**SORU 1**

Bir e-posta sunucusunda Naive Bayes tabanlı bir spam filtresi çalışmaktadır. Filtre eğitim verilerinden aşağıdaki istatistikleri öğrenmiştir:

| **Bilgi** | **Değer** |
|---|---|
| Gelen bir e-postanın spam olma olasılığı | %20 |
| Spam olduğu bilinen bir e-postada "ücretsiz" kelimesinin geçme olasılığı | %60 |
| Normal olduğu bilinen bir e-postada "ücretsiz" kelimesinin geçme olasılığı | %10 |
| Spam olduğu bilinen bir e-postada "tıklayın" kelimesinin geçme olasılığı | %70 |
| Normal olduğu bilinen bir e-postada "tıklayın" kelimesinin geçme olasılığı | %5 |

Naive Bayes'te kelimeler birbirinden koşullu bağımsız kabul edilir. İki kelime birlikte geçiyorsa olasılıklar çarpılır. Payda her sınıf için ortak olduğundan sadece pay hesaplanarak karşılaştırma yapılabilir.

Yeni gelen bir e-postada hem "ücretsiz" hem de "tıklayın" kelimeleri geçmektedir. Bu e-postanın spam olduğuna dair olasılığı kaçtır?

A)%91
B)%84
C)%75
D)%60
E)%95

**ÇÖZÜM**

P(Spam) = 0.20, P(Normal) = 0.80
P("ücretsiz" ve "tıklayın" | Spam) = 0.60 × 0.70 = 0.42
P("ücretsiz" ve "tıklayın" | Normal) = 0.10 × 0.05 = 0.005
Pay_Spam = 0.42 × 0.20 = 0.084
Pay_Normal = 0.005 × 0.80 = 0.004
Toplam = 0.084 + 0.004 = 0.088
P(Spam | "ücretsiz", "tıklayın") = 0.084 / 0.088 ≈ 0.9545 → %95

**CEVAP: E**

**SORU 2**

Bir şirketin veri tabanında A, B, C, D ve E olmak üzere beş çalışan kaydı bulunmaktadır. Her kayıt için aktif (1) veya pasif (0) durumu tutulmaktadır. Sistem yöneticisi veri tabanında tutarsızlık olduğunu fark etmiş ve kayıtları incelemiştir.

İnceleme sonucunda şu kısıtların her zaman geçerli olması gerektiği anlaşılmıştır:

| Kısıt No | Kural |
|---|---|
| K1 | A aktifse B de aktif olmalıdır. |
| K2 | B aktifse C pasif olmalıdır. |
| K3 | C veya D'den en az biri aktif olmalıdır. |
| K4 | D aktifse E de aktif olmalıdır. |
| K5 | E aktifse A pasif olmalıdır. |

Mevcut kayıtlarda A, B, D ve E aktif; C pasiftir.

Mevcut kayıtta kaç kısıt ihlal edilmektedir ve tutarsızlığı gidermek için hangi tek değişiklik yeterlidir?

A)1 kısıt ihlali var; A pasif yapılmalıdır.
B) 2 kısıt ihlali var; E pasif yapılmalıdır.
C) 1 kısıt ihlali var; E pasif yapılmalıdır.
D) 2 kısıt ihlali var; A pasif yapılmalıdır.
E) 3 kısıt ihlali var; tek değişiklikle düzeltilemez.

**ÇÖZÜM**

Mevcut durum: A=1, B=1, C=0, D=1, E=1
Her kısıtı kontrol edelim:
K1: A aktif → B aktif mi? B=1 ✓ K2: B aktif → C pasif mi? C=0 ✓ K3: C veya D aktif mi? D=1 ✓ K4: D aktif → E aktif mi? E=1 ✓ K5: E aktif → A pasif mi? A=1 ✗ ← ihlal!
Yalnızca K5 ihlal edilmiş → 1 kısıt ihlali var.
Şimdi tek değişiklikle düzeltilip düzeltilemeyeceğini kontrol edelim:
A pasif yapılsın (A=0): K1: A=0 → B için kısıt yok ✓ K2: B=1 → C=0 ✓ K3: D=1 ✓ K4: D=1 → E=1 ✓ K5: E=1 → A=0 ✓ Tüm kısıtlar sağlanır ✓ Seçenek 2 — E pasif yapılsın (E=0): K4: D=1 → E aktif olmalı, ama E=0 ✗ Yeni ihlal yaratır!
Doğru cevap: 1 kısıt ihlali vardır (K5) ve yalnızca A'yı pasif yapmak tüm kısıtları sağlar. E'yi pasif yapmak K4'ü ihlal edeceğinden geçerli bir çözüm değildir.
**CEVAP: A**

**SORU 3**

Bir güvenlik sisteminde mesajlar üç aşamalı bir dönüşüm boru hattından (pipeline) geçirilerek şifrelenmektedir. Her aşama bir fonksiyonla tanımlanmıştır:

$f : \mathbb{R} \to \mathbb{R}$, $f(x) = 2x + 3$ (ölçekleme ve öteleme)
$g : \mathbb{R} \to \mathbb{R}$, $g(x) = (x - 1) / 4$ (normalleştirme)
$h : \mathbb{R} \to \mathbb{R}$, $h(x) = x^3 + 2$ (küpleme katmanı)

Şifreleme işlemi sırasıyla önce f, sonra g, son olarak h olarak uygulanır ($h \circ g \circ f$). Şifresi çözülmek istenen bir mesajın şifreli değeri $y = 10$ olarak alınmıştır. Buna göre orijinal mesaj x değeri nedir?

A) $x = 3$
B) $x = 5$
C) $x = 7$
D) $x = 9$
E) $x = 11$

**ÇÖZÜM**

Her fonksiyonun tersini bulalım:

$f(x) = 2x + 3 \to f^{-1}(x) = (x - 3) / 2$

$g(x) = (x-1)/4 \to g^{-1}(x) = 4x + 1$

$h(x) = x^3 + 2 \to h^{-1}(x) = \sqrt[3]{(x - 2)}$

$(h \circ g \circ f)^{-1} = f^{-1} \circ g^{-1} \circ h^{-1}$ Yani: önce $h^{-1}$, sonra $g^{-1}$, son olarak $f^{-1}$ uygulanır.

$y = 10$ üzerinde sırasıyla uygulayalım:

$h^{-1}(10) = \sqrt[3]{(10 - 2)} = \sqrt[3]{8} = 2$

$g^{-1}(2) = 4 \cdot 2 + 1 = 9$

$f^{-1}(9) = (9 - 3) / 2 = 3$

Doğrulama (ileri yönde kontrol):

$f(3) = 2 \cdot 3 + 3 = 9$ $g(9) = (9-1)/4 = 8/4 = 2$ $h(2) = 2^3 + 2 = 10$ ✓

**CEVAP: A**

**SORU 4**

$A = \{1, 2, 3, 4\}$ kümesi üzerinde iki bağıntı tanımlanmıştır:

$R = \{(a, b) \in A \times A \mid a + b \text{ asal sayıdır}\}$
$S = \{(a, b) \in A \times A \mid a \cdot b \text{ çift sayıdır}\}$

Bileşke bağıntı $S \circ R$ şu şekilde tanımlanır:
$(S \circ R) = \{(a, c) \mid \exists\, b \in A : (a, b) \in R \text{ ve } (b, c) \in S\}$

Buna göre, $S \circ R$ bağıntısının eleman sayısı kaçtır ve $(2, 2) \in S \circ R$ midir?

A) 10 eleman, $(2,2) \in S \circ R$
B) 12 eleman, $(2,2) \in S \circ R$
C) 12 eleman, $(2,2) \notin S \circ R$
D) 14 eleman, $(2,2) \in S \circ R$
E) 16 eleman, $(2,2) \notin S \circ R$

## ÇÖZÜM

R'yi bulalım (a + b asal):

1+1=2 ✓ 1+2=3 ✓ 1+3=4 ✗ 1+4=5 ✓ 2+1=3 ✓ 2+2=4 ✗ 2+3=5 ✓ 2+4=6 ✗ 3+1=4 ✗ 3+2=5 ✓ 3+3=6 ✗ 3+4=7 ✓ 4+1=5 ✓ 4+2=6 ✗ 4+3=7 ✓ 4+4=8 ✗

R = {(1,1),(1,2),(1,4),(2,1),(2,3),(3,2),(3,4),(4,1),(4,3)}

S'yi bulalım (a · b çift):

a·b çift olması için en az biri çift olmalıdır. Tek·Tek = Tek → S dışı: (1,1),(1,3),(3,1),(3,3)

S = A×A \ {(1,1),(1,3),(3,1),(3,3)} = {(1,2),(1,4),(2,1),(2,2),(2,3),(2,4),
(3,2),(3,4),(4,1),(4,2),(4,3),(4,4)}

S ∘ R'yi bulalım (her a için R'deki b'yi bulup S'de ilerleyelim):

a=1: R'de b∈{1,2,4} b=1 → S'de (1,c): c∈{2,4} b=2 → S'de (2,c): c∈{1,2,3,4} b=4 → S'de (4,c): c∈{1,2,3,4} → (1,c): c∈{1,2,3,4} [4 çift]

a=2: R'de b∈{1,3} b=1 → S'de (1,c): c∈{2,4} b=3 → S'de (3,c): c∈{2,4} → (2,c): c∈{2,4} [2 çift]

a=3: R'de b∈{2,4} b=2 → S'de (2,c): c∈{1,2,3,4} b=4 → S'de (4,c): c∈{1,2,3,4} → (3,c): c∈{1,2,3,4} [4 çift]

a=4: R'de b∈{1,3} b=1 → S'de (1,c): c∈{2,4} b=3 → S'de (3,c): c∈{2,4} → (4,c): c∈{2,4} [2 çift]

Toplam: 4+2+4+2 = 12 eleman.  (2,2) ∈ S ∘ R mi? a=2 için S ∘ R'de c∈{2,4} → (2,2) ∈ S ∘ R

**CEVAP: B**

**SORU 5**

$(1 + x)^{10} \cdot (1 + x^2)^5$ açılımında $x^4$ teriminin katsayısı kaçtır?

