Bölüm 7: Sosyal Seçim ve Oylama

Algoritmik Oyun Kuramı

Sosyal seçim · Çoğulluk ve mutlak çoğunluk · Borda ve onay · Anında ikinci tur · Condorcet ve Copeland · Puanlama kuralları · Condorcet döngüsü · Kemeny · Stratejik oy · Arrow ve Gibbard–Satterthwaite · May teoremi · Tek tepeli tercihler · Hesaplamalı sosyal seçim

Sercan KÜLCÜ · Giresun Üniversitesi · Bilgisayar Mühendisliği

© Sercan Külcü. Tüm hakları saklıdır. · Sunumun etkileşimli HTML sürümü: https://sercankulcu.github.io/files/agt/slides/Bolum_07_Sosyal_Secim_ve_Oylama.html

Oyunla Başlangıç: Condorcet Paradoksu

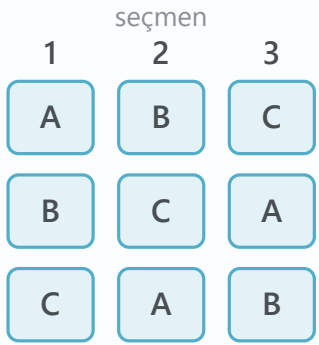
Oyunun kuralı

Her seçmen adayları sıralar; her aday çifti ikili çoğunlukla karşılaştırılır ve bütün rakiplerini yenen aday Condorcet kazananı olur.

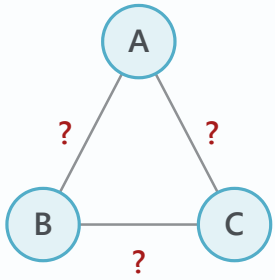
Merak sorusu

- Her seçmenin sıralaması kendi içinde tutarlıysa, topluluğun tercihi de tutarlı olmak zorunda mı?
- Oylamaların sırasını siz belirleseydiniz istediğiniz adayı kazandırabilir miydiniz?

[Condorcet Paradoksu oyunu →]



üstteki aday
daha çok tercih edilir



ikili çoğunlukta
kim kimi yener?

SOSYAL SEÇİM

Öğrenme Çıktıları

Sayım

Aynı tercih profilinde mutlak çoğunluk, çoğulluk, Borda, onay, anında ikinci tur ve Condorcet kurallarını adım adım uygulamak.

Paradokslar

Condorcet döngüsünü, gündem sırasının etkisini ve IRV'nin monotonluk ihlalini göstermek; Kemeny sıralamasını bulmak.

Stratejik oy

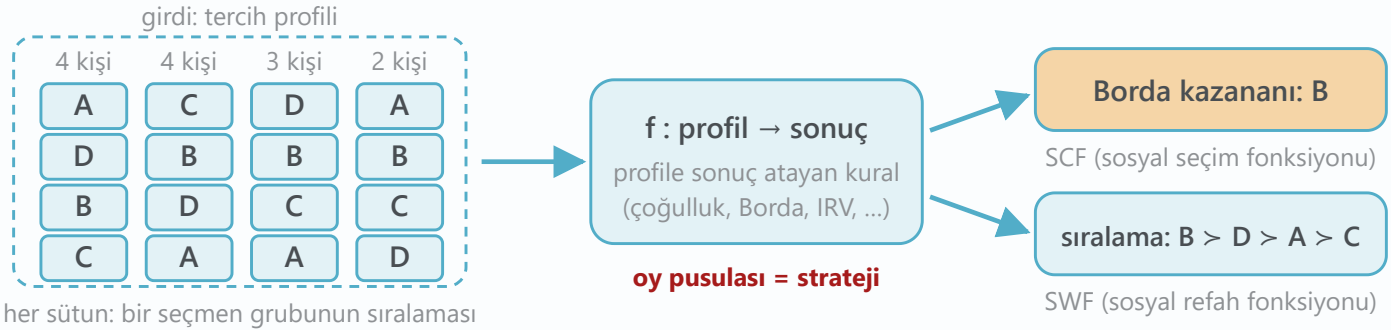
Çoğullukta taktik oyu ve Borda'da "gömme" manipülasyonunu sayısal olarak göstermek.

Teoremler

Arrow, Gibbard–Satterthwaite ve May teoremlerinin ne söylediğini açıklamak; tek tepeli tercihlerle kaçış yolunu tanımak.

Sosyal Seçim (Social Choice) Nedir?

- Sosyal seçim, bireysel tercihleri **toplu bir karara** veya toplumsal bir sıralamaya dönüştüren kuralları inceler.
- Amaç tek bir evrensel adalet tanımı bulmak değil; kuralların **hangi ölçütleri sağladığını** ve hangi ödünleşimleri yarattığını anlamaktır.
- Uygulamalar: seçimler, komite ve kulüp kararları, sınıf etkinliği seçimi, ortak kaynak tahsisi, çevrim içi anketler.



Mekanizma tasarımının parasız özel hâli: ödeme yok, yalnızca sıralamalar var.

SOSYAL SEÇİM

Örnek Tercih Profili

sıra	4 kişi	4 kişi	3 kişi	2 kişi
1.	A	C	D	A
2.	D	B	B	B
3.	B	D	C	C
4.	C	A	A	D

Tercih profili (preference profile); kuralların hepsi bu profille karşılaştırılır.

13 seçmen, 4 aday

Aynı sıralamayı veren seçmenler bir sütunda toplanmıştır.

Okuma

1. sütun: 4 seçmen için A ≻ D ≻ B ≻ C.

Mutlak çoğunluk eşiği

13 oyun yarısından fazlası: **7 oy**.

Aynı profile uygulanan kurallar **dört farklı kazanan** verir.

OYLAMA KURALLARI

Temel Oylama Kuralları

Kural	Seçmen ne bildirir?	Kazanan
Mutlak çoğunluk (majority)	Tek aday	Oyların **yarısından fazlasını** alan (yoksa kazanan yok)
Çoğulluk (plurality)	Tek aday	En çok birinci tercih alan; yarıyı geçmesi gerekmez
Borda sayımı	Tam sıralama	m aday için sıradaki yer m−1, m−2, …, 0 puan; en yüksek toplam
Onay (approval)	Onaylanan aday kümesi	En çok onay alan
Anında ikinci tur (IRV)	Tam sıralama	En az birinci tercih alan elenir, oyları aktarılır; çoğunluğa dek tekrarlanır
Condorcet	Tam sıralama	İkili karşılaştırmada **her rakibini yenen** aday (her zaman olmayabilir)

Dikkat: "en çok oyu alan kazanır" çoğulluğun tanımıdır; mutlak çoğunluk oyların %50'sinden fazlasını ister.

Mutlak Çoğunluk ve Çoğulluk Adım Adım

sıra	4 kişi	4 kişi	3 kişi	2 kişi
1.	A	C	D	A
2.	D	B	B	B
3.	B	D	C	C
4.	C	A	A	D

aday	oy
A	?
B	?
C	?
D	?

Mutlak çoğunlukta ve çoğullukta her seçmen yalnızca **birinci tercihini** bildirir; sıralamanın geri kalanı kullanılmaz.

Mutlak Çoğunluk ve Çoğulluk Adım Adım

sıra	4 kişi	4 kişi	3 kişi	2 kişi
1.	**A**	**C**	**D**	**A**
2.	D	B	B	B
3.	B	D	C	C
4.	C	A	A	D

aday	birinci tercih
A	**6**
B	0
C	4
D	3

Birinci sıra sayılır: A = 4 + 2 = 6, C = 4, D = 3, B = 0.

Mutlak Çoğunluk ve Çoğulluk Adım Adım

sıra	4 kişi	4 kişi	3 kişi	2 kişi
1.	**A**	**C**	**D**	**A**
2.	D	B	B	B
3.	B	D	C	C
4.	C	A	A	D

aday	birinci tercih
A	**6**
B	0
C	4
D	3

Mutlak çoğunluk: 6 < 7 ⇒ **kazanan yok**. Çoğulluk: en çok oy alan **A** kazanır (oyların yalnızca 6/13'ü).

Mutlak Çoğunluk ve Çoğulluk Adım Adım

sıra	4 kişi	4 kişi	3 kişi	2 kişi
1.	**A**	**C**	~~D~~	**A**
2.	~~D~~	~~B~~	~~B~~	~~B~~
3.	~~B~~	~~D~~	**C**	C
4.	C	A	A	~~D~~

aday	ikinci tur
A	6
~~B~~	—
C	**7**
~~D~~	—

İki turlu sistem (ör. cumhurbaşkanlığı seçimi): ilk iki A ve C ikinci tura kalır. A = 6, C = 7 ⇒ **C** kazanır.

Borda Sayımı Adım Adım

sıra	4 kişi	4 kişi	3 kişi	2 kişi
1.	A	C	D	A
2.	D	B	B	B
3.	B	D	C	C
4.	C	A	A	D

aday	toplam
A	0
B	0
C	0
D	0

Borda: 4 aday için 1. sıra **3**, 2. sıra **2**, 3. sıra **1**, 4. sıra **0** puan. Her grubun katkısı = seçmen sayısı × puan.

Borda Sayımı Adım Adım

sıra	4 kişi	4 kişi	3 kişi	2 kişi
1.	**A**	C	D	A
2.	D	B	B	B
3.	B	D	C	C
4.	C	A	A	D

aday	G1	toplam
A	12	12
B	4	4
C	0	0
D	8	8

G1–G4: grupların katkısı

1. grup (4 seçmen, A ≻ D ≻ B ≻ C): A: 4×3 = 12, D: 4×2 = 8, B: 4×1 = 4, C: 4×0 = 0.

Borda Sayımı Adım Adım

sıra	4 kişi	4 kişi	3 kişi	2 kişi
1.	A	**C**	D	A
2.	D	B	B	B
3.	B	D	C	C
4.	C	A	A	D

aday	G1	G2	toplam
A	12	0	12
B	4	8	12
C	0	12	12
D	8	4	12

G1–G4: grupların katkısı

2. grup (4 seçmen, C ≻ B ≻ D ≻ A): C: 4×3 = 12, B: 4×2 = 8, D: 4×1 = 4, A: 4×0 = 0.

Borda Sayımı Adım Adım

sıra	4 kişi	4 kişi	3 kişi	2 kişi
1.	A	C	**D**	A
2.	D	B	B	B
3.	B	D	C	C
4.	C	A	A	D

aday	G1	G2	G3	toplam
A	12	0	0	12
B	4	8	6	18
C	0	12	3	15
D	8	4	9	21

G1–G4: grupların katkısı

3. grup (3 seçmen, D ≻ B ≻ C ≻ A): D: 3×3 = 9, B: 3×2 = 6, C: 3×1 = 3, A: 3×0 = 0.

Borda Sayımı Adım Adım

sıra	4 kişi	4 kişi	3 kişi	2 kişi
1.	A	C	D	**A**
2.	D	B	B	B
3.	B	D	C	C
4.	C	A	A	D

aday	G1	G2	G3	G4	toplam
A	12	0	0	6	18
B	4	8	6	4	22
C	0	12	3	2	17
D	8	4	9	0	21

G1–G4: grupların katkısı

4. grup (2 seçmen, A ≻ B ≻ C ≻ D): A: 2×3 = 6, B: 2×2 = 4, C: 2×1 = 2, D: 2×0 = 0.

Borda Sayımı Adım Adım

sıra	4 kişi	4 kişi	3 kişi	2 kişi
1.	A	C	D	A
2.	D	B	B	B
3.	B	D	C	C
4.	C	A	A	D

aday	G1	G2	G3	G4	toplam
A	12	0	0	6	18
B	**4**	**8**	**6**	**4**	**22**
C	0	12	3	2	17
D	8	4	9	0	21

Toplam: B = 22, D = 21, A = 18, C = 17 ⇒ Borda kazananı **B**; oysa B'yi kimse birinci sıraya koymadı!

Onay Oylaması (Approval Voting)

sıra	4 kişi	4 kişi	3 kişi	2 kişi
1.	A	C	D	A
2.	D	B	B	B
3.	B	D	C	C
4.	C	A	A	D

aday	onay
A	?
B	?
C	?
D	?

Onay oylamasında seçmen kabul ettiği **tüm** adayları işaretler. Varsayım: her seçmen ilk iki tercihini onaylıyor.

Onay Oylaması (Approval Voting)

sıra	4 kişi	4 kişi	3 kişi	2 kişi
1.	**A**	**C**	**D**	**A**
2.	**D**	**B**	**B**	**B**
3.	B	D	C	C
4.	C	A	A	D

aday	onay
A	6
B	**9**
C	4
D	7

Onaylar: A = 4 + 2 = 6, B = 4 + 3 + 2 = 9, C = 4, D = 4 + 3 = 7 ⇒ onay kazananı **B** (herkes ilk ikisini onaylarsa onay oylaması 2-onay kuralına dönüşür).

Onay Oylaması (Approval Voting)

sıra	4 kişi	4 kişi	3 kişi	2 kişi
1.	**A**	**C**	**D**	**A**
2.	D	B	B	B
3.	B	D	C	C
4.	C	A	A	D

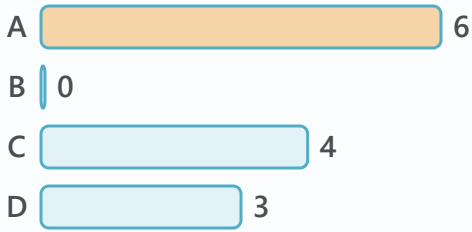
aday	onay (yalnız 1.)
A	**6**
B	0
C	4
D	3

Sonuç onay eşiğine bağlıdır: herkes yalnızca birinci tercihini onaylasaydı onay = çoğulluk olur ve A kazanırdı.

Pozisyonel Puanlama Kuralları

Pozisyonel puanlama kuralı (scoring rule): m aday için bir vektör $s = (s_1 \geq s_2 \geq \ldots \geq s_m)$; k. sıraya konan aday s_k puan alır. Çoğulluk, k-onay, Borda ve veto bu ailenin üyeleridir.



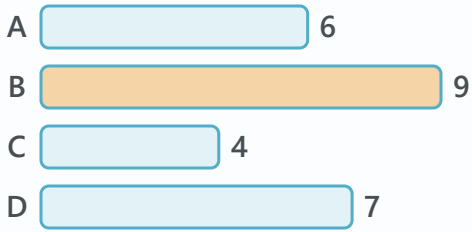


s = (1, 0, 0, 0) (çoğulluk): A = 6, C = 4, D = 3, B = 0 ⇒ kazanan **A**. Turuncu hücreler puan alan sıralar.

