesikli kırmızı: dürüst teklif. Hiçbir teklif, hiçbir B için onu geçemez.

Sonuç, ortak değerler, bütçe kısıtları veya birbirine bağlı değerler altında aynı biçimde geçerli olmayabilir. Vickrey, VCG'nin tek nesneli özel durumudur.

Adım Adım: Vickrey'de Dürüstlüğün İspatı



Strateji	B (rakip)	Sonuç	Fayda	Dürüste göre
Dürüst b = 70	60	kazanır, 60 öder	**10**	—

Değerin v = 70. B = max b_{-i}, diğerlerinin en yüksek teklifi (bilmiyorsun). Dürüst teklifte B = 60 ise kazanır, 60 ödersin → 10.

Adım 1 / 8

v = 70 için tüm b, B ∈ {0, …, 120} çiftleri denendiğinde hiçbir b dürüst teklifi geçemiyor.

Adım Adım: Vickrey'de Dürüstlüğün İspatı

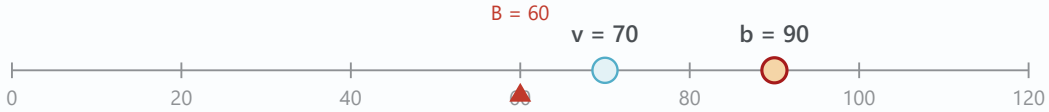


Strateji	B (rakip)	Sonuç	Fayda	Dürüste göre
Dürüst b = 70	60	kazanır, 60 öder	**10**	—
Dürüst b = 70	80	kaybeder	**0**	—

Dürüst teklifte B = 80 ise kaybedersin → 0. Dürüst teklif hep doğru kararı verir: kâr varsa kazan, zarar varsa kaybet.

v = 70 için tüm b, B ∈ {0, …, 120} çiftleri denendiğinde hiçbir b dürüst teklifi geçemiyor.

Adım Adım: Vickrey'de Dürüstlüğün İspatı



Strateji	B (rakip)	Sonuç	Fayda	Dürüste göre
Dürüst b = 70	60	kazanır, 60 öder	**10**	—
Dürüst b = 70	80	kaybeder	**0**	—
Fazla b = 90	60	kazanır, 60 öder	**10**	aynı

Fazla teklif b = 90, B = 60 (< v): yine kazanırsın, yine 60 ödersin → 10. Fark yok.

v = 70 için tüm b, B ∈ {0, …, 120} çiftleri denendiğinde hiçbir b dürüst teklifi geçemiyor.

Adım Adım: Vickrey'de Dürüstlüğün İspatı



Strateji	B (rakip)	Sonuç	Fayda	Dürüste göre
Dürüst b = 70	60	kazanır, 60 öder	**10**	—
Dürüst b = 70	80	kaybeder	**0**	—
Fazla b = 90	60	kazanır, 60 öder	**10**	aynı
Fazla b = 90	80	kazanır, 80 öder	**−10**	**daha kötü**

b = 90, B = 80 (v ile b arasında): kazanırsın ama **80 ödersin** → **−10**. Dürüst olsan 0 alırdın. Zarar!

v = 70 için tüm b, B ∈ {0, …, 120} çiftleri denendiğinde hiçbir b dürüst teklifi geçemiyor.

Adım Adım: Vickrey'de Dürüstlüğün İspatı



Strateji	B (rakip)	Sonuç	Fayda	Dürüste göre
Dürüst b = 70	60	kazanır, 60 öder	**10**	—
Dürüst b = 70	80	kaybeder	**0**	—
Fazla b = 90	60	kazanır, 60 öder	**10**	aynı
Fazla b = 90	80	kazanır, 80 öder	**−10**	**daha kötü**
Fazla b = 90	95	kaybeder	**0**	aynı

b = 90, B = 95 (> b): kaybedersin → 0, dürüstle aynı.

v = 70 için tüm b, B ∈ {0, …, 120} çiftleri denendiğinde hiçbir b dürüst teklifi geçemiyor.

Adım Adım: Vickrey'de Dürüstlüğün İspatı

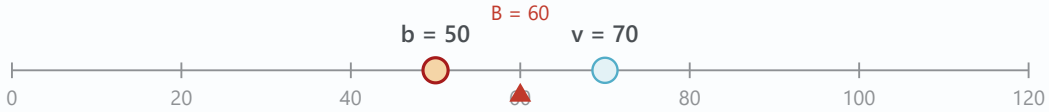


Strateji	B (rakip)	Sonuç	Fayda	Dürüste göre
Dürüst b = 70	60	kazanır, 60 öder	**10**	—
Dürüst b = 70	80	kaybeder	**0**	—
Fazla b = 90	60	kazanır, 60 öder	**10**	aynı
Fazla b = 90	80	kazanır, 80 öder	**−10**	**daha kötü**
Fazla b = 90	95	kaybeder	**0**	aynı
Eksik b = 50	40	kazanır, 40 öder	**30**	aynı

Eksik teklif b = 50, B = 40 (< b): kazanırsın, 40 ödersin → 30. Dürüstle aynı (ödemeni b değil B belirler).

v = 70 için tüm b, B ∈ {0, …, 120} çiftleri denendiğinde hiçbir b dürüst teklifi geçemiyor.

Adım Adım: Vickrey'de Dürüstlüğün İspatı



Strateji	B (rakip)	Sonuç	Fayda	Dürüste göre
Dürüst b = 70	60	kazanır, 60 öder	**10**	—
Dürüst b = 70	80	kaybeder	**0**	—
Fazla b = 90	60	kazanır, 60 öder	**10**	aynı
Fazla b = 90	80	kazanır, 80 öder	**−10**	**daha kötü**
Fazla b = 90	95	kaybeder	**0**	aynı
Eksik b = 50	40	kazanır, 40 öder	**30**	aynı
Eksik b = 50	60	kaybeder	**0**	**daha kötü**

b = 50, B = 60 (b ile v arasında): **kaybedersin** → **0**; dürüst olsan 70 − 60 = 10 kazanırdın. Fırsat kaçtı!

v = 70 için tüm b, B ∈ {0, …, 120} çiftleri denendiğinde hiçbir b dürüst teklifi geçemiyor.

Adım Adım: Vickrey'de Dürüstlüğün İspatı



Strateji	B (rakip)	Sonuç	Fayda	Dürüste göre
Dürüst b = 70	60	kazanır, 60 öder	**10**	—
Dürüst b = 70	80	kaybeder	**0**	—
Fazla b = 90	60	kazanır, 60 öder	**10**	aynı
Fazla b = 90	80	kazanır, 80 öder	**−10**	**daha kötü**
Fazla b = 90	95	kaybeder	**0**	aynı
Eksik b = 50	40	kazanır, 40 öder	**30**	aynı
Eksik b = 50	60	kaybeder	**0**	**daha kötü**
Eksik b = 50	80	kaybeder	**0**	aynı

b = 50, B = 80: kaybedersin, dürüstle aynı. Sonuç: sapma hiçbir durumda daha iyi değil, bazı durumlarda kötü → **b = v zayıf baskın**.

v = 70 için tüm b, B ∈ {0, …, 120} çiftleri denendiğinde hiçbir b dürüst teklifi geçemiyor.

Oyuna Dönüş: Vickrey Açık Artırması

Kuramın cevabı: değerinizi yazın

Açılıştaki zarflar: siz 70, rakipler 60 ve 45. Dürüst teklifle kazanır, 60 ödersiniz: kazanç 10. 65 ya da 100 yazsanız da aynı 60'ı ödersiniz; 55 yazarsanız nesne 60'a gider ve 10'luk kazancı kaçırırsınız.

Teklifiniz yalnızca kazanıp kazanmayacağınızı belirler, fiyatı rakiplerin en yüksek teklifi belirler: **dürüst teklif zayıf baskındır**. Aynı 70, birinci fiyatta kazansanız bile 70 − 70 = 0 kazanç bırakır; orada teklifi değerin altına çekmek gerekir.

Gerçekte

Laboratuvar deneylerinde ikinci fiyat kapalı zarf açık artırmasında değerin üstünde teklif vermek yaygındır ve tekrarla da kolay kaybolmaz; aynı deneklerin yükselen fiyatlı İngiliz açık artırmasında dengeye çok daha hızlı yaklaştığı görülmüştür (Kagel, Harstad ve Levin, 1987).

Gündelik hayatta: eBay'in otomatik (vekil) teklif sisteminde kazanan, ikinci en yüksek teklifi bir artış miktarı kadar geçen fiyatı öder; bu, ikinci fiyat kuralına çok yakındır.

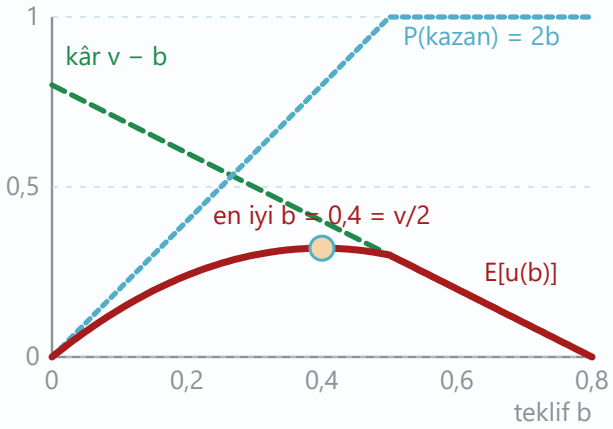
Teklif değerden yukarı ya da aşağı çekildiğinde kazanç hiçbir B için artmaz; yalnızca bazı B değerlerinde azalır.

Birinci Fiyatta Teklif Kırpma (Bid Shading)

- Birinci fiyatta dürüst teklif anlamsızdır: kazansan bile v − v = 0 fayda.
- Teklifi düşürmek kazanınca kârı artırır, ama kazanma olasılığını azaltır:

$$E[u(b)] = (v - b) \cdot P(b \text{ ile kazanmak})$$

- Denge teklifi bu ödünleşimi dengeler; diğerlerinin nasıl teklif verdiğine bağlıdır → **Bayes–Nash dengesi**.
- Hollanda açık artırması stratejik olarak aynıdır: “fiyat kaça inince kabul ederim?” = kırpılmış teklif.



İki teklifçi, v = 0,8; rakip değerinin yarısını teklif ediyor.

Adım Adım: Denge Teklifi b = (n − 1)v / n

1. Diğer n − 1 teklifçi $b_j = \beta \cdot v_j$ teklif etsin ($v_j \sim U[0, 1]$).

Varsayım: rakipler değerlerinin sabit bir oranını teklif ediyor.

n = 2, 3, 5, 10 ve v = 0,3; 0,6; 0,8 için E[u(b)] ızgarada (10.001 nokta) maksimize edilince en iyi teklif (n−1)v/n ile çakışıyor.

Adım Adım: Denge Teklifi b = (n − 1)v / n

1. Diğer n − 1 teklifçi $b_j = \beta \cdot v_j$ teklif etsin ($v_j \sim U[0, 1]$).

2. $P(\text{kazan}) = P(\beta\, v_j < b \text{ her } j \text{ için}) = (b/\beta)^{n-1}$

Her rakip için $P(v_j < b/\beta) = b/\beta$; bağımsızlıktan çarpılır.

n = 2, 3, 5, 10 ve v = 0,3; 0,6; 0,8 için E[u(b)] ızgarada (10.001 nokta) maksimize edilince en iyi teklif (n−1)v/n ile çakışıyor.

Adım Adım: Denge Teklifi b = (n − 1)v / n

1. Diğer n − 1 teklifçi $b_j = \beta \cdot v_j$ teklif etsin ($v_j \sim U[0, 1]$).
2. $P(\text{kazan}) = P(\beta\, v_j < b \text{ her } j \text{ için}) = (b/\beta)^{n-1}$
3. $E[u(b)] = (v - b) \cdot (b/\beta)^{n-1}$

Beklenen fayda: kazanınca elde edilen kâr × kazanma olasılığı.

n = 2, 3, 5, 10 ve v = 0,3; 0,6; 0,8 için E[u(b)] ızgarada (10.001 nokta) maksimize edilince en iyi teklif (n−1)v/n ile çakışıyor.

Adım Adım: Denge Teklifi b = (n − 1)v / n

1. Diğer n − 1 teklifçi $b_j = \beta \cdot v_j$ teklif etsin ($v_j \sim U[0, 1]$).
2. $P(\text{kazan}) = P(\beta\, v_j < b \text{ her } j \text{ için}) = (b/\beta)^{n-1}$
3. $E[u(b)] = (v - b) \cdot (b/\beta)^{n-1}$
4. $dE[u(b)]/db = [\, -b + (n - 1)(v - b)\,] \cdot b^{n-2}/\beta^{n-1} = 0$

$(v - b)$'nin türevi -1, b^{n-1}'in türevi $(n - 1)b^{n-2}$; çarpım kuralıyla toplanıp ortak çarpan b^{n-2} dışarı alınır. En büyük noktada türev sıfırdır.

n = 2, 3, 5, 10 ve v = 0,3; 0,6; 0,8 için E[u(b)] ızgarada (10.001 nokta) maksimize edilince en iyi teklif (n−1)v/n ile çakışıyor.

Adım Adım: Denge Teklifi b = (n − 1)v / n

1. Diğer n − 1 teklifçi $b_j = \beta \cdot v_j$ teklif etsin ($v_j \sim U[0, 1]$).
2. $P(\text{kazan}) = P(\beta\, v_j < b \text{ her } j \text{ için}) = (b/\beta)^{n-1}$
3. $E[u(b)] = (v - b) \cdot (b/\beta)^{n-1}$
4. $dE[u(b)]/db = [\, -b + (n - 1)(v - b)\,] \cdot b^{n-2}/\beta^{n-1} = 0$
5. $-b + (n - 1)(v - b) = 0 \Rightarrow$ **$b = (n - 1)\, v / n$**

Sadeleştirince çözüm β'dan bağımsız çıkar.

n = 2, 3, 5, 10 ve v = 0,3; 0,6; 0,8 için E[u(b)] ızgarada (10.001 nokta) maksimize edilince en iyi teklif (n−1)v/n ile çakışıyor.

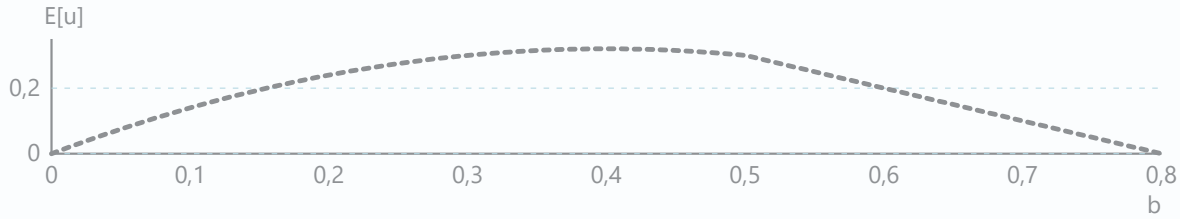
Adım Adım: Denge Teklifi b = (n − 1)v / n

1. Diğer n − 1 teklifçi $b_j = \beta \cdot v_j$ teklif etsin ($v_j \sim U[0, 1]$).
2. $P(\text{kazan}) = P(\beta\, v_j < b \text{ her } j \text{ için}) = (b/\beta)^{n-1}$
3. $E[u(b)] = (v - b) \cdot (b/\beta)^{n-1}$
4. $dE[u(b)]/db = [\, -b + (n - 1)(v - b)\,] \cdot b^{n-2}/\beta^{n-1} = 0$
5. $-b + (n - 1)(v - b) = 0 \Rightarrow$ **$b = (n - 1)\, v / n$**
6. Tutarlılık: $\beta = (n - 1)/n$ ✓ $\quad n = 2 \Rightarrow b = v/2$

Benim en iyi tepkim de aynı biçimde → simetrik Bayes–Nash dengesi. İki kişide değerin yarısı.

n = 2, 3, 5, 10 ve v = 0,3; 0,6; 0,8 için E[u(b)] ızgarada (10.001 nokta) maksimize edilince en iyi teklif (n−1)v/n ile çakışıyor.

Kod İzleme: `en_iyi_teklif(0.8, 2)`



Rakipler b = v/2 teklif ediyor (β = 0,5). Izgara: b = 0, 0,1, …, 0,8.

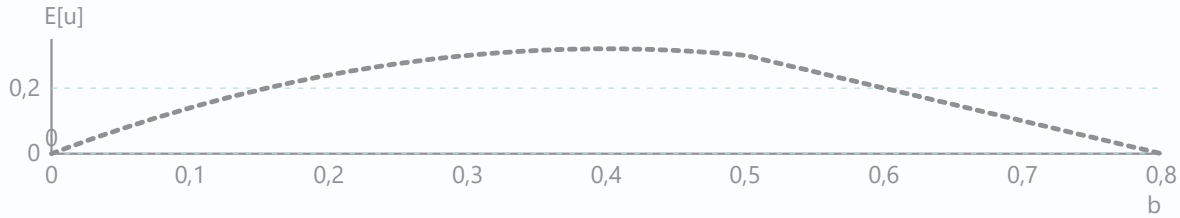
Adım 1 / 11

```
1  def en_iyi_teklif(v, n, adim=0.1):
▶     beta = (n - 1) / n        # rakipler beta*v
▶     en, b_en = -1, 0
▶     b = 0.0
5     while b <= v:
6       p = min(1, b / beta) ** (n - 1)  # P(kazan)
7       eu = (v - b) * p
8       if eu > en:
9         en, b_en = eu, b
10      b = round(b + adim, 10)
11    return b_en, en
```

DEĞİŞKENLER

`v = 0,8` `n = 2` `beta = 0,5` `en = -1` `b_en = 0` `b = 0`

Kod İzleme: `en_iyi_teklif(0.8, 2)`



b = 0: P(kazan) = 0, E[u] = (0,8 − 0)·0 = **0** → yeni en iyi.