A) 210
B) 335
C) 420
D) 445
E) 480

## ÇÖZÜM

Her iki açılımın genel terimleri:

$(1+x)^{10}$ → x^j terimi: C(10,j) , j = 0,1,...,10

$(1+x^2)^5$ → x^2m terimi: C(5,m) , m = 0,1,...,5

j + 2m = 4 koşulunu sağlayan çiftler:

m=0, j=4: C(10,4) · C(5,0) = 210 · 1 = 210

m=1, j=2: C(10,2) · C(5,1) = 45 · 5 = 225

m=2, j=0: C(10,0) · C(5,2) = 1 · 10 = 10

Toplam: 210 + 225 + 10 = 445

**CEVAP: D**

## SORU 6

Bir ağ protokolünde her paket 10 bitlik veri taşımaktadır. İletim sırasında her bit bağımsız olarak bozulabilir. $(1+x)^{10}$ açılımındaki $x^k$ katsayısı, tam olarak k bitinin bozulduğu farklı paket sayısını vermektedir.

Örneğin $x^2$ katsayısı olan C(10,2) = 45, tam 2 bitin bozulduğu 45 farklı paket konfigürasyonu olduğunu ifade eder.

Protokolün hata tespit mekanizması şu özelliğe dayanmaktadır:
- Çift sayıda bozuk bit içeren paketler tespit edilemez (bu paketler geçerli görünür).
- Tek sayıda bozuk bit içeren paketler tespit edilir (bu paketler reddedilir).

Tespit edilemeyen hatalı paket sayısı ile tespit edilen hatalı paket sayısının farkı kaçtır? (Hiç hatası olmayan paket ve tüm bitleri bozulmuş paket hariç tutulacaktır.)

A) 0
B) 2
C) −2
D) 10
E) 512

**ÇÖZÜM**

$x=1$: $(1+1)^{10} = 2^{10} = 1024 \rightarrow C(10,0)+C(10,1)+...+C(10,10) = 1024$

$x=-1$: $(1-1)^{10} = 0 \rightarrow C(10,0)-C(10,1)+C(10,2)-...+C(10,10) = 0$

İki denklemi toplayalım: $1024 + 0 = 2\cdot[C(10,0)+C(10,2)+C(10,4)+C(10,6)+C(10,8)+C(10,10)]$

Çift indeksli toplam = $512 = 2^9$

İki denklemi çıkaralım: $1024 - 0 = 2\cdot[C(10,1)+C(10,3)+C(10,5)+C(10,7)+C(10,9)]$

Tek indeksli toplam = $512 = 2^9$

k=0 ve k=10 hariç tutalım: Tespit edilemeyen (çift, $k\neq0,10$): $512 - C(10,0) - C(10,10) = 512 - 1 - 1 = 510$

Tespit edilen (tek): 512

Fark = $510 - 512 = -2$. Çift ve tek indeksli katsayılar toplamı her zaman eşittir ($2^9 = 512$). Ancak k=0 ve k=10 çift olduğu için onları çıkartınca fark −2 olur.

**CEVAP: C**

**SORU 7**

Bir işletim sistemi, bellek adreslerini 7 slotluk bir hash tablosuna dağıtmaktadır. 7 asal sayı olduğu için hash fonksiyonu olarak şu kullanılmaktadır:

h(adres) = adres mod 7

$2^{100}$ numaralı bellek adresi hash tablosunun kaçıncı slotuna düşer?

A) 0. slot (h = 0)
B) 1. slot (h = 1)
C) 2. slot (h = 2)
D) 4. slot (h = 4)
E) 6. slot (h = 6)

**ÇÖZÜM**

Periyodu tespiti:

$2^1 \bmod 7 = 2$

$2^2 \bmod 7 = 4$

$2^3 \bmod 7 = 8 \bmod 7 = 1$

$2^4 \bmod 7 = 2$ periyot başa döndü. Dolayısı ile periyot = 3.

100 mod 3 hesabı: $100 = 33 \times 3 + 1$. Bu durumda $100 \bmod 3 = 1$

$2^{100} \equiv 2\text{^}(100 \bmod 3) = 2^1 = 2 \pmod 7$

$h(2^{100}) = 2 \rightarrow$ 2. slot

**CEVAP: C**

## SORU 8

$(x) + (x - 3) = 2$ denklemini sağlayan x değeri nedir?

A) 1
B) 2
C) 3
D) 4
E) 5

**ÇÖZÜM**

$\log_2(x(x-3)) = 2 \Longrightarrow x(x-3) = 4 \Longrightarrow x^2-3x-4=0 \Longrightarrow (x-4)(x+1)=0$. x>3 koşulundan x=4.

**CEVAP: D**

## SORU 9

Bir yazılım mühendisi üç farklı alt işlemden oluşan bir algoritma analiz etmektedir:

İşlem 1: 8'li arama ağacında $n^3$ düğüm aranıyor → $T_1(n) = \log_8(n^3)$ adım
İşlem 2: Her adımda 4 seçenek arasından 16 tanesi değerlendiriliyor → $T_2 = \log_4(16)$ sabit çarpan
İşlem 3: n/4 boyutlu bir veri kümesi ikiye bölünerek işleniyor → $T_3(n) = \log_2(n/4)$ adım

Toplam çalışma süresi aşağıdaki gibi tanımlanmıştır:
$T(n) = T_1(n) \cdot T_2 - T_3(n)$

Buna göre T(n) ifadesinin $\log_2(n)$ cinsinden sadeleştirilmiş hali aşağıdakilerden hangisidir?

A) $\log_2(n) + 1$
B) $\log_2(n) + 2$
C) $2 \cdot \log_2(n)$
D) $(3/2) \cdot \log_2(n)$
E) $\log_2(n) + 4$

**ÇÖZÜM**

$\log_8(n^3) = \log_2(n^3) / \log_2(8) = 3 \cdot \log_2(n) / 3 = \log_2(n)$

$\log_4(16) = \log_2(16) / \log_2(4) = 4 / 2 = 2$

$\log_2(n/4) = \log_2(n) - \log_2(4) = \log_2(n) - 2$

$T(n) = \log_8(n^3) \cdot \log_4(16) - \log_2(n/4) = \log_2(n) \cdot 2 - (\log_2(n) - 2) = 2 \cdot \log_2(n) - \log_2(n) + 2 = \log_2(n) + 2$

**CEVAP: B**

**SORU 10**

$P(x) = x^3 - 6x^2 + 11x - 6$ polinomu $(x-1)$ ile bölündüğünde kalan sıfırdır. $P(x)$'in tam çarpanları hangisidir?

A) $(x-1)(x-2)(x-4)$
B) $(x-1)(x-2)(x-3)$
C) $(x+1)(x-2)(x-3)$
D) $(x-1)(x+2)(x-3)$
E) $(x-1)(x^2+5x+6)$

## ÇÖZÜM

$P(1)=1-6+11-6=0$.

Polinom bölme: $x^3-6x^2+11x-6 = (x-1)(x^2-5x+6) = (x-1)(x-2)(x-3)$

**CEVAP: B**

## SORU 11

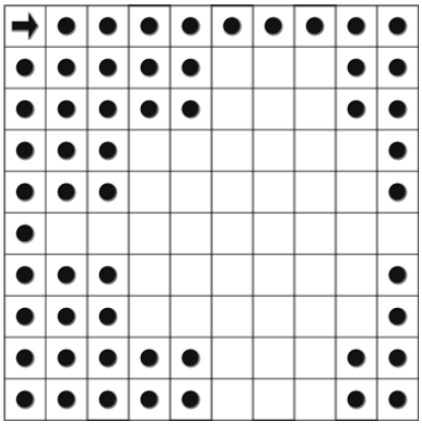
10x10'luk bir tahta üzerinde oynanan bir oyunda oyuncu her turda bir zar atarak gelen zar değerine göre hareket etmektedir. Oyuncu bulunduğu karede sağ, sol, yukarı veya aşağı yönlerinden birine bakıyor kabul edilir. Oyuncunun hareketi sonucunda tahtanın bir kenarından çıkması durumunda karşı kenardan tekrar tahtaya girdiği kabul edilmektedir. Örneğin, tahtanın 10. sütununda bulunan bir oyuncu 2 birim sağa gitmesi gerektiğinde tahtanın sağ kenarından çıkıp sol kenarından tahtaya girecek ve bulunduğu satırın 2. sütununa gidecektir. Tahtanın sol üst köşesinde sağa bakarak başlayan bir oyuncu, arka arkaya 4 zar atarak oynadığında tahtadaki karelerden kaç tanesine ulaşmış olamaz?

Zarların karşılıkları:
1: 2 kare ilerle,
2: saat yönünde 90 derece dön,
3: 1 kare geri git
4: saatin tersi yönünde 90 derece dön,
5: bekle
6: 1 kare ilerle

A) 45
B) 47
C) 50
D) 53
E) 55

**ÇÖZÜM**

Bu soru hızlıca 10x10 luk bir tahta çizerek ve hamleler bu tahta üzerinde işaretlenerek çözülebilir. Aşağıdaki şekilde ulaşılabilen kareler gösterilmiştir. 45 kareye ulaşılamadığı görülmektedir.



**CEVAP**: A

**SORU 12**

Bir öğrenci a,b,c,d,e,f,i harflerini istediği miktarda kullanarak yazabileceği tüm 5 harfli kelimelerin sayısını merak etmektedir. Bu kelimeleri türetirken öğrencinin takip etmesi gereken kurallar şunlardır:

1. kelime iki sessiz harfle başlayamaz veya bitemez
2. iki sesli harf yan yana gelemez
3. üç sessiz harf yanyana gelemez

Bu kurallar dahilinde öğrencinin yazabileceği kaç farklı kelime vardır?

