

Ayrik Sistemler

İçin İleri Olasılık

235112004

M.Cüneyd
Kurtbaş

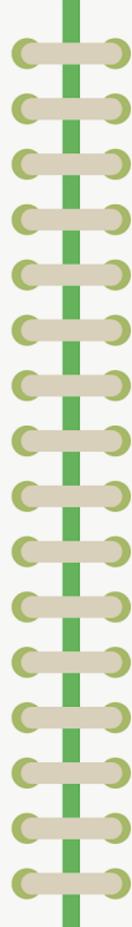


Sunum



Güvercin Yuvası İlkesi

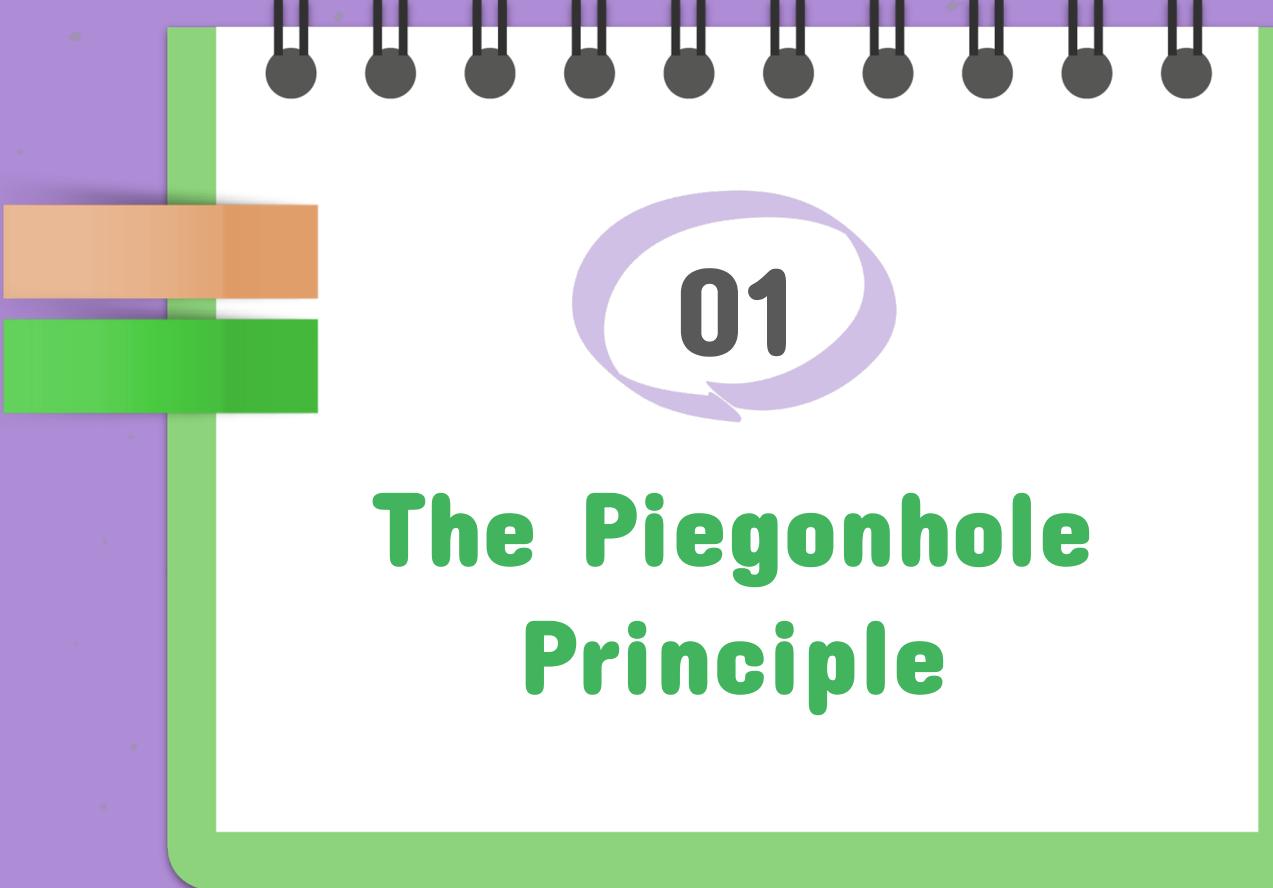
Permutasyon



Kombinasyon

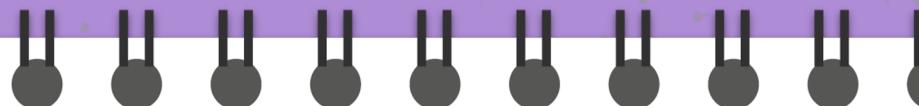
Kodların Çalıştırılması





01

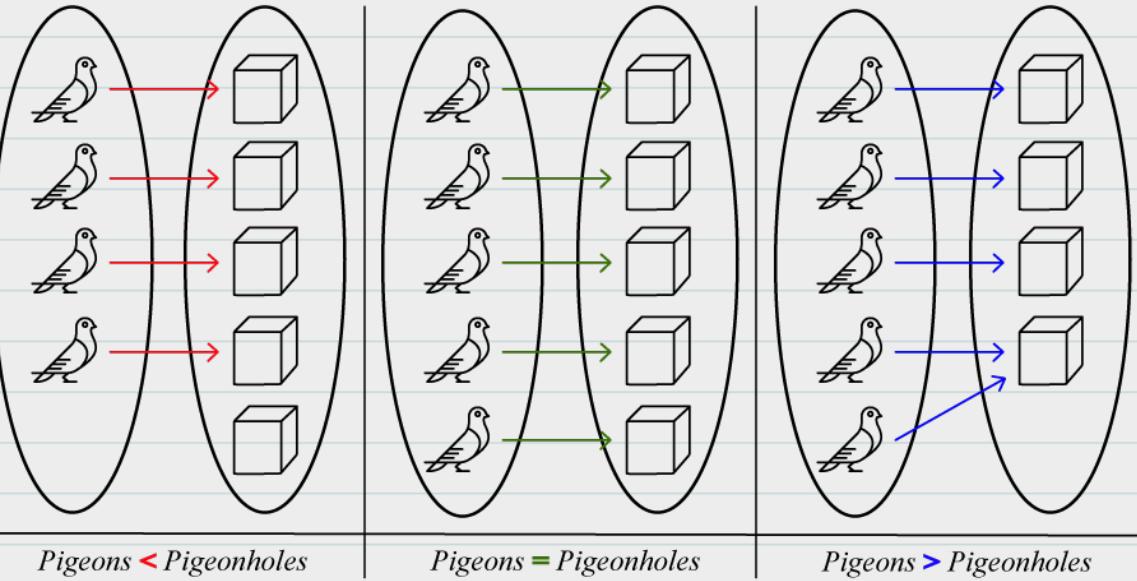
The Piegonghole Principle



Nedir?

"Güvercin Yuvası İlkesi", birçok matematiksel ve mantıksal bağlama uygulanabilen temel bir ilkedir. Bu ilke, bir grup öğe arasında ilişkiyi gösterir ve basitçe şöyle ifade edilir: Eğer nesne sayısı bir kutunun nesne kapasitesinden fazlaysa, en az bir kutuda birden fazla nesne olacağı kesindir.