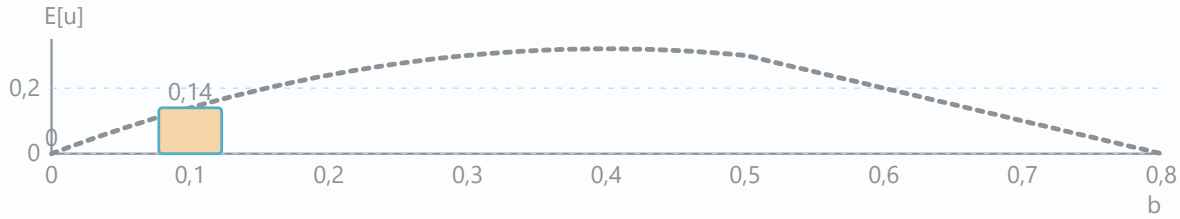
Adım 2 / 11

```
1  def en_iyi_teklif(v, n, adim=0.1):
2    beta = (n - 1) / n        # rakipler beta*v
3    en, b_en = -1, 0
4    b = 0.0
5    while b <= v:
▶      p = min(1, b / beta) ** (n - 1)  # P(kazan)
▶      eu = (v - b) * p
8      if eu > en:
▶        en, b_en = eu, b
10     b = round(b + adim, 10)
11   return b_en, en
```

DEĞİŞKENLER

`v = 0,8` `n = 2` `beta = 0,5` `b = 0` `p = 0` `eu = 0` `en = 0` `b_en = 0`

Kod İzleme: `en_iyi_teklif(0.8, 2)`



b = 0,1: P(kazan) = 0,2, E[u] = (0,8 − 0,1)·0,2 = **0,14** → yeni en iyi.

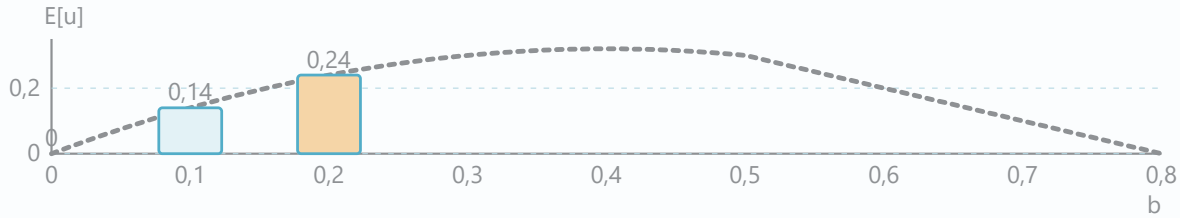
Adım 3 / 11

```
1  def en_iyi_teklif(v, n, adim=0.1):
2    beta = (n - 1) / n        # rakipler beta*v
3    en, b_en = -1, 0
4    b = 0.0
5    while b <= v:
▶      p = min(1, b / beta) ** (n - 1)  # P(kazan)
▶      eu = (v - b) * p
8      if eu > en:
▶        en, b_en = eu, b
10     b = round(b + adim, 10)
11   return b_en, en
```

DEĞİŞKENLER

`v = 0,8` `n = 2` `beta = 0,5` `b = 0,1`
`p = 0,2` `eu = 0,14` `en = 0,14`
`b_en = 0,1`

Kod İzleme: `en_iyi_teklif(0.8, 2)`



b = 0,2: P(kazan) = 0,4, E[u] = (0,8 − 0,2)·0,4 = **0,24** → yeni en iyi.

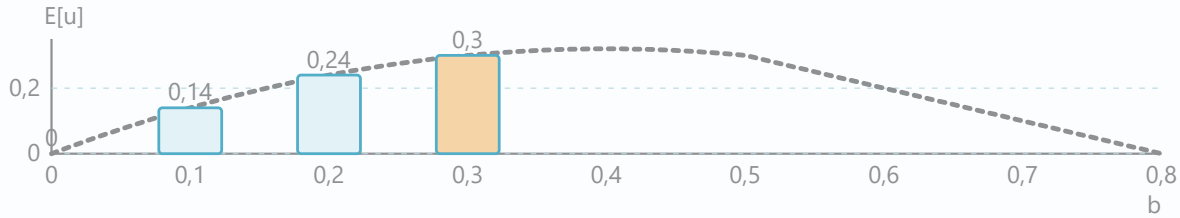
Adım 4 / 11

```
1  def en_iyi_teklif(v, n, adim=0.1):
2    beta = (n - 1) / n        # rakipler beta*v
3    en, b_en = -1, 0
4    b = 0.0
5    while b <= v:
▶      p = min(1, b / beta) ** (n - 1)  # P(kazan)
▶      eu = (v - b) * p
8      if eu > en:
▶        en, b_en = eu, b
10     b = round(b + adim, 10)
11   return b_en, en
```

DEĞİŞKENLER

`v = 0,8` `n = 2` `beta = 0,5` `b = 0,2` `p = 0,4` `eu = 0,24` `en = 0,24` `b_en = 0,2`

Kod İzleme: `en_iyi_teklif(0.8, 2)`



b = 0,3: P(kazan) = 0,6, E[u] = (0,8 − 0,3)·0,6 = **0,3** → yeni en iyi.

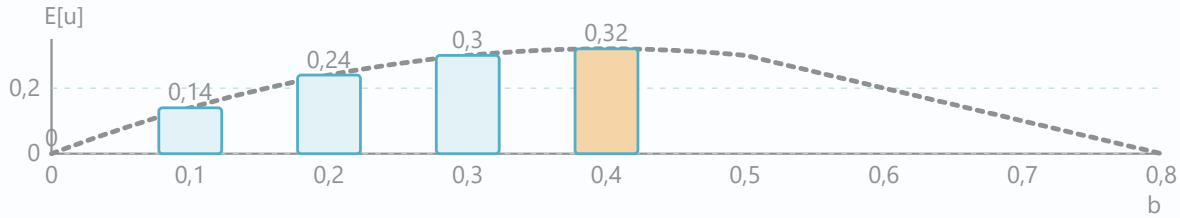
Adım 5 / 11

```
1  def en_iyi_teklif(v, n, adim=0.1):
2    beta = (n - 1) / n        # rakipler beta*v
3    en, b_en = -1, 0
4    b = 0.0
5    while b <= v:
▶      p = min(1, b / beta) ** (n - 1)  # P(kazan)
▶      eu = (v - b) * p
8      if eu > en:
▶        en, b_en = eu, b
10     b = round(b + adim, 10)
11   return b_en, en
```

DEĞİŞKENLER

`v = 0,8` `n = 2` `beta = 0,5` `b = 0,3`
`p = 0,6` `eu = 0,3` `en = 0,3`
`b_en = 0,3`

Kod İzleme: `en_iyi_teklif(0.8, 2)`



b = 0,4: P(kazan) = 0,8, E[u] = (0,8 − 0,4)·0,8 = **0,32** → yeni en iyi.

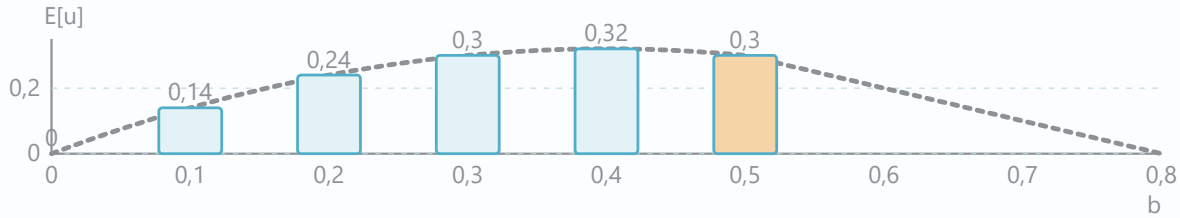
Adım 6 / 11

```
1  def en_iyi_teklif(v, n, adim=0.1):
2    beta = (n - 1) / n        # rakipler beta*v
3    en, b_en = -1, 0
4    b = 0.0
5    while b <= v:
▶      p = min(1, b / beta) ** (n - 1)  # P(kazan)
▶      eu = (v - b) * p
8      if eu > en:
▶        en, b_en = eu, b
10     b = round(b + adim, 10)
11   return b_en, en
```

DEĞİŞKENLER

`v = 0,8` `n = 2` `beta = 0,5` `b = 0,4`
`p = 0,8` `eu = 0,32` `en = 0,32`
`b_en = 0,4`

Kod İzleme: `en_iyi_teklif(0.8, 2)`



b = 0,5: P(kazan) = 1, E[u] = (0,8 − 0,5)·1 = **0,3** < 0,32: en iyi değişmez.

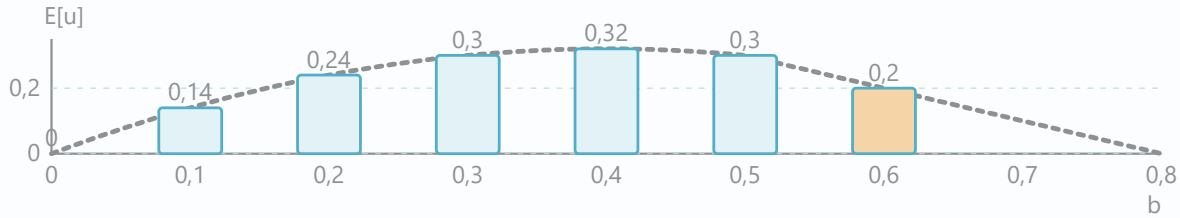
Adım 7 / 11

```
1  def en_iyi_teklif(v, n, adim=0.1):
2    beta = (n - 1) / n        # rakipler beta*v
3    en, b_en = -1, 0
4    b = 0.0
5    while b <= v:
▶      p = min(1, b / beta) ** (n - 1)  # P(kazan)
▶      eu = (v - b) * p
8      if eu > en:
9        en, b_en = eu, b
10     b = round(b + adim, 10)
11   return b_en, en
```

DEĞİŞKENLER

`v = 0,8` `n = 2` `beta = 0,5` `b = 0,5`
`p = 1` `eu = 0,3` `en = 0,32` `b_en = 0,4`

Kod İzleme: `en_iyi_teklif(0.8, 2)`



b = 0,6: P(kazan) = 1, E[u] = (0,8 − 0,6)·1 = **0,2** < 0,32: en iyi değişmez.

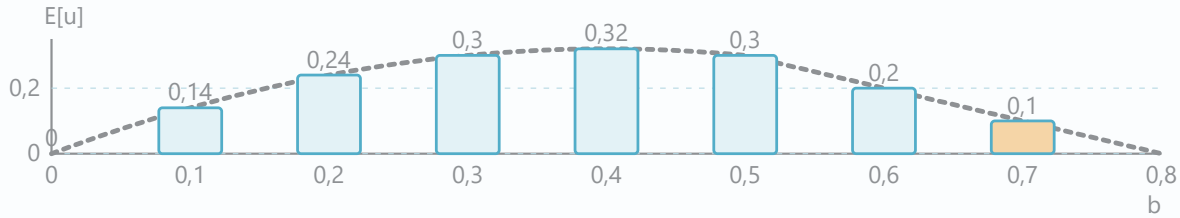

```
1  def en_iyi_teklif(v, n, adim=0.1):
2    beta = (n - 1) / n        # rakipler beta*v
3    en, b_en = -1, 0
4    b = 0.0
5    while b <= v:
▶      p = min(1, b / beta) ** (n - 1)  # P(kazan)
▶      eu = (v - b) * p
8      if eu > en:
9        en, b_en = eu, b
10     b = round(b + adim, 10)
11   return b_en, en
```

DEĞİŞKENLER

`v = 0,8` `n = 2` `beta = 0,5` `b = 0,6`

`p = 1` `eu = 0,2` `en = 0,32` `b_en = 0,4`

Kod İzleme: `en_iyi_teklif(0.8, 2)`



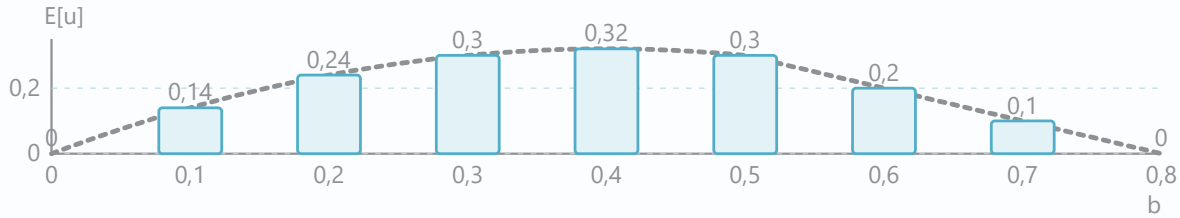
b = 0,7: P(kazan) = 1, E[u] = (0,8 − 0,7)·1 = **0,1** < 0,32: en iyi değişmez.


```
1  def en_iyi_teklif(v, n, adim=0.1):
2    beta = (n - 1) / n        # rakipler beta*v
3    en, b_en = -1, 0
4    b = 0.0
5    while b <= v:
▶      p = min(1, b / beta) ** (n - 1)  # P(kazan)
▶      eu = (v - b) * p
8      if eu > en:
9        en, b_en = eu, b
10     b = round(b + adim, 10)
11   return b_en, en
```

DEĞİŞKENLER

`v = 0,8` `n = 2` `beta = 0,5` `b = 0,7`
`p = 1` `eu = 0,1` `en = 0,32` `b_en = 0,4`

Kod İzleme: `en_iyi_teklif(0.8, 2)`



b = 0,8: P(kazan) = 1, E[u] = (0,8 − 0,8)·1 = **0** < 0,32: en iyi değişmez.

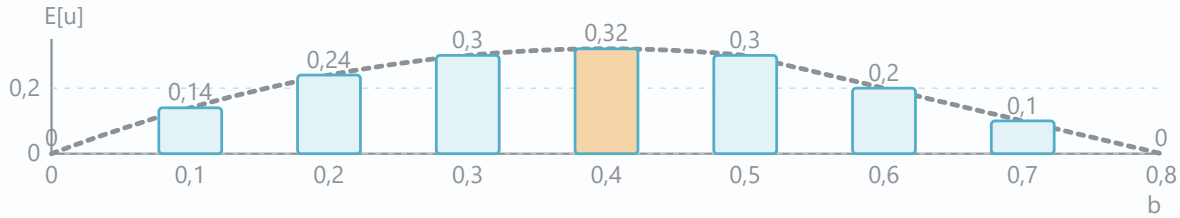
Adım 10 / 11

```
1  def en_iyi_teklif(v, n, adim=0.1):
2    beta = (n - 1) / n        # rakipler beta*v
3    en, b_en = -1, 0
4    b = 0.0
5    while b <= v:
▶      p = min(1, b / beta) ** (n - 1)  # P(kazan)
▶      eu = (v - b) * p
8      if eu > en:
9        en, b_en = eu, b
10     b = round(b + adim, 10)
11   return b_en, en
```

DEĞİŞKENLER

`v = 0,8` `n = 2` `beta = 0,5` `b = 0,8`
`p = 1` `eu = 0` `en = 0,32` `b_en = 0,4`

Kod İzleme: `en_iyi_teklif(0.8, 2)`



Sonuç: **b = 0,4 = v/2**, E[u] = 0,32. Düşük teklif kazanmayı, yüksek teklif kârı öldürür.