A) 2016
B) 2160
C) 1872
D) 1728
E) 2304

**ÇÖZÜM**

Sesli harfleri o, sessiz harfleri x ile kodlarsak öğrencinin yazabileceği kelime örüntüleri şunlardır: oxoxo, oxxox, xoxox, xoxxo. Her örüntü için alternatifleri sayarsak:

- oxoxo: 3.4.3.4.3 = 432
- oxxox: 3.4.4.3.4 = 576
- xoxox: 4.3.4.3.4 = 576
- xoxxo: 4.3.4.4.3 = 576

Toplamda 432+576+576+576 = 2160 farklı kelime yazılabilir.

**CEVAP**: B

## SORU 13

Bir öğrenci aşağıda verilen algoritmayı bir dairesel array üzerinde çalıştırmak istemektedir. k elemanlı dairesel bir array'de i<=k olmak kaydıyla k+i'inci eleman i'inci eleman kabul edilir. Array başlangıçta 8 büyüklüğünde olup, içinde 1'den 8'e kadar sırayla sayılar tutulmaktadır. *pointer* başlangıçta 1. elemana işaret etmekte olup, *pointer*'da tutulan değer 9'dur (1. eleman arraydeki ilk elemandır).

Algoritma'da kullanılan yardımcı işlemler:

- `ilerle(x)`: pointer'ı x eleman ilerlet (saat yönünde).
- `değistir()`: pointer'ın işaret ettiği elemanın değeriyle pointer'da tutulan değeri birbiriyle değiştir.
- `p_deger()`: pointer'da tutulan değeri dön

```
1. p_deger() tek mi?
     1.1. evet:
        1.1.1. degistir()
        1.1.2. ilerle(p_deger())
     1.2. hayır:
        1.2.1. ilerle(p_deger() / 2)
        1.2.2. degistir()
2. p_deger() == 9 ise dur; değilse 1.'e git
```

Algoritma çalıştırılıp durduğunda arrayin son hali hangisinde verildiği gibi olur?

A) sonsuza kadar çalışmaya devam eder.
B) [1,2,3,4,5,6,7,8]
C) [6,1,3,8,5,4,7,2]
D) [6,1,3,2,5,4,7,8]
E) [6,1,2,4,5,3,7,8]

**ÇÖZÜM**

Step 1: Array: [1, 2, 3, 4, 5, 6, 7, 8], p: [@1, 9]
Step 2: Array: [9, 2, 3, 4, 5, 6, 7, 8], p: [@2, 1]
Step 3: Array: [9, 1, 3, 4, 5, 6, 7, 8], p: [@4, 2]
Step 4: Array: [9, 1, 3, 4, 2, 6, 7, 8], p: [@5, 5]
Step 5: Array: [9, 1, 3, 4, 5, 6, 7, 8], p: [@7, 2]
Step 6: Array: [9, 1, 3, 4, 5, 6, 7, 2], p: [@8, 8]
Step 7: Array: [9, 1, 3, 8, 5, 6, 7, 2], p: [@4, 4]
Step 8: Array: [9, 1, 3, 8, 5, 4, 7, 2], p: [@6, 6]
Step 9: Array: [6, 1, 3, 8, 5, 4, 7, 2], p: [@1, 9]

**CEVAP**: C

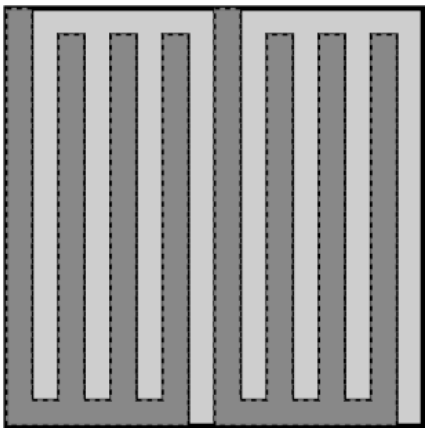
**SORU 14**

Bir kenarı 1 birim olan bir kare 4 özdeş parçaya bölündüğünde her bir parçanın çevresi en fazla kaç birim olabilir?

A) 2
B) 2.5
C) 4
D) 6.32
E) sonsuz

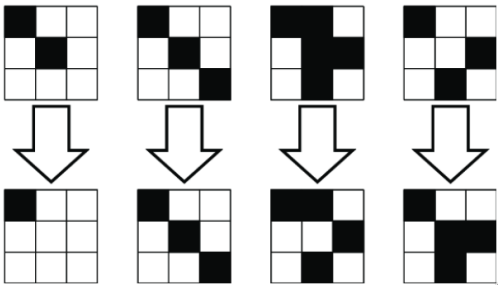
**ÇÖZÜM**

Aşağıdaki çözümde görüleceği üzere kareyi bu şekilde 4 eşit parçaya bölmek mümkündür. Burada her bir parçadaki "diş" sayısı istenildiği kadar arttırılabilir. Bu sayede istenirse sonsuz dişe sahip parçalar elde edilebileceğinden cevap E şıkkıdır.



**CEVAP**: E

**SORU 15**

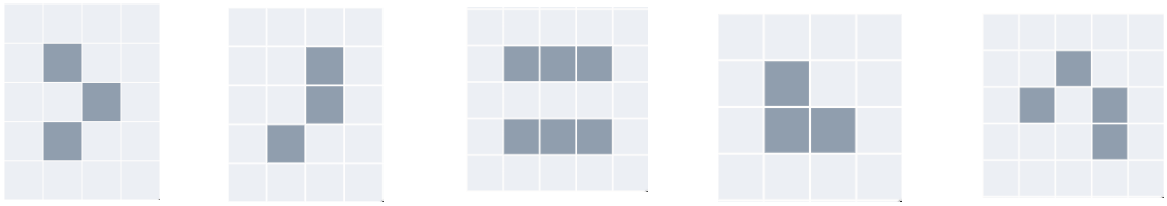


Conway'in "hayat oyunu"nda, eş kare hücrelere bölünmüş (grid) sonsuz büyüklükteki bir tahtada temel birkaç kural yardımı ile basit bir hayat simülasyonu yapılır. Her bir hücre ya boştur, ya da içinde bir canlı organizma yaşamaktadır. Oyun tur bazlıdır, her turda kurallar tüm kareler için hesaplanır ve bir sonraki tura geçilir. Tüm kurallar tüm karelere eş zamanlı olarak uygulanır. Kurallar şu şekildedir:

- 1 veya 0 komşusu canlı olan karelerdeki canlılar ölür
- 2 veya 3 komşusu canlı olan karelerdeki canlılar yaşamaya devam eder
- 4 veya daha fazla komşusu canlı olan karelerdeki canlılar ölür
- 3 komşusu olan her boş karede yeni bir canlı yaşamaya başlar

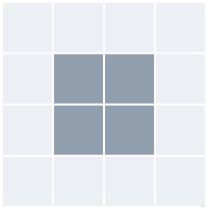
Aşağıdaki başlangıçlardan hangisi sonsuza kadar yaşayan bir sistem üretir? (koyu renk kareler canlı karelerdir) (tahtada en az bir canlı kare bulunması yaşamın devam ettiği anlamına gelir)

A) B) C) D) E)



**ÇÖZÜM**

Verilen başlangıçlardan yalnızca D şıkkındaki başlangıç sonsuz bir yaşam döngüsü üretir. Diğerleri birkaç tur sonunda yok olurlar. D şıkkındaki başlangıcın ulaştığı sonsuz yaşam döngüsü aşağıdadır:



**CEVAP**: D

**[Soru 16-17 için açıklama]**

Bir tamsayı dizisini soldan sağa işleyen bir yığın algoritması tanımlayalım. Yığın başlangıçta boştur. Dizinin elemanları sırayla alınır ve her eleman **x** için aşağıdaki kurallar uygulanır.

• Yığın boş ise **x** yığıta eklenir.

• Yığın boş değilse ve yığıtın üstündeki eleman **x**'e eşitse, üstteki eleman silinir.

• Aksi halde, yığıtın üstündeki eleman |**üstteki eleman − x**| değeri ile değiştirilir.

Dizinin tüm elemanları işlendiğinde algoritma sonlanır.

Örnek olarak **{5, 1, 4}** dizisi işlendiğinde yığın sırasıyla **{5}**, sonra **{4}**, sonra **{}** olur.

## SORU 16

A: {7, 3, 10, 6, 4, 2, 8} olduğunda algoritma sonlandığında yığıtın üstündeki eleman ne olur?

A) 2
B) 4
C) 6
D) 8
E) 10

## ÇÖZÜM

Başlangıç: {}

7 alınır → {7}
3 alınır → üst eleman |7−3| = 4 olur → {4}
10 alınır → üst eleman |4−10| = 6 olur → {6}
6 alınır → üstteki elemanla eşit, silinir → {}
4 alınır → {4}
2 alınır → üst eleman |4−2| = 2 olur → {2}
8 alınır → üst eleman |2−8| = 6 olur → {6}

Son durumda üstteki eleman **6**'dır.

**CEVAP**: C

**SORU 17**

A: {4, 9, 5, 1, 8, 2, 6} dizisinin sonuna tek bir sayı eklenecektir. Aşağıdakilerden hangisi eklenirse algoritma sonlandığında yığın boş olur?

A) 1
B) 2
C) 3
D) 4
E) 5

**ÇÖZÜM**

Önce temel dizi işlenir:
{}
4 → {4}
9 → {5}
5 → {}
1 → {1}
8 → {7}
2 → {5}
6 → {1}

Yani {4, 9, 5, 1, 8, 2, 6} işlendiğinde sonda yığında **1** kalır.

Şimdi sonuna bir sayı eklenirse:

- 1 eklenirse: üstteki 1 ile eşit olduğu için silinir, yığın boş olur.
- 2 eklenirse: $|1-2| = 1$ kalır.
- 3 eklenirse: $|1-3| = 2$ kalır.
- 4 eklenirse: $|1-4| = 3$ kalır.
- 5 eklenirse: $|1-5| = 4$ kalır.

Dolayısıyla yalnızca **1** eklendiğinde yığın boş olur.