Pozisyonel Puanlama Kuralları

Pozisyonel puanlama kuralı (scoring rule): m aday için bir vektör $s = (s_1 \geq s_2 \geq \ldots \geq s_m)$; k. sıraya konan aday s_k puan alır. Çoğulluk, k-onay, Borda ve veto bu ailenin üyeleridir.





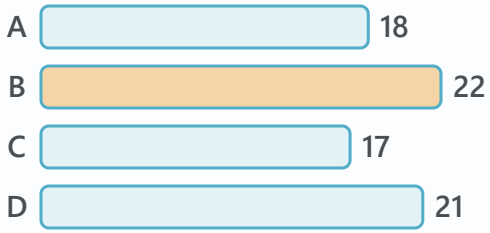
s = (1, 1, 0, 0) (2-onay (ilk iki)): B = 9, D = 7, A = 6, C = 4 ⇒ kazanan **B**.

Pozisyonel Puanlama Kuralları

Pozisyonel puanlama kuralı (scoring rule): m aday için bir vektör $s = (s_1 \geq s_2 \geq \ldots \geq s_m)$; k. sıraya konan aday s_k puan alır. Çoğulluk, k-onay, Borda ve veto bu ailenin üyeleridir.



Borda

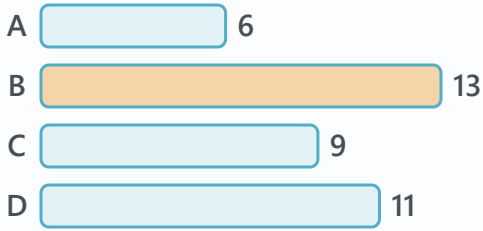


$s = (3, 2, 1, 0)$ (Borda): B = 22, D = 21, A = 18, C = 17 ⇒ kazanan **B**.

Pozisyonel Puanlama Kuralları

Pozisyonel puanlama kuralı (scoring rule): m aday için bir vektör $s = (s_1 \geq s_2 \geq \ldots \geq s_m)$; k. sıraya konan aday s_k puan alır. Çoğulluk, k-onay, Borda ve veto bu ailenin üyeleridir.



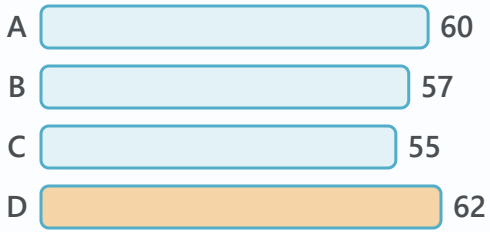


$s = (1, 1, 1, 0)$ (veto (sonuncuya 0)): B = 13, D = 11, C = 9, A = 6 ⇒ kazanan **B**.

Pozisyonel Puanlama Kuralları

Pozisyonel puanlama kuralı (scoring rule): m aday için bir vektör $s = (s_1 \geq s_2 \geq \ldots \geq s_m)$; k. sıraya konan aday s_k puan alır. Çoğulluk, k-onay, Borda ve veto bu ailenin üyeleridir.





s = (10, 5, 3, 0) (başka bir vektör): D = 62, A = 60, B = 57, C = 55 ⇒ kazanan **D**. Bu vektör Condorcet kazananı D'yi seçer.

OYLAMA KURALLARI

Anında İkinci Tur (IRV) Adım Adım

sıra	4 kişi	4 kişi	3 kişi	2 kişi
1.	A	C	D	A
2.	D	B	B	B
3.	B	D	C	C
4.	C	A	A	D

aday	1. tur
A	?
B	?
C	?
D	?

Anında ikinci tur (instant-runoff, tercihli oy): her turda seçmenin **hâlâ yarışan** en üst adayı sayılır; çoğunluk yoksa en az oylu elenir.

Anında İkinci Tur (IRV) Adım Adım

sıra	4 kişi	4 kişi	3 kişi	2 kişi
1.	**A**	**C**	**D**	**A**
2.	D	B	B	B
3.	B	D	C	C
4.	C	A	A	D

aday	1. tur
A	6
B	0
C	4
D	3

1. tur: A = 6, B = 0, C = 4, D = 3. Çoğunluk yok (7 gerekir).

Anında İkinci Tur (IRV) Adım Adım

sıra	4 kişi	4 kişi	3 kişi	2 kişi
1.	**A**	**C**	**D**	**A**
2.	D	~~B~~	~~B~~	~~B~~
3.	~~B~~	D	C	C
4.	C	A	A	D

aday	1. tur
A	6
B	0
C	4
D	3

En az oylu **B** (0 oy) elenir. Onu birinci sıraya koyan yok, aktarılacak oy da yok.

Anında İkinci Tur (IRV) Adım Adım

sıra	4 kişi	4 kişi	3 kişi	2 kişi
1.	**A**	**C**	**D**	**A**
2.	D	~~B~~	~~B~~	~~B~~
3.	~~B~~	D	C	C
4.	C	A	A	D

aday	2. tur
A	6
B	—
C	4
D	3

2. tur: A = 6, C = 4, D = 3. Çoğunluk yok (7 gerekir).

Anında İkinci Tur (IRV) Adım Adım

sıra	4 kişi	4 kişi	3 kişi	2 kişi
1.	**A**	**C**	~~D~~	**A**
2.	~~D~~	~~B~~	~~B~~	~~B~~
3.	~~B~~	~~D~~	**C**	C
4.	C	A	A	~~D~~

aday	2. tur
A	6
B	—
C	4
D	3

En az oylu **D** (3) elenir. Oyları sıradaki yarışan adaya aktarılır: 3 oy D → C.

Anında İkinci Tur (IRV) Adım Adım

sıra	4 kişi	4 kişi	3 kişi	2 kişi
1.	**A**	**C**	~~D~~	**A**
2.	~~D~~	~~B~~	~~B~~	~~B~~
3.	~~B~~	~~D~~	**C**	C
4.	C	A	A	~~D~~

aday	3. tur
A	6
B	—
C	**7**
D	—

3. tur: A = 6, C = 7. 7 ≥ 7 ⇒ **C** mutlak çoğunlukla kazanır.

Condorcet: İkili Karşılaştırmalar Adım Adım

sıra	4 kişi	4 kişi	3 kişi	2 kişi
1.	A	C	D	A
2.	D	B	B	B
3.	B	D	C	C
4.	C	A	A	D

x ╲ y	A	B	C	D	galibiyet
A	—				0
B		—			0
C			—		0
D				—	0

Condorcet: her aday çifti için "kaç seçmen x'i y'ye tercih ediyor?" Tablo hücresi (x, y) = x'i y'ye tercih eden seçmen sayısı. 4 aday ⇒ 6 karşılaştırma.

Condorcet: İkili Karşılaştırmalar Adım Adım

sıra	4 kişi	4 kişi	3 kişi	2 kişi
1.	**A**	C	D	**A**
2.	D	**B**	**B**	**B**
3.	**B**	D	C	C
4.	C	**A**	**A**	D

x ╲ y	A	B	C	D	galibiyet
A	—	6			0
B	**7**	—			1
C			—		0
D				—	0

A – B: A'yı tercih eden 6, B'yi tercih eden 7 ⇒ **B** kazanır.

Condorcet: İkili Karşılaştırmalar Adım Adım

sıra	4 kişi	4 kişi	3 kişi	2 kişi
1.	**A**	**C**	D	**A**
2.	D	B	B	B
3.	B	D	**C**	**C**
4.	**C**	**A**	**A**	D

x ╲ y	A	B	C	D	galibiyet
A	—	6	6		0
B	7	—			1
C	**7**		—		1
D				—	0

A – C: A'yı tercih eden 6, C'yi tercih eden 7 ⇒ **C** kazanır.

Condorcet: İkili Karşılaştırmalar Adım Adım

sıra	4 kişi	4 kişi	3 kişi	2 kişi
1.	**A**	C	**D**	**A**
2.	**D**	B	B	B
3.	B	**D**	C	C
4.	C	**A**	**A**	**D**

x ╲ y	A	B	C	D	galibiyet
A	—	6	6	6	0
B	7	—			1
C	7		—		1
D	**7**			—	1

A – D: A'yı tercih eden 6, D'yi tercih eden 7 ⇒ **D** kazanır.

Condorcet: İkili Karşılaştırmalar Adım Adım

sıra	4 kişi	4 kişi	3 kişi	2 kişi
1.	A	**C**	D	A
2.	D	**B**	**B**	**B**
3.	**B**	D	**C**	**C**
4.	**C**	A	A	D

x ╲ y	A	B	C	D	galibiyet
A	—	6	6	6	0
B	7	—	**9**		2
C	7	4	—		1
D	7			—	1

B – C: B'yi tercih eden 9, C'yi tercih eden 4 ⇒ **B** kazanır.

Condorcet: İkili Karşılaştırmalar Adım Adım

sıra	4 kişi	4 kişi	3 kişi	2 kişi
1.	A	C	**D**	A
2.	**D**	**B**	**B**	**B**
3.	**B**	**D**	C	C
4.	C	A	A	**D**

x ╲ y	A	B	C	D	galibiyet
A	—	6	6	6	0
B	7	—	9	6	2
C	7	4	—		1
D	7	**7**		—	2

B – D: B'yi tercih eden 6, D'yi tercih eden 7 ⇒ **D** kazanır.

Condorcet: İkili Karşılaştırmalar Adım Adım

sıra	4 kişi	4 kişi	3 kişi	2 kişi
1.	A	**C**	**D**	A
2.	**D**	B	B	B
3.	B	**D**	**C**	**C**
4.	**C**	A	A	**D**

x ╲ y	A	B	C	D	galibiyet
A	—	6	6	6	0
B	7	—	9	6	2
C	7	4	—	6	1
D	7	7	**7**	—	3

C – D: C'yi tercih eden 6, D'yi tercih eden 7 ⇒ **D** kazanır.

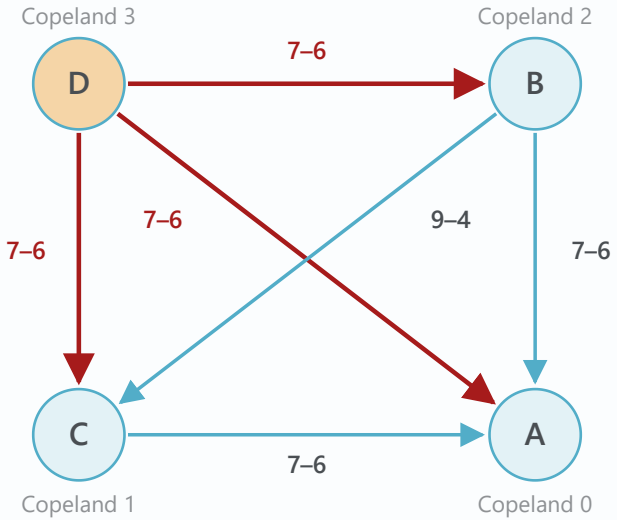
Condorcet: İkili Karşılaştırmalar Adım Adım

sıra	4 kişi	4 kişi	3 kişi	2 kişi
1.	A	C	D	A
2.	D	B	B	B
3.	B	D	C	C
4.	C	A	A	D

x ╲ y	A	B	C	D	galibiyet
A	—	6	6	6	0
B	7	—	9	6	2
C	7	4	—	6	1
D	**7**	**7**	**7**	**—**	**3**

D üç rakibini de 7–6 yener: Condorcet kazananı D. A ise herkese yenilir (Condorcet kaybedeni), ama çoğulluğu kazanmıştı!

İkili Çoğunluk Çizgesi ve Copeland



- Her aday bir düğüm; **x** → **y** oku: çoğunluk x'i y'ye tercih ediyor (etiket: oy sayıları).
- Bu çizge bir **turnuvadır**: her çift arasında tam olarak bir ok vardır (seçmen sayısı tek olduğu için eşitlik yok).
- **Condorcet kazananı** = hiç ok almayan düğüm (D). Condorcet kaybedeni = hep ok alan (A).
- **Copeland** kuralı: çıkan ok sayısı (galibiyet) en çok olan kazanır ⇒ D (3) ≻ B (2) ≻ C (1) ≻ A (0).

Her aday en az bir ok alıyorsa (hiç ok almayan düğüm yoksa) Condorcet kazananı yoktur; tüm adayları kapsayan bir **döngü** buna yol açar.

OYLAMA KURALLARI

Aynı Profil, Dört Farklı Kazanan

Kural	Sayım	Kazanan
Mutlak çoğunluk	A 6 < 7	yok
Çoğulluk	A 6, C 4, D 3, B 0	**A**
İki turlu sistem	C 7 – A 6	**C**
Anında ikinci tur	C 7 – A 6 (3. tur)	**C**
Borda	B 22, D 21, A 18, C 17	**B**
2-onay (ilk iki)	B 9, D 7, A 6, C 4	**B**
Condorcet	D 3 galibiyet	**D**
Kemeny	D ≻ B ≻ C ≻ A (44 uyum)	**D**

Ders 1

"Halkın iradesi" kurala bağlıdır: kural, seçmenler kadar sonucu belirler.

Ders 2

Çoğulluk Condorcet *kaybedenini* (A) seçebilir; IRV Condorcet *kazananını* (D) eleyebilir.

Kod İzleme: `borda(oylar)`

4 kişi	A	D	B	C
4 kişi	C	B	D	A
3 kişi	D	B	C	A
2 kişi	A	B	C	D

A	0
B	0
C	0
D	0

oylar: satır = seçmen grubu; 1. sıra 3 puan … 4. sıra 0

oylar = 4 grup (satırlar). `puan` sözlüğü boş başlar; her aday için puan birikecek.

Adım 1 / 11

```
▶ def borda(oylar):
▶   puan = {}
3   for adet, sira in oylar:
4     m = len(sira)
5     for k, aday in enumerate(sira):
6       p = adet * (m - 1 - k)
7       puan[aday] = puan.get(aday, 0) + p
8   kazanan = max(puan, key=puan.get)
9   return kazanan, puan
```

DEĞİŞKENLER

`puan = {}`

Kod İzleme: `borda(oylar)`

4 kişi	A	D	B	C
4 kişi	C	B	D	A
3 kişi	D	B	C	A
2 kişi	A	B	C	D

A	0
B	0
C	0
D	0

oylar: satır = seçmen grubu; 1. sıra 3 puan … 4. sıra 0

1. grup: 4 seçmen, sıralama ADBC. m = 4 aday: sıradaki yer k için puan m − 1 − k.