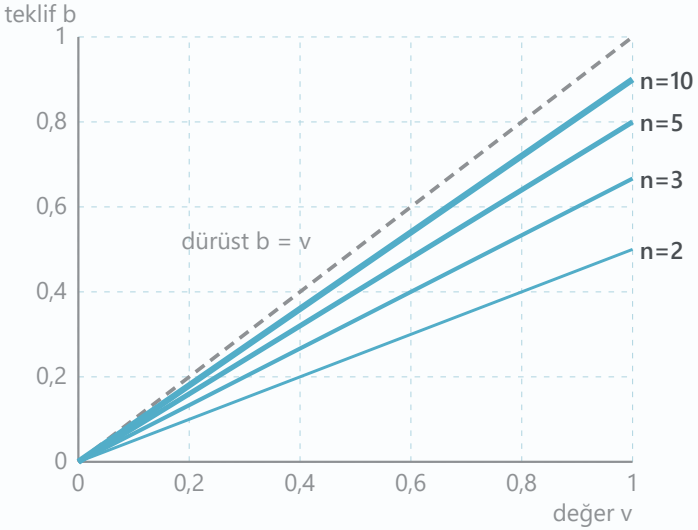
Adım 11 / 11

```
1  def en_iyi_teklif(v, n, adim=0.1):
2    beta = (n - 1) / n        # rakipler beta*v
3    en, b_en = -1, 0
4    b = 0.0
5    while b <= v:
6      p = min(1, b / beta) ** (n - 1)  # P(kazan)
7      eu = (v - b) * p
8      if eu > en:
9        en, b_en = eu, b
10     b = round(b + adim, 10)
▶    return b_en, en
```

DEĞİŞKENLER

`v = 0,8` `n = 2` `b_en = 0,4` `en = 0,32`

Rekabet Arttıkça Kırpma Azalır



n	b(v)	v = 0,8 → b	kırpma
2	1/2·v	0,4	0,4
3	2/3·v	0,53	0,27
5	4/5·v	0,64	0,16
10	9/10·v	0,72	0,08

Çok teklifçi varken kazanmak için değere yakın teklif gerekir; n → ∞ iken b → v.

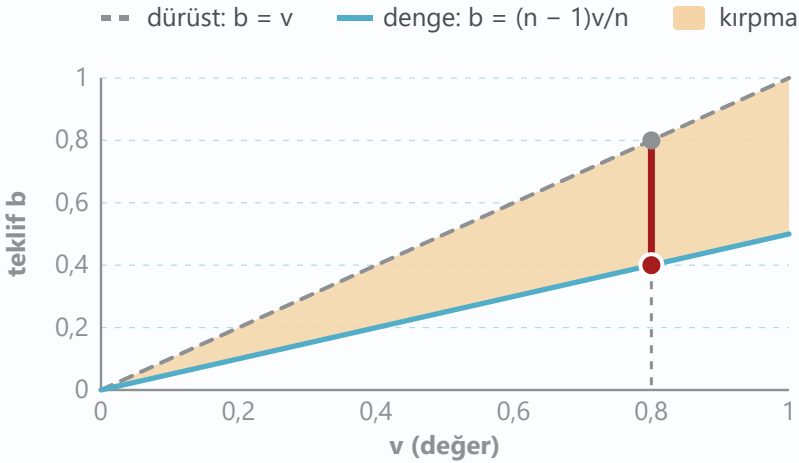
KAPALI ZARF STRATEJİLERİ

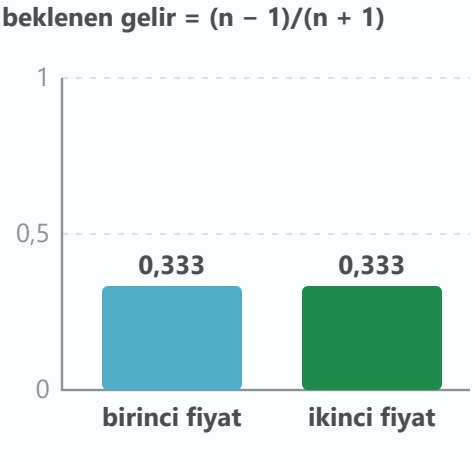
Teklifçi Sayısı Arttıkça Teklif Kırpma

Denge teklifi **b(v) = (n − 1)v / n** · kırpma = v − b(v) = v/n · beklenen gelir = **(n − 1)/(n + 1)**

n = 2

v = 0,8: teklif b = **0,4** | kırpma v/n = **0,4** | beklenen gelir = **1/3 ≈ 0,333** (iki kuralda aynı)





Teklifçi sayısı arttıkça rekabet kırpmayı azaltır ve teklif değere yaklaşır; beklenen gelir birinci ve ikinci fiyat kuralında aynıdır (gelir eşdeğerliği) ve n ile birlikte artar.

Adım Adım: İki Teklifçide Beklenen Gelir = 1/3

Kural	Satıcının geliri	Beklenen gelir

İki teklifçi, değerler $v_1, v_2 \sim U[0, 1]$ bağımsız. Soru: hangi kural satıcıya daha çok kazandırır?

Adım Adım: İki Teklifçide Beklenen Gelir = 1/3

Kural	Satıcının geliri	Beklenen gelir
İkinci fiyat	$\min(v_1, v_2)$	$E[\min] = \int_0^1 (1 - t)^2 \, dt = 1/3$

İkinci fiyat: herkes dürüst, satıcı küçük değeri alır. İki değerin küçüğü t'yi aşıyorsa ikisi de aşar: $P(\min > t) = (1 - t)\cdot(1 - t)$. $E[X] = \int_0^1 P(X > t) \, dt$ ile beklenti **1/3**.

Adım Adım: İki Teklifçide Beklenen Gelir = 1/3

Kural	Satıcının geliri	Beklenen gelir
İkinci fiyat	$\min(v_1, v_2)$	$E[\min] = \int_0^1 (1 - t)^2\, dt = 1/3$
Birinci fiyat	$\max(v_1, v_2) / 2$	$E[\max] = \int_0^1 (1 - t^2)\, dt = 2/3 \rightarrow 1/3$

Birinci fiyat: denge $b = v/2$, kazanan büyük değerli. $P(\max < t) = t^2$, yani $P(\max > t) = 1 - t^2$ $\rightarrow E[\max] = 2/3$, gelir **1/3**.

Adım Adım: İki Teklifçide Beklenen Gelir = 1/3

Kural	Satıcının geliri	Beklenen gelir
İkinci fiyat	$\min(v_1, v_2)$	$E[\min] = \int_0^1 (1 - t)^2\, dt = 1/3$
Birinci fiyat	$\max(v_1, v_2) / 2$	$E[\max] = \int_0^1 (1 - t^2)\, dt = 2/3 \rightarrow 1/3$
İngiliz	≈ ikinci fiyat	1/3

İngiliz açık artırması özel değerlerde ikinci fiyatla aynı sonucu verir → 1/3.

Adım Adım: İki Teklifçide Beklenen Gelir = 1/3

Kural	Satıcının geliri	Beklenen gelir
İkinci fiyat	$\min(v_1, v_2)$	$E[\min] = \int_0^1 (1 - t)^2\, dt = 1/3$
Birinci fiyat	$\max(v_1, v_2) / 2$	$E[\max] = \int_0^1 (1 - t^2)\, dt = 2/3 \rightarrow 1/3$
İngiliz	≈ ikinci fiyat	1/3
Hollanda	≡ birinci fiyat	1/3

Hollanda açık artırması birinci fiyatla stratejik olarak özdeş → 1/3. **Dört kural, aynı beklenen gelir.**

Adım Adım: İki Teklifçide Beklenen Gelir = 1/3

Kural	Satıcının geliri	Beklenen gelir
İkinci fiyat	$\min(v_1, v_2)$	$E[\min] = \int_0^1 (1 - t)^2\, dt = 1/3$
Birinci fiyat	$\max(v_1, v_2) / 2$	$E[\max] = \int_0^1 (1 - t^2)\, dt = 2/3 \rightarrow 1/3$
İngiliz	≈ ikinci fiyat	1/3
Hollanda	≡ birinci fiyat	1/3
Simülasyon	200 bin çekiliş	İkinci fiyat 0,333 · birinci fiyat 0,333

Monte Carlo simülasyonu sonucu doğruluyor: tek tek açık artırmalarda gelir farklı, **ortalamada eşit**.

Gelir Eşdeğerliği Teoremi (Revenue Equivalence)

- Bağımsız özel değerler, risk nötr simetrik katılımcılar ve aynı tahsis kuralı (en yüksek değerli kazanır) altında standart açık artırmalar **aynı beklenen geliri** sağlar (Vickrey 1961; Myerson 1981; Riley ve Samuelson 1981).
- **Neden?** Myerson ödeme formülü $p(v) = v \cdot x(v) - \int_0^v x(z)\,dz$, beklenen ödemeyi yalnızca tahsis kuralı $x(\cdot)$ ve en düşük tipin ödemesi (0) ile belirler: aynı $x(\cdot) \rightarrow$ aynı $p(\cdot) \rightarrow$ aynı gelir. $n = 2$, $U[0, 1]$: $x(v) = v$, $p(v) = v^2 - v^2/2 = v^2/2$; birinci fiyatta: teklif $v/2$ × kazanma olasılığı v; ikinci fiyatta: kazanma olasılığı v × kazanınca ödenen ortalama rakip değeri $v/2$. Üçü aynı.
- Riskten kaçınma, bütçe kısıtları, ortak değerler veya asimetrik bilgi bu sonucu değiştirebilir.

n	Kuram: (n−1)/(n+1)	İkinci fiyat (simülasyon)	Birinci fiyat (simülasyon)
2	1/3 = 0,33	0,332	0,333
3	1/2 = 0,5	0,500	0,500
4	3/5 = 0,6	0,600	0,600
5	2/3 = 0,67	0,668	0,667

Bu nedenle açık artırma biçimi yalnızca gelirle değil, sadelik, şeffaflık ve katılım teşvikiyle birlikte seçilir.

Varsayımı Değiştir: Ya Teklifçiler Riskten Kaçınırsa?

İki teklifçi, değerler U[0, 1]. Kazananın kârı x = v − fiyat, faydası **$u(x) = x^{\rho}$** ($0 < \rho \leq 1$). $\rho = 1$: risk nötr; $\rho < 1$: **riskten kaçınan** ($\rho = ½$ ise $u(x) = \sqrt{x}$).

Teklifçiler riskten kaçınırsa **hangi kural satıcıya daha çok kazandırır?**

Varsayımı Değiştir: Ya Teklifçiler Riskten Kaçınırsa?

İki teklifçi, değerler U[0, 1]. Kazananın kârı x = v − fiyat, faydası $\mathbf{u(x) = x^{\rho}}$ ($0 < \rho \leq 1$). ρ = 1: risk nötr; ρ < 1: **riskten kaçınan** (ρ = ½ ise u(x) = √x).

Teklifçiler riskten kaçınırsa **hangi kural satıcıya daha çok kazandırır?**

İkinci fiyat: değişmez

Fiyatı teklif değil, rakibin teklifi belirler. Dürüst teklif hâlâ zayıf baskın. Gelir: E[min] = 1/3.

Birinci fiyat: teklif yükselir

Kaybetme riskini azaltmak için teklif artar: $\mathbf{b(v) = v / (1 + \rho)}$. ρ = 1: v/2; ρ = ½: 2v/3.

Varsayımı Değiştir: Ya Teklifçiler Riskten Kaçınırsa?

İki teklifçi, değerler U[0, 1]. Kazananın kârı x = v − fiyat, faydası **$u(x) = x^{\rho}$** ($0 < \rho \leq 1$). $\rho = 1$: risk nötr; $\rho < 1$: **riskten kaçınan** ($\rho = ½$ ise $u(x) = \sqrt{x}$).

Teklifçiler riskten kaçınırsa **hangi kural satıcıya daha çok kazandırır?**

İkinci fiyat: değişmez
Fiyatı teklif değil, rakibin teklifi belirler. Dürüst teklif hâlâ zayıf baskın. Gelir: E[min] = 1/3.

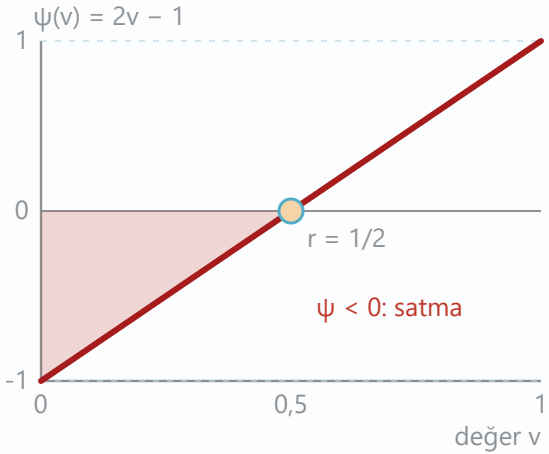
Birinci fiyat: teklif yükselir
Kaybetme riskini azaltmak için teklif artar: **$b(v) = v / (1 + \rho)$**. $\rho = 1$: v/2; $\rho = ½$: 2v/3.

Beklenen gelir	**$\rho = 1$**	**$\rho = ½$**
İkinci fiyat	1/3 ≈ 0,333	1/3 ≈ 0,333
Birinci fiyat (E[max] = 2/3)	(2/3)·(1/2) = 1/3	**(2/3)·(2/3) = 4/9 ≈ 0,444**

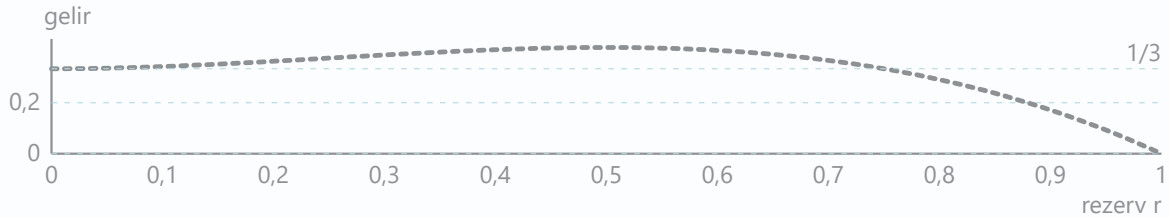
Riskten kaçınma gelir eşdeğerliğini bozar: birinci fiyat daha çok kazandırır.

Gelir Maksimizasyonu: Myerson ve Rezerv Fiyatı

- Satıcı verimliliği değil **beklenen geliri** maksimize etmek isterse? (Myerson, 1981)
- **Sanal değer** $\psi(v) = v - (1 - F(v))/f(v)$. $F(v)$: değerin v'den küçük olma olasılığı; $f(v)$: yoğunluk. $U[0, 1]$'de $F(v) = v$, $f(v) = 1 \Rightarrow \psi(v) = 2v - 1$.
- Beklenen gelir, kazananın sanal değerinin beklentisine eşittir. En iyi mekanizma: nesneyi en büyük ψ'ye ver, $\psi < 0$ ise satma.
- ψ artan ve teklifçiler simetrikse bu, **rezerv fiyatlı ikinci fiyat** açık artırmasıdır: $\psi(r) = 0 \Rightarrow 2r - 1 = 0 \Rightarrow r = 1/2$ (n'den bağımsız).
- İki teklifçide rezerv r ile beklenen gelir $1/3 + r^2 - 4r^3/3$; $r = 1/2$'de $1/3 \rightarrow$ **5/12 ≈ 0,417**. Bedeli: bazen hiç satılmaz (verimlilik kaybı).



Kod İzleme: `en_iyi_rezerv()`



Rezerv fiyatı r = 0, 0,1, …, 1 ızgarasında denenecek; `gelir(r)` 100 × 100 değer ızgarasında ortalama ödemeyi hesaplar.

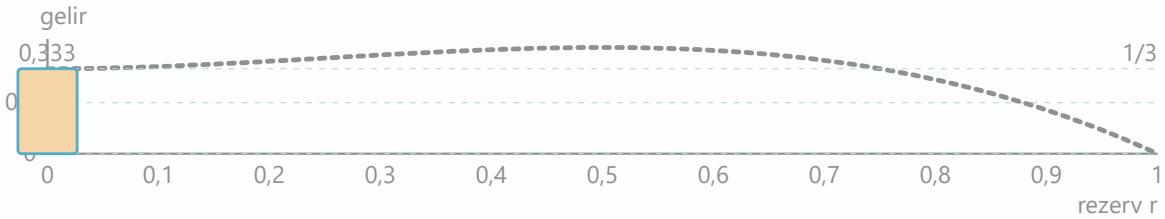
Adım 1 / 13

```
1  def gelir(r, m=100):      # 2 teklifçi, U[0,1]
2    v = [(i + .5) / m for i in range(m)]
3    top = sum(max(min(a, c), r) for a in v
4              for c in v if max(a, c) >= r)
5    return top / m**2       # ortalama ödeme
6  def en_iyi_rezerv(adim=0.1):
▶    en, r_en = -1, None
8    for k in range(round(1/adim) + 1):
9      R = gelir(k * adim)
10     if R > en: en, r_en = R, k * adim
11   return r_en, en
```

DEĞİŞKENLER

`en = -1` `r_en = None`

Kod İzleme: `en_iyi_rezerv()`



r = 0: gelir(r) = **0,333** → yeni en iyi. Rezervsiz = ikinci fiyat: 1/3.