**CEVAP**: A

**[Soru 18-19 için açıklama]**

Bir bilgisayar oyununda ekran üzerinde yan yana dizilmiş **n** adet sayı blokundan oluşan bir dizi bulunmaktadır. Oyuncu, her hamlede yan yana duran ve değerleri birbirine eşit olan iki blok seçer; bu iki blok silinerek yerlerine değerlerinin **1** fazlası olan yeni bir blok eklenir. Sağda kalan bloklar sola kayarak dizinin bütünlüğü korunur.

Oyuncu, yan yana duran eşit değerli blok kalmayana kadar bu işlemi tekrarlar. Birden fazla birleştirme seçeneği olması durumunda, oyuncu istediği çifti seçmekte özgürdür.

Örnek olarak başlangıç dizisi **A = [2, 1, 1, 2]** olsun:

- Oyuncu **1** ve **1**'i birleştirirse dizi **[2, 2, 2]** olur. Ardından **2**'leri birleştirerek **[3, 2]** veya **[2, 3]** elde eder (kalan eleman sayısı 2).
- Eğer dizi **A = [1, 1, 1, 1]** olsaydı, sırasıyla **[2, 1, 1]**, **[2, 2]** ve nihayetinde **[3]** elde edilirdi (kalan eleman sayısı 1).

## SORU 18

Başlangıç dizisi **A = [3, 2, 2, 1, 1, 2, 3, 4]** olarak verildiğinde, en zekice seçimler yapıldığında oyun bittiğinde dizide en az kaç eleman kalır?

A) 1
B) 2
C) 3
D) 4
E) 5

**ÇÖZÜM**

Birleşme işleminin matematiksel bir özelliği bulunmaktadır. Aynı değere sahip iki elemanın (örneğin **k** ve **k**) birleşerek **k+1** değerini alması, ikili (binary) sayı sistemindeki toplama işlemiyle aynı mantıktadır (**2^k + 2^k = 2^{k+1}**).

Dizideki her **x** değerini **2^x** olarak kabul edip topladığımızda, işlemlerin sırası ne olursa olsun elde edilen bu toplam sabit kalacaktır.

Başlangıç dizisi: **A = [3, 2, 2, 1, 1, 2, 3, 4]**

Toplam = **2^3 + 2^2 + 2^2 + 2^1 + 2^1 + 2^2 + 2^3 + 2^4 = 8 + 4 + 4 + 2 + 2 + 4 + 8 + 16 = 48** olur.

48 sayısı ikili sistemde **32 + 16**, yani **2^5 + 2^4** şeklinde ifade edilebilir. Başka bir deyişle, toplam değeri 48 olan bir sistem minimum iki parçaya (**5** ve **4** değerli iki blok) indirgenebilir. Sonucun tek bir blok (kalan 1) olabilmesi için toplamın tam olarak **2^n** formatında olması gerekirdi.

Adım adım simülasyonla bu sonuca şu şekilde ulaşılabilir:

- 1, 1 birleştirilir → **2**: Dizi **[3, 2, 2, 2, 2, 3, 4]** olur.
- En soldaki **2, 2** birleştirilir → **[3, 3, 2, 2, 3, 4]**
- En soldaki **3, 3** birleştirilir → **[4, 2, 2, 3, 4]**
- Kalan **2, 2** birleştirilir → **[4, 3, 3, 4]**
- 3, 3 birleştirilir → **[4, 4, 4]**
- 4, 4 birleştirilir → **[5, 4]**. Kalan eleman sayısı **2**'dir. Diğer şıklar teorik alt sınır olan 2'den büyüktür veya (A şıkkı gibi) matematiksel olarak imkansızdır.

**CEVAP**: B

**SORU 19**

Başlangıç dizisi **A = [3, 2, 1, 1, 1, 1, 1, 1, 2, 3]** olarak verildiğinde, en kötü senaryoda (rastgele ve hatalı seçimler sonucunda) oyun tıkandığında dizide en fazla kaç eleman kalabilir?

A) 6
B) 7
C) 8
D) 9
E) 10

**ÇÖZÜM**

Oyunun sonlanması veya tıkanması durumu, dizi içerisinde yan yana konumlanmış ve sayısal değerleri özdeş olan herhangi bir blok çiftinin kalmamasıyla gerçekleşir. Verilen başlangıç dizisi **10** elemandan oluşmaktadır ve **A = [3, 2, 1, 1, 1, 1, 1, 1, 2, 3]** formundadır.

Dizinin merkezinde **6** adet **1** değeri ardışık şekilde sıralanmıştır. En kötü senaryo analizinde (oyunun mümkün olan en yüksek eleman sayısında kilitlenmesi hedefiyle), bu **1**'lerin arasına farklı değerlerin sızmasını sağlayarak zincirleme birleşme potansiyelini minimize etmeliyiz. Bilindiği üzere her birleştirme hamlesi, dizideki toplam eleman miktarını **1** birim eksiltmektedir.

Şayet tek bir hamle ile **9** elemana düşülürse, mevcut yapıda tüm **1**'lerin birbirinden izole edilmesi teknik olarak mümkün değildir; mutlaka komşu **1** çiftleri kalacaktır. Dolayısıyla **9** elemanlı bir dizide tıkanma yaşanması matematiksel bir imkansızlıktır.

İki birleşme yapılıp **8** elemana inildiğinde, merkezdeki altı adet **1**'den iki çiftin birleştirilmesiyle elimizde fazladan iki adet **2** ve iki adet **1** kalır. Orijinal dizideki mevcut iki adet **2** ile birlikte toplamda dört adet **2** elde edilir. Bu dört adet **2** değerinin yan yana gelmesini önlemek adına aralarına en az üç adet farklı değer yerleştirilmelidir. Ancak elde kalan iki adet **1** bu izolasyon için sayıca yetersizdir ve en az iki adet **2** mutlaka temas

edecektir. Bu sebeple oyun **8** elemanda da kilitlenemez; oluşan bu **2**'ler mecburen birleşecektir.

Zorunlu olan bu üçüncü hamleyi de içeren ve sayıların optimum şekilde yalıtıldığı simülasyon basamakları şöyledir:

- **Başlangıç Durumu:**
  Dizi: [3, 2, 1, 1, 1, 1, 1, 1, 2, 3]
- **1. Adım:** Ortadaki bloklardan 4. ve 5. indeksteki **1**'ler eşleşir (yeni değer **2**).
  Dizi: [3, 2, 1, **2**, 1, 1, 1, 2, 3]
- **2. Adım:** Kalan ardışık bloklardan 6. ve 7. indeksteki 1'ler eşleşir (yeni değer **2**).
  Dizi: [3, 2, 1, 2, **2**, 1, 2, 3]
- **3. Adım (Zorunlu):** Yan yana gelen **2**'ler kural gereği birleşmek zorundadır (yeni değer **3**). Dizi: [3, 2, 1, **3**, 1, 2, 3]

Bu final tablosunda (komşuluk dizilimi 3-2, 2-1, 1-3, 3-1, 1-2, 2-3 şeklindedir) yan yana aynı değerli blok kalmadığı için oyun kilitlenir. Neticede 10 elemanla başlayan süreçte, en yüksek yalıtım başarısıyla dahi maksimum **7** elemanlı bir tıkanma noktasına varılabilir.

**CEVAP**: B

## SORU 20

Bir depo otomasyon sisteminde görevli robot kol, yan yana dizilmiş **n** adet farklı ağırlıktaki koliyi hafiften ağıra doğru (soldan sağa) sıralama ile yükümlüdür. Robotun icra edebildiği yegane işlem, dizinin en solundan başlayarak belirlenen bir **k.** sıradaki koliye kadar olan bölümü tersyüz etmektir.

Robotun sıralama algoritması adım adım şu kuralları uygular:

1. Sıralanmamış kısımdaki en ağır koliyi tespit et.
2. Eğer bu koli halihazırda sıralanmamış kısmın en sağında ise 5. adıma geç.
3. Eğer koli en sol (1. sıra) konumda değilse, kolinin bulunduğu yere kadar olan alt diziyi ters çevirerek en başa getir.
4. En sol tarafa gelmiş olan koliyi, sıralanmamış kısmın sonuna yerleştirmek için ilgili alt diziyi ters çevir.
5. Sıralanmış kabul edilen koli sayısını bir artır ve kalan koliler için 1. adıma dön.

Başlangıç ağırlık dizisi **A = [12, 5, 18, 4, 15, 8, 10]** olarak verilmiştir.

En ağır **iki** koli kendi nihai yerlerine yerleştirildiği anda (üçüncü döngüye geçilmeden hemen önce), dizide baştan (soldan) 2. sırada hangi koli bulunur?

A) 5
B) 8
C) 10
D) 12
E) 15

## ÇÖZÜM

Algoritmanın çalışma prensibi, her adımda sıralanmamış kısımdaki en büyük elemanı seçerek önce en başa, ardından da sıralanmamış kısmın sonuna taşımaya dayanmaktadır.

**Başlangıç Durumu:**

Dizi: [12, 5, 18, 4, 15, 8, 10]
Sıralanmamış Kısım: 7 eleman.

**1. Döngü (En ağır 1. kolinin yerleştirilmesi):**
- En ağır koli 18'dir (3. sırada).
- 18 en sağda değildir.
- Baştan 3. elemana kadar ters çevir: [12, 5, 18] → [18, 5, 12, 4, 15, 8, 10]
- Tüm sıralanmamış kısmı (7 eleman) ters çevir: [10, 8, 15, 4, 12, 5, 18]
- 18 yerleşti, sıralanmamış kısım 6 elemana düştü.

**2. Döngü (En ağır 2. kolinin yerleştirilmesi):**
- Kalan kısımdaki en ağır koli 15'tir (3. sırada).
- 15 en sağda değildir.

- Baştan 3. elemana kadar ters çevir: [10, 8, 15] → [15, 8, 10, 4, 12, 5, 18]
- Sıralanmamış kısmı (ilk 6 eleman) ters çevir: [5, 12, 4, 10, 8, 15, 18]
- 15 yerleşti, sıralanmamış kısım 5 elemana düştü.

İkinci kolinin yerleştirilmesi tamamlandığında dizinin güncel hali: [5, 12, 4, 10, 8, 15, 18] şeklindedir. Bu durumda soldan 2. koli **12**'dir.