Adım 2 / 11

```
1 def borda(oylar):
2   puan = {}
▶   for adet, sira in oylar:
▶     m = len(sira)
5     for k, aday in enumerate(sira):
6       p = adet * (m - 1 - k)
7       puan[aday] = puan.get(aday, 0) + p
8   kazanan = max(puan, key=puan.get)
9   return kazanan, puan
```

DEĞİŞKENLER

`adet = 4` `sira = "ADBC"` `m = 4` `puan = {}`

Kod İzleme: `borda(oylar)`

4 kişi	A	D	B	C
4 kişi	C	B	D	A
3 kişi	D	B	C	A
2 kişi	A	B	C	D

A	12
B	0
C	0
D	0

oylar: satır = seçmen grubu; 1. sıra 3 puan … 4. sıra 0

k = 0: A 1. sırada ⇒ p = 4 × (3 − 0) = **12**; A toplamı 12.

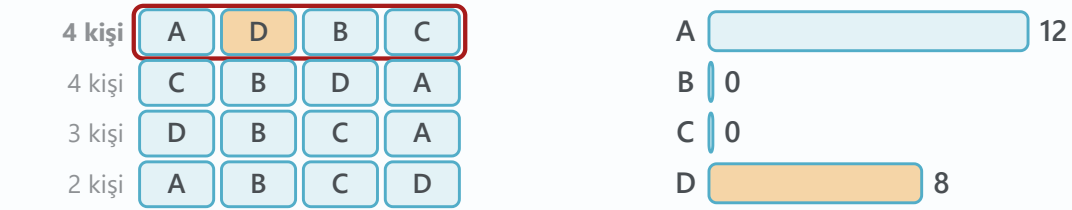
Adım 3 / 11

```
1 def borda(oylar):
2   puan = {}
3   for adet, sira in oylar:
4     m = len(sira)
▶     for k, aday in enumerate(sira):
▶       p = adet * (m - 1 - k)
▶       puan[aday] = puan.get(aday, 0) + p
8   kazanan = max(puan, key=puan.get)
9   return kazanan, puan
```

DEĞİŞKENLER

`adet = 4` `sira = "ADBC"` `k = 0` `aday = A` `p = 12` `puan = {A: 12}`

Kod İzleme: `borda(oylar)`



oylar: satır = seçmen grubu; 1. sıra 3 puan … 4. sıra 0

k = 1: D 2. sırada ⇒ p = 4 × (3 − 1) = **8**; D toplamı 8.

Adım 4 / 11

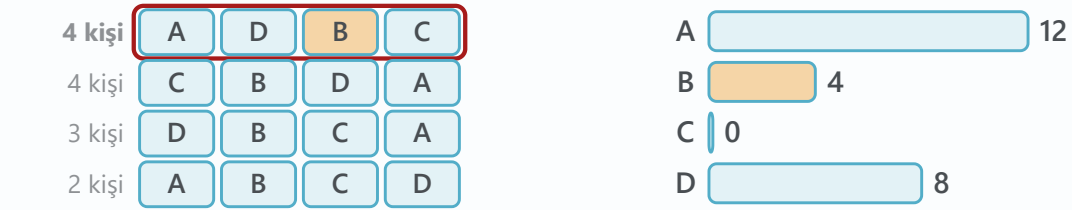
```
1  def borda(oylar):
2      puan = {}
3      for adet, sira in oylar:
4          m = len(sira)
▶          for k, aday in enumerate(sira):
▶              p = adet * (m - 1 - k)
▶              puan[aday] = puan.get(aday, 0) + p
8      kazanan = max(puan, key=puan.get)
9      return kazanan, puan
```

DEĞİŞKENLER

`adet = 4` `sira = "ADBC"` `k = 1` `aday = D` `p = 8`

`puan = {A: 12, D: 8}`

Kod İzleme: `borda(oylar)`



oylar: satır = seçmen grubu; 1. sıra 3 puan … 4. sıra 0

k = 2: B 3. sırada ⇒ p = 4 × (3 − 2) = **4**; B toplamı 4.

Adım 5 / 11

```
1 def borda(oylar):
2   puan = {}
3   for adet, sira in oylar:
4     m = len(sira)
      for k, aday in enumerate(sira):
        p = adet * (m - 1 - k)
        puan[aday] = puan.get(aday, 0) + p
8   kazanan = max(puan, key=puan.get)
9   return kazanan, puan
```

DEĞİŞKENLER

`adet = 4` `sira = "ADBC"` `k = 2` `aday = B` `p = 4`

`puan = {A: 12, D: 8, B: 4}`

Kod İzleme: `borda(oylar)`

4 kişi	A	D	B	C
4 kişi	C	B	D	A
3 kişi	D	B	C	A
2 kişi	A	B	C	D

A	12
B	4
C	0
D	8

oylar: satır = seçmen grubu; 1. sıra 3 puan … 4. sıra 0

k = 3: C 4. sırada ⇒ p = 4 × (3 − 3) = **0**; C toplamı 0.

Adım 6 / 11

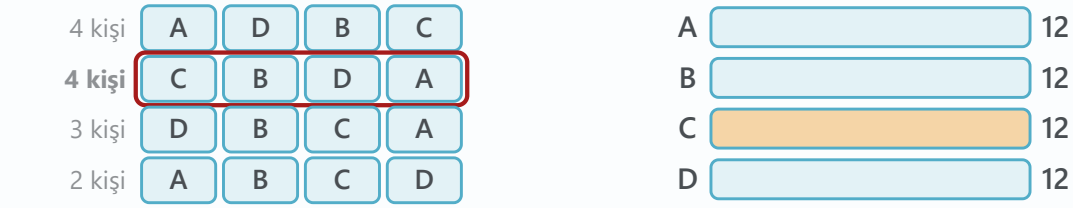
```
1 def borda(oylar):
2   puan = {}
3   for adet, sira in oylar:
4     m = len(sira)
▶     for k, aday in enumerate(sira):
▶       p = adet * (m - 1 - k)
▶       puan[aday] = puan.get(aday, 0) + p
8   kazanan = max(puan, key=puan.get)
9   return kazanan, puan
```

DEĞİŞKENLER

`adet = 4` `sira = "ADBC"` `k = 3` `aday = C` `p = 0`

`puan = {A: 12, D: 8, B: 4, C: 0}`

Kod İzleme: `borda(oylar)`



oylar: satır = seçmen grubu; 1. sıra 3 puan … 4. sıra 0

2. grup (4 × CBDA): iç döngü 4 kez döner; C +12, B +8, D +4, A +0.

Adım 7 / 11

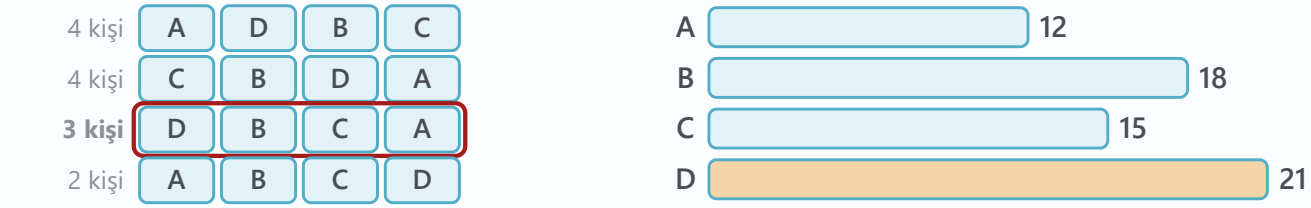
```
1 def borda(oylar):
2   puan = {}
▶   for adet, sira in oylar:
▶     m = len(sira)
▶     for k, aday in enumerate(sira):
▶       p = adet * (m - 1 - k)
▶       puan[aday] = puan.get(aday, 0) + p
8   kazanan = max(puan, key=puan.get)
9   return kazanan, puan
```

DEĞİŞKENLER

`adet = 4` `sira = "CBDA"` `m = 4`

`puan = {A: 12, D: 12, B: 12, C: 12}`

Kod İzleme: `borda(oylar)`



oylar: satır = seçmen grubu; 1. sıra 3 puan … 4. sıra 0

3. grup (3 × DBCA): iç döngü 4 kez döner; D +9, B +6, C +3, A +0.

Adım 8 / 11

```
1 def borda(oylar):
2   puan = {}
▶   for adet, sira in oylar:
▶     m = len(sira)
▶     for k, aday in enumerate(sira):
▶       p = adet * (m - 1 - k)
▶       puan[aday] = puan.get(aday, 0) + p
8   kazanan = max(puan, key=puan.get)
9   return kazanan, puan
```

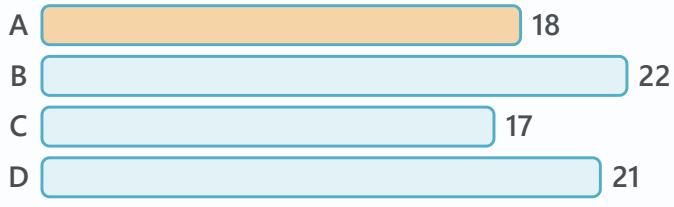
DEĞİŞKENLER

`adet = 3` `sira = "DBCA"` `m = 4`

`puan = {A: 12, D: 21, B: 18, C: 15}`

Kod İzleme: `borda(oylar)`

4 kişi	A	D	B	C
4 kişi	C	B	D	A
3 kişi	D	B	C	A
2 kişi	A	B	C	D



oylar: satır = seçmen grubu; 1. sıra 3 puan … 4. sıra 0

4. grup (2 × ABCD): iç döngü 4 kez döner; A +6, B +4, C +2, D +0.

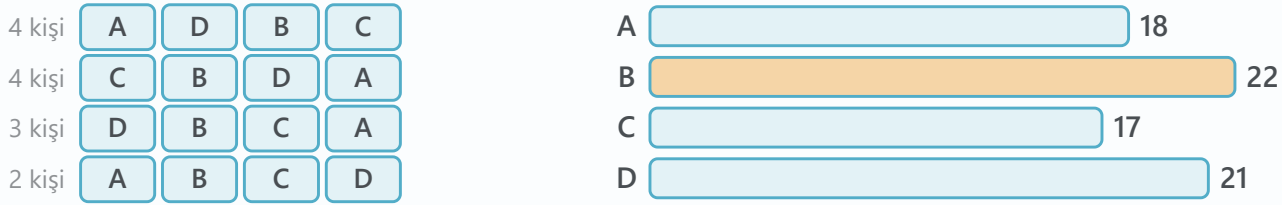

```
1 def borda(oylar):
2   puan = {}
▶   for adet, sira in oylar:
▶     m = len(sira)
▶     for k, aday in enumerate(sira):
▶       p = adet * (m - 1 - k)
▶       puan[aday] = puan.get(aday, 0) + p
8   kazanan = max(puan, key=puan.get)
9   return kazanan, puan
```

DEĞİŞKENLER

`adet = 2` `sira = "ABCD"` `m = 4`

`puan = {A: 18, D: 21, B: 22, C: 17}`

Kod İzleme: `borda(oylar)`



oylar: satır = seçmen grubu; 1. sıra 3 puan ... 4. sıra 0

`max`: puanı en yüksek aday **B** (22); hiçbir seçmen B'yi birinci sıraya koymadı. Eşitlikte `max` ilk gördüğünü seçer: gerçek kurallarda eşitlik bozma kuralı gerekir.

Adım 10 / 11

```
1 def borda(oylar):
2   puan = {}
3   for adet, sira in oylar:
4     m = len(sira)
5     for k, aday in enumerate(sira):
6       p = adet * (m - 1 - k)
7       puan[aday] = puan.get(aday, 0) + p
▶   kazanan = max(puan, key=puan.get)
9   return kazanan, puan
```

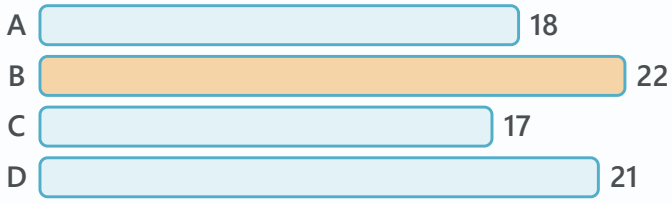
DEĞİŞKENLER

`puan = {A: 18, D: 21, B: 22, C: 17}`

`kazanan = B`

Kod İzleme: `borda(oylar)`

4 kişi	A	D	B	C
4 kişi	C	B	D	A
3 kişi	D	B	C	A
2 kişi	A	B	C	D



oylar: satır = seçmen grubu; 1. sıra 3 puan … 4. sıra 0

Karmaşıklık: n seçmen grubu, m aday ⇒ O(n·m). Sayım kolaydır; zor olan kuralın özellikleridir.

Adım 11 / 11

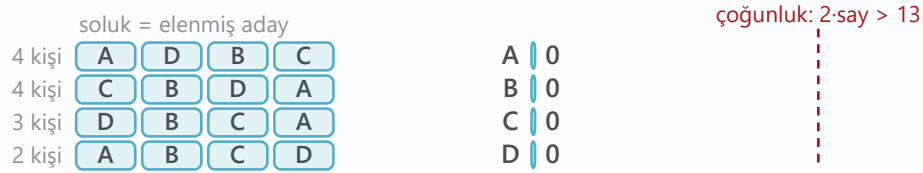
```
1 def borda(oylar):
2   puan = {}
3   for adet, sira in oylar:
4     m = len(sira)
5     for k, aday in enumerate(sira):
6       p = adet * (m - 1 - k)
7       puan[aday] = puan.get(aday, 0) + p
8   kazanan = max(puan, key=puan.get)
▶   return kazanan, puan
```

DEĞİŞKENLER

`puan = {A: 18, D: 21, B: 22, C: 17}`

`kazanan = B`

Kod İzleme: `ikinci_tur(oylar, adaylar)`



Tüm adaylar yarışta. Her tur: say, çoğunluğu kontrol et, yoksa en zayıfı ele.

Adım 1 / 15

```
▶ def ikinci_tur(oylar, adaylar):
▶   kalan = set(adaylar)
3   while True:
4     say = {a: 0 for a in kalan}
5     for adet, sira in oylar:
6       ilk = next(a for a in sira if a in kalan)
7       say[ilk] += adet
8     lider = max(say, key=say.get)
9     if 2 * say[lider] > sum(say.values()):
10      return lider
11    kaybeden = min(say, key=say.get)
12    kalan.remove(kaybeden)
```

DEĞİŞKENLER

`kalan = {A, B, C, D}`

Kod İzleme: `ikinci_tur(oylar, adaylar)`



1. tur başlar: yarışan her aday için sayaç 0.