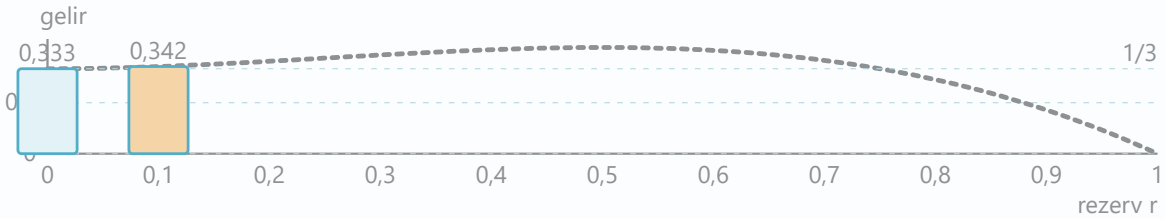
Adım 2 / 13

```
1  def gelir(r, m=100):      # 2 teklifçi, U[0,1]
2    v = [(i + .5) / m for i in range(m)]
3    top = sum(max(min(a, c), r) for a in v
4              for c in v if max(a, c) >= r)
5    return top / m**2       # ortalama ödeme
6  def en_iyi_rezerv(adim=0.1):
7    en, r_en = -1, None
8    for k in range(round(1/adim) + 1):
▶      R = gelir(k * adim)
▶      if R > en: en, r_en = R, k * adim
11   return r_en, en
```

DEĞİŞKENLER

`k = 0` `R = 0,333` `en = 0,333` `r_en = 0`

Kod İzleme: `en_iyi_rezerv()`



r = 0,1: gelir(r) = **0,342** → yeni en iyi.

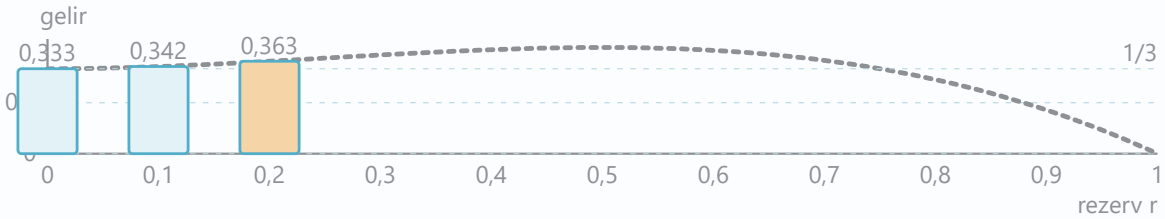
Adım 3 / 13

```
1  def gelir(r, m=100):       # 2 teklifçi, U[0,1]
2    v = [(i + .5) / m for i in range(m)]
3    top = sum(max(min(a, c), r) for a in v
4              for c in v if max(a, c) >= r)
5    return top / m**2          # ortalama ödeme
6  def en_iyi_rezerv(adim=0.1):
7    en, r_en = -1, None
8    for k in range(round(1/adim) + 1):
▶      R = gelir(k * adim)
▶      if R > en: en, r_en = R, k * adim
11   return r_en, en
```

DEĞİŞKENLER

`k = 1` `R = 0,342` `en = 0,342` `r_en = 0,1`

Kod İzleme: `en_iyi_rezerv()`



r = 0,2: gelir(r) = **0,363** → yeni en iyi.

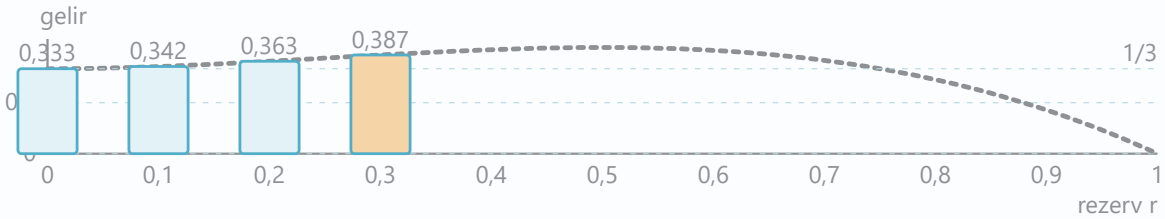
Adım 4 / 13

```
1  def gelir(r, m=100):      # 2 teklifçi, U[0,1]
2    v = [(i + .5) / m for i in range(m)]
3    top = sum(max(min(a, c), r) for a in v
4              for c in v if max(a, c) >= r)
5    return top / m**2       # ortalama ödeme
6  def en_iyi_rezerv(adim=0.1):
7    en, r_en = -1, None
8    for k in range(round(1/adim) + 1):
▶      R = gelir(k * adim)
▶      if R > en: en, r_en = R, k * adim
11   return r_en, en
```

DEĞİŞKENLER

`k = 2` `R = 0,363` `en = 0,363` `r_en = 0,2`

Kod İzleme: `en_iyi_rezerv()`



r = 0,3: gelir(r) = **0,387** → yeni en iyi.

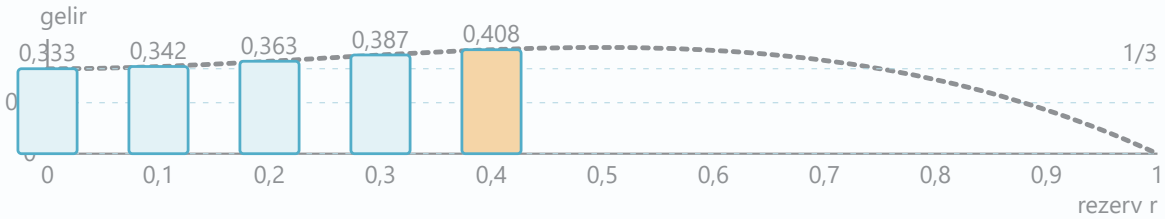
Adım 5 / 13

```
1  def gelir(r, m=100):       # 2 teklifçi, U[0,1]
2    v = [(i + .5) / m for i in range(m)]
3    top = sum(max(min(a, c), r) for a in v
4              for c in v if max(a, c) >= r)
5    return top / m**2          # ortalama ödeme
6  def en_iyi_rezerv(adim=0.1):
7    en, r_en = -1, None
8    for k in range(round(1/adim) + 1):
▶      R = gelir(k * adim)
▶      if R > en: en, r_en = R, k * adim
11   return r_en, en
```

DEĞİŞKENLER

`k = 3` `R = 0,387` `en = 0,387` `r_en = 0,3`

Kod İzleme: `en_iyi_rezerv()`



r = 0,4: gelir(r) = **0,408** → yeni en iyi.

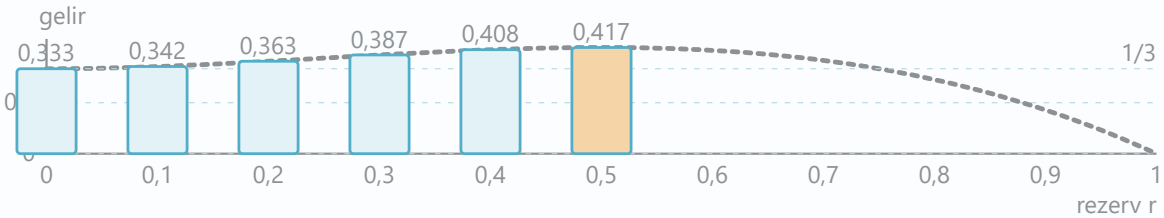
Adım 6 / 13

```
1  def gelir(r, m=100):       # 2 teklifçi, U[0,1]
2    v = [(i + .5) / m for i in range(m)]
3    top = sum(max(min(a, c), r) for a in v
4              for c in v if max(a, c) >= r)
5    return top / m**2          # ortalama ödeme
6  def en_iyi_rezerv(adim=0.1):
7    en, r_en = -1, None
8    for k in range(round(1/adim) + 1):
▶      R = gelir(k * adim)
▶      if R > en: en, r_en = R, k * adim
11   return r_en, en
```

DEĞİŞKENLER

`k = 4` `R = 0,408` `en = 0,408` `r_en = 0,4`

Kod İzleme: `en_iyi_rezerv()`



r = 0,5: gelir(r) = **0,417** → yeni en iyi.

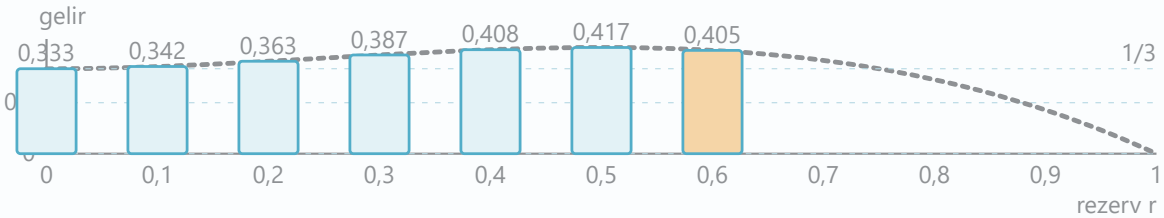
Adım 7 / 13

```
1  def gelir(r, m=100):       # 2 teklifçi, U[0,1]
2    v = [(i + .5) / m for i in range(m)]
3    top = sum(max(min(a, c), r) for a in v
4              for c in v if max(a, c) >= r)
5    return top / m**2          # ortalama ödeme
6  def en_iyi_rezerv(adim=0.1):
7    en, r_en = -1, None
8    for k in range(round(1/adim) + 1):
▶      R = gelir(k * adim)
▶      if R > en: en, r_en = R, k * adim
11   return r_en, en
```

DEĞİŞKENLER

`k = 5` `R = 0,417` `en = 0,417` `r_en = 0,5`

Kod İzleme: `en_iyi_rezerv()`



r = 0,6: gelir(r) = **0,405** < 0,417.

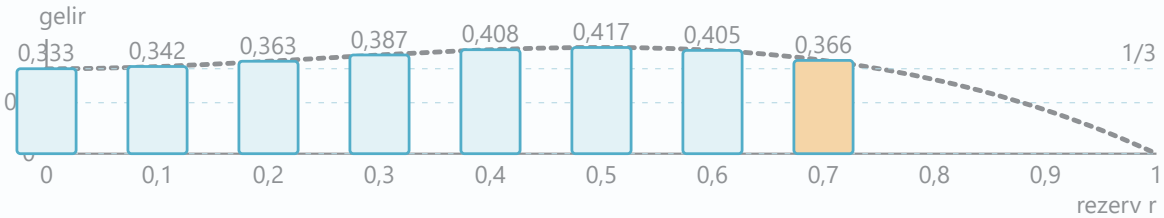
Adım 8 / 13

```
1  def gelir(r, m=100):       # 2 teklifçi, U[0,1]
2    v = [(i + .5) / m for i in range(m)]
3    top = sum(max(min(a, c), r) for a in v
4              for c in v if max(a, c) >= r)
5    return top / m**2          # ortalama ödeme
6  def en_iyi_rezerv(adim=0.1):
7    en, r_en = -1, None
8    for k in range(round(1/adim) + 1):
▶      R = gelir(k * adim)
▶      if R > en: en, r_en = R, k * adim
11   return r_en, en
```

DEĞİŞKENLER

`k = 6` `R = 0,405` `en = 0,417` `r_en = 0,5`

Kod İzleme: `en_iyi_rezerv()`



r = 0,7: gelir(r) = **0,366** < 0,417.

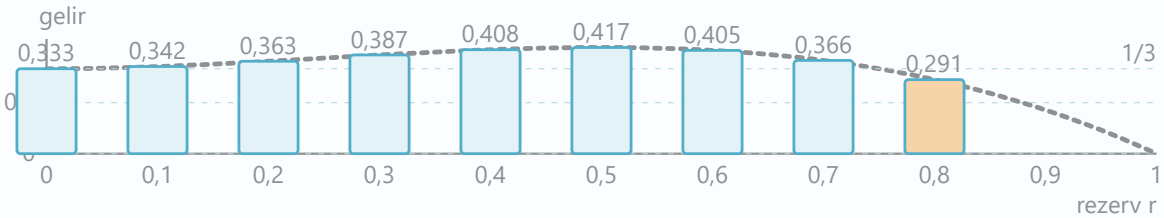
Adım 9 / 13

```
1  def gelir(r, m=100):       # 2 teklifçi, U[0,1]
2    v = [(i + .5) / m for i in range(m)]
3    top = sum(max(min(a, c), r) for a in v
4              for c in v if max(a, c) >= r)
5    return top / m**2          # ortalama ödeme
6  def en_iyi_rezerv(adim=0.1):
7    en, r_en = -1, None
8    for k in range(round(1/adim) + 1):
▶      R = gelir(k * adim)
▶      if R > en: en, r_en = R, k * adim
11   return r_en, en
```

DEĞİŞKENLER

`k = 7` `R = 0,366` `en = 0,417` `r_en = 0,5`

Kod İzleme: `en_iyi_rezerv()`



r = 0,8: gelir(r) = **0,291** < 0,417.

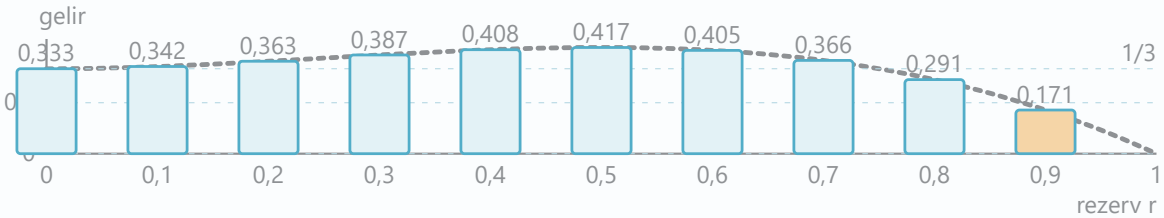
Adım 10 / 13

```
1  def gelir(r, m=100):      # 2 teklifçi, U[0,1]
2    v = [(i + .5) / m for i in range(m)]
3    top = sum(max(min(a, c), r) for a in v
4              for c in v if max(a, c) >= r)
5    return top / m**2       # ortalama ödeme
6  def en_iyi_rezerv(adim=0.1):
7    en, r_en = -1, None
8    for k in range(round(1/adim) + 1):
▶      R = gelir(k * adim)
▶      if R > en: en, r_en = R, k * adim
11   return r_en, en
```

DEĞİŞKENLER

`k = 8` `R = 0,291` `en = 0,417` `r_en = 0,5`

Kod İzleme: `en_iyi_rezerv()`



r = 0,9: gelir(r) = **0,171** < 0,417.

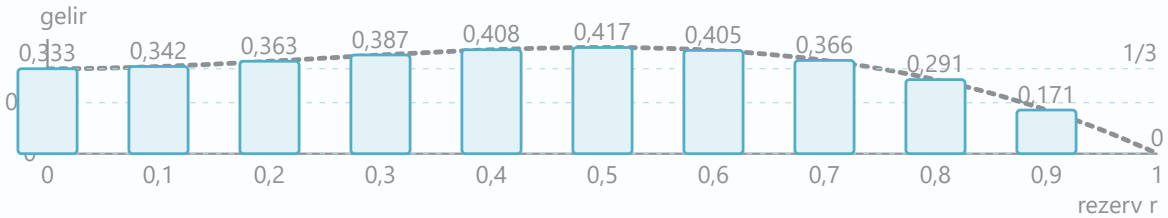
Adım 11 / 13

```
1  def gelir(r, m=100):      # 2 teklifçi, U[0,1]
2    v = [(i + .5) / m for i in range(m)]
3    top = sum(max(min(a, c), r) for a in v
4              for c in v if max(a, c) >= r)
5    return top / m**2       # ortalama ödeme
6  def en_iyi_rezerv(adim=0.1):
7    en, r_en = -1, None
8    for k in range(round(1/adim) + 1):
▶      R = gelir(k * adim)
▶      if R > en: en, r_en = R, k * adim
11   return r_en, en
```

DEĞİŞKENLER

`k = 9` `R = 0,171` `en = 0,417` `r_en = 0,5`

Kod İzleme: `en_iyi_rezerv()`



r = 1: gelir(r) = **0** < 0,417.

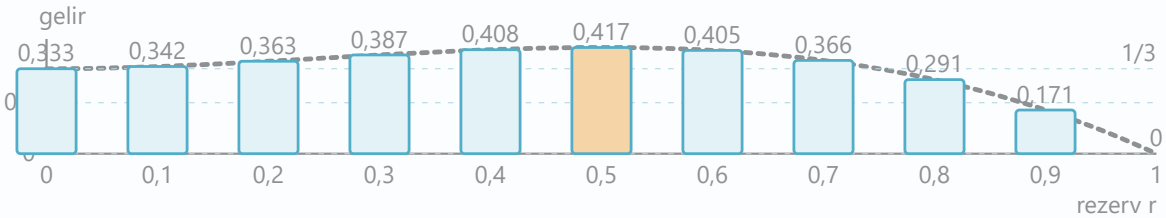
Adım 12 / 13

```
1  def gelir(r, m=100):       # 2 teklifçi, U[0,1]
2    v = [(i + .5) / m for i in range(m)]
3    top = sum(max(min(a, c), r) for a in v
4              for c in v if max(a, c) >= r)
5    return top / m**2          # ortalama ödeme
6  def en_iyi_rezerv(adim=0.1):
7    en, r_en = -1, None
8    for k in range(round(1/adim) + 1):
▶      R = gelir(k * adim)
▶      if R > en: en, r_en = R, k * adim
11   return r_en, en
```

DEĞİŞKENLER

`k = 10` `R = 0` `en = 0,417` `r_en = 0,5`

Kod İzleme: `en_iyi_rezerv()`



En iyi rezerv **r = 0,5**: gelir ≈ 0,417 = 5/12 > 1/3. Yüksek r sık sık satışsız kalır.