**CEVAP**: D

**SORU 21**

8 farklı kitap bir rafa dizilecektir. Ancak belirli 3 kitap yan yana gelmemelidir. Kaç farklı diziliş vardır?

A) $8!-3!\cdot 6!$
B) $8!-3!\cdot 6!\cdot 2$
C) $8!-6!$
D) $8!-3!\cdot 5!$
E) Hiçbiri

**ÇÖZÜM**
Toplam diziliş sayısı: 8!
İstenmeyen durum: Bu 3 belirli kitap yan yana olsun. Bu 3 kitabı tek bir blok gibi düşünelim. O zaman elimizde:

- 1 blok
- diğer 5 kitap

olmak üzere toplam 6 nesne olur. Bunlar: 6! şekilde dizilir. Ama blok içindeki 3 kitap kendi arasında da 3! şekilde sıralanabilir.

Dolayısıyla istenmeyen durum sayısı: $6!\cdot 3!6!$ olur.

Bu durumda istenen sonuç:

$8!-6!\cdot 3!$

**CEVAP**: A

**SORU 22**

10 kişilik bir gruptan 4 kişilik bir komite seçilecektir. Ancak bu gruptaki iki kişi A ve B aynı anda komitede bulunamaz. Kaç farklı seçim yapılabilir?

A) 70
B) 140
C) 154
D) 182
E) 308

**ÇÖZÜM**

Önce kısıtsız seçim sayısı:

$$\binom{10}{4} = \frac{10 \times 9 \times 8 \times 7}{4 \times 3 \times 2 \times 1} = 210$$

Şimdi istenmeyen durum: A ve B birlikte komitede olsun.

A ve B zaten seçilmiş olur. Geriye 8 kişiden 2 kişi daha seçilir:

$$\binom{8}{2} = \frac{8 \times 7}{2 \times 1} = 28$$

Dolayısıyla istenen sayı:

210 - 28 = 182

**CEVAP**: D

**SORU 23**

Bir üniversite senatosunda 6 Profesör ve 8 Doçent bulunmaktadır. Bu gruptan 5 kişilik bir araştırma komisyonu seçilecektir. Ancak seçim için şu kısıtlamalar getirilmiştir:

- Komisyonda en az 2 Profesör bulunmak zorundadır.
- Profesörler arasındaki A ve B şahısları, aynı komisyonda beraber yer alamazlar.

Bu koşullara uygun kaç farklı komisyon kurulabilir?

A) 1120
B) 1306
C) 1406
D) 1526
E) 1746

**ÇÖZÜM**

1. Adım: Genel Durum (En az 2 Prof Şartı)

- (2P, 3D): C(6,2) x C(8,3) = 15 x 56 = 840
- (3P, 2D): C(6,3) x C(8,2) = 20 x 28 = 560
- (4P, 1D): C(6,4) x C(8,1) = 15 x 8 = 120
- (5P, 0D): C(6,5) x C(8,0) = 6 x 1 = 6
- Toplam: 1526

2. Adım: Yasaklı Durum (A ve B'nin beraber olduğu ve "en az 2 Prof" şartını sağlayanlar)

A ve B zaten 2 Profesördür. Onları komisyona sabitlediğimizde kalan 3 kişiyi kalan 12 kişiden seçeriz. Ancak "en az 2 Prof" şartı her halükarda sağlanmış olur:

- (A, B Sabit + Geri kalan 12 kişiden 3 kişi): $C(12,3) = \frac{12 \times 11 \times 10}{3 \times 2 \times 1} = 220$

3. Adım: Sonuç

- 1526 - 220 = 1306

**CEVAP**: B

**SORU 24**

5 Elma ve 5 portakal bir sıraya dizilecektir. Elmalar ve portakallar alternatif dizilecektir. Yani sıranın bir yerinde elma varsa sonra portakal sonra elma vs. şeklinde bir dizilim olacaktır. Kaç farklı dizilim mümkündür?

A) 252
B) 14400
C) 28800
D) 3628800
E) Hiçbiri

**ÇÖZÜM**

Elmaların ve portakalların alternatif (bir elma, bir portakal veya bir portakal, bir elma şeklinde) dizilmesi için iki temel senaryo vardır:

1. Senaryo: Dizilimin Elma ile Başlaması (E-P-E-P-E-P-E-P-E-P)

- Elmaların yerleşimi: 5 elma kendi aralarında 5! farklı şekilde dizilir.
- Portakalların yerleşimi: 5 postakal kendi aralarında 5! farklı şekilde dizilir.
- Bu senaryo için toplam dizilim: 5!

2. Senaryo: Dizilimin Portakal ile Başlaması (P-E-P-E-P-E-P-E-P-E)

- Portakalların yerleşimi: 5 portakal kendi aralarında 5! farklı şekilde dizilir.
- Elmaların yerleşimi: 5 elma kendi aralarında 5! farklı şekilde dizilir.
- Bu senaryo için toplam dizilim: 5!

**Toplam Sonuç**

Bu iki senaryo birbirinden bağımsız (ayrık) olduğu için toplarız:

$$(5! \cdot 5!) + (5! \cdot 5!) = 2 \cdot (5! \cdot 5!)$$

$$2 \cdot (120 \cdot 120) = 2 \cdot 14.400 = 28.800$$

**CEVAP**: C

## SORU 25

1’den 200’e kadar olan (1 ve 200 dahil) doğal sayılar arasından kaç tanesi 2, 3 ve 5 sayılarının hiçbirine bölünmez?

A) 52
B) 53
C) 54
D) 55
E) 56

### ÇÖZÜM

2, 3 ve 5'in en küçük ortak katı 30'dur. 1 ile 30 arasında bu sayılara bölünmeyenleri sayarsak:

{1, 7, 11, 13, 17, 19, 23, 29} olmak üzere 8 tanedir.

- 200 içinde kaç tane 30 var? 200 / 30 = 6.
- 6 x 8 = 48 sayı buradan gelir.
- Geriye kalan son 20 sayıya (181-200 arası) manuel bakarsak: 181, 187, 191, 193, 197, 199 olmak üzere 6 tane de oradan gelir.

Yani, 48 + 6 = 54.

**CEVAP**: C

## SORU 26

Yönlü ve döngüsüz bir çizge (DAG) için aşağıdakilerden hangisi topologik sıralamanın tek (unique) olması için gerekli ve yeterlidir?

A) Çizge bağlıdır
B) Her düğüme gelen kenar sayısı (indegree) $\leq 1$’dir
C) Her ardışık düğüm çifti arasında bir kenar vardır
D) Çizge bir ağaçtır
E) Tüm düğümlerden çıkan kenar sayısı (outdegree) $\geq 1$’dir

### ÇÖZÜM

Tek topolojik sıralama $\Leftrightarrow$ sıralamadaki her i için $u_i \rightarrow u_{i+1}$ arasında tek bir kenar vardır.

**CEVAP**: C

## SORU 27

Genişlik öncelikli arama (Breadth First Search - BFS) ve Derinlik öncelikli arama (Depth First Search - DFS) ile ilgili aşağıdakilerden hangisi her zaman doğrudur?

A) DFS ağacı minimum yüksekliktedir.
B) BFS ağacı minimum yüksekliktedir.
C) DFS aynı zamanda başlangıç düğümünden diğer düğümlere olan en kısa yolu bulur.
D) BFS ağırlıklı çizgelere uygulandığında aynı zamanda başlangıç düğümünden diğer düğümlere olan ağırlıklı en kısa yolu bulur.
E) DFS her zaman BFS'ten daha hızlıdır.

**ÇÖZÜM**

BFS algoritmasının çalışma mantığına göre düğümler seviye seviye ziyaret edilir. Bu nedenle algoritma sonunda oluşan BFS ağacının minimum yükseklikte olması garantidir.

**CEVAP**: B

## SORU 28

Bir yönlü çizge veriliyor. Bu çizgenin güçlü bağlı bileşenleri (SCC) bulunuyor ve her bir SCC, tek bir düğüm olacak şekilde sıkıştırılıyor. Bu şekilde elde edilen yeni çizgeye yoğuşum çizgesi denir.

Yoğuşum çizgesinde kenarlar şu şekilde oluşturulur: Orijinal çizgede farklı iki SCC arasında, birinciden ikinciye yönlü bir kenar varsa, yoğuşum çizgesinde de bu iki SCC'ye karşılık gelen düğümler arasında yönlü bir kenar bulunur.

Bu yoğuşum çizgesi için aşağıdakilerden hangisi her zaman doğrudur?

A) Her zaman bağlıdır.
B) Her zaman iki bölümlü (bipartite)'tır.
C) Her zaman yönsüzdür.
D) Hiçbir zaman döngü içermez.
E) Her zaman ağaçtır.

**ÇÖZÜM**

SCC'ler tek düğüm yapılınca döngü kalmaz kalmaz. Yani yoğuşum çizgesi döngü içermez. Eğer içerirse o zaman ilk çizgenin SCC'leri yanlış hesaplanmış manasındadır.

**CEVAP**: D

**SORU 29**

Bir çizgede, V düğüm ve E kenar sayısını temsil etmek üzere “Seyrek (sparse)” çizgelerde kenar sayısı düğüm sayısına yakındır (yani çok fazla kenar yoktur).

Bir çizgeyi temsil etmek için iki yöntem vardır:

Komşuluk Matrisi (Adjacency Matrix)
Komşuluk Listesi (Adjacency List)

Seyrek bir çizgede, Komşuluk Matrisi yerine Komşuluk Listesi kullanmanın en önemli avantajı aşağıdakilerden hangisidir?