Adım 2 / 15

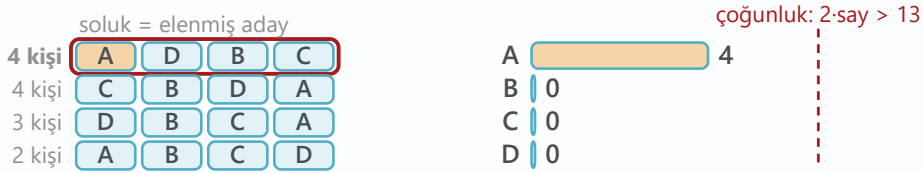
```
1  def ikinci_tur(oylar, adaylar):
2      kalan = set(adaylar)
▶      while True:
▶          say = {a: 0 for a in kalan}
5          for adet, sira in oylar:
6              ilk = next(a for a in sira if a in kalan)
7              say[ilk] += adet
8          lider = max(say, key=say.get)
9          if 2 * say[lider] > sum(say.values()):
10             return lider
11         kaybeden = min(say, key=say.get)
12         kalan.remove(kaybeden)
```

DEĞİŞKENLER

`kalan = {A, B, C, D}`

`say = {A: 0, B: 0, C: 0, D: 0}`

Kod İzleme: `ikinci_tur(oylar, adaylar)`



ADBC: yarışan ilk aday A ⇒ say[A] += 4.

Adım 3 / 15

```
1  def ikinci_tur(oylar, adaylar):
2      kalan = set(adaylar)
3      while True:
4          say = {a: 0 for a in kalan}
▶          for adet, sira in oylar:
▶              ilk = next(a for a in sira if a in kalan)
▶              say[ilk] += adet
8          lider = max(say, key=say.get)
9          if 2 * say[lider] > sum(say.values()):
10             return lider
11         kaybeden = min(say, key=say.get)
12         kalan.remove(kaybeden)
```

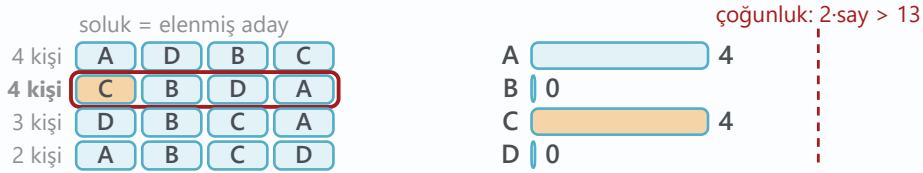
DEĞİŞKENLER

`kalan = {A, B, C, D}` `adet = 4`

`sira = "ADBC"` `ilk = A`

`say = {A: 4, B: 0, C: 0, D: 0}`

Kod İzleme: `ikinci_tur(oylar, adaylar)`



CBDA: yarışan ilk aday C ⇒ say[C] += 4.

Adım 4 / 15

```
1  def ikinci_tur(oylar, adaylar):
2      kalan = set(adaylar)
3      while True:
4          say = {a: 0 for a in kalan}
▶          for adet, sira in oylar:
▶              ilk = next(a for a in sira if a in kalan)
▶              say[ilk] += adet
8          lider = max(say, key=say.get)
9          if 2 * say[lider] > sum(say.values()):
10             return lider
11         kaybeden = min(say, key=say.get)
12         kalan.remove(kaybeden)
```

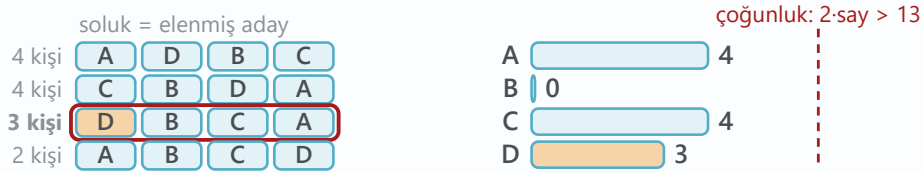
DEĞİŞKENLER

`kalan = {A, B, C, D}` `adet = 4`

`sira = "CBDA"` `ilk = C`

`say = {A: 4, B: 0, C: 4, D: 0}`

Kod İzleme: `ikinci_tur(oylar, adaylar)`



DBCA: yarışan ilk aday D ⇒ say[D] += 3.

Adım 5 / 15

```
1  def ikinci_tur(oylar, adaylar):
2    kalan = set(adaylar)
3    while True:
4      say = {a: 0 for a in kalan}
▶      for adet, sira in oylar:
▶        ilk = next(a for a in sira if a in kalan)
▶        say[ilk] += adet
8      lider = max(say, key=say.get)
9      if 2 * say[lider] > sum(say.values()):
10       return lider
11     kaybeden = min(say, key=say.get)
12     kalan.remove(kaybeden)
```

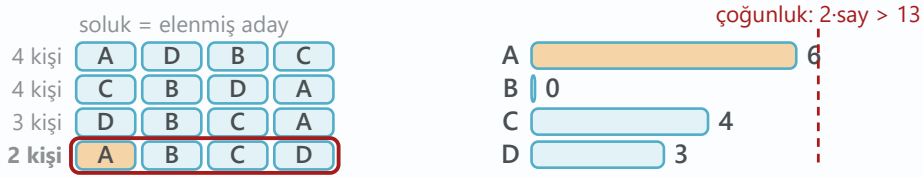
DEĞİŞKENLER

`kalan = {A, B, C, D}` `adet = 3`

`sira = "DBCA"` `ilk = D`

`say = {A: 4, B: 0, C: 4, D: 3}`

Kod İzleme: `ikinci_tur(oylar, adaylar)`



ABCD: yarışan ilk aday A ⇒ say[A] += 2.

Adım 6 / 15

```
1  def ikinci_tur(oylar, adaylar):
2      kalan = set(adaylar)
3      while True:
4          say = {a: 0 for a in kalan}
▶          for adet, sira in oylar:
▶              ilk = next(a for a in sira if a in kalan)
▶              say[ilk] += adet
8          lider = max(say, key=say.get)
9          if 2 * say[lider] > sum(say.values()):
10             return lider
11         kaybeden = min(say, key=say.get)
12         kalan.remove(kaybeden)
```

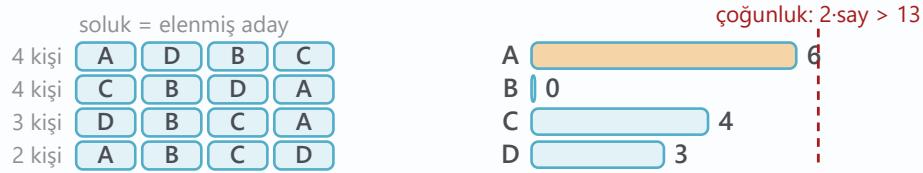
DEĞİŞKENLER

`kalan = {A, B, C, D}` `adet = 2`

`sira = "ABCD"` `ilk = A`

`say = {A: 6, B: 0, C: 4, D: 3}`

Kod İzleme: `ikinci_tur(oylar, adaylar)`



lider = A: 2 × 6 = 12 ≤ 13 ⇒ çoğunluk yok.

Adım 7 / 15

```
1  def ikinci_tur(oylar, adaylar):
2      kalan = set(adaylar)
3      while True:
4          say = {a: 0 for a in kalan}
5          for adet, sira in oylar:
6              ilk = next(a for a in sira if a in kalan)
7              say[ilk] += adet
▶          lider = max(say, key=say.get)
▶          if 2 * say[lider] > sum(say.values()):
10             return lider
11         kaybeden = min(say, key=say.get)
12         kalan.remove(kaybeden)
```

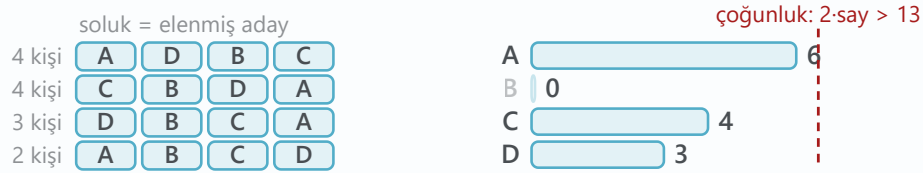
DEĞİŞKENLER

`kalan = {A, B, C, D}`

`say = {A: 6, B: 0, C: 4, D: 3}`

`lider = A`

Kod İzleme: `ikinci_tur(oylar, adaylar)`



En az oylu **B** (0) elenir; kalan = {A, C, D}. Eşitlikte `min` ve `max` ilk gördüğünü seçer: gerçek kurallarda eşitlik bozma kuralı gerekir.

Adım 8 / 15

```
1  def ikinci_tur(oylar, adaylar):
2    kalan = set(adaylar)
3    while True:
4      say = {a: 0 for a in kalan}
5      for adet, sira in oylar:
6        ilk = next(a for a in sira if a in kalan)
7        say[ilk] += adet
8      lider = max(say, key=say.get)
9      if 2 * say[lider] > sum(say.values()):
10       return lider
▶      kaybeden = min(say, key=say.get)
▶      kalan.remove(kaybeden)
```

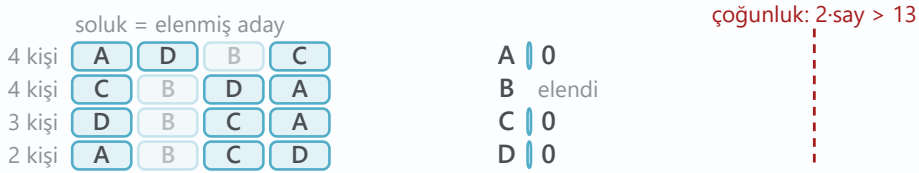
DEĞİŞKENLER

`kalan = {A, C, D}`

`say = {A: 6, B: 0, C: 4, D: 3}`

`kaybeden = B`

Kod İzleme: `ikinci_tur(oylar, adaylar)`



2. tur başlar: yarışan her aday için sayaç 0.

Adım 9 / 15

```
1  def ikinci_tur(oylar, adaylar):
2      kalan = set(adaylar)
▶      while True:
▶          say = {a: 0 for a in kalan}
5          for adet, sira in oylar:
6              ilk = next(a for a in sira if a in kalan)
7              say[ilk] += adet
8          lider = max(say, key=say.get)
9          if 2 * say[lider] > sum(say.values()):
10             return lider
11         kaybeden = min(say, key=say.get)
12         kalan.remove(kaybeden)
```

DEĞİŞKENLER

`kalan = {A, C, D}`

`say = {A: 0, C: 0, D: 0}`

Kod İzleme: `ikinci_tur(oylar, adaylar)`



2. tur sayımı: elenen B'yi birinci sıraya koyan yoktu; sayılar değişmedi.

Adım 10 / 15

```
1  def ikinci_tur(oylar, adaylar):
2    kalan = set(adaylar)
3    while True:
4      say = {a: 0 for a in kalan}
▶      for adet, sira in oylar:
▶        ilk = next(a for a in sira if a in kalan)
▶        say[ilk] += adet
8      lider = max(say, key=say.get)
9      if 2 * say[lider] > sum(say.values()):
10       return lider
11     kaybeden = min(say, key=say.get)
12     kalan.remove(kaybeden)
```

DEĞİŞKENLER

`kalan = {A, C, D}`

`say = {A: 6, C: 4, D: 3}`

Kod İzleme: `ikinci_tur(oylar, adaylar)`



lider = A: 2 × 6 = 12 ≤ 13 ⇒ çoğunluk yok.

Adım 11 / 15

```
1  def ikinci_tur(oylar, adaylar):
2      kalan = set(adaylar)
3      while True:
4          say = {a: 0 for a in kalan}
5          for adet, sira in oylar:
6              ilk = next(a for a in sira if a in kalan)
7              say[ilk] += adet
▶          lider = max(say, key=say.get)
▶          if 2 * say[lider] > sum(say.values()):
10             return lider
11         kaybeden = min(say, key=say.get)
12         kalan.remove(kaybeden)
```

DEĞİŞKENLER

`kalan = {A, C, D}`

`say = {A: 6, C: 4, D: 3}`

`lider = A`

Kod İzleme: `ikinci_tur(oylar, adaylar)`



En az oylu **D** (3) elenir; kalan = {A, C}.

Adım 12 / 15

```
1  def ikinci_tur(oylar, adaylar):
2    kalan = set(adaylar)
3    while True:
4      say = {a: 0 for a in kalan}
5      for adet, sira in oylar:
6        ilk = next(a for a in sira if a in kalan)
7        say[ilk] += adet
8      lider = max(say, key=say.get)
9      if 2 * say[lider] > sum(say.values()):
10       return lider
▶      kaybeden = min(say, key=say.get)
▶      kalan.remove(kaybeden)
```

DEĞİŞKENLER

`kalan = {A, C}`

`say = {A: 6, C: 4, D: 3}`

`kaybeden = D`

Kod İzleme: `ikinci_tur(oylar, adaylar)`



3. tur başlar: yarışan her aday için sayaç 0.


```
1  def ikinci_tur(oylar, adaylar):
2      kalan = set(adaylar)
▶      while True:
▶          say = {a: 0 for a in kalan}
5          for adet, sira in oylar:
6              ilk = next(a for a in sira if a in kalan)
7              say[ilk] += adet
8          lider = max(say, key=say.get)
9          if 2 * say[lider] > sum(say.values()):
10             return lider
11         kaybeden = min(say, key=say.get)
12         kalan.remove(kaybeden)
```

DEĞİŞKENLER

`kalan = {A, C}`

`say = {A: 0, C: 0}`

Kod İzleme: `ikinci_tur(oylar, adaylar)`



DBCA: yarışan ilk aday C ⇒ say[C] += 3. (D elendiği için oy C'ye geçti.)

Adım 14 / 15

```
1  def ikinci_tur(oylar, adaylar):
2    kalan = set(adaylar)
3    while True:
4      say = {a: 0 for a in kalan}
▶      for adet, sira in oylar:
▶        ilk = next(a for a in sira if a in kalan)
▶        say[ilk] += adet
8      lider = max(say, key=say.get)
9      if 2 * say[lider] > sum(say.values()):
10       return lider
11     kaybeden = min(say, key=say.get)
12     kalan.remove(kaybeden)
```

DEĞİŞKENLER

`kalan = {A, C}` `adet = 3`

`sira = "DBCA"` `ilk = C`

`say = {A: 4, C: 7}`

Kod İzleme: `ikinci_tur(oylar, adaylar)`



lider = C: 2 × 7 = 14 > 13 ⇒ **C** döndürülür.