The Pigeonhole Principle

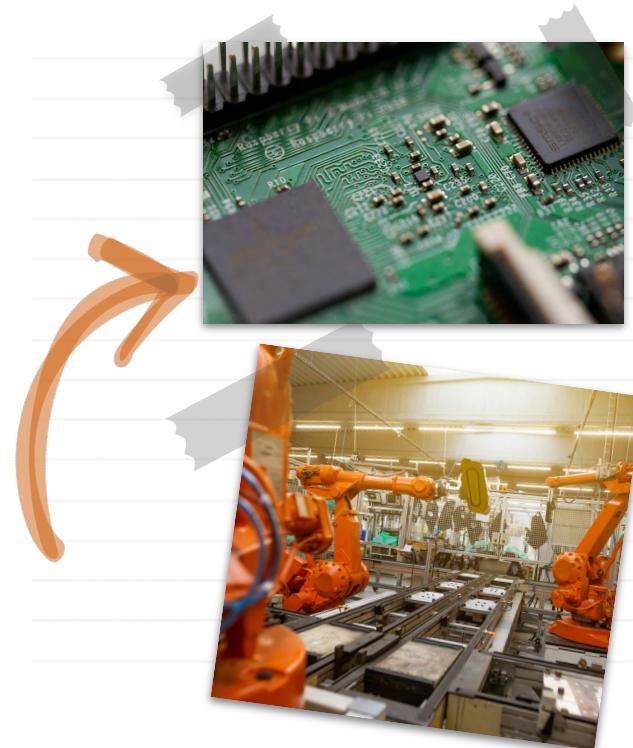


Problem

Elektronik cihaz üretilen bir fabrikada;

Her ürünün **1 harf** ve **3 rakamdan** oluşan bir model kodu bulunmaktadır. Üretilen cihazlardan **en az 3 adet** cihazın aynı model koduna sahip olmasını **garanti** edebilmek için; en az kaç adet cihaz üretilmiş olmalıdır?

Kısıtlar: Model kodları **harf ile başlamakta** ve alfabetden yalnızca **6 harf** kullanılmaktadır.



Yaklaşım



En az 3 cihazın;
aynı model koduna sahip
olmasını garanti etmek



- Yuva Sayısı = 6.000

$$\text{Harf} \times \text{Rakam} \times \text{Rakam} \times \text{Rakam}$$
$$6 \times 10 \times 10 \times 10$$

Örneğin; A322, X095

- Güvercin Sayısı = 12.001

$$6.000 + 6.000 + 1$$

The Pigeonhole Principle

```
def en_az_guvercin_sayisi(harf_sayisi, rakam_sayisi, garanti_edilen):  
    toplam_uretim = 0  
  
    for i in range(1, garanti_edilen):  
  
        kombinasyon = harf_sayisi * rakam_sayisi * rakam_sayisi * rakam_sayisi  
  
        toplam_uretim += kombinasyon  
  
    return toplam_uretim +1
```

The Pigeonhole Principle

```
harf_sayisi = 6
```

```
rakam_sayisi = 10
```

```
garanti_edilen = 3
```

```
harf_paleti = {  
    "harfler": ["A", "B", "C", "X", "Y", "Z"],  
}
```

```
kod_list = []
```

```
guvercin_sayisi = en_az_guvercin_sayisi(harf_sayisi, rakam_sayisi, garanti_edilen)
```

```
print("En az 3 aynı kod üretmeyi garanti etmek için gerekli üretim:", guvercin_sayisi)
```

The Piegonhole Principle

```
def kod_olustur():

    rastgele_harf = random.choice(harf_paleti["harfler"])

    rastgele_sayı = random.randint(0, 999)

    guvercin_kod = f"{rastgele_harf}{rastgele_sayı:03}"

    return guvercin_kod
```

The Pigeonhole Principle

#Yuva Sayısı < Güvercin Sayısı

```
while len(kod_list) < guvercin_sayisi:  
    kod_list.append(kod_olustur())
```

The Pigeonhole Principle

```
# Rastgele Üretilen 4 Haneli Kodlardan Tekrar Edenleri Listeleme

ayni_kodlar = []
tekrar_sayisi = {}

for kod in kod_list:

    if kod_list.count(kod) > 1 and kod not in ayni_kodlar:

        ayni_kodlar.append(kod)

        tekrar_sayisi[kod] = kod_list.count(kod)

for ayni_kod in ayni_kodlar:

    print(f"{ayni_kod}: {tekrar_sayisi[ayni_kod]}")
```

The Piegonhole Principle

```
from collections import Counter

eleman_sayisi = Counter(kod_list)

ayni_kodlar = sorted(eleman_sayisi.items(), key=lambda x: x[1], reverse=True)

print("Üretilen Tüm Kodlar:")

for eleman, sayi in ayni_kodlar:

    print(f"{eleman}: {sayi}")
```



Permutation



Nedir?