A) İki düğüm arasında kenar olup olmadığını her zaman sabit sürede kontrol edebilmek
B) Kullanılan belleği, düğüm sayısının karesi kadar yer kaplamak yerine, yalnızca düğüm ve kenar sayısı kadar olacak şekilde azaltmak
C) Bir düğümün kaç komşusu olduğunu bulmak için, sadece o düğüme ait küçük bir listeyi incelemek yerine tüm düğümler üzerinden tek tek kontrol yapmak zorunda kalmak
D) Yeni bir kenar eklerken çok daha fazla işlem yapmak zorunda kalmak
E) Çok sayıda kenarı olan yoğun çizgelerde daha hızlı çalışmak

**ÇÖZÜM**

Seyrek (sparse) bir çizgede kenar sayısı düğüm sayısına yakın ve azdır ($E \approx V$); komşuluk matrisi ise tüm olası düğüm çiftleri için yer ayırdığı için $V \times V$ boyutunda çok büyük bir bellek kullanırken, komşuluk listesi sadece gerçekten var olan kenarları saklar ve bu nedenle yaklaşık düğüm ve kenar sayısı kadar ($V + E$) bellek kullanır; dolayısıyla özellikle seyrek çizgelerde komşuluk listesi, gereksiz boş bağlantıları tutmadığı için çok daha az bellek kullanarak önemli bir alan tasarrufu sağlar.

**CEVAP**: B

## SORU 30

Aşağıdaki yönlü çizgenin komşuluk listeleri aşağıda verilen sırayla gezilmektedir ve Derinlik Öncelikli Arama (Depth First Search - DFS), düğüm A'dan başlatılmaktadır.

- A: B, C
- B: D, E
- C: F
- D: C, F
- E: F, G
- F: H
- G: D, H
- H: —

DFS sırasında bir düğüm ilk kez görüldüğünde ona bir keşif zamanı d[u], o düğümün tüm komşuları tamamlandığında ise bir bitiş zamanı f[u] veriliyor. Sayaç her keşif ve her bitişte 1 artıyor; başlangıçta sayacın değeri 0'dır.

Buna göre aşağıdakilerden hangisi doğrudur?

A) d[F]=6 ve f[F]=9
B) d[C]=4 ve f[C]=7
C) d[E]=10 ve f[E]=15
D) d[G]=11 ve f[G]=14
E) d[D]=3 ve f[D]=10

**ÇÖZÜM**

DFS, A düğümünden başlatılır ve komşular verilen sırayla gezilir. Sayaç her keşif ve her bitiş anında 1 artırılır. Buna göre önce A keşfedilir ve d[A]=1 olur; A'dan B'ye gidilir ve d[B]=2; B'den D'ye gidilir ve d[D]=3; D'den C'ye gidilir ve d[C]=4; C'den F'ye gidilir ve d[F]=5; F'den H'ye gidilir ve d[H]=6 olur. H'nin komşusu olmadığı için f[H]=7, ardından F tamamlanır ve f[F]=8, sonra C tamamlanır ve f[C]=9, ardından D tamamlanır ve f[D]=10. B'ye geri dönülerek E'ye gidilir ve d[E]=11; E'den F zaten ziyaret edilmiş olduğu için geçilir, sonra G keşfedilir ve d[G]=12; G'nin komşuları D ve H daha önce ziyaret edildiğinden G tamamlanır ve f[G]=13, ardından E tamamlanır ve f[E]=14, sonra B tamamlanır ve f[B]=15. Son olarak A'nın diğer komşusu C zaten ziyaret edilmiş olduğundan A tamamlanır ve f[A]=16. Böylece sadece E şıkkında verilen zaman doğrudur.

**CEVAP**: E

## SORU 31

Kenar ağırlıkları pozitif olan bir çizge verilmiştir. Her düğüm (ing: vertex) bir şehri, her kenar (ing: edge) ise iki şehir arasındaki kara yolu mesafesini temsil etmektedir. Bir $s$ şehrinden $t$ şehrine gitmek için aşağıdaki arama yöntemi kullanılmaktadır:

- Arama yöntemi: Derinlik Öncelikli Arama (ing: Depth-First Search, DFS)
- Her düğüm için bir $h(n)$ değeri tanımlıdır.
- $h(n)$ düğüm $n$'den hedef $t$'ye olan kuş uçuşu mesafe tahminini verir; bu nedenle $h(t) = 0$ dır.
- Bir düğüm genişletildiğinde, komşular küçük $h(n)$ değerli olan önce ziyaret edilecek şekilde önceliklendirilir.
- DFS yığını (ing: stack) son-giren-ilk-çıkar (ing: Last-in-first-out, LIFO) olduğu için, bu önceliği sağlamak amacıyla komşular yığına büyük $h(n)$'den küçük $h(n)$'e doğru eklenir.
- Algoritma hedef düğüm $t$'ye ulaştığı anda durur ve bulduğu yolu döner.

*En kısa yol*, toplam kenar maliyeti en küçük yol anlamındadır. Yukarıda tarif edilen bu algoritma ile ilgili aşağıdakilerden hangisi doğrudur?

A) Kuş uçuşu mesafe kullanıldığı için algoritma her zaman en kısa yolu bulur.
B) Algoritma, tüm yolları sistematik olarak denediği için her zaman en iyi çözümü bulur.
C) $h(n)$ kullanımı algoritmanın davranışını değiştirmez, sonuçlar standart DFS ile aynıdır.
D) Algoritma, Genişlik Öncelikli Arama (ing: Breadth-First Search, BFS) gibi çalıştığı için en kısa yolu bulur.
E) Algoritma, standart DFS'e göre genellikle daha hızlı sonuç verebilir ve bazı durumlarda daha iyi (daha kısa) bir yol bulabilir, ancak en kısa yolu garanti etmez.

**ÇÖZÜM**

Tanımlanan algoritma aslında klasik Derinlik Öncelikli Arama'nın (DFS'in) bir varyantıdır ve sadece komşuların ziyaret sırasını $h(n)$ heuristiğine göre değiştirir; yani arama uzayını tamamen değiştirmez, yalnızca *daha umut verici* görünen dalları önce dener. Bu sayede bazı durumlarda hedefe daha hızlı ulaşabilir ve tesadüfen daha kısa bir yol bulabilir. Ancak algoritma hedefe ulaştığı anda durduğu ve tüm olası yolları sistematik olarak karşılaştırmadığı için, bulunan yolun toplam maliyet açısından en kısa yol olduğu garanti edilemez. Dolayısıyla bu yöntem performansı pratikte iyileştirebilir ama optimal çözüm garantisi vermez; bu da **E** şıkkında ifade edilen durumdur.

**CEVAP**: E

**[Soru 32-50 için açıklama]**

- Soruları C programlama dili çerçevesinde cevaplayınız.
- Derleyici olarak gcc kullanıldığını varsayınız.
- Gerekli tüm başlık (header) dosyalarının verilen programa dahil edildiğini varsayınız.

## SORU 32

Kenar uzunluğu 2 olan bir kare, koordinat düzleminde orijin merkezlidir. Bu durumda bu karenin köşeleri $(-1,-1), (-1,1), (1,-1), (1,1)$ noktalarındadır. Bu karenin içinde, yarıçapı 1 olan ve merkezi yine orijinde bulunan bir çember yer almaktadır. Aşağıdaki C programlama dilinde yazılmış olan bilgisayar programı $N$ defa işleyen bir döngü içinde (a) kare içinde eşit olasılıkla rastgele bir konumda bir nokta üretir, (b) bu noktalardan kaç tanesinin çemberin içinde kaldığını $M$ tamsayı değişkeni içinde sayar. Program döngü bittikten sonra (4.0 * M) / N değerini döner. *İpucu:* M / N oranı üzerinden düşününüz.

```
double hesapla(int N) {
    int M = 0;
    for (int i = 0; i < N; i++) {
        double x = -1.0 + 2.0 * rand() / (double)RAND_MAX;
        double y = -1.0 + 2.0 * rand() / (double)RAND_MAX;
        if (x * x + y * y <= 1.0) M++;
    }
    return (4.0 * M) / N;
}
```

N çok büyük olduğunda (yani $N \to \infty$), fonksiyonun döndüğü değer aşağıdakilerden hangisine en yakındır?

**Not**: rand() fonksiyonu 0 ile RAND_MAX arasında rastgele bir tamsayı üretir. RAND_MAX, rand() fonksiyonunun üretebileceği en büyük değeri temsil eder.

A) $\pi/4$
B) $2$
C) $3$
D) $\pi$
E) $4$

**ÇÖZÜM**

Bu sorunun bağlamı algoritmik, özü ise matematikseldir. Her iki yaklaşımı bir araya getirmeyi gerektirmektedir. Karenin alanı $a^2 = 2^2 = 4$'tür. Çemberin alanı ise $\pi r^2 = \pi 1^2 = \pi$'dir. Noktalar kare içinde eşit olasılıkla dağıtıldığı için $\frac{M}{N} \approx \frac{çemberin\ alan}{karenin\ alan} = \frac{\pi}{4}$ olur yani $\frac{M}{N} \approx \frac{\pi}{4}$. Buradan $\frac{4M}{N} \approx \pi$ olur. Dolayısı ile döndürülen değer $N$ arttıkça π değerine yaklaşır.

**CEVAP**: D

**SORU 33**

Bir merdivende 0'dan $n$'e kadar (0 ve $n$ dahil) numaralandırılmış basamaklar vardır. Verilen $c$ dizisi için $c[i]$ değeri $i$ numaralı basamağa basmanın maliyetini göstermektedir. Bir kişi ilk başta 0. basamakta sıfır maliyet ile durmaktadır. Bu kişi her hareketinde yalnızca bir basamak veya iki basamak çıkabilmektedir. $F(i)$ ile $i$. basamağa en az toplam maliyetle ulaşma değerini aşağıdaki şekilde tanımlanmıştır:

$$F(0) = 0, \quad F(1) = c[1], \qquad F(i) = min(F(i-1), F(i-2)) + c[i] \quad (i \geq 2 \ için)$$

Bu bağıntıyı verimli bir şekilde hesaplamak için aşağıdaki kod kullanılıyor:

```
#define ATANMAMIS -1

int F(int i, int c[], int memo[]) {
  if (i == 0) return 0;
  if (i == 1) return c[1];
  if (memo[i] == ATANMAMIS) {
    int a = F(i - 1, c, memo);
    int b = F(i - 2, c, memo);
    memo[i] = min(a, b) + c[i];
  }
  return memo[i];
}
```

Başlangıçta memo dizisindeki tüm elemanlar ATANMAMIS değerine ayarlanmıştır. $A$ ve $B$, $n$ 'den bağımsız tamsayı sabit iki değer olsun. F(n, c, memo) çağrısı yapıldığında, memo dolu olup hemen dönen çağrılar da dahil olmak üzere, F(n, c, memo) fonksiyonu toplam kaç defa çağrılmaktadır?