Adım 13 / 13

```
1  def gelir(r, m=100):      # 2 teklifçi, U[0,1]
2    v = [(i + .5) / m for i in range(m)]
3    top = sum(max(min(a, c), r) for a in v
4              for c in v if max(a, c) >= r)
5    return top / m**2       # ortalama ödeme
6  def en_iyi_rezerv(adim=0.1):
7    en, r_en = -1, None
8    for k in range(round(1/adim) + 1):
9      R = gelir(k * adim)
10     if R > en: en, r_en = R, k * adim
▶    return r_en, en
```

DEĞİŞKENLER

`r_en = 0,5` `en = 0,417`

GELİR VE BİLGİ

Rezervin Bedeli: Gelir Artar, Verimlilik Düşer

İki teklifçi, değerler U[0, 1], rezerv fiyatı r olan ikinci fiyat açık artırması. İki değer de r'nin altındaysa nesne satılmaz.

r	Beklenen gelir $1/3 + r^2 - 4r^3/3$	Hiç satılmama olasılığı r^2	Beklenen sosyal refah kaybı $2r^3/3$
0	1/3 ≈ 0,333	0	0
0,25	3/8 = 0,375	1/16 = %6,25	1/96 ≈ 0,010
0,5	**5/12 ≈ 0,417**	**1/4 = %25**	**1/12 ≈ 0,083**
0,75	1/3 ≈ 0,333	9/16 = %56,25	9/32 ≈ 0,281

- **Satılmama:** iki değerin ikisi de r'nin altında: $r \cdot r = r^2$.
- **Refah kaybı:** nesneyi değeri en büyük teklifçi almalıdır; bu değer r'nin altındaysa kaybolur. En büyük değerin yoğunluğu 2t: kayıp $\int_0^r t \cdot 2t\, dt = 2r^3/3$.
- Rezervsiz sosyal refah: E[max] = 2/3.

r = 0,5 geliri maksimize eder, ama nesne %25 olasılıkla satılmaz ve beklenen sosyal refah 2/3'ten 7/12'ye düşer (%12,5 kayıp).

Tümü Öder ve Çok Birimli Açık Artırmalar

Tümü öder (all-pay) açık artırması

En yüksek teklif kazanır ama *herkes* teklifini öder: lobi, ar-ge yarışı, seçim kampanyası.
İki teklifçi, U[0, 1]: dengede $b(v) = v^2/2$. Değer 0,8 → teklif 0,32 (birinci fiyatta 0,4). Kazanma olasılığı düşük tipler neredeyse hiç teklif vermez.
Beklenen gelir yine 1/3 (**gelir eşdeğerliği**): tip v'nin beklenen ödemesi $v^2/2$, birinci ve ikinci fiyattakiyle aynı.

Çok birimli açık artırmalar

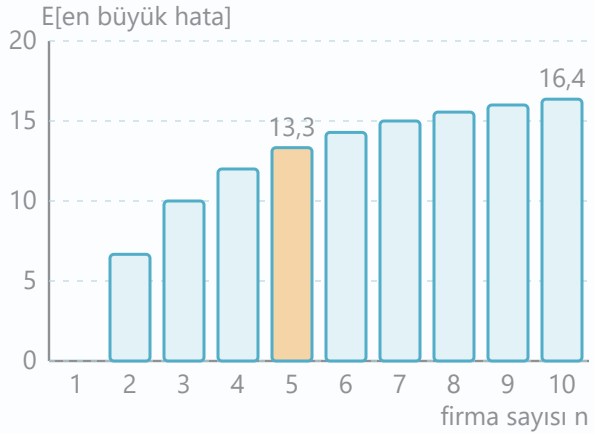
Özdeş birimlerde (devlet tahvili) kazananlar ya aynı fiyatı ya da kendi tekliflerini öder; birlikte değerli nesnelerde (frekans lisansları) paket teklifler kullanılır.

Ek okuma: Bulow ve Klemperer (1996): bir ek teklifçi çekmek, en iyi rezerv fiyatından daha çok kazandırır.

Tümü öder modeli, ödülü tek kişinin kazandığı yarışmaları açıklar: değerler eşit ve herkesçe biliniyorsa beklenen toplam harcama ödülün değerine eşitlenir.

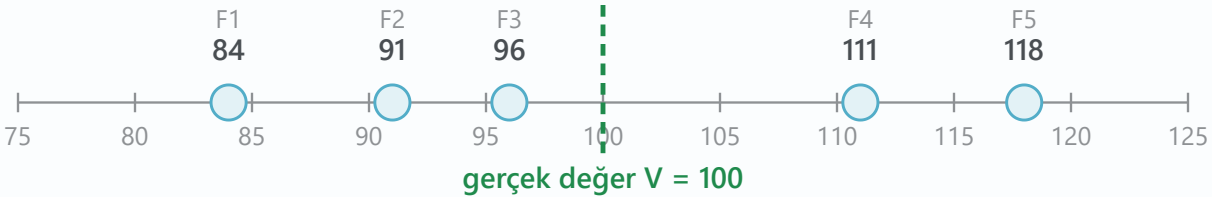
Kazananın Laneti (Winner's Curse)

- Ortak değerli bir açık artırmada en yüksek teklif, değeri **en fazla tahmin eden** katılımcıdan gelir.
- Kazanan, tahmin hatasını hesaba katmazsa nesnenin gerçek değerinden fazla ödeme yapabilir.
- Tahminler ortalamada doğru olsa bile, *en büyük* tahmin sistematik olarak yüksektir.
- Katılımcılar bu riski azaltmak için tekliflerini gözlemlerinin altında tutar ve bilgi kalitesini dikkate alır.
- İlk kez Meksika Körfezi petrol sahası açık artırmalarında belgelendi (Capen, Clapp, Campbell, 1971).



Tahmin hataları U[−20, 20]: kazananın beklenen fazla tahmini 20(n−1)/(n+1). Rakip arttıkça lanet büyür (simülasyon da aynı değerleri verir).

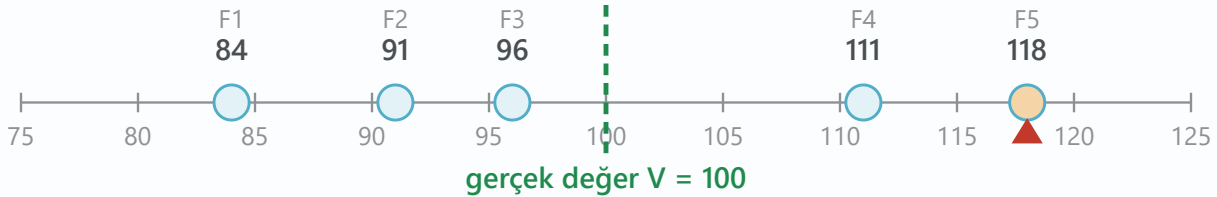
Adım Adım: Petrol Sahası Açık Artırması



Durum	Kazanan / ödeme	Kâr

Sahanın gerçek değeri V = 100 (kimse bilmiyor). Beş firma tahmin yapar: 84, 91, 96, 111, 118. Ortalama **100**: tahminler ortalamada doğru.

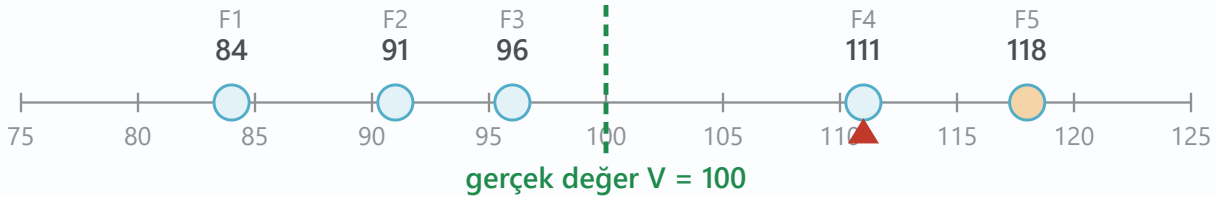
Adım Adım: Petrol Sahası Açık Artırması



Durum	Kazanan / ödeme	Kâr
Birinci fiyat, kırpılmamış teklif	F5 → 118	**−18**

Herkes tahmini kadar teklif verirse (birinci fiyat) en iyimser F5 kazanır ve 118 öder → kâr **−18**.

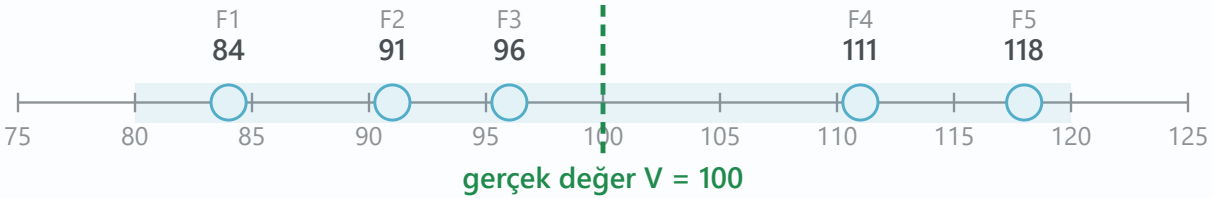
Adım Adım: Petrol Sahası Açık Artırması



Durum	Kazanan / ödeme	Kâr
Birinci fiyat, kırpılmamış teklif	F5 → 118	**−18**
İkinci fiyat, kırpılmamış teklif	F5 → 111	**−11**

İkinci fiyatta bile F5, 111 öder → kâr **−11**. Kazanmak, “en çok abartan ben miyim?” sorusuna evet demektir.

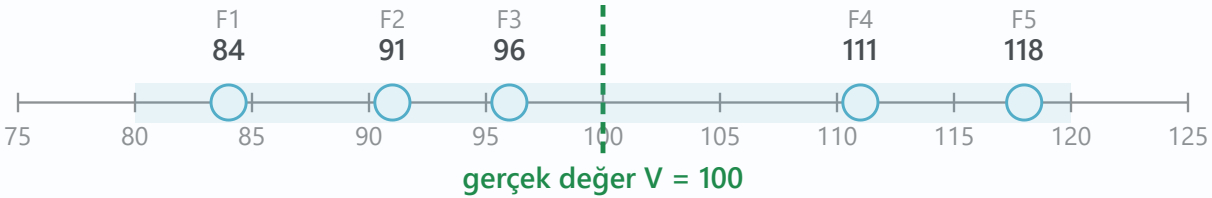
Adım Adım: Petrol Sahası Açık Artırması



Durum	Kazanan / ödeme	Kâr
Birinci fiyat, kırpılmamış teklif	F5 → 118	**−18**
İkinci fiyat, kırpılmamış teklif	F5 → 111	**−11**
En büyük hatanın beklentisi (n = 5)	20·4/6	**≈ 13,3**

Hatalar U[−20, 20] ise beş tahminin en büyüğünün hatası ortalamada 20·4/6 ≈ **13,3**. Birinci fiyatta tahmini kadar teklifin beklenen kârı ≈ −13,3; ikinci fiyatta ödenen ikinci büyük tahminin hatası ortalamada 20·(n − 3)/(n + 1) = 20·2/6 ≈ 6,7 olduğundan ≈ −6,7.

Adım Adım: Petrol Sahası Açık Artırması



Durum	Kazanan / ödeme	Kâr
Birinci fiyat, kırpılmamış teklif	F5 → 118	**−18**
İkinci fiyat, kırpılmamış teklif	F5 → 111	**−11**
En büyük hatanın beklentisi (n = 5)	20·4/6	**≈ 13,3**
Simülasyon, birinci fiyat, kırpılmamış teklif	ortalama	**−13,3**
Simülasyon, birinci fiyat, 13,3 kırpılmış	ortalama	**0**

Birinci fiyat simülasyonu (100 bin açık artırma): tahmini kadar teklifte ortalama kâr −13,3; teklifi 13,3 kırpınca ortalama kâr ≈ 0. 10 firmada gereken kırpma 16,4: rakip arttıkça **daha çok** kırp.

Çift Taraflı Açık Artırma (Double Auction)

- Birden çok alıcı ve birden çok satıcı aynı anda emir verir: alıcılar **alış teklifi** (bid), satıcılar **satış isteği** (ask).
- Borsalar, elektrik piyasaları, karbon izni piyasaları bu yapıdadır.
- Eşleştirme: alış teklifleri azalan, satış istekleri artan sıralanır; alış teklifi ≥ satış isteği olduğu sürece çiftler eşleşir.
- Fiyat: eşleşen son çift ile ilk eşleşmeyen çift arasındaki aralıktan (ör. orta nokta) tek bir piyasa fiyatı seçilir.
- Toplam artık (surplus; alıcı + satıcı artığı) = eşleşen çiftlerde Σ (alış − satış) → mümkün olan en büyük değer.

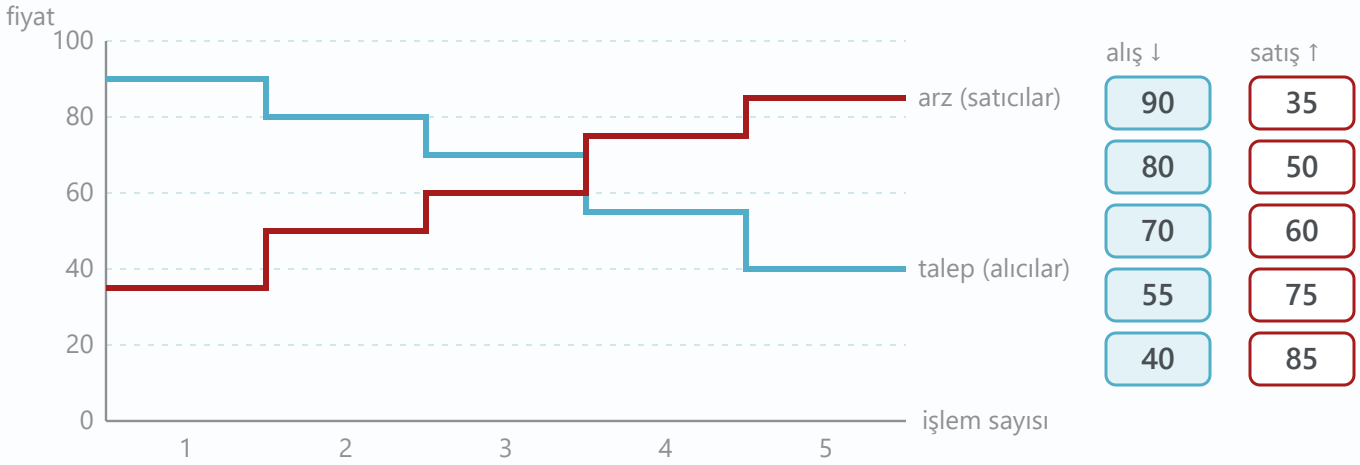
Adım Adım: Çift Taraflı Açık Artırmada Eşleştirme



alış	satış
70	60
40	85
90	35
55	75
80	50

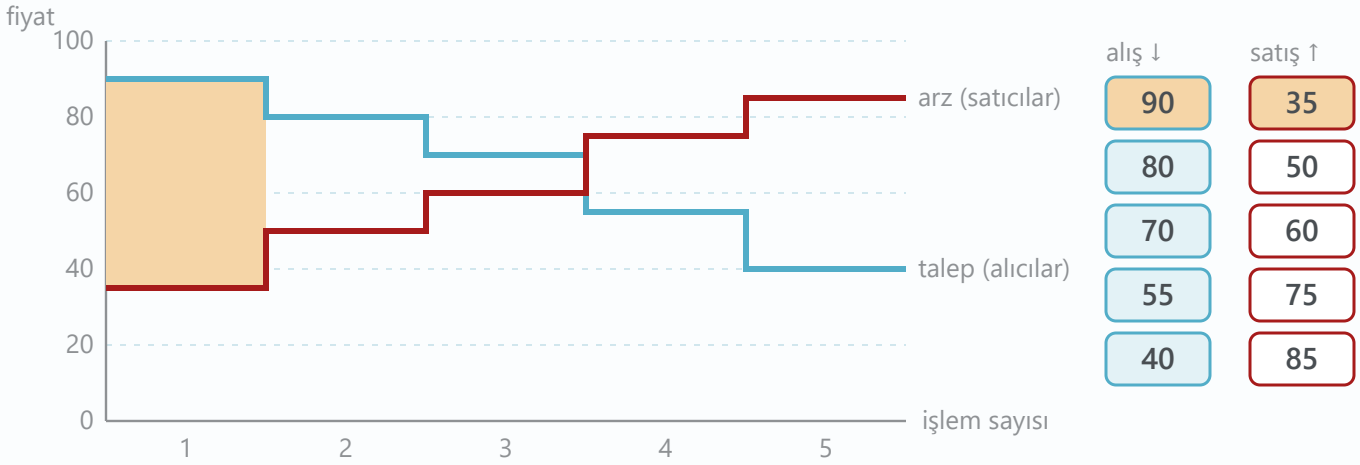
Emir defteri: alıcılar 70, 40, 90, 55, 80 ödemeye; satıcılar 60, 85, 35, 75, 50 almaya razı.