Belirli bir sıralamayla düzenlenmiş nesnelerin farklı düzenlemelerinin sayısını ifade eder.

Yani, bir grup öğe arasında yapılan farklı sıralamaların toplam sayısıdır. Permutasyon, matematikte, istatistikte, bilgisayar biliminde ve diğer birçok alanda çeşitli problemleri çözmek için kullanılır.

Permutation



Choose only two



$$nPr = P_r^n = P(n, r) = \frac{n!}{(n - r)!}$$

Problem

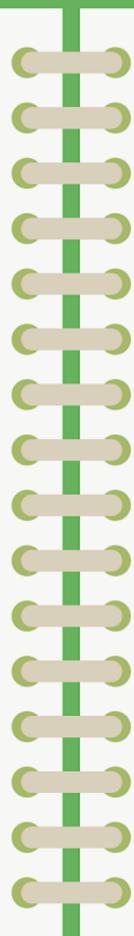
Kullanıcı tarafından girilen ve benzersiz harflerden oluşan bir String ifadesinin **harfleri birer kez kullanılarak** ;

Anlamlı ya da anlamsız kelimeler **1'den başlayarak** ardışık numaralandırılmış kartlara **alfabetik** olarak yazılacaktır.

Buna göre kullanıcının verdiği String ifade kaç numaralı kartta yazılacaktır ?



Yaklaşım



Örneğin < E L İ F > kelimesi;

1 numaralı kart: E F İ L

2 numaralı kart: E F L İ

3 numaralı kart: E İ F L

4 numaralı kart: E İ L F

5 numaralı kart: E L F İ

6 numaralı kart: E L İ F

Permutation

```
input_string = "CUNEYD"  
  
sirali_string = ""  
  
def sirala():  
  
    sirali_string = sorted(input_string)  
  
    for char in sirali_string:  
  
        print(char, end=" ")  
  
    return sirali_string
```

Permutation

```
def index_bul(sirali_string, aranan_karakter):

    try:

        index = sirali_string.index(aranan_karakter)

        return index

    except ValueError:

        return -1
```

Permutation

```
karakter_sayisi = len(input_string)  
sirali_string = sirala()
```



C D E N U Y



Permutation

```
def faktoriyel(n):  
    if n == 0:  
        return 1  
    else:  
        return n * faktoriyel(n-1)
```

Permutation

```
toplam = 0

for i in range (0, karakter_sayisi):

    index = index_bul(sirali_string, input_string[i])

    if index > 0:
        toplam = toplam + (index * faktoriyel(karakter_sayisi - 1))
        karakter_sayisi = karakter_sayisi - 1

    else:
        toplam +=0
        karakter_sayisi = karakter_sayisi - 1

    sirali_string.remove(input_string[i])
```

Permutation

```
print (f"{input_string} İFADESI {toplam+1}. KARTTA YAZILACAKTIR.")
```



88. KART





03

Combination



Nedir?

Kombinasyon, bir kümeden belirli bir sayıda öğenin seçilmesiyle oluşturulan farklı grupların sayısını ifade eder. Kombinasyonda seçilen öğelerin hangi sırayla seçildiği önemli değildir, sadece seçilen öğelerin kendisi önemlidir. Olasılık teorisi, istatistik ve kombinatorikte sıkça kullanılır.

Combination



10 meyveden 4 tanesini nasıl seçersin?