**Not**: Bu koddaki `min(a, b)` fonksiyonu verilen a ve b parametrelerinden minimumunu dönmek üzere tanımlandığını kabul edin.

A) $A$

B) $Alog(n) + B$

C) $An + B$

D) $An^2 + B$

E) $Af^n + B \ (1 < f \leq 2)$

**ÇÖZÜM**

Fonksiyon ilk kez $F(n)$ için çağrıldığında, hesaplama zinciri $F(n-1), F(n-2), \ldots, F(1), F(0)$ şeklinde aşağı doğru iner ve bu sırada her $i > 0$ için sonuç memo[i] içine bir kez yazılır ($F(0) \rightarrow 0$). Daha sonra yapılan çağrılarda, eğer bir $i$ değeri daha önce hesaplanmışsa memo[i] üzerinden doğrudan döner ve yeni alt çağrılar oluşturulmaz. Böylece her $i$ için özyineleme yapılmasına gerek kalmadığı için çağrı sayısı $n$ ile orantılı hale gelir. Bu nedenle karmaşıklık doğrusal olup $An + B$ biçimindedir. Burada $A$ sıfırdan büyük bit tamsayı olup $B$ herhangi bir tamsayı olabilir. Soruyu doğru cevaplamak için $A$ ve $B$ değerlerini hesaplanmasına gerek yoktur. Doğru cevap **C** şıkkıdır.

**CEVAP**: C

**SORU 34**

```
struct YonelimliCevreleyenKutu {
  double merkezX;
  double merkezY;
  float aci;
  unsigned short genislik;   // desimetre
  unsigned short yukseklik;  // desimetre
};
```

Varsayımlar

- double: 8 byte
- float: 4 byte
- unsigned short: 2 byte
- hizalama: 8 byte
- sistem: 64-bit

Yukarda verilen yapı için aşağıdaki ifadelerden hangisi tamamen doğrudur?

(1 metre = 10 desimetre, padding = dolgu, alignment = hizalama)

A) Toplam boyut 24 byte'tır. Dolgu yoktur. `genislik` en fazla 6553.5 metre tutabilir.
B) Toplam boyut 24 byte'tır. Dolgu yoktur. `genislik` en fazla 65535 metre tutabilir.
C) Toplam boyut 28 byte'tır ve dolgu yoktur. `genislik` en fazla 6553.5 metre tutabilir.
D) Toplam boyut 32 byte'tır. Sonda 4 byte dolgu vardır. `genislik` en fazla 6553.5 metre tutabilir.
E) Toplam boyut 32 byte'tır. Sonda 8 byte dolgu vardır. `genislik` en fazla 6553.5 metre tutabilir.

**ÇÖZÜM**

Bu yapıda alanlar, verilen hizalama kurallarına göre belleğe sırasıyla yerleştirilir. double türündeki merkezX ve merkezY değişkenleri 8'er byte olup 8 byte hizalama gerektirir ve sırasıyla offset 0–7 ve 8–15 aralıklarını doldurur. Ardından gelen float aci 4 byte'tır ve 4 byte hizalamaya uygundur; 16–19 aralığına yerleşir ve herhangi bir dolgu gerektirmez. Sonraki iki unsigned short alan (her biri 2 byte) doğal olarak 2 byte hizalamaya uygun şekilde 20–21 ve 22–23 aralıklarına yerleşir. Bu yerleşimde ne alanlar arasında ne de yapı sonunda ek bir dolgu (padding) oluşur. Toplam boyut 24 byte olur ve bu değer zaten en büyük hizalama birimi olan 8'in katıdır. Ayrıca genislik değişkeni unsigned short olduğundan maksimum 65535 değerini tutabilir; bu değer desimetre cinsinden olduğuna göre 65535 / 10 = 6553.5 metre eder. Dolayısıyla hem bellek yerleşimi hem de birim dönüşümü açısından tamamen doğru olan seçenek **A**'dır.

**CEVAP**: A

## SORU 35

Aşağıdaki fonksiyonun işlevi nedir?

```
int bitUzerindeIslemYap(int x) {
  return x & (x - 1);
}
```

A) Sayının sağdan itibaren ilk 1 değerindeki bitini izole eder (diğer tüm bitleri 0 yapar)
B) Sayının sağdan itibaren ilk 1 değerindeki bitini 0 yapar
C) Sayının tek olup olmadığını kontrol eder
D) Sayının tüm bitlerini ters çevirir
E) Sayıyı iki katına çıkarır

**ÇÖZÜM**

x & (x - 1) ifadesi, sayının ikili (binary) gösteriminde sağdan itibaren ilk 1 değerindeki bitini 0 yapar. Bunun nedeni, x - 1 işleminin sayının en sağdaki 1 değerindeki bitini 0'a çevirip, onun sağındaki tüm bitleri 1 yapmasıdır. Ardından x ile x - 1 arasında yapılan bit düzeyinde AND (&) işlemi, bu en sağdaki 1 bitini sıfırlar ve diğer bitleri aynen korur. Böylece sonuçta yalnızca ilk 1 değerindeki biti (en sağdaki) temizlenmiş, diğer bitler değişmeden kalmış olur. Doğru cevap **B** şıkkıdır.

**CEVAP**: B

**SORU 36**

```
int main() {
    int i, j;
    for (i = 0; i < 4; i++) {
        for (j = i; j < 5; j++) {
            if ((i + j) % 2 == 0)
                printf("*");
        }
    }
    return 0;
}
```

Yukarıdaki program ekrana kaç tane yıldız "*" basar?

A) 5
B) 6
C) 7
D) 8
E) 9

**ÇÖZÜM**

Programda dış döngü i = 0, 1, 2, 3 değerleri için çalışırken iç döngü j = i'den başlayarak 4'e kadar (j < 5) her i için 5 - i kadar iterasyon yapar. Her bir (i, j) çifti için i + j toplamının çift olması durumunda ekrana bir yıldız basılır. i = 0 iken j = 0, 2, 4 için toplam çift olduğundan 3 yıldız; i = 1 iken j = 1 ve 3 için 2 yıldız; i = 2 iken j = 2 ve 4 için 2 yıldız; i = 3 iken sadece j = 3 için toplam çift olduğundan 1 yıldız basılır. Bu değerler toplandığında 3 + 2 + 2 + 1 = 8 yıldız elde edilir.

**CEVAP**: D

**SORU 37**

```
int temp() {
    int a[5] = {1, 3, 5, 7, 9};
    int *p = a;

    int t = 0;
    int i;
    for (i = 0; i < 5; i += ?) {
        t += *(p + i);
    }

    return t;
}
```

Yukarıdaki fonksiyonun 15 değerini dönmesi için for döngüsü içindeki **?** yerine aşağıdakilerden hangisi yazılmalıdır?

A) 0
B) 1
C) 2
D) 3
E) 4

**ÇÖZÜM**

Fonksiyonun 15 değerini döndürmesi için ? yerine 2 yazılmalıdır. Bunun nedeni, döngünün i = 0'dan başlayıp her adımda i += 2 ile ilerlemesi ve i < 5 koşulu nedeniyle i = 0, 2, 4 değerlerini almasıdır. Bu durumda t değişkenine sırasıyla a[0] = 1, a[2] = 5 ve a[4] = 9 eklenir ve toplam 1 + 5 + 9 = 15 elde edilir. Diğer adım değerleri (örneğin 1, 3, 4) ya tüm elemanları toplar (25), ya sadece iki eleman toplar (1+7=8, 1+9=10) ya da tek eleman (1) topladığı için 15 sonucunu vermez. Bu nedenle doğru cevap 2'dir.

**CEVAP**: C

**SORU 38**

```
int f(int n) {
    if (n <= 1) return 1;
    return f(n - 1) + f(n - 2);
}

int main() {
    printf("%d\n", f(5));
    return 0;
}
```

Yukarıdaki kod hangi çıktıyı üretir?

A) 5
B) 8
C) 13
D) 15
E) Hiçbiri

**ÇÖZÜM**

Bu kod, klasik bir Fibonacci benzeri özyinelemeli (recursive) fonksiyonu tanımlamaktadır. Fonksiyon, n 1 veya daha küçük olduğunda 1 döndürmekte, aksi halde f(n-1) ile f(n-2)'yi toplamaktadır. Bu durumda f(0)=1, f(1)=1, f(2)=f(1)+f(0)=1+1=2, f(3)=f(2)+f(1)=2+1=3, f(4)=f(3)+f(2)=3+2=5 ve f(5)=f(4)+f(3)=5+3=8 olarak hesaplanır. Sonuçta main fonksiyonu f(5) çağrısı yaparak ekrana 8 yazdırır.

**CEVAP**: B

**SORU 39**

```
void f(int *x, int n) {
    if (n == 0) return;
    *x += n;
    f(x, n - 1);
    *x -= 1;
}

int main() {
    int x = 0;
    f(&x, 4);
    printf("%d\n", x);
    return 0;
}
```

Yukarıdaki kod hangi çıktıyı üretir?

A) 6
B) 8
C) 12
D) 0
E) Program hata vereceğinden çıktı oluşmaz

**ÇÖZÜM**

Bu kod, özyinelemeli bir fonksiyon olan f aracılığıyla x işaretçisi üzerinden değer güncellemeleri yapmaktadır. f(&x, 4) çağrısı ile başlayan süreçte, n=4 için *x += n ile x önce 4 olur, ardından f(x,3) çağrılır; bu çağrıda x'e 3 eklenir (7 olur), sonra f(x,2) ile 2 eklenir (9 olur), f(x,1) ile 1 eklenir (10 olur) ve f(x,0) base case'e ulaşılıp geri dönülür. Dönüş sırasında her seviyede *x -= 1 çalıştığı için, en içten dışa doğru sırasıyla 10'dan 1 çıkarılır (9), ardından 9'dan 1 çıkarılır (8), ardından 8'den 1 çıkarılır (7), ve en dıştaki f(&x,4) seviyesinde 7'den 1 çıkarılır ve x son değeri 6 olur. Bu nedenle programın çıktısı 6'dır.