Adım 15 / 15

```
1  def ikinci_tur(oylar, adaylar):
2      kalan = set(adaylar)
3      while True:
4          say = {a: 0 for a in kalan}
5          for adet, sira in oylar:
6              ilk = next(a for a in sira if a in kalan)
7              say[ilk] += adet
▶          lider = max(say, key=say.get)
▶          if 2 * say[lider] > sum(say.values()):
▶              return lider
11         kaybeden = min(say, key=say.get)
12         kalan.remove(kaybeden)
```

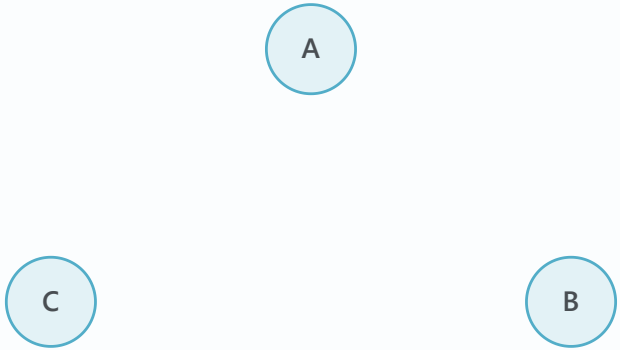
DEĞİŞKENLER

`kalan = {A, C}`

`say = {A: 6, C: 7}` `lider = C`

Condorcet Döngüsü Adım Adım

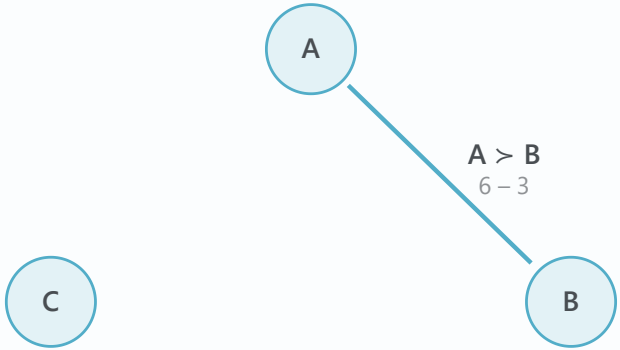
sıra	4 kişi	3 kişi	2 kişi
1.	A	B	C
2.	B	C	A
3.	C	A	B



9 seçmen, 3 aday. Her seçmenin sıralaması tek başına tutarlıdır (geçişli).

Condorcet Döngüsü Adım Adım

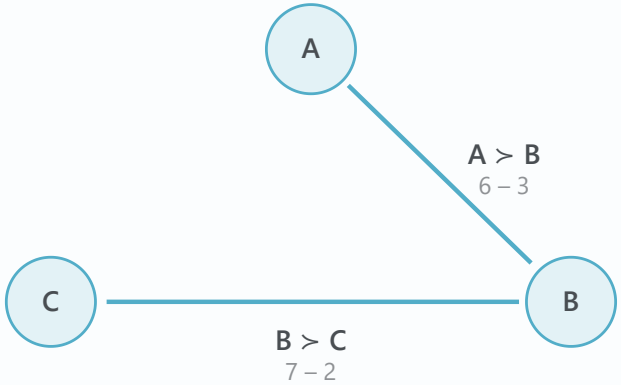
sıra	4 kişi	3 kişi	2 kişi
1.	**A**	**B**	C
2.	B	C	**A**
3.	C	A	B



A – B: A'yı tercih eden 4 + 2 = 6, B'yi tercih eden 3 ⇒ çoğunluk **A ≻ B**.

Condorcet Döngüsü Adım Adım

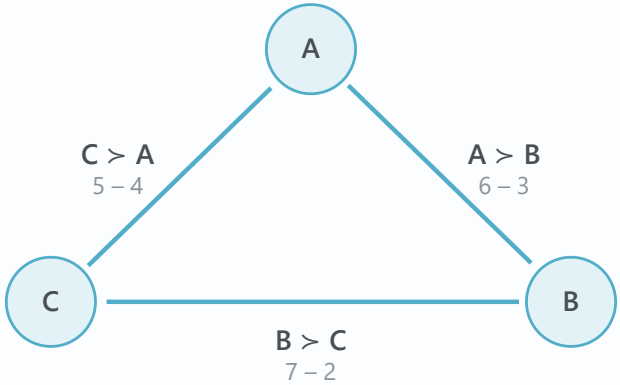
sıra	4 kişi	3 kişi	2 kişi
1.	A	**B**	**C**
2.	**B**	C	A
3.	C	A	B



B – C: 4 + 3 = 7'ye 2 ⇒ çoğunluk **B ≻ C**. Geçişlilik beklentisi: A ≻ C.

Condorcet Döngüsü Adım Adım

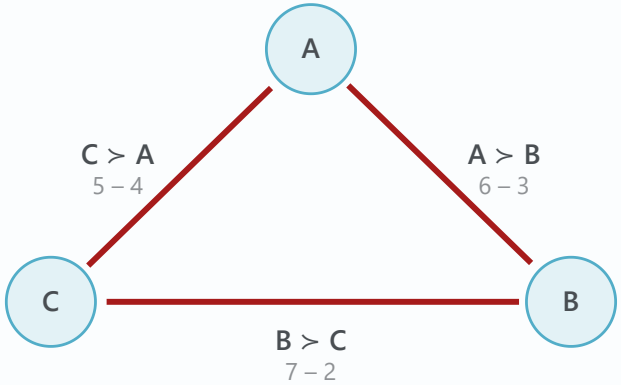
sıra	4 kişi	3 kişi	2 kişi
1.	**A**	B	**C**
2.	B	**C**	A
3.	C	A	B



C – A: 3 + 2 = 5'e 4 ⇒ çoğunluk **C ≻ A**!

Condorcet Döngüsü Adım Adım

sıra	4 kişi	3 kişi	2 kişi
1.	A	B	C
2.	B	C	A
3.	C	A	B



A ≻ B ≻ C ≻ A: toplumsal tercih **döngüseldir**; her adayı yenen bir çoğunluk vardır, Condorcet kazananı yoktur.

CONDORCET PARADOKSU

Gündem Sırası Kazananı Belirler

Aynı döngüsel profilde ikili elemeli oylama (sequential pairwise voting): önce iki aday oylanır, kazanan sıradakiyle karşılaşır.

Gündem 1: (A – B), sonra – C

- A – B: 6 – 3 ⇒ **A**
- A – C: 4 – 5 ⇒ **C**

Kazanan: **C**

Önce A ile B oylanır (A kazanır), kazanan A son olarak C ile karşılaşır ve kaybeder: **C**.

Adım 1 / 3

Üç gündemin üçü de farklı kazanan üretir: gündemi belirleyen (ör. komite başkanı) sonucu seçebilir. Sona bırakılan aday avantajlıdır.

CONDORCET PARADOKSU

Gündem Sırası Kazananı Belirler

Aynı döngüsel profilde ikili elemeli oylama (sequential pairwise voting): önce iki aday oylanır, kazanan sıradakiyle karşılaşır.

Gündem 1: (A – B), sonra – C

- A – B: 6 – 3 ⇒ **A**
- A – C: 4 – 5 ⇒ **C**

Kazanan: **C**

Gündem 2: (B – C), sonra – A

- B – C: 7 – 2 ⇒ **B**
- B – A: 3 – 6 ⇒ **A**

Kazanan: **A**

Önce B – C (B kazanır), sonra B – A: A 6–3 kazanır: **A**.

Üç gündemin üçü de farklı kazanan üretir: gündemi belirleyen (ör. komite başkanı) sonucu seçebilir. Sona bırakılan aday avantajlıdır.

CONDORCET PARADOKSU

Gündem Sırası Kazananı Belirler

Aynı döngüsel profilde ikili elemeli oylama (sequential pairwise voting): önce iki aday oylanır, kazanan sıradakiyle karşılaşır.

Gündem 1: (A – B), sonra – C

- A – B: 6 – 3 ⇒ **A**
- A – C: 4 – 5 ⇒ **C**

Kazanan: **C**

Gündem 2: (B – C), sonra – A

- B – C: 7 – 2 ⇒ **B**
- B – A: 3 – 6 ⇒ **A**

Kazanan: **A**

Gündem 3: (A – C), sonra – B

- A – C: 4 – 5 ⇒ **C**
- C – B: 2 – 7 ⇒ **B**

Kazanan: **B**

Önce A – C (C kazanır), sonra C – B: B 7–2 kazanır: **B**.

Üç gündemin üçü de farklı kazanan üretir: gündemi belirleyen (ör. komite başkanı) sonucu seçebilir. Sona bırakılan aday avantajlıdır.

Oyuna Dönüş: Condorcet Paradoksu

Kuramın cevabı: hayır

Oyunun Seviye 1 profili (A≻B≻C, B≻C≻A, C≻A≻B): A–B 2–1, B–C 2–1, C–A 2–1. Toplumsal tercih döngüseldir; Condorcet kazananı yoktur ve gündemi kuran sonucu seçer.

Gündem	İlk oylama	Kazanan
(A–B), sonra C	A	**C**
(B–C), sonra A	B	**A**
(A–C), sonra B	C	**B**

Ne kadar sık?

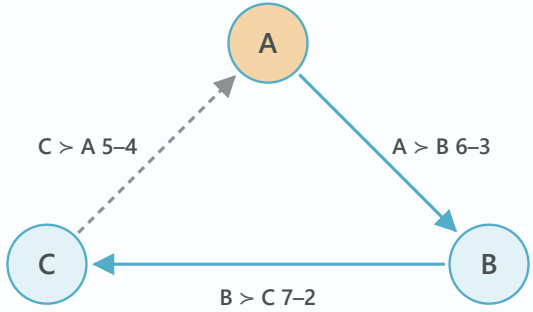
Üç seçmen ve üç adayda her sıralama eşit olasılıklıysa $6^3 = 216$ profilin 12'si döngüdür: %5,6. Seçmen sayısı çok büyüdükçe bu oran yaklaşık %8,8'e çıkar (Guilbaud, 1952).

Gerçekte: Seçim ve anket verilerinde döngüler bu "rastgele tercih" hesabının öngördüğünden de seyrek görülür; gerçek tercihler rastgele değildir, birbirine benzer (Regenwetter vd., 2006).

Döngüde gündemi belirleyen kazananı belirler: üç olası gündemin her birinde farklı bir aday kazanır ve sona bırakılan aday avantajlıdır.

Kemeny Kuralı: En Uyumlu Sıralama

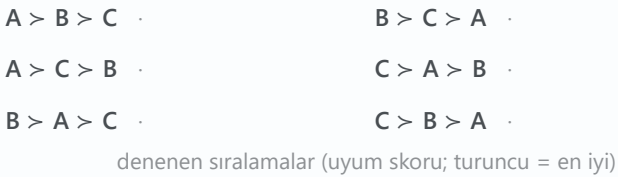
- Kemeny (1959): seçmenlerin ikili tercihleriyle **en çok uyuşan** toplumsal sıralamayı seç.
- Bir sıralamanın uyumu: o sıralamada x'in y'den önce geldiği her çift için x'i y'ye tercih eden seçmen sayılarının toplamı.
- Condorcet kazananı varsa başa yerleşir: ana profilde D ≻ B ≻ C ≻ A (44).
- Döngüde de karar verir: **en zayıf** çoğunluğu bozar ⇒ A ≻ B ≻ C (17).
- En iyi sıralamayı bulmak **NP-zordur** (Bartholdi, Tovey, Trick, 1989).



kesikli: Kemeny'nin bozduğu en zayıf çoğunluk (5–4)

Uygulama: sıralama birleştirme (rank aggregation; Dwork vd., 2001): arama motoru sonuç listeleri, yarışma ve komite sıralamaları.

Kod İzleme: `kemeny(oylar, adaylar)`



denenen sıralamalar (uyum skoru; turuncu = en iyi)

oylar = [(4, "ABC"), (3, "BCA"), (2, "CAB")] (döngü profili). En iyi sıralama henüz yok; skor −1.

Adım 1 / 14

```
1  from itertools import permutations
2  def kemeny(oylar, adaylar):
▶    en_iyi, en_skor = None, -1
4    for sira in permutations(adaylar):
5      skor = 0
6      for i, x in enumerate(sira):
7        for y in sira[i + 1:]:
8          skor += sum(n for n, oy in oylar
9                      if oy.index(x) < oy.index(y))
10     if skor > en_skor:
11       en_iyi, en_skor = sira, skor
12   return en_iyi, en_skor
```

DEĞİŞKENLER

`en_iyi = None` `en_skor = −1`

Kod İzleme: `kemeny(oylar, adaylar)`



sira = A ≻ B ≻ C: 3 çift için uyuşan seçmenler 6 + 4 + 7 = **17**.

Adım 2 / 14

```
1  from itertools import permutations
2  def kemeny(oylar, adaylar):
3    en_iyi, en_skor = None, -1
▶    for sira in permutations(adaylar):
▶      skor = 0
▶      for i, x in enumerate(sira):
▶        for y in sira[i + 1:]:
▶          skor += sum(n for n, oy in oylar
▶                      if oy.index(x) < oy.index(y))
10     if skor > en_skor:
11       en_iyi, en_skor = sira, skor
12   return en_iyi, en_skor
```

DEĞİŞKENLER

`sira = (A, B, C)` `skor = 17`

`en_iyi = None` `en_skor = -1`

Kod İzleme: `kemeny(oylar, adaylar)`



17 > önceki en iyi ⇒ en_iyi = A ≻ B ≻ C.

Adım 3 / 14

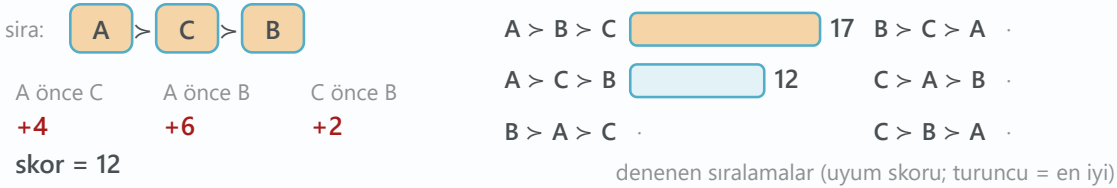
```
1  from itertools import permutations
2  def kemeny(oylar, adaylar):
3    en_iyi, en_skor = None, -1
4    for sira in permutations(adaylar):
5      skor = 0
6      for i, x in enumerate(sira):
7        for y in sira[i + 1:]:
8          skor += sum(n for n, oy in oylar
9                      if oy.index(x) < oy.index(y))
▶      if skor > en_skor:
▶        en_iyi, en_skor = sira, skor
12   return en_iyi, en_skor
```

DEĞİŞKENLER

`sira = (A, B, C)` `skor = 17`

`en_iyi = (A, B, C)` `en_skor = 17`

Kod İzleme: `kemeny(oylar, adaylar)`



sira = A ≻ C ≻ B: 3 çift için uyuşan seçmenler 4 + 6 + 2 = **12**.