Adım Adım: Çift Taraflı Açık Artırmada Eşleştirme



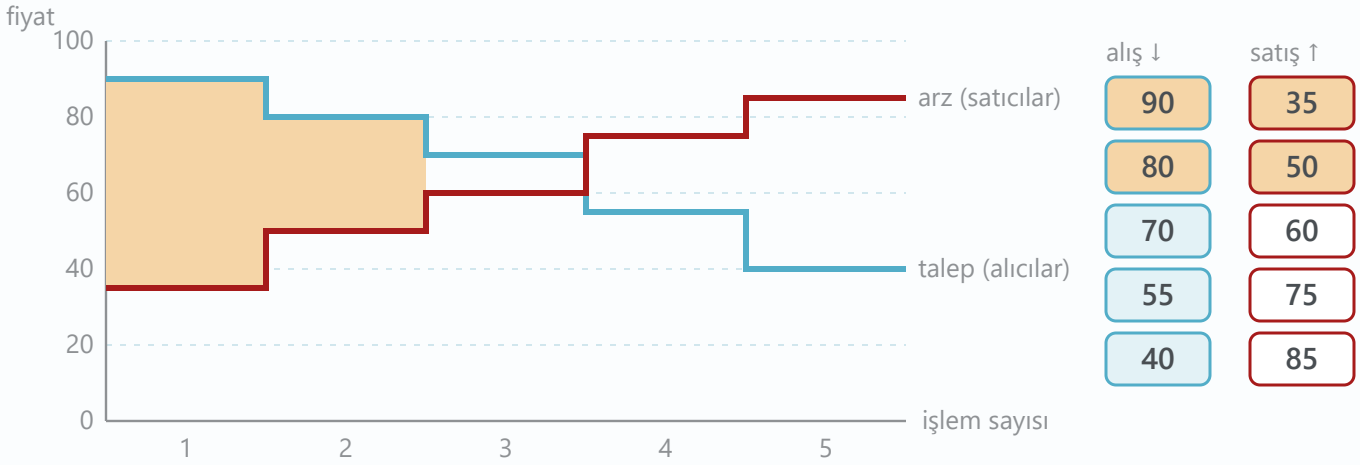
Alış teklifleri **azalan** (talep eğrisi), satış istekleri **artan** (arz eğrisi) sıralanır.

Adım Adım: Çift Taraflı Açık Artırmada Eşleştirme



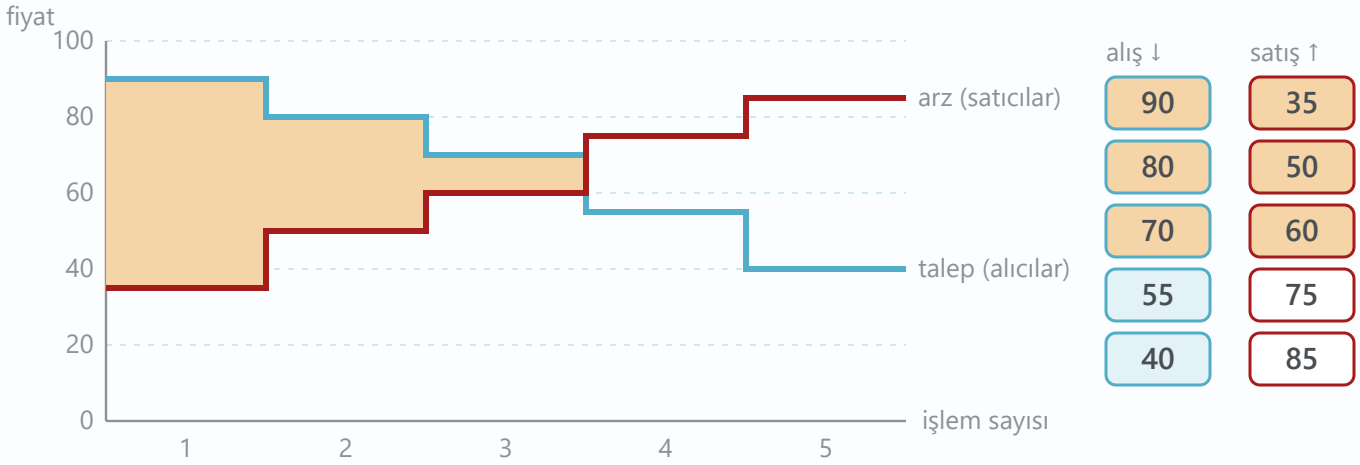
Çift 1: 90 ≥ 35 → **eşleşir**; bu işlemin yarattığı artık 90 − 35 = 55.

Adım Adım: Çift Taraflı Açık Artırmada Eşleştirme



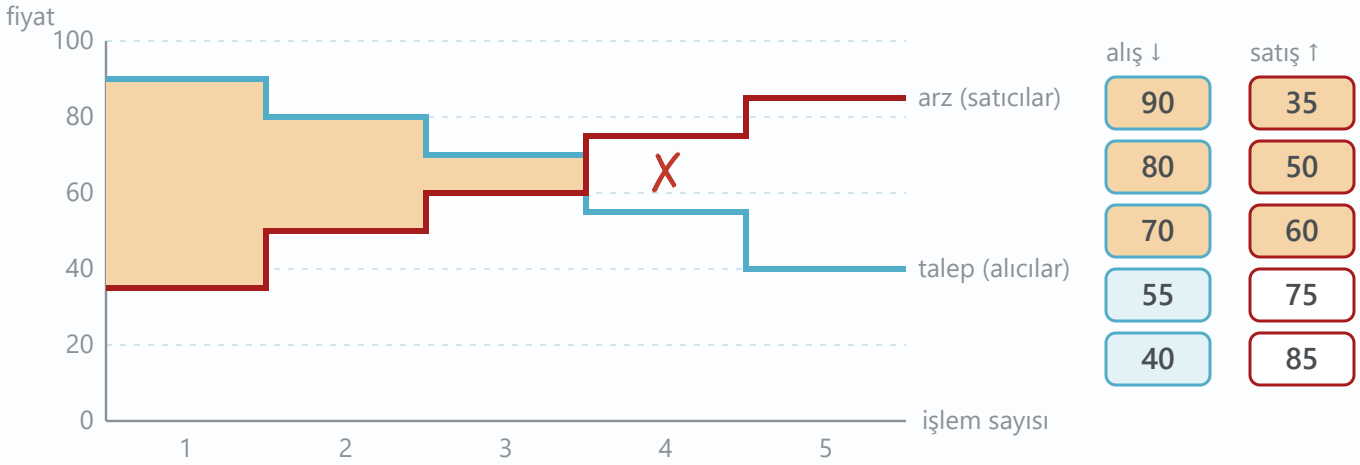
Çift 2: 80 ≥ 50 → **eşleşir**; bu işlemin yarattığı artık 80 − 50 = 30.

Adım Adım: Çift Taraflı Açık Artırmada Eşleştirme



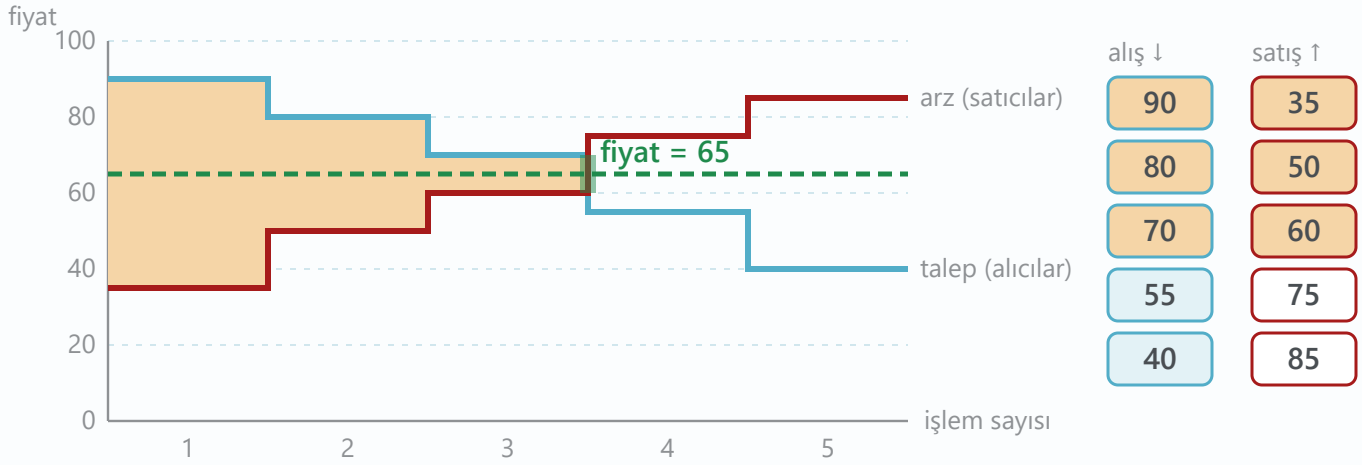
Çift 3: 70 ≥ 60 → **eşleşir**; bu işlemin yarattığı artık 70 − 60 = 10.

Adım Adım: Çift Taraflı Açık Artırmada Eşleştirme



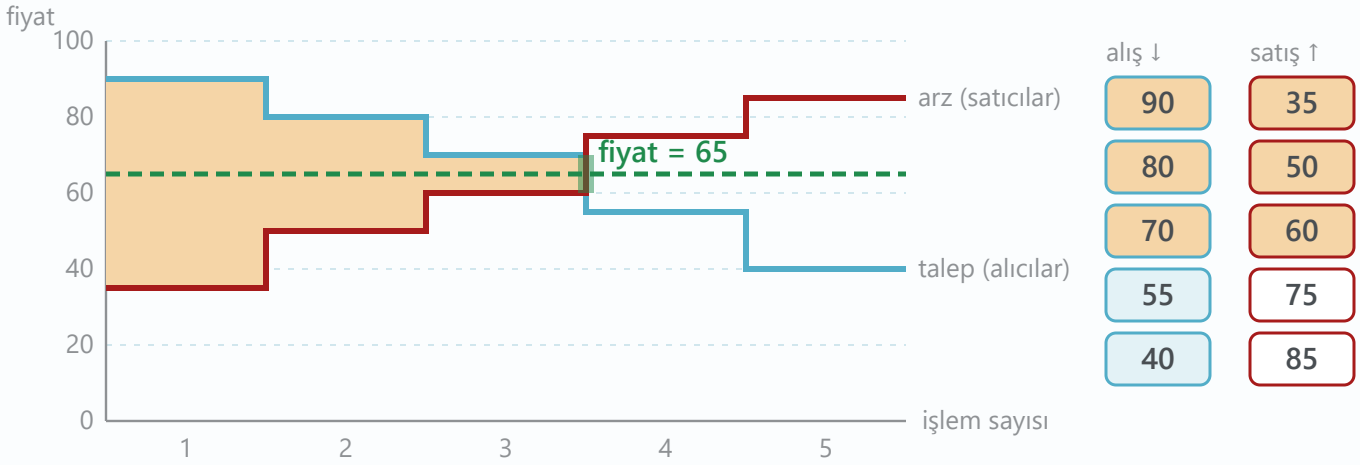
Çift 4: 55 < 75 → alıcı satıcının istediğini ödemez; **dur**. 3 işlem gerçekleşir.

Adım Adım: Çift Taraflı Açık Artırmada Eşleştirme



Fiyat aralığı: [max(60, 55), min(70, 75)] = [60, 70]. Orta nokta **fiyat = 65**; eşleşen herkes bu fiyattan işlem yapar.

Adım Adım: Çift Taraflı Açık Artırmada Eşleştirme



Alıcı artığı (90−65)+(80−65)+(70−65) = 45, satıcı artığı (65−35)+(65−50)+(65−60) = 50; toplam **95** = olası en büyük toplam artık.

Kod İzleme: `cift_tarafli(alis, satis)`

b	90	80	70	55	40
s	60	85	35	75	50

Alış teklifleri büyükten küçüğe sıralanır: en istekli alıcı başa.

Adım 1 / 14

```
1  def cift_tarafli(alis, satis):
▶    b = sorted(alis, reverse=True)  # alıcı
3    s = sorted(satis)               # satıcı
4    k, n = 0, min(len(b), len(s))
5    while k < n and b[k] >= s[k]:
6      k += 1                    # çift eşleşti
7    if k == 0:
8      return 0, None
9    lo = max([s[k-1]] + b[k:k+1])
10   hi = min([b[k-1]] + s[k:k+1])
11   return k, (lo + hi) / 2
```

DEĞİŞKENLER

`alis = [70, 40, 90, 55, 80]`

`b = [90, 80, 70, 55, 40]`

Kod İzleme: `cift_tarafli(alis, satis)`

b	90	80	70	55	40
s	35	50	60	75	85

Satış istekleri küçükten büyüğe: en ucuz satıcı başa.

Adım 2 / 14

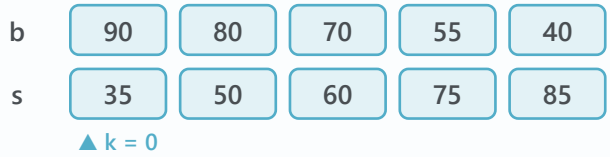
```
1  def cift_tarafli(alis, satis):
2    b = sorted(alis, reverse=True)  # alıcı
▶    s = sorted(satis)               # satıcı
4    k, n = 0, min(len(b), len(s))
5    while k < n and b[k] >= s[k]:
6      k += 1                    # çift eşleşti
7    if k == 0:
8      return 0, None
9    lo = max([s[k-1]] + b[k:k+1])
10   hi = min([b[k-1]] + s[k:k+1])
11   return k, (lo + hi) / 2
```

DEĞİŞKENLER

`b = [90, 80, 70, 55, 40]`

`s = [35, 50, 60, 75, 85]`

Kod İzleme: `cift_tarafli(alis, satis)`



k: eşleşen çift sayısı; n: kısa listenin uzunluğu.

Adım 3 / 14

```
1  def cift_tarafli(alis, satis):
2    b = sorted(alis, reverse=True)  # alıcı
3    s = sorted(satis)               # satıcı
▶    k, n = 0, min(len(b), len(s))
5    while k < n and b[k] >= s[k]:
6      k += 1                    # çift eşleşti
7    if k == 0:
8      return 0, None
9    lo = max([s[k-1]] + b[k:k+1])
10   hi = min([b[k-1]] + s[k:k+1])
11   return k, (lo + hi) / 2
```

DEĞİŞKENLER

`b = [90, 80, 70, 55, 40]`

`s = [35, 50, 60, 75, 85]` `k = 0`

`n = 5`

Kod İzleme: `cift_tarafli(alis, satis)`

b	90	80	70	55	40
s	35	50	60	75	85

▲ k = 0

Koşul: b[0] = 90 ≥ s[0] = 35? **Evet**.

Adım 4 / 14

```
1  def cift_tarafli(alis, satis):
2    b = sorted(alis, reverse=True)  # alıcı
3    s = sorted(satis)               # satıcı
4    k, n = 0, min(len(b), len(s))
▶    while k < n and b[k] >= s[k]:
6      k += 1                        # çift eşleşti
7    if k == 0:
8      return 0, None
9    lo = max([s[k-1]] + b[k:k+1])
10   hi = min([b[k-1]] + s[k:k+1])
11   return k, (lo + hi) / 2
```

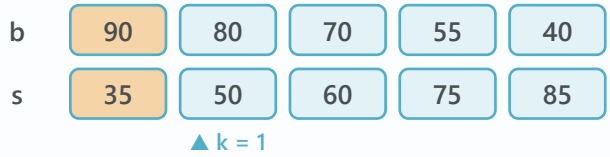
DEĞİŞKENLER

`b = [90, 80, 70, 55, 40]`

`s = [35, 50, 60, 75, 85]` `k = 0`

`n = 5`

Kod İzleme: `cift_tarafli(alis, satis)`



Çift eşleşti → **k = 1**.

Adım 5 / 14

```
1  def cift_tarafli(alis, satis):
2    b = sorted(alis, reverse=True)  # alıcı
3    s = sorted(satis)               # satıcı
4    k, n = 0, min(len(b), len(s))
5    while k < n and b[k] >= s[k]:
▶      k += 1                      # çift eşleşti
7    if k == 0:
8      return 0, None
9    lo = max([s[k-1]] + b[k:k+1])
10   hi = min([b[k-1]] + s[k:k+1])
11   return k, (lo + hi) / 2
```

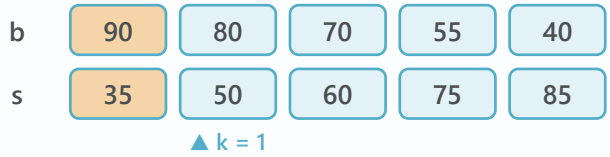
DEĞİŞKENLER

`b = [90, 80, 70, 55, 40]`

`s = [35, 50, 60, 75, 85]` `k = 1`

`n = 5`

Kod İzleme: `cift_tarafli(alis, satis)`



Koşul: b[1] = 80 ≥ s[1] = 50? **Evet**.


```
1  def cift_tarafli(alis, satis):
2    b = sorted(alis, reverse=True)  # alıcı
3    s = sorted(satis)               # satıcı
4    k, n = 0, min(len(b), len(s))
▶   while k < n and b[k] >= s[k]:
6      k += 1                      # çift eşleşti
7    if k == 0:
8      return 0, None
9    lo = max([s[k-1]] + b[k:k+1])
10   hi = min([b[k-1]] + s[k:k+1])
11   return k, (lo + hi) / 2
```

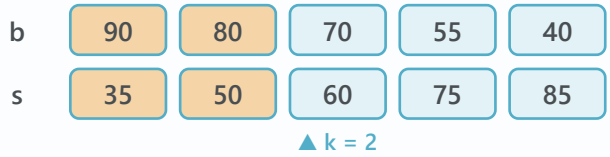
DEĞİŞKENLER

`b = [90, 80, 70, 55, 40]`

`s = [35, 50, 60, 75, 85]` `k = 1`

`n = 5`

Kod İzleme: `cift_tarafli(alis, satis)`



Çift eşleşti → **k = 2**.