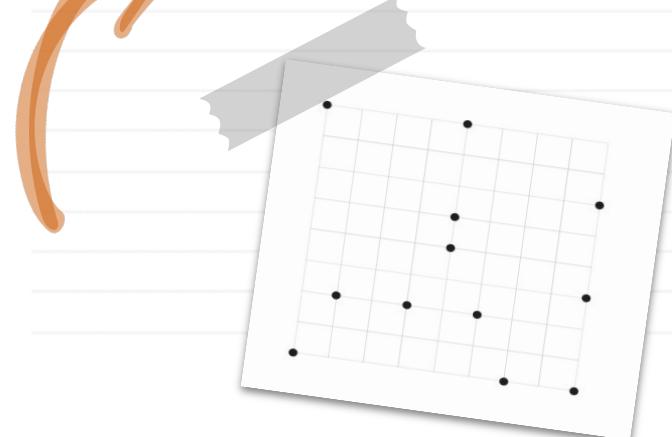
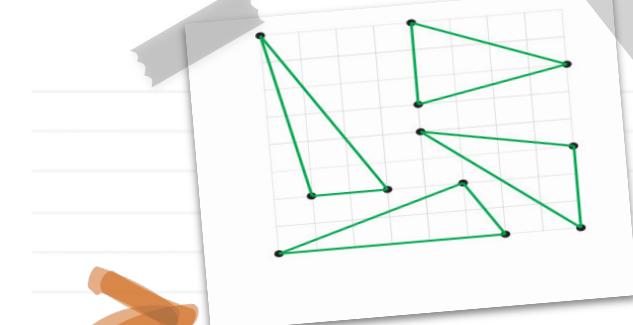
$${}^n C_r = \frac{n!}{r!(n-r)!}$$

$${}^{10} C_4 = C(n, r) = C(10, 4)$$

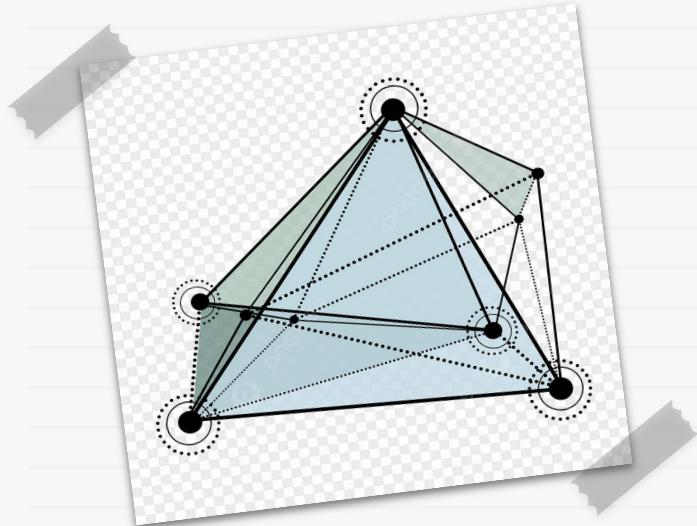
Problem

7 tanesi doğrusal olmak üzere;

Toplam 17 nokta ile
en çok kaç farklı üçgen
çizilebilir?



Yaklaşım



17 nokta üzerinde herhangi 3 nokta ile çizilebilecek üçgen sayısızı:

$$C(17, 3)$$

Kendi içinde üçgen oluşturamayacak olan 7 doğrusal nokta için:

$$C(7, 3)$$

Çizilebilek **en fazla** üçgen sayısı:

$$C(17, 3) - C(7, 3)$$

Combination

```
def kombinasyon(n, k):

    if k < 0 or k > n:

        return 0

    sonuc = 1

    for i in range(min(k, n - k)):

        sonuc *= (n - i)

        sonuc /= (i + 1)

    return sonuc
```

Combination

```
ucgen_kombinasyonu = kombinasyon(17, 3)  
dogrusal_noktalarin_kombinasyonu = kombinasyon(7, 3)  
en_cok_ucgen = ucgen_kombinasyonu - dogrusal_noktalarin_kombinasyonu  
print("Farklı çizilebilecek en fazla üçgen sayısı:", en_cok_ucgen)
```



645



Tesekkürler

235112004

M. Cüneyd
Kurtbas