Adım 4 / 14

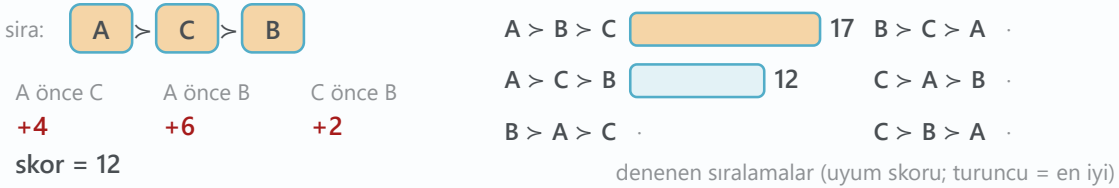
```
1  from itertools import permutations
2  def kemeny(oylar, adaylar):
3    en_iyi, en_skor = None, -1
▶    for sira in permutations(adaylar):
▶      skor = 0
▶      for i, x in enumerate(sira):
▶        for y in sira[i + 1:]:
▶          skor += sum(n for n, oy in oylar
▶                      if oy.index(x) < oy.index(y))
10     if skor > en_skor:
11       en_iyi, en_skor = sira, skor
12   return en_iyi, en_skor
```

DEĞİŞKENLER

`sira = (A, C, B)` `skor = 12`

`en_iyi = (A, B, C)` `en_skor = 17`

Kod İzleme: `kemeny(oylar, adaylar)`



12 ≤ 17: en iyi değişmez.

Adım 5 / 14

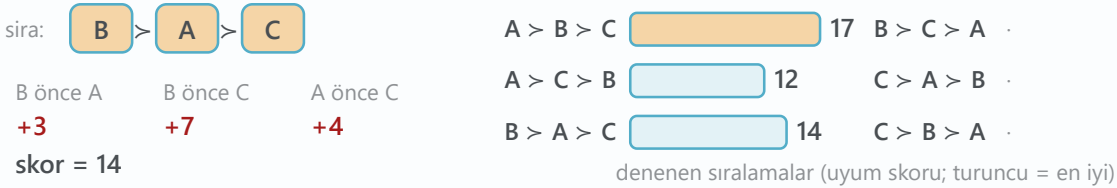
```
1  from itertools import permutations
2  def kemeny(oylar, adaylar):
3    en_iyi, en_skor = None, -1
4    for sira in permutations(adaylar):
5      skor = 0
6      for i, x in enumerate(sira):
7        for y in sira[i + 1:]:
8          skor += sum(n for n, oy in oylar
9                      if oy.index(x) < oy.index(y))
▶      if skor > en_skor:
11       en_iyi, en_skor = sira, skor
12   return en_iyi, en_skor
```

DEĞİŞKENLER

`sira = (A, C, B)` `skor = 12`

`en_iyi = (A, B, C)` `en_skor = 17`

Kod İzleme: `kemeny(oylar, adaylar)`



sira = B ≻ A ≻ C: 3 çift için uyuşan seçmenler 3 + 7 + 4 = **14**.

Adım 6 / 14

```
1  from itertools import permutations
2  def kemeny(oylar, adaylar):
3    en_iyi, en_skor = None, -1
▶    for sira in permutations(adaylar):
▶      skor = 0
▶      for i, x in enumerate(sira):
▶        for y in sira[i + 1:]:
▶          skor += sum(n for n, oy in oylar
▶                      if oy.index(x) < oy.index(y))
10     if skor > en_skor:
11       en_iyi, en_skor = sira, skor
12   return en_iyi, en_skor
```

DEĞİŞKENLER

`sira = (B, A, C)` `skor = 14`

`en_iyi = (A, B, C)` `en_skor = 17`

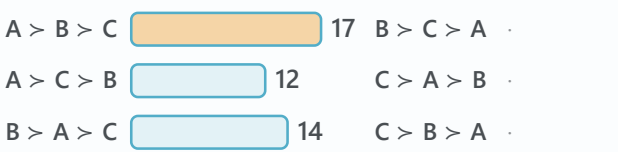
Kod İzleme: `kemeny(oylar, adaylar)`

sira: B ≻ A ≻ C

B önce A	B önce C	A önce C
+3	+7	+4

skor = 14

14 ≤ 17: en iyi değişmez.



denenen sıralamalar (uyum skoru; turuncu = en iyi)

Adım 7 / 14

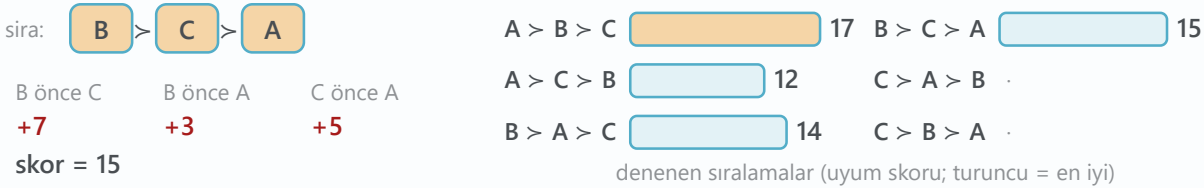
```
1  from itertools import permutations
2  def kemeny(oylar, adaylar):
3    en_iyi, en_skor = None, -1
4    for sira in permutations(adaylar):
5      skor = 0
6      for i, x in enumerate(sira):
7        for y in sira[i + 1:]:
8          skor += sum(n for n, oy in oylar
9                      if oy.index(x) < oy.index(y))
▶      if skor > en_skor:
11       en_iyi, en_skor = sira, skor
12   return en_iyi, en_skor
```

DEĞİŞKENLER

`sira = (B, A, C)` `skor = 14`

`en_iyi = (A, B, C)` `en_skor = 17`

Kod İzleme: `kemeny(oylar, adaylar)`



sira = B ≻ C ≻ A: 3 çift için uyuşan seçmenler 7 + 3 + 5 = **15**.

Adım 8 / 14

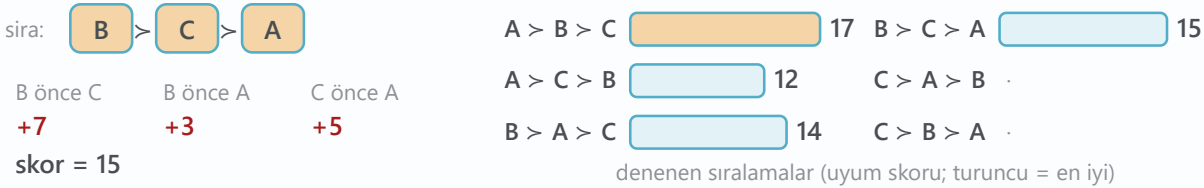
```
1  from itertools import permutations
2  def kemeny(oylar, adaylar):
3    en_iyi, en_skor = None, -1
▶    for sira in permutations(adaylar):
▶      skor = 0
▶      for i, x in enumerate(sira):
▶        for y in sira[i + 1:]:
▶          skor += sum(n for n, oy in oylar
▶                      if oy.index(x) < oy.index(y))
10     if skor > en_skor:
11       en_iyi, en_skor = sira, skor
12   return en_iyi, en_skor
```

DEĞİŞKENLER

`sira = (B, C, A)` `skor = 15`

`en_iyi = (A, B, C)` `en_skor = 17`

Kod İzleme: `kemeny(oylar, adaylar)`



15 ≤ 17: en iyi değişmez.

Adım 9 / 14

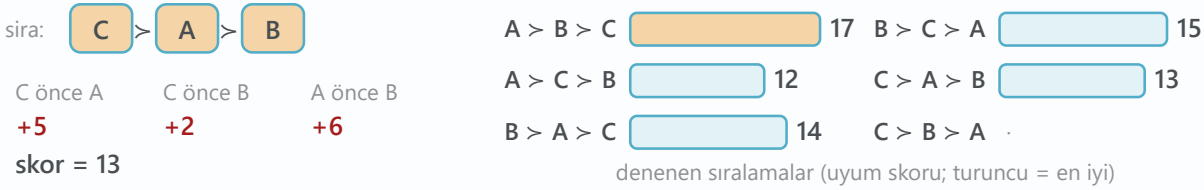
```
1  from itertools import permutations
2  def kemeny(oylar, adaylar):
3    en_iyi, en_skor = None, -1
4    for sira in permutations(adaylar):
5      skor = 0
6      for i, x in enumerate(sira):
7        for y in sira[i + 1:]:
8          skor += sum(n for n, oy in oylar
9                      if oy.index(x) < oy.index(y))
▶      if skor > en_skor:
11       en_iyi, en_skor = sira, skor
12   return en_iyi, en_skor
```

DEĞİŞKENLER

`sira = (B, C, A)` `skor = 15`

`en_iyi = (A, B, C)` `en_skor = 17`

Kod İzleme: `kemeny(oylar, adaylar)`



sira = C ≻ A ≻ B: 3 çift için uyuşan seçmenler 5 + 2 + 6 = **13**.

Adım 10 / 14

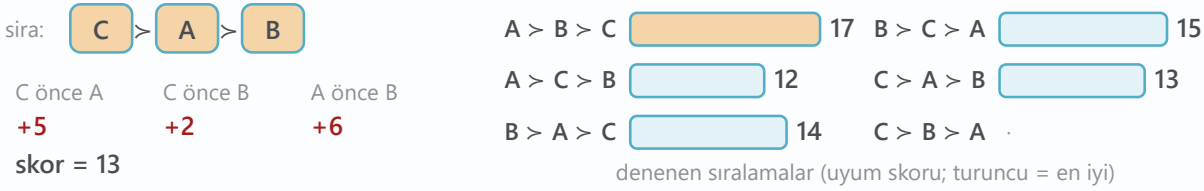
```
1  from itertools import permutations
2  def kemeny(oylar, adaylar):
3    en_iyi, en_skor = None, -1
▶    for sira in permutations(adaylar):
▶      skor = 0
▶      for i, x in enumerate(sira):
▶        for y in sira[i + 1:]:
▶          skor += sum(n for n, oy in oylar
▶                      if oy.index(x) < oy.index(y))
10     if skor > en_skor:
11       en_iyi, en_skor = sira, skor
12   return en_iyi, en_skor
```

DEĞİŞKENLER

`sira = (C, A, B)` `skor = 13`

`en_iyi = (A, B, C)` `en_skor = 17`

Kod İzleme: `kemeny(oylar, adaylar)`



13 ≤ 17: en iyi değişmez.

Adım 11 / 14

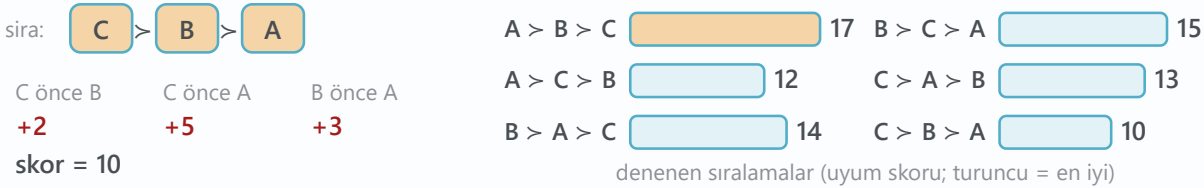
```
1  from itertools import permutations
2  def kemeny(oylar, adaylar):
3    en_iyi, en_skor = None, -1
4    for sira in permutations(adaylar):
5      skor = 0
6      for i, x in enumerate(sira):
7        for y in sira[i + 1:]:
8          skor += sum(n for n, oy in oylar
9                      if oy.index(x) < oy.index(y))
▶      if skor > en_skor:
11       en_iyi, en_skor = sira, skor
12   return en_iyi, en_skor
```

DEĞİŞKENLER

`sira = (C, A, B)` `skor = 13`

`en_iyi = (A, B, C)` `en_skor = 17`

Kod İzleme: `kemeny(oylar, adaylar)`



sira = C ≻ B ≻ A: 3 çift için uyuşan seçmenler 2 + 5 + 3 = **10**.

Adım 12 / 14

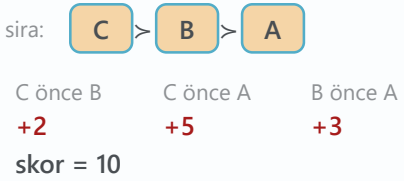
```
1  from itertools import permutations
2  def kemeny(oylar, adaylar):
3    en_iyi, en_skor = None, -1
▶    for sira in permutations(adaylar):
▶      skor = 0
▶      for i, x in enumerate(sira):
▶        for y in sira[i + 1:]:
▶          skor += sum(n for n, oy in oylar
▶                      if oy.index(x) < oy.index(y))
10     if skor > en_skor:
11       en_iyi, en_skor = sira, skor
12   return en_iyi, en_skor
```

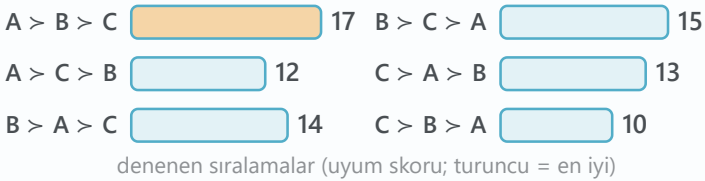
DEĞİŞKENLER

`sira = (C, B, A)` `skor = 10`

`en_iyi = (A, B, C)` `en_skor = 17`

Kod İzleme: `kemeny(oylar, adaylar)`





denenen sıralamalar (uyum skoru; turuncu = en iyi)

10 ≤ 17: en iyi değişmez.