Adım 7 / 14

```
1  def cift_tarafli(alis, satis):
2    b = sorted(alis, reverse=True)  # alıcı
3    s = sorted(satis)               # satıcı
4    k, n = 0, min(len(b), len(s))
5    while k < n and b[k] >= s[k]:
▶      k += 1                      # çift eşleşti
7    if k == 0:
8      return 0, None
9    lo = max([s[k-1]] + b[k:k+1])
10   hi = min([b[k-1]] + s[k:k+1])
11   return k, (lo + hi) / 2
```

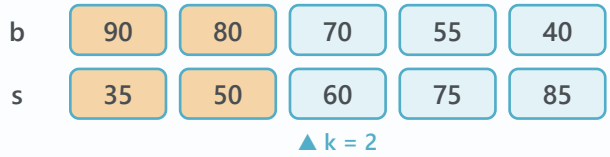
DEĞİŞKENLER

`b = [90, 80, 70, 55, 40]`

`s = [35, 50, 60, 75, 85]` `k = 2`

`n = 5`

Kod İzleme: `cift_tarafli(alis, satis)`



Koşul: b[2] = 70 ≥ s[2] = 60? **Evet**.

Adım 8 / 14

```
1  def cift_tarafli(alis, satis):
2    b = sorted(alis, reverse=True)  # alıcı
3    s = sorted(satis)               # satıcı
4    k, n = 0, min(len(b), len(s))
▶   while k < n and b[k] >= s[k]:
6      k += 1                        # çift eşleşti
7    if k == 0:
8      return 0, None
9    lo = max([s[k-1]] + b[k:k+1])
10   hi = min([b[k-1]] + s[k:k+1])
11   return k, (lo + hi) / 2
```

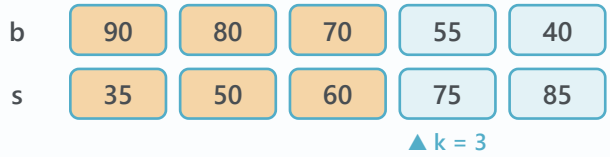
DEĞİŞKENLER

`b = [90, 80, 70, 55, 40]`

`s = [35, 50, 60, 75, 85]` `k = 2`

`n = 5`

Kod İzleme: `cift_tarafli(alis, satis)`



Çift eşleşti → **k = 3**.

Adım 9 / 14

```
1  def cift_tarafli(alis, satis):
2    b = sorted(alis, reverse=True)  # alıcı
3    s = sorted(satis)               # satıcı
4    k, n = 0, min(len(b), len(s))
5    while k < n and b[k] >= s[k]:
▶      k += 1                      # çift eşleşti
7    if k == 0:
8      return 0, None
9    lo = max([s[k-1]] + b[k:k+1])
10   hi = min([b[k-1]] + s[k:k+1])
11   return k, (lo + hi) / 2
```

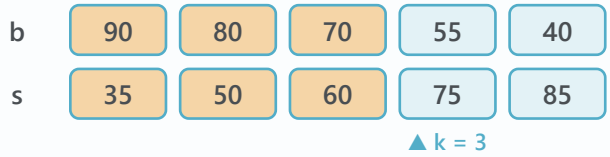
DEĞİŞKENLER

`b = [90, 80, 70, 55, 40]`

`s = [35, 50, 60, 75, 85]` `k = 3`

`n = 5`

Kod İzleme: `cift_tarafli(alis, satis)`



Koşul: b[3] = 55 ≥ s[3] = 75? **Hayır**. Döngü biter.

Adım 10 / 14

```
1  def cift_tarafli(alis, satis):
2    b = sorted(alis, reverse=True)  # alıcı
3    s = sorted(satis)               # satıcı
4    k, n = 0, min(len(b), len(s))
▶  while k < n and b[k] >= s[k]:
6      k += 1                        # çift eşleşti
7    if k == 0:
8      return 0, None
9    lo = max([s[k-1]] + b[k:k+1])
10   hi = min([b[k-1]] + s[k:k+1])
11   return k, (lo + hi) / 2
```

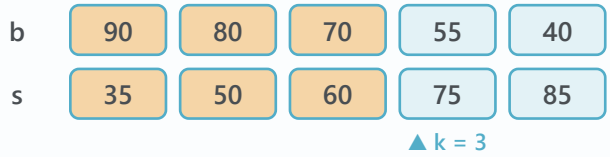
DEĞİŞKENLER

`b = [90, 80, 70, 55, 40]`

`s = [35, 50, 60, 75, 85]` `k = 3`

`n = 5`

Kod İzleme: `cift_tarafli(alis, satis)`



k ≠ 0: en az bir işlem var, fiyat hesaplanır.

Adım 11 / 14

```
1  def cift_tarafli(alis, satis):
2      b = sorted(alis, reverse=True)  # alıcı
3      s = sorted(satis)               # satıcı
4      k, n = 0, min(len(b), len(s))
5      while k < n and b[k] >= s[k]:
6          k += 1                      # çift eşleşti
▶      if k == 0:
8          return 0, None
9      lo = max([s[k-1]] + b[k:k+1])
10     hi = min([b[k-1]] + s[k:k+1])
11     return k, (lo + hi) / 2
```

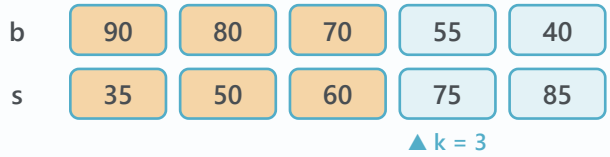
DEĞİŞKENLER

`b = [90, 80, 70, 55, 40]`

`s = [35, 50, 60, 75, 85]` `k = 3`

`n = 5`

Kod İzleme: `cift_tarafli(alis, satis)`



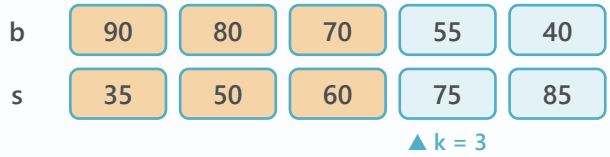
lo = max(s[k−1], b[k]) = max(60, 55) = **60**: son satıcı zarar etmemeli, dışarıdaki alıcı almak istememeli.


```
1  def cift_tarafli(alis, satis):
2    b = sorted(alis, reverse=True)  # alıcı
3    s = sorted(satis)               # satıcı
4    k, n = 0, min(len(b), len(s))
5    while k < n and b[k] >= s[k]:
6      k += 1                    # çift eşleşti
7    if k == 0:
8      return 0, None
▶    lo = max([s[k-1]] + b[k:k+1])
10   hi = min([b[k-1]] + s[k:k+1])
11   return k, (lo + hi) / 2
```

DEĞİŞKENLER

`k = 3` `lo = 60`

Kod İzleme: `cift_tarafli(alis, satis)`



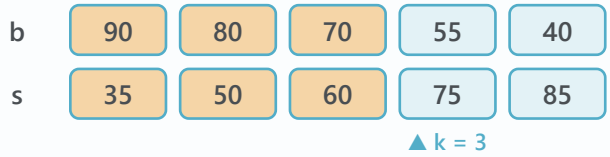
hi = min(b[k−1], s[k]) = min(70, 75) = **70**.


```
1  def cift_tarafli(alis, satis):
2    b = sorted(alis, reverse=True)  # alıcı
3    s = sorted(satis)               # satıcı
4    k, n = 0, min(len(b), len(s))
5    while k < n and b[k] >= s[k]:
6      k += 1                    # çift eşleşti
7    if k == 0:
8      return 0, None
9    lo = max([s[k-1]] + b[k:k+1])
▶    hi = min([b[k-1]] + s[k:k+1])
11   return k, (lo + hi) / 2
```

DEĞİŞKENLER

`k = 3` `lo = 60` `hi = 70`

Kod İzleme: `cift_tarafli(alis, satis)`



Sonuç: **3 işlem, fiyat (60 + 70)/2 = 65**.

Adım 14 / 14

```
1  def cift_tarafli(alis, satis):
2    b = sorted(alis, reverse=True)  # alıcı
3    s = sorted(satis)               # satıcı
4    k, n = 0, min(len(b), len(s))
5    while k < n and b[k] >= s[k]:
6      k += 1                    # çift eşleşti
7    if k == 0:
8      return 0, None
9    lo = max([s[k-1]] + b[k:k+1])
10   hi = min([b[k-1]] + s[k:k+1])
▶    return k, (lo + hi) / 2
```

DEĞİŞKENLER

`k = 3` `lo = 60` `hi = 70` `p = 65.0`

Çift Taraflı Piyasada Teşvikler

İmkânsızlık

İki taraflı ticarette verimlilik + IC + IR + bütçe dengesi birlikte sağlanamaz (Myerson–Satterthwaite, 1983).

Orta nokta kuralı

Verimli ve bütçe dengeli, ama IC değil: marjinal alıcı teklifini düşürüp fiyatı aşağı çekebilir.

McAfee (1992) kuralı

Dışarıda kalan ilk alıcı ile ilk satıcının ortalaması p_0 olsun. p_0 eşleşen son alıcı ile satıcının arasına düşerse herkes p_0'dan işlem yapar; düşmezse son (en az kazanç getiren) çift çıkarılır. Dürüstlük baskındır, bedeli bazen bir işlem.

Örneğimizde

$p_0 = (55 + 75)/2 = 65 \in [60, 70] \rightarrow$ üç işlem de 65'ten gerçekleşir; hiçbir işlem kaybolmaz.

Kod İzleme: `mcafee(alis, satis)`

	1. çift	2. çift	3. çift	4. çift
alış b	90	80	70	40
satış s	30	50	65	85

Yeni örnek (alıcılar 90, 80, 70, 40; satıcılar 30, 50, 65, 85): alış azalan, satış artan sıralanır; sona bekçiler (0 ve ∞) eklenir.

Adım 1 / 11

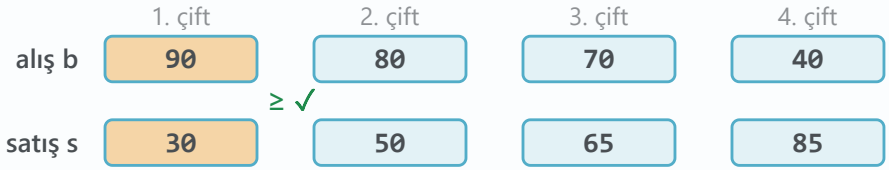
```
1  def mcafee(alis, satis):
▶    b = sorted(alis, reverse=True) + [0]
▶    s = sorted(satis) + [float('inf')]
4    k = 0
5    while b[k] >= s[k]:
6      k += 1                  # verimli işlem
7    if k == 0: return 0, None, None
8    p0 = (b[k] + s[k]) / 2    # dışarıdaki çift
9    if s[k-1] <= p0 <= b[k-1]:
10     return k, p0, p0        # tek fiyat
11   return k - 1, b[k-1], s[k-1]  # iki fiyat
```

DEĞİŞKENLER

`b = [90, 80, 70, 40, 0]`

`s = [30, 50, 65, 85, inf]`

Kod İzleme: `mcafee(alis, satis)`



b[0] = 90 ≥ s[0] = 30? **Evet**: verimli çift.

Adım 2 / 11

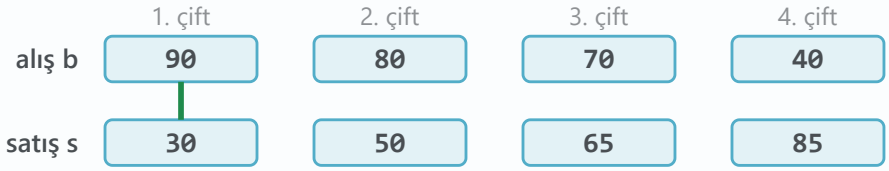
```
1  def mcafee(alis, satis):
2    b = sorted(alis, reverse=True) + [0]
3    s = sorted(satis) + [float('inf')]
4    k = 0
▶    while b[k] >= s[k]:
6      k += 1                  # verimli işlem
7    if k == 0: return 0, None, None
8    p0 = (b[k] + s[k]) / 2    # dışarıdaki çift
9    if s[k-1] <= p0 <= b[k-1]:
10     return k, p0, p0        # tek fiyat
11   return k - 1, b[k-1], s[k-1]  # iki fiyat
```

DEĞİŞKENLER

`b = [90, 80, 70, 40, 0]`

`s = [30, 50, 65, 85, inf]` `k = 0`

Kod İzleme: `mcafee(alis, satis)`



k = 1: 1 verimli işlem bulundu.

Adım 3 / 11

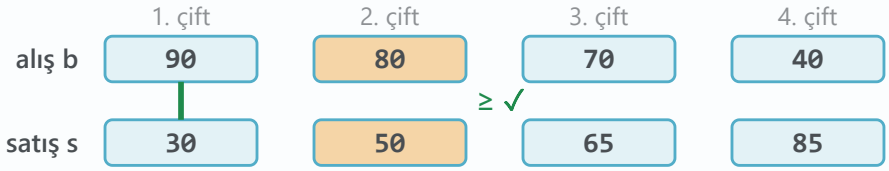
```
1  def mcafee(alis, satis):
2    b = sorted(alis, reverse=True) + [0]
3    s = sorted(satis) + [float('inf')]
4    k = 0
5    while b[k] >= s[k]:
▶      k += 1                  # verimli işlem
7    if k == 0: return 0, None, None
8    p0 = (b[k] + s[k]) / 2   # dışarıdaki çift
9    if s[k-1] <= p0 <= b[k-1]:
10     return k, p0, p0       # tek fiyat
11   return k - 1, b[k-1], s[k-1]  # iki fiyat
```

DEĞİŞKENLER

`b = [90, 80, 70, 40, 0]`

`s = [30, 50, 65, 85, inf]` `k = 1`

Kod İzleme: `mcafee(alis, satis)`



b[1] = 80 ≥ s[1] = 50? **Evet**: verimli çift.

Adım 4 / 11

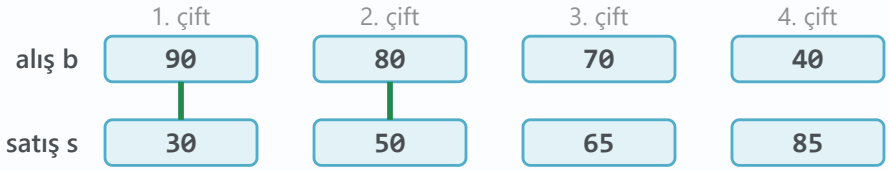
```
1  def mcafee(alis, satis):
2    b = sorted(alis, reverse=True) + [0]
3    s = sorted(satis) + [float('inf')]
4    k = 0
▶   while b[k] >= s[k]:
6      k += 1                  # verimli işlem
7    if k == 0: return 0, None, None
8    p0 = (b[k] + s[k]) / 2    # dışarıdaki çift
9    if s[k-1] <= p0 <= b[k-1]:
10     return k, p0, p0        # tek fiyat
11   return k - 1, b[k-1], s[k-1]  # iki fiyat
```

DEĞİŞKENLER

`b = [90, 80, 70, 40, 0]`

`s = [30, 50, 65, 85, inf]` `k = 1`

Kod İzleme: `mcafee(alis, satis)`



k = 2: 2 verimli işlem bulundu.

Adım 5 / 11

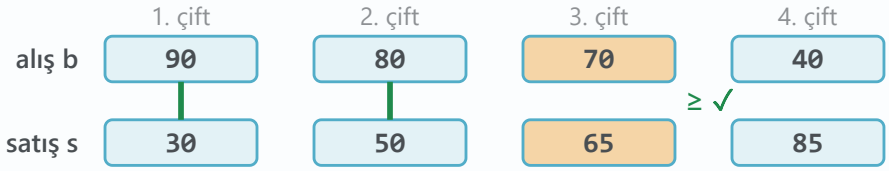
```
1  def mcafee(alis, satis):
2    b = sorted(alis, reverse=True) + [0]
3    s = sorted(satis) + [float('inf')]
4    k = 0
5    while b[k] >= s[k]:
▶      k += 1                  # verimli işlem
7    if k == 0: return 0, None, None
8    p0 = (b[k] + s[k]) / 2   # dışarıdaki çift
9    if s[k-1] <= p0 <= b[k-1]:
10     return k, p0, p0       # tek fiyat
11   return k - 1, b[k-1], s[k-1]  # iki fiyat
```

DEĞİŞKENLER

`b = [90, 80, 70, 40, 0]`

`s = [30, 50, 65, 85, inf]` `k = 2`

Kod İzleme: `mcafee(alis, satis)`



b[2] = 70 ≥ s[2] = 65? **Evet**: verimli çift.