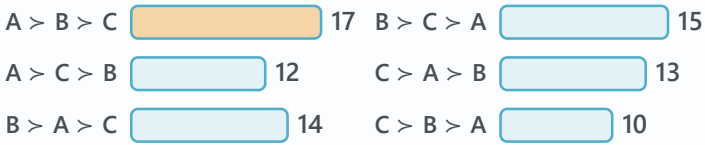

```
1  from itertools import permutations
2  def kemeny(oylar, adaylar):
3    en_iyi, en_skor = None, -1
4    for sira in permutations(adaylar):
5      skor = 0
6      for i, x in enumerate(sira):
7        for y in sira[i + 1:]:
8          skor += sum(n for n, oy in oylar
9                      if oy.index(x) < oy.index(y))
▶      if skor > en_skor:
11       en_iyi, en_skor = sira, skor
12   return en_iyi, en_skor
```

DEĞİŞKENLER

`sira = (C, B, A)` `skor = 10`

`en_iyi = (A, B, C)` `en_skor = 17`

Kod İzleme: `kemeny(oylar, adaylar)`



denenen sıralamalar (uyum skoru; turuncu = en iyi)

Sonuç A ≻ B ≻ C (17): yalnızca en zayıf çoğunluk C ≻ A (5–4) bozuldu. m aday için m! sıra denenir.

Adım 14 / 14

```
1  from itertools import permutations
2  def kemeny(oylar, adaylar):
3    en_iyi, en_skor = None, -1
4    for sira in permutations(adaylar):
5      skor = 0
6      for i, x in enumerate(sira):
7        for y in sira[i + 1:]:
8          skor += sum(n for n, oy in oylar
9                      if oy.index(x) < oy.index(y))
10     if skor > en_skor:
11       en_iyi, en_skor = sira, skor
▶    return en_iyi, en_skor
```

DEĞİŞKENLER

`en_iyi = (A, B, C)` `en_skor = 17`

Kurallar Hangi Ölçütleri Sağlar?

Kural	Condorcet kazananını seçer	Condorcet kaybedeni seçilmez	Monoton	Stratejiye dayanıklı
Çoğulluk	**hayır**	**hayır (A)**	**evet**	**hayır**
Borda	**hayır (B)**	**evet**	**evet**	**hayır**
Anında ikinci tur	**hayır (C)**	**evet**	**hayır**	**hayır**
Onay	**hayır**	**hayır**	**evet**	**hayır**
Condorcet (Copeland)	**evet**	**evet**	**evet**	**hayır**
Kemeny	**evet**	**evet**	**evet**	**hayır**

- **Monotonluk:** kazananı bir pusulada yukarı taşımak onu kaybettirmemeli. IRV bunu ihlal edebilir: yukarı taşıma, eleme sırasını değiştirir.
- Condorcet yöntemleri döngüde kazanan bulamaz; ek bir eşitlik bozma kuralı gerekir. **Maksimin** (maximin, Simpson) kuralı her adayın ikili karşılaştırmalarda aldığı en düşük oy sayısını bulur; bu sayı en yüksek olan aday kazanır: örnek profilde A 6, B 6, C 4, D 7 → D. Condorcet kazananı varsa onu seçer, döngüde de karar verir.
- Stratejiye dayanıklı: hiçbir seçmen yanlış sıralama bildirerek daha iyi sonuç alamaz. Son sütundaki "hayır"ların nedeni genel bir teoremdir (Gibbard–Satterthwaite).

CONDORCET PARADOKSU

Anında İkinci Turda Monotonluk İhlali

sıra	6 kişi	6 kişi	5 kişi
1.	**A**	**B**	**C**
2.	B	C	A
3.	C	A	B

aday	1. tur
A	6
B	6
C	5

17 seçmen, çoğunluk için 9 oy gerekir. 1. tur: A = 6, B = 6, C = 5 ⇒ çoğunluk yok, en az oylu **C** elenir.

Anında İkinci Turda Monotonluk İhlali

sıra	6 kişi	6 kişi	5 kişi
1.	**A**	**B**	~~C~~
2.	B	C	**A**
3.	C	A	B

aday	2. tur
A	**11**
B	6
C	—

C'nin 5 oyu sıradaki A'ya geçer: A = 11, B = 6 ⇒ **A kazanır**.

Anında İkinci Turda Monotonluk İhlali

sıra	6+2 kişi	4 kişi	5 kişi
1.	**A**	B	C
2.	B	C	A
3.	C	A	B

aday	—
A	?
B	?
C	?

Seçimden önce A kampanya yapar: B ≻ C ≻ A diyen 6 kişiden **2'si** A'yı başa alır (A ≻ B ≻ C). A'nın desteği yalnızca **arttı**.

Anında İkinci Turda Monotonluk İhlali

sıra	6+2 kişi	4 kişi	5 kişi
1.	**A**	**B**	**C**
2.	B	C	A
3.	C	A	B

aday	1. tur
A	8
B	4
C	5

Yeni 1. tur: A = 8, B = 4, C = 5 ⇒ bu kez en az oylu **B** elenir (C değil).

Anında İkinci Turda Monotonluk İhlali

sıra	6+2 kişi	4 kişi	5 kişi
1.	**A**	~~B~~	**C**
2.	B	**C**	A
3.	C	A	B

aday	2. tur
A	8
B	—
C	**9**

B'nin 4 oyu C'ye geçer: C = 9, A = 8 ⇒ **C kazanır**. A, daha çok destek aldığı için kaybetti: IRV **monoton değildir**.

STRATEJİK OY

Çoğullukta Taktik Oy (Oy Bölünmesi)

sıra	4 kişi	4 kişi	3 kişi	2 kişi
1.	**A**	**C**	**D**	**A**
2.	D	B	B	B
3.	B	D	C	C
4.	C	A	A	D

aday	çoğulluk
A	**6**
B	0
C	4
D	3

Dürüst oylamada çoğulluk kazananı A. 3. grup ($D \succ B \succ C \succ A$) için A **en kötü** aday!

Adım 1 / 3

"Boşa oy vermeme" davranışı: seçmen favorisi yerine kazanma şansı olan, daha az kötü bulduğu adaya oy verir (Duverger yasası: çoğulluk iki partili sisteme iter).

STRATEJİK OY

Çoğullukta Taktik Oy (Oy Bölünmesi)

sıra	4 kişi	4 kişi	3 kişi	2 kişi
1.	A	C	**D**	A
2.	D	B	B	B
3.	B	D	**C**	C
4.	C	A	A	D

aday	çoğulluk
A	**6**
B	0
C	4
D	3

3. grubun favorisi D'nin (3 oy) kazanma şansı yok. A'yı durdurabilecek aday C'dir ve bu grup C'yi A'ya tercih eder.

Adım 2 / 3

"Boşa oy vermeme" davranışı: seçmen favorisi yerine kazanma şansı olan, daha az kötü bulduğu adaya oy verir (Duverger yasası: çoğulluk iki partili sisteme iter).

STRATEJİK OY

Çoğullukta Taktik Oy (Oy Bölünmesi)

sıra	4 kişi	4 kişi	3 kişi	2 kişi
1.	A	C	**C**	A
2.	D	B	D	B
3.	B	D	B	C
4.	C	A	A	D

aday	çoğulluk
A	6
B	0
C	**7**
D	0

3. grup C'ye oy verir (pusula $C \succ D \succ B \succ A$): $C = 7$, $A = 6 \Rightarrow$ **C** kazanır; bu grup için A'dan daha iyi.

Adım 3 / 3

"Boşa oy vermeme" davranışı: seçmen favorisi yerine kazanma şansı olan, daha az kötü bulduğu adaya oy verir (Duverger yasası: çoğulluk iki partili sisteme iter).

Borda'da Stratejik Oy Adım Adım

sıra	4 kişi	4 kişi	3 kişi	2 kişi
1.	A	C	D	A
2.	D	B	B	B
3.	B	D	C	C
4.	C	A	A	D

aday	dürüst
A	18
B	**22**
C	17
D	21

Dürüst Borda sayımı: B = 22, D = 21. 3. grup (D ≻ B ≻ C ≻ A) favorisi D'nin 1 puan farkla kaybettiğini görüyor.

Borda'da Stratejik Oy Adım Adım

sıra	4 kişi	4 kişi	3 kişi	2 kişi
1.	A	C	D	A
2.	D	B	**C**	B
3.	B	D	**B**	C
4.	C	A	A	D

aday	dürüst
A	18
B	**22**
C	17
D	21

Manipülasyon ("gömme", burying): 3 seçmen asıl rakip B'yi 2. sıradan 3. sıraya iter; pusula D ≻ C ≻ B ≻ A.

Borda'da Stratejik Oy Adım Adım

sıra	4 kişi	4 kişi	3 kişi	2 kişi
1.	A	C	D	A
2.	D	B	**C**	B
3.	B	D	**B**	C
4.	C	A	A	D

aday	dürüst	fark
A	18	0
B	22	−3
C	17	+3
D	21	0

Her manipülatör için B 2 yerine 1, C 1 yerine 2 puan alır: B −3, C +3.

Borda'da Stratejik Oy Adım Adım

sıra	4 kişi	4 kişi	3 kişi	2 kişi
1.	A	C	D	A
2.	D	B	C	B
3.	B	D	B	C
4.	C	A	A	D

aday	dürüst	stratejik
A	18	18
B	22	19
C	17	20
D	**21**	**21**

Yeni toplamlar: D = 21, C = 20, B = 19, A = 18 ⇒ **D** kazanır. 3. grup yalan söyleyerek favorisini seçtirdi.

STRATEJİK OY

Stratejik Oy Sistemi Ne Kadar Bozar?

Çoğulluk

Oy bölünmesi: benzer iki aday oyları paylaşır, üçüncüsü kazanır. Taktik oy yaygın ve basittir.

Borda

Rakibi en alta gömmek çok etkilidir; herkes gömerse sıralama seçmenlerin gerçek tercihlerinden uzaklaşır.

Anında ikinci tur

Manipülasyon daha zor hesaplanır ama **mümkündür**; ayrıca monoton değildir. "IRV stratejik oya kapalıdır" yaygın ama yanlış bir kanıdır.

Hesaplama bariyeri

Bazı kurallarda (ör. IRV ve Copeland'in bir türevi) kârlı manipülasyonu bulmak NP-zordur (Bartholdi vd., 1989–91).

Etki tercih dağılımına bağlıdır: yakın yarışlarda stratejik oy kazananı değiştirebilir. Şeffaf kurallar ve anlaşılır bilgi bazı manipülasyonları azaltır, ama hiçbir genel kural stratejik davranışı her durumda ortadan kaldıramaz.

İMKÂNSIZLIK TEOREMLERİ

Arrow'un İmkânsızlık Teoremi: Koşullar

Toplumsal **sıralama** üreten bir kural (sosyal refah fonksiyonu) için makul görünen koşullar:

Evrensel alan (unrestricted domain)
Her tercih profili kabul edilir; sonuç tam ve geçişli bir sıralamadır.

Pareto (oybirliği)
Herkes x'i y'ye tercih ediyorsa toplum da x'i y'ye tercih eder.

İlgisiz alternatiflerden bağımsızlık (IIA)
x ile y'nin toplumsal sırası yalnızca seçmenlerin x–y sırasına bağlıdır.

Diktatör yok
Tercihi her profilde toplumun tercihi olan tek bir seçmen yoktur.

Arrow (1951): En az **3 aday** varsa bu dört koşulu birlikte sağlayan hiçbir kural yoktur. (2 adayda basit çoğunluk hepsini sağlar: May teoremi.)

Borda ve İlgisiz Alternatifler

sıra	4 kişi	4 kişi	3 kişi	2 kişi
1.	A	C	D	A
2.	D	B	B	B
3.	B	D	C	C
4.	C	A	A	D

aday	4 aday
A	18
B	**22**
C	17
D	**21**

Borda, 4 adayla: B = 22 > D = 21; toplum **B'yi D'nin önüne** koyar.

Borda ve İlgisiz Alternatifler

sıra	4 kişi	4 kişi	3 kişi	2 kişi
1.	A	∈	D	A
2.	D	B	B	B
3.	B	D	∈	∈
4.	∈	A	A	D

aday	4 aday
A	18
B	**22**
C	17
D	**21**

Şimdi C yarıştan çekilsin. C ne kazanıyordu ne de ikinciydi: B–D sıralaması için "ilgisiz" bir aday.

Borda ve İlgisiz Alternatifler

sıra	4 kişi	4 kişi	3 kişi	2 kişi
1.	A	B	D	A
2.	D	D	B	B
3.	B	A	A	D

aday	4 aday	C yok
A	18	12
B	**22**	**13**
C	17	
D	**21**	**14**

Yeni Borda (2, 1, 0): D = 14 > B = 13. Hiçbir seçmenin B–D sırası değişmedi (B'yi tercih eden 6, D'yi tercih eden 7), ama toplumsal sıra döndü: Borda **IIA'yı ihlal eder**. Arrow'un IIA'sı aday kümesi sabitken de bozulur. 4 kişilik A ≻ D ≻ B ≻ C grubu yalnız C ile B'nin yerini değiştirirse (A ≻ D ≻ C ≻ B) B 22'den 18'e düşer, D 21'de kalır (C de 21'e çıkar): B–D sırası yine döner.

İMKÂNSIZLIK TEOREMLERİ

Arrow Teoreminin Pratik Anlamı

Söylediği

Toplu karar kuralları bütün makul ölçütleri aynı anda sağlayamaz; her kural bir koşuldan ödün verir.

Söylemediği

Oylamanın anlamsız olduğunu ya da belirli bir kuralı yasakladığını söylemez.

Kim neden ödün verir?

Borda: IIA. Condorcet/çoğunluk sıralaması: geçişlilik (döngü). Diktatörlük: adalet.

Kurumlar için

Hangi ölçüte öncelik verildiği açıkça belirlenmeli ve belgelenmelidir.

Kaçış yolları koşulları gevşetmektir: tercih alanını kısıtlamak (tek tepeli tercihler), sıralama yerine yoğunluk bilgisi (puanlama, para) kullanmak.

Gibbard–Satterthwaite Teoremi

Teorem (Gibbard 1973, Satterthwaite 1975): En az **3 aday** varsa; her profilde tek kazanan seçen, her adayın kazanabildiği (örtenlik) ve **stratejiye dayanıklı** (kimse yalan söyleyerek daha iyi sonuç alamaz) her kural **diktatörlüktür**.

Örnek profildeki kanıtlar

Çoğullukta 3. grubun C'ye oyu; Borda'da B'nin gömülmesi.

Arrow ile ilişki

Arrow sıralamalar, G–S tek kazanan içindir; G–S, Arrow'dan türetilebilir.

Mekanizma tasarımı

Parasız ortamda teşvik uyumlu genel kural yoktur. Para varsa VCG mümkündür.

- Tasarımın hedefi stratejik davranışı yok etmek değil **sınırlandırmaktır**: az bilgi, hesaplama zorluğu, kısıtlı tercih alanı.

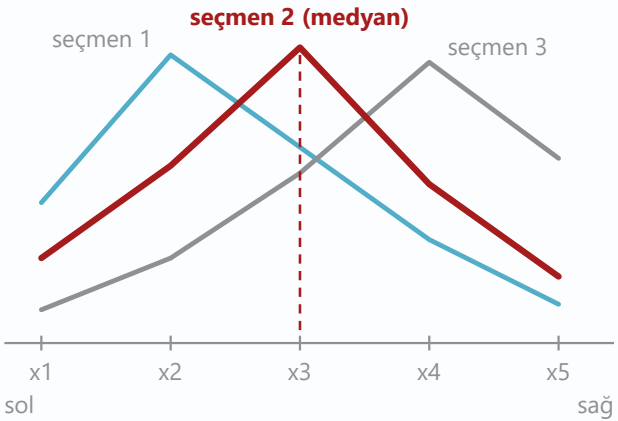
May Teoremi: İki Aday İçin Basit Çoğunluk

İki aday (x, y) varken hangi kural "adil"dir? May (1952) üç koşulu sağlayan kuralın **yalnızca basit çoğunluk** olduğunu gösterdi.

Koşul	Anlamı	İhlal eden kural
Anonimlik	Seçmenlerin isimleri önemsiz; oylar yer değiştirse sonuç aynı	Diktatör; ağırlıklı oy
Tarafsızlık	Adayların isimleri önemsiz; iki aday simetrik muamele görür	"Statüko ⅔ olmadan değişmez" (nitelikli çoğunluk)
Pozitif duyarlılık	Beraberlikte ya da x kazanırken bir seçmen x'e geçerse x kesin kazanır	"Her zaman yazı-tura"; "tek sayıda oy alan kazanır"

İki adayda kural tartışması biter; zorluklar (döngü, Arrow, G–S) **üç ve daha fazla** adayla başlar.

Kaçış Yolu: Tek Tepeli Tercihler



- Adaylar bir eksende sıralıysa (sol–sağ, bütçe miktarı, konum) ve her seçmenin en çok sevdiği bir konum (tepesi) varsa ve aday bu tepeden uzaklaştıkça seçmenin faydası azalıyorsa tercihler **tek tepelidir**.
- **Medyan seçmen teoremi** (Black, 1948): medyan seçmenin tepesi Condorcet kazananıdır; döngü oluşmaz.
- "Tepeleri bildir, medyanı seç" kuralı **stratejiye dayanıklıdır**: G–S alan kısıtlanınca geçerliliğini yitirir.

Tesis yerleşimindeki medyan mekanizması bu fikrin uygulamasıdır.

Dört Sonuç Yan Yana

Teorem	Varsayımlar	Sonuç	Kaçış yolu
Arrow (1951)	≥ 3 aday, evrensel alan, Pareto, IIA	Kural diktatörlüktür	Tercih alanını kısıtlamak (tek tepeli); IIA'dan vazgeçmek; sıralama yerine kardinal bilgi (puan, yoğunluk)
Gibbard–Satterthwaite (1973, 1975)	≥ 3 aday, evrensel alan, örten (her aday kazanabilir), stratejiye dayanıklı	Kural diktatörlüktür	Tek tepeli tercihler; rastgele kurallar (ör. rastgele diktatör); para (VCG)
May (1952)	2 aday; anonimlik, tarafsızlık, pozitif duyarlılık	Kural basit çoğunluktur (ve tersi)	Gerekmez: imkânsızlık değil, **karakterizasyon**
Tek tepeli tercihler (Black, 1948)	Adaylar bir eksende, her seçmenin tek tepesi var, tek sayıda seçmen	Medyan seçmenin tepesi Condorcet kazananıdır; medyan kuralı stratejiye dayanıklıdır	Kendisi bir kaçış yoludur: evrensel alan varsayımını kısıtlar

Arrow ve G–S'de diktatörlük, varsayımların hepsi birlikte istendiğinde çıkar; bir varsayım gevşeyince olumlu sonuçlar mümkün olur.

Hesaplamalı Sosyal Seçim

Bilgisayar bilimi, ekonomi ve sosyal seçimin kesişimi: kurallar yalnızca hangi ölçütleri sağladıklarına göre değil, **hesaplama maliyetine** göre de incelenir.

Kazananı hesaplamak

Çoğulluk ve Borda n seçmen, m aday için $O(nm)$, Copeland $O(nm^2)$ sürede hesaplanır. Kemeny ve Dodgson (Condorcet kazananı olmak için en az sayıda komşu yer değiştirmesi gereken aday kazanır) kurallarında kazananı bulmak NP-zordur (Bartholdi, Tovey, Trick, 1989).

Manipülasyonu hesaplamak

Tek manipülatör çoğulluk ve Borda'da kârlı pusulayı açgözlü bir algoritmayla polinom sürede bulur; IRV'de bu sorun NP-zordur. Zorluk bir “bariyer”dir ama yalnızca en kötü durum için.

Tercih çıkarımı

m aday için tam sıralama $\log_2(m!)$ bit bilgidir ($m = 10$'da ≈ 22 bit). Seçmenlere adım adım sorular sorup yalnızca kazananı belirlemeye yetecek bilgiyi toplamak iletişimi azaltır.

Adil bölüşüm ve yargı birleştirme

Bu alanın iki konusu daha.

HESAPLAMALI SOSYAL SEÇİM

Yargı Birleştirme: Doktrin Paradoksu

Üç yargıçlı mahkeme: davalı ancak **p** (sözleşme geçerli) **ve q** (sözleşme ihlal edildi) ise tazminat öder.

Yargıç	p	q	Sonuç: p ∧ q (p ve q)
1. yargıç	evet	evet	evet
2. yargıç	evet	hayır	hayır
3. yargıç	hayır	evet	hayır
Çoğunluk	**evet (2–1)**	**evet (2–1)**	**hayır (1–2)**

- Her yargıç tutarlıdır, ama önermeler tek tek oylanınca çoğunluk "p doğru, q doğru, p ∧ q yanlış" der: **tutarsız** bir toplu yargı (Kornhauser ve Sager, 1986).
- Gerekçelere göre karar verilirse davalı öder, sonuca göre karar verilirse ödemez: sonuç yine **usule** bağlıdır.
- List ve Pettit (2002): anonim, sistematik ve her zaman tutarlı bir birleştirme kuralı yoktur; Arrow teoreminin yargılar için karşılığı.

ÖZET

Gerçek Dünyada Oylama Kuralları

İki turlu sistem
Türkiye ve Fransa cumhurbaşkanlığı: ilk turda mutlak çoğunluk yoksa ilk iki aday yarışır.

Anında ikinci tur
Avustralya Temsilciler Meclisi, bazı ABD şehirleri; seçmen tam sıralama verir.

Çoğulluk
Birleşik Krallık ve ABD'de tek üyeli seçim bölgeleri.

Borda benzeri puanlama
Eurovision (12, 10, 8, ...), spor ödülleri (ör. en değerli oyuncu oylamaları).

Onay
Bazı bilimsel dernek seçimleri; "uygun olduğum saatler" anketleri.

Çevrim içi
Reddit ve forum oylamaları, topluluk anketleri; sınıf ve kulüp kararları.

ÖZET

Birden Çok Sandalye: Nispi Temsil ve D'Hondt

- Tek kazanan yerine S sandalye dağıtılır; hedef, sandalye payının oy payına yakın olmasıdır (**nispi temsil**).
- **D'Hondt**: her partinin oyu 1, 2, 3, … ile bölünür; en büyük S bölüm sandalye alır. Türkiye'de milletvekili seçimlerinde her seçim çevresinde D'Hondt, ayrıca ülke barajı uygulanır.
- Örnek: 6 sandalye, oylar A 42 000, B 31 000, C 15 000, D 12 000. En büyük altı bölüm 42 000, 31 000, 21 000, 15 500, 15 000, 14 000 → **A 3, B 2, C 1, D 0**.
- A %42 oyla sandalyelerin %50'sini alır; D %12 oyla hiç almaz. D'Hondt büyük partileri hafifçe kayırır. En büyük kalan (Hare) yönteminde her parti oy payı × 6'nın tam kısmı kadar sandalye alır, kalan sandalyeler küsuratı en büyük partilere gider: aynı oylarla A 2, B 2, C 1, D 1.

Bölen	A	B	C	D
÷1	**42.000**	**31.000**	**15.000**	12.000
÷2	**21.000**	**15.500**	7.500	6.000
÷3	**14.000**	10.333	5.000	4.000
÷4	10.500	7.750	3.750	3.000
Sandalye	**3**	**2**	**1**	**0**

Vurgulu hücreler: en büyük altı bölüm.

Tek kazananlı kurallardaki gibi burada da her yöntem bir ödünleşimdir: D'Hondt hükümet kararlılığını, en büyük kalan yöntemi küçük partilerin temsilini öne çıkarır.

ÖZET

Hangi Oylama Kuralı Adil Kabul Edilebilir?

- Adalet değerlendirmesi temsil, çoğunluk desteği, azınlıkların korunması ve stratejiye dayanıklılık gibi ölçütlere bağlıdır.
- **Çoğulluk** basittir ve hızlıdır, ancak oy bölünmesine ve çoğunluk desteği olmayan kazanana yol açabilir (A: 6/13).
- **Borda** daha fazla tercih bilgisi kullanır ve uzlaşmacı adayı öne çıkarır (B), ancak stratejik sıralamaya açıktır.
- **Anında ikinci tur** çoğunluk desteği arar, fakat monotonluk ve stratejik oy sorunlarını tamamen çözmez.
- **Condorcet yöntemleri** ikili çoğunluğa saygı gösterir, ama döngüde karar veremez.

Kural seçimi kararın bağlamına ve öncelikli ölçütlere göre yapılmalıdır: kulüp başkanlığı için hızlı çoğulluk, sınıf etkinliği için sıralamalı bir kural daha dengeli olabilir.

ÖZET

Özet

Kurallar

Mutlak çoğunluk, çoğulluk, Borda, onay, anında ikinci tur, Condorcet, Kemeny: aynı profil 4 farklı kazanan.

Condorcet paradoksu

Geçişli bireysel tercihler döngüsel toplumsal tercih üretebilir; gündemi kontrol eden sonucu seçer. Kemeny en zayıf çoğunluğu bozar; IRV monoton değildir.

Stratejik oy

Çoğullukta taktik oy, Borda'da gömme: dürüstlük her kuralda en iyi strateji değildir.

Arrow

≥ 3 adayda evrensel alan + Pareto + IIA + diktatör yok birlikte sağlanamaz.

Gibbard–Satterthwaite

≥ 3 adayda stratejiye dayanıklı örten kural = diktatörlük.

May ve tek tepelilik

2 adayda basit çoğunluk tek adil kural; tek tepeli tercihlerde medyan dayanıklıdır.

ÖZET

Etkileşimli Oyunlar

Voting Rules

Aynı tercih profilinde çoğulluk, Borda, IRV ve Condorcet kazananları.

Condorcet Paradoksu

İkili karşılaştırmalarda döngü ve gündem sırasının etkisi.

Facility Location

Tek tepeli tercihlerde medyan mekanizması: yalan söylemek neden işe yaramaz?

ÖZET

Örnek Problemler

1. 9 seçmen: 4 × A ≻ B ≻ C, 3 × C ≻ B ≻ A, 2 × B ≻ C ≻ A. Çoğulluk, IRV, Borda ve Condorcet kazananları?
2. 1. sorudaki A seçmenleri B'yi gömerek (A ≻ C ≻ B) Borda'da ne elde eder?
3. Döngü profilinde (4 ABC, 3 BCA, 2 CAB) B'nin kazanması için gündem nasıl olmalı?
4. 1. sorudaki profilin Kemeny sıralaması nedir? (6 sıralamanın uyum skorunu karşılaştırın.)
5. "Statüko, ancak ⅔ oyla değişir" kuralı May teoreminin hangi koşulunu ihlal eder?
6. 11 seçmen: 2 × ACB, 3 × BAC, 2 × BCA, 4 × CBA. Condorcet kazananı kim? Hangi $s = (1, a, 0)$ kuralı ($0 \le a \le 1$) onu seçer?

▶ **Cevapları göster**

Kaynaklar

- F. Brandt, V. Conitzer, U. Endriss, J. Lang, A. D. Procaccia (ed.), *Handbook of Computational Social Choice*, Cambridge University Press, 2016.
- N. Nisan, T. Roughgarden, É. Tardos, V. Vazirani (ed.), *Algorithmic Game Theory*, Cambridge University Press, 2007 (Bölüm 9).
- K. J. Arrow, *Social Choice and Individual Values*, Wiley, 1951.
- A. Gibbard, "Manipulation of voting schemes: a general result", *Econometrica* 41(4), 1973.
- M. A. Satterthwaite, "Strategy-proofness and Arrow's conditions", *J. Economic Theory* 10, 1975.
- K. O. May, "A set of independent necessary and sufficient conditions for simple majority decision", *Econometrica* 20(4), 1952.
- D. Black, "On the rationale of group decision-making", *J. Political Economy* 56(1), 1948.
- J. J. Bartholdi, C. A. Tovey, M. A. Trick, "The computational difficulty of manipulating an election" ve "Voting schemes for which it can be difficult to tell who won the election", *Social Choice and Welfare* 6, 1989 · M. Balinski, H. P. Young, *Fair Representation*, Yale University Press, 1982 (nispi temsil).
- J. G. Kemeny, "Mathematics without numbers", *Daedalus* 88(4), 1959; C. Dwork, R. Kumar, M. Naor, D. Sivakumar, "Rank aggregation methods for the web", WWW 2001.
- A. Karlin, Y. Peres, *Game Theory, Alive*, AMS, 2017 (sosyal seçim ve oylama bölümü); D. Easley, J. Kleinberg, *Networks, Crowds, and Markets*, Cambridge University Press, 2010 (Bölüm 23: Voting).
- Ders sayfaları: A. Procaccia, 15-896 (CMU) · N. Shah, CSC304 (Toronto).

Son

Bölüm 7: Sosyal Seçim ve Oylama

Sonraki bölüm: Bölüm 8: Ağ Tasarımı ve Yönlendirme

Çalışma soruları · Terim sözlüğü