Adım 6 / 11

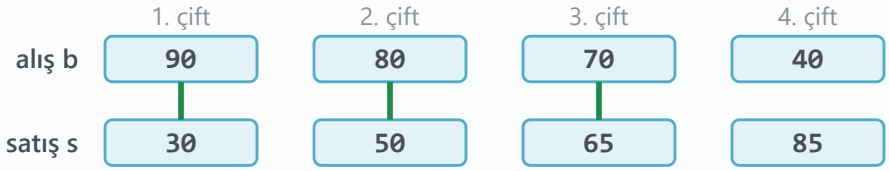
```
1  def mcafee(alis, satis):
2    b = sorted(alis, reverse=True) + [0]
3    s = sorted(satis) + [float('inf')]
4    k = 0
▶   while b[k] >= s[k]:
6      k += 1                  # verimli işlem
7    if k == 0: return 0, None, None
8    p0 = (b[k] + s[k]) / 2    # dışarıdaki çift
9    if s[k-1] <= p0 <= b[k-1]:
10     return k, p0, p0        # tek fiyat
11   return k - 1, b[k-1], s[k-1]  # iki fiyat
```

DEĞİŞKENLER

`b = [90, 80, 70, 40, 0]`

`s = [30, 50, 65, 85, inf]` `k = 2`

Kod İzleme: `mcafee(alis, satis)`



k = 3: 3 verimli işlem bulundu.

Adım 7 / 11

```
1  def mcafee(alis, satis):
2    b = sorted(alis, reverse=True) + [0]
3    s = sorted(satis) + [float('inf')]
4    k = 0
5    while b[k] >= s[k]:
▶      k += 1                  # verimli işlem
7    if k == 0: return 0, None, None
8    p0 = (b[k] + s[k]) / 2   # dışarıdaki çift
9    if s[k-1] <= p0 <= b[k-1]:
10     return k, p0, p0       # tek fiyat
11   return k - 1, b[k-1], s[k-1]  # iki fiyat
```

DEĞİŞKENLER

`b = [90, 80, 70, 40, 0]`

`s = [30, 50, 65, 85, inf]` `k = 3`

Kod İzleme: `mcafee(alis, satis)`



b[3] = 40 ≥ s[3] = 85? **Hayır**: döngü biter.

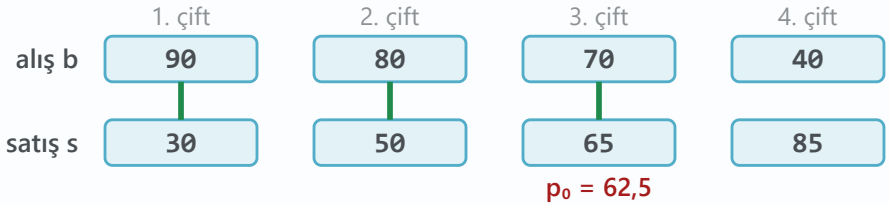

```
1  def mcafee(alis, satis):
2    b = sorted(alis, reverse=True) + [0]
3    s = sorted(satis) + [float('inf')]
4    k = 0
▶    while b[k] >= s[k]:
6      k += 1                  # verimli işlem
7    if k == 0: return 0, None, None
8    p0 = (b[k] + s[k]) / 2    # dışarıdaki çift
9    if s[k-1] <= p0 <= b[k-1]:
10     return k, p0, p0        # tek fiyat
11   return k - 1, b[k-1], s[k-1]  # iki fiyat
```

DEĞİŞKENLER

`b = [90, 80, 70, 40, 0]`

`s = [30, 50, 65, 85, inf]` `k = 3`

Kod İzleme: `mcafee(alis, satis)`



Dışarıda kalan ilk çiftten fiyat adayı: $p_0 = (40 + 85)/2 =$ **62,5**.

Adım 9 / 11

```
1  def mcafee(alis, satis):
2    b = sorted(alis, reverse=True) + [0]
3    s = sorted(satis) + [float('inf')]
4    k = 0
5    while b[k] >= s[k]:
6      k += 1                  # verimli işlem
7    if k == 0: return 0, None, None
▶   p0 = (b[k] + s[k]) / 2    # dışarıdaki çift
9    if s[k-1] <= p0 <= b[k-1]:
10     return k, p0, p0        # tek fiyat
11   return k - 1, b[k-1], s[k-1]  # iki fiyat
```

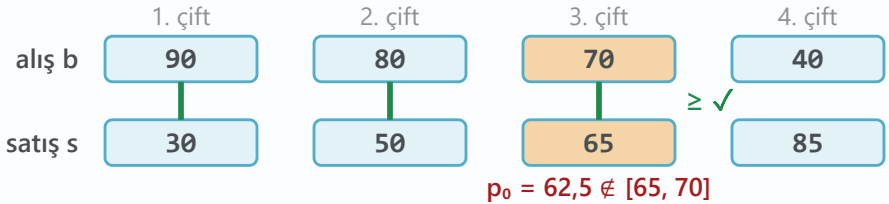
DEĞİŞKENLER

`b = [90, 80, 70, 40, 0]`

`s = [30, 50, 65, 85, inf]` `k = 3`

`p0 = 62,5`

Kod İzleme: `mcafee(alis, satis)`



$p_0 \in [s[2], b[2]] = [65, 70]$ mı? **Hayır** (62,5 < 65): 3. satıcı zarar ederdi.

Adım 10 / 11

```
1  def mcafee(alis, satis):
2    b = sorted(alis, reverse=True) + [0]
3    s = sorted(satis) + [float('inf')]
4    k = 0
5    while b[k] >= s[k]:
6      k += 1                  # verimli işlem
7    if k == 0: return 0, None, None
8    p0 = (b[k] + s[k]) / 2    # dışarıdaki çift
▶    if s[k-1] <= p0 <= b[k-1]:
10     return k, p0, p0        # tek fiyat
11   return k - 1, b[k-1], s[k-1]  # iki fiyat
```

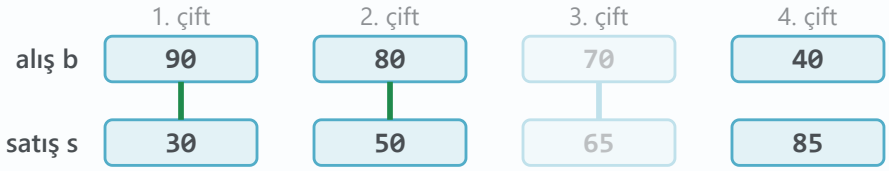
DEĞİŞKENLER

`b = [90, 80, 70, 40, 0]`

`s = [30, 50, 65, 85, inf]` `k = 3`

`p0 = 62,5`

Kod İzleme: `mcafee(alis, satis)`



2 işlem: alıcı 70, satıcı 65

En zayıf çift (70, 65) çıkarılır: **2 işlem**; alıcılar 70 öder, satıcılar 65 alır. Artan 2·5 = 10 piyasa yapıcıda kalır.

Adım 11 / 11

```
1  def mcafee(alis, satis):
2    b = sorted(alis, reverse=True) + [0]
3    s = sorted(satis) + [float('inf')]
4    k = 0
5    while b[k] >= s[k]:
6      k += 1                  # verimli işlem
7    if k == 0: return 0, None, None
8    p0 = (b[k] + s[k]) / 2    # dışarıdaki çift
9    if s[k-1] <= p0 <= b[k-1]:
10     return k, p0, p0        # tek fiyat
▶    return k - 1, b[k-1], s[k-1]  # iki fiyat
```

DEĞİŞKENLER

`b = [90, 80, 70, 40, 0]`

`s = [30, 50, 65, 85, inf]` `k = 3`

`p0 = 62,5`

Açık Artırma ve Piyasa Tasarımı

- Piyasa tasarımı, katılımcıların kimlerle ve hangi kurallarla eşleşeceğini belirler.
- Temel hedefler verimlilik, teşvik uyumluluğu, adalet, kararlılık ve uygulanabilirliktir; kurallar stratejiye dayanıklı olmalıdır.
- Açık artırmalar, kıt kaynakları fiyat ve tekliflere göre tahsis eden piyasa tasarımı araçlarıdır.
- Elektrik piyasaları, emisyon izinleri, kamu ihaleleri ve frekans lisansları fiyatlı uygulamalardır; para kullanılmayan piyasalarda fiyatın yerini öncelik ve eşleştirme kuralları alır.
- Gelir, verimlilik, basitlik ve stratejiye dayanıklılık arasında ödünleşimler bulunabilir.

Kuramdan Uygulamaya: Kısa Tarihçe

Eski bir kurum

Herodot, MÖ 5. yüzyılda Babil'deki açık artırmaları anlatır. MS 193'te Pretoryen Muhafızları Roma tahtını açık artırmayla Didius Julianus'a sattı.

Kuram

Vickrey (1961) dört türü karşılaştırıp gelir eşdeğerliğini buldu; Myerson (1981) geliri maksimize eden açık artırmayı tanımladı. Nobel: Vickrey 1996, Myerson 2007.

Spektrum açık artırmaları

ABD FCC 1994'te ilk spektrum açık artırmasını Milgrom, Wilson ve McAfee'nin önerdiği eşzamanlı çok turlu artırmayla yaptı. Milgrom ve Wilson: Nobel 2020.

Eşleştirme piyasaları

Roth ve Shapley (Nobel 2012): fiyatın kullanılmadığı piyasalarda kararlı eşleştirme ve takas kuralları.

Tasarımcının araçları: tahsis ve ödeme kuralı, **bilgi açıklama** (hangi teklifler, ne zaman gösterilir?), katılım koşulları. Tasarımlar uygulamadan önce laboratuvar deneyleri ve saha verisiyle sınanır.

DEĞERLENDİRME

Vickrey Açık Artırmaları Neden Az Kullanılıyor?

- **Güven**: katılımcıların ikinci en yüksek teklifin doğru açıklandığına güvenmesi gerekir (sahte teklif riski).
- **Sezgisellik**: “ikinci fiyatı öde” kuralı çoğu insana doğal gelmez; laboratuvarda fazla teklif sık görülür.
- **Model varsayımları**: ortak veya birbirine bağlı değerlerde dürüst teklifin baskın olduğu sonucu geçerliliğini yitirebilir.
- **Çok nesne**: bütçe kısıtları ve tamamlayıcı ürünler ödeme kuralını karmaşıklaştırır (VCG).
- **Satıcı kaygısı**: düşük gerçekleşen fiyat veya danışıklı teklif riski.

Bu nedenlerle pratik tasarımlar çoğu zaman dürüstlük teşvikiyle uygulama kolaylığı arasında denge kurar (ör. İngiliz açık artırması aynı sonucu şeffaf biçimde verir). Ayrıntı: Rothkopf, Teisberg ve Kahn (1990); Ausubel ve Milgrom (2006).

DEĞERLENDİRME

Tartışma: Hangi Açık Artırma, Ne Kadar Adil?

Hangi türe katılmak isterdiniz?

- İngiliz: şeffaf, fiyatı adım adım görmek avantajlı.
- Vickrey: dürüst teklif en iyisi, strateji kurmaya gerek yok.
- Hollanda / birinci fiyat: hızlı ama kırpma miktarını tahmin etmek gerekir.
- Tercih kişisel stratejiye ve risk iştahına bağlıdır.

Açık artırmalar ve piyasa tasarımı adil mi?

- Kurallar önceden belirlenir → öngörülebilirlik; manipülasyon azaltılabilir.
- Ama bütçesi büyük oyuncular avantajlıdır; katılım erişimi ve bilgi eşitsizliği önemlidir.
- Satıcının geliri ile fırsat eşitliği farklı hedeflerdir; “herkes için tam adalet” garanti değildir.

DEĞERLENDİRME

Özet: Dört Tür Yan Yana

Tür	Öder	Denge teklifi (IPV, U[0,1], n kişi)	Eşdeğeri
İngiliz	son rakibin çekildiği fiyat	değerine kadar kal	ikinci fiyat
İkinci fiyat	ikinci en yüksek teklif	$b = v$ (zayıf baskın)	İngiliz
Hollanda	kabul edilen fiyat	$b = (n-1)v/n$	birinci fiyat
Birinci fiyat	kendi teklifi	$b = (n-1)v/n$	Hollanda

Gelir eşdeğerliği

Dördünde de beklenen gelir $(n-1)/(n+1)$; iki kişide 1/3.

Kazananın laneti

Ortak değerde kazanan fazla ödeyebilir; rakip sayısı arttıkça daha çok kırp.

Çift taraflı

Alış teklifleri azalan, satış istekleri artan eşle; fiyat [lo, hi] aralığından.

DEĞERLENDİRME

Örnek Problemler

1. Sınırlar 40, 65, 80, 90. Dört kuralda kazanan kim, ne öder (herkes sınırını teklif ediyor)?
2. n = 4 teklifçi, değerler U[0, 1]. Değeri 0,6 olan teklifçinin birinci fiyattaki denge teklifi nedir?
3. Üç teklifçi, U[0, 1]: ikinci fiyat açık artırmasının beklenen geliri nedir?
4. Vickrey'de değerin 50, rakiplerin en yüksek teklifi 55. 60 teklif edersen ne olur?
5. Alıcılar 100, 70, 45; satıcılar 30, 60, 80. Kaç işlem olur, orta nokta fiyatı nedir?
6. İki teklifçi, U[0, 1], ikinci fiyat, rezerv r = 0,4. Beklenen gelir nedir? Rezervsizden fazla mı?

▶ **Cevapları göster**

DEĞERLENDİRME

Etkileşimli Simülatörler

İngiliz Açık Artırması

Fiyat yükselirken kal ya da çekil.

Hollanda Açık Artırması

Düşen fiyatta ne zaman “kabul” demeli?

Birinci Fiyat Kapalı Zarf

Teklif kırpma: kâr ile kazanma olasılığı arasındaki ödünleşim.

Vickrey (İkinci Fiyat)

Dürüst teklif neden en iyisidir?

Çift Taraflı Piyasa

Alıcı ve satıcı emirleriyle piyasa takası.

Monopoly

İsteğe bağlı: satılmayan arsalar İngiliz açık artırmasıyla satılır.

Kaynaklar (1/2)

- V. Krishna, *Auction Theory*, 2. baskı, Academic Press, 2010 · P. Milgrom, *Putting Auction Theory to Work*, Cambridge University Press, 2004.
- T. Roughgarden, *Twenty Lectures on Algorithmic Game Theory*, Cambridge University Press, 2016; ders sayfası: CS364A · N. Nisan, T. Roughgarden, É. Tardos, V. Vazirani (ed.), *Algorithmic Game Theory*, Cambridge University Press, 2007 (Bölüm 9 ve 13).
- A. Karlin, Y. Peres, *Game Theory, Alive*, American Mathematical Society, 2017 (açık artırma bölümleri) · D. Easley, J. Kleinberg, *Networks, Crowds, and Markets*, Cambridge University Press, 2010 (Bölüm 9: Auctions).
- W. Vickrey, "Counterspeculation, Auctions, and Competitive Sealed Tenders", *Journal of Finance*, 1961 · R. Myerson, "Optimal Auction Design", *Mathematics of Operations Research*, 1981.
- J. G. Riley, W. F. Samuelson, "Optimal Auctions", *AER* 71(3), 1981 · J. Bulow, P. Klemperer, "Auctions versus Negotiations", *AER* 86(1), 1996.
- M. H. Rothkopf, T. J. Teisberg, E. P. Kahn, "Why Are Vickrey Auctions Rare?", *J. Political Economy* 98(1), 1990 · L. M. Ausubel, P. Milgrom, "The Lovely but Lonely Vickrey Auction", *Combinatorial Auctions*, MIT Press, 2006.

Kaynaklar (2/2)

- R. P. McAfee, “A Dominant Strategy Double Auction”, *Journal of Economic Theory*, 1992 · E. Capen, R. Clapp, W. Campbell, “Competitive Bidding in High-Risk Situations”, *Journal of Petroleum Technology*, 1971.
- J. H. Kagel, R. M. Harstad, D. Levin, “Information Impact and Allocation Rules in Auctions with Affiliated Private Values: A Laboratory Study”, *Econometrica* 55(6), 1987 · R. B. Myerson, M. A. Satterthwaite, “Efficient Mechanisms for Bilateral Trading”, *Journal of Economic Theory* 29(2), 1983.
- Ders sayfaları: N. Shah, CSC304 (Toronto) · M. Dinitz (Johns Hopkins) · M. Hoefer (Frankfurt).

Son

Bölüm 5: Açık Artırmalar

Sonraki bölüm: Bölüm 6: Çevrim İçi Platformlar

Çalışma soruları · Terim sözlüğü