**CEVAP**: A

**SORU 40**

```
void foo(int n) {
    if (n <= 0) return;
    foo(n - 1);
    printf("*");
    foo(n - 1);
}
```

Yukarıdaki fonksiyon `foo(3);` şeklinde çağrılırsa ekrana kaç tane yıldız `"*"` yazdırılır?

A) 3
B) 5
C) 7
D) 8
E) 15

**ÇÖZÜM**

Fonksiyon foo(3) şeklinde çağrıldığında, özyineleme ağacı oluşur. foo(3), önce foo(2) çağırır, sonra bir yıldız basar, ardından tekrar foo(2) çağırır. Her bir foo(2) çağrısı ise önce foo(1) çağırır, bir yıldız basar, sonra tekrar foo(1) çağırır. Her bir foo(1) çağrısı da önce foo(0) çağırır (hiçbir şey yapmaz), bir yıldız basar, sonra tekrar foo(0) çağırır. Bu durumda bir foo(1) çağrısı tam olarak 1 yıldız basar. Dolayısıyla bir foo(2) çağrısı, iki foo(1) çağrısı (toplam 2 yıldız) ve aradaki 1 yıldız olmak üzere toplam 3 yıldız basar. foo(3) ise iki foo(2) çağrısı (toplam 6 yıldız) ve aradaki 1 yıldız olmak üzere toplam 7 yıldız basar. Bu nedenle ekrana 7 adet yıldız (*) yazdırılır.

**CEVAP**: C

**SORU 41**

```
int main() {
    char s[] = "OLIMPIYAT";
    int i, j;

    for (i = 0, j = strlen(s) - 1; i < j; i += 2, j -= 3) {
        char t = s[i];
        s[i] = s[j];
        s[j] = t;
    }

    for (i = 0; s[i] != '\0'; i++) {
        if (s[i] == 'I' || s[i] == 'A')
            s[i] = '*';
    }

    printf("%s", s);
    return 0;
}
```

Yukarıdaki kodun çıktısı nedir?

A) `TL*MP*Y*O`
B) `TL*MPY*O*`
C) `TO*MP*Y*L`
D) `TL*MPIYAO`
E) `OL*MP*Y*T`

## ÇÖZÜM

Dizinin başlangıç durumunu temsil eden karakter dizisi şu şekildedir:
`OLIMPIYAT`
İlk döngü bloğu neticesinde indisler üzerinden karakter değişimleri gerçekleştirilir:

- `i = 0, j = 8` için → `O` ile `T` yer değiştirir: `TLIMPIYAO`
- `i = 2, j = 5` için → `I` ile `I` yer değiştirir (içerik değişmez): `TLIMPIYAO`

İkinci döngü safhasında ise dizideki tüm `I` ve `A` karakterleri `*` simgesiyle güncellenir:

`TL*MP*Y*O`

**CEVAP**: A

**SORU 42**

```
void f(int n) {
    printf("*");
    if (n <= 1) return;
    f(n / 2);
    if (n % 2 == 0)
        f(n / 2 - 1);
}

int main() {
    f(10);
    return 0;
}
```

Yukarıdaki kod çalıştığında ekrana kaç adet yıldız "*" karakteri yazdırılır?

A) 8
B) 9
C) 10
D) 11
E) 12

**ÇÖZÜM**

`f(n)` fonksiyonu her çağrıldığında ekrana basılan toplam yıldız sayısını `T(n)` ile ifade edelim.

Fonksiyon gövdesi incelendiğinde, her özyineli çağrının başlangıcında bir adet "*" karakterinin yazdırıldığı görülmektedir.

- `T(0) = 1`
- `T(1) = 1`
- `T(2) = 1 + T(1) + T(0) = 3`
- `T(4) = 1 + T(2) + T(1) = 1 + 3 + 1 = 5`
- `T(5) = 1 + T(2) = 4` (n tek olduğu için)
- `T(10) = 1 + T(5) + T(4) = 1 + 4 + 5 = 10`

**CEVAP**: C

**SORU 43**

```
int main() {
    int A[3][3] = {{1,2,3},{4,5,6},{7,8,9}};
    int i, j;

    for (i = 0; i < 3; i++) {
        for (j = 0; j < 3; j++) {
            if (i < j)
                A[j][i] = A[i][j] - A[j][i];
        }
    }

    for (i = 0; i < 3; i++)
        printf("%d ", A[i][0] + A[i][1] + A[i][2]);

    return 0;
}
```

Yukarıdaki kodun çıktısı nedir?

A) 6 9 3
B) 6 3 9
C) 9 6 3
D) 3 9 6
E) 6 9 6

**ÇÖZÜM**

Dizinin başlangıç durumunu temsil eden matris şu şekilde tanımlanmıştır:

```
1 2 3
4 5 6
7 8 9
```

Algoritma uyarınca sadece `i < j` koşulu sağlandığında matris elemanları üzerinde şu güncellemeler gerçekleştirilir:

- `i = 0, j = 1` için → `A[1][0] = 2 - 4 = -2`
- `i = 0, j = 2` için → `A[2][0] = 3 - 7 = -4`
- `i = 1, j = 2` için → `A[2][1] = 6 - 8 = -2`

Bu işlemler neticesinde matrisin son hali şu şekli alır:

```
 1  2  3
-2  5  6
-4 -2  9
```

Programın son aşamasında satır elemanları toplanarak sonuçlar yazdırılır:

1. 1. satır için → `1 + 2 + 3 = 6`
2. 2. satır için → `-2 + 5 + 6 = 9`
3. 3. satır için → `-4 - 2 + 9 = 3`

**CEVAP**: A

**SORU 44**

```
int f(int n) {
    int s = 0;
    while (n > 0) {
        s += n & 1;
        n >>= 1;
    }
    return s;
}
```

Yukarıdaki `f` fonksiyonunun amacı ne olabilir?

A) `n` sayısının onluk tabandaki basamak sayısını hesaplamak
B) `n` sayısının ikilik sistem gösterimindeki 1 bitlerinin sayısını hesaplamak
C) `n` sayısının en büyük asal bölenini bulmak
D) `n` sayısının tek olup olmadığını belirlemek
E) `n` sayısının 2'nin kuvveti olup olmadığını kontrol etmek

**ÇÖZÜM**

Fonksiyonun işleyiş mantığı şu şekilde özetlenebilir:

- `n & 1` işlemi ile `n` sayısının en düşük anlamlı bitinin (LSB) `1` olup olmadığı denetlenir.
- İlgili bitin `1` olması durumunda `s` değişkeni artırılarak sayaç güncellenir.
- `n >>= 1` komutu vasıtasıyla sayı bir bit sağa kaydırılarak bir sonraki basamağa geçilir.
- Bu döngü neticesinde sayının ikilik tabandaki tüm bitleri sırayla işleme alınmış olur.

Örnek olarak `13` sayısı ikilik sistemde `1101` şeklinde temsil edilir. Bünyesindeki toplam '1' biti sayısı 3 olduğu için fonksiyon `3` değerini dönecektir.

**CEVAP**: B

**SORU 45**

```
int main() {
    int a[] = {2, 4, 6, 8, 10};
    int *p = a + 4;

    while (p > a) {
        *(p - 1) = *p - *(p - 1);
        p--;
    }

    return 0;
}
```

Yukarıdaki kodun çalıştırılması neticesinde aşağıdakilerden hangisi doğru olur?

A) `a[0]` değeri `-8` olur.
B) `a[4]` değeri `2` olur.
C) Dizi elemanlarının toplamı `-10` olur.
D) Dizide negatif eleman kalmaz.
E) `a[2]` değeri `-6` olur.

**ÇÖZÜM**

Dizinin başlangıç değerleri `{2, 4, 6, 8, 10}` olarak atanmıştır.

İşaretçi değişkeni `p = a + 4` ifadesiyle başlangıçta dizinin son elemanını (`a[4]`) gösterecek şekilde konumlandırılmıştır.

Döngü adımları `p > a` koşulu sağlandığı müddetçe şu şekilde çalıştırılır:

- `a[3] = a[4] - a[3] = 10 - 8 = 2`
- `a[2] = a[3] - a[2] = 2 - 6 = -4`
- `a[1] = a[2] - a[1] = -4 - 4 = -8`
- `a[0] = a[1] - a[0] = -8 - 2 = -10`

Bu işlemler neticesinde dizinin nihai içeriği `{-10, -8, -4, 2, 10}` halini alır.

**CEVAP**: C

**SORU 46**

```
int main() {
    int arr[10] = {2, 3, 5, 1, 9, 2, 8, 4, 0, 7};

    int i = 0;
    int t = 0;

    do {
        int g = arr[i];

        t += g;

        arr[i] = (arr[i] + 1) % 10;
        i = g;

    } while (i != 0);

    printf("%d\n", t);
    return 0;
}
```

Yukarıdaki program çalıştırıldığında ekrana hangi çıktıyı yazdırır?

A) 15
B) 17
C) 23
D) 29
E) 41

## ÇÖZÜM

### Adım Adım İzleme Tablosu:

- Başlangıç: `i = 0`, `toplam_yol = 0`
- **1. Tur:**
  - Bulunulan İndeks: `i = 0`
  - Okunan Değer (`gecici`): `arr[0]` yani **2**.
  - `toplam_yol` = 0 + 2 = **2**.
  - Mutasyon: `arr[0] = (2 + 1) % 10 = 3` olur. (Artık `arr[0]` 3'tür).
  - Yeni İndeks: `i = 2`'ye zıplanır.
- **2. Tur:**
  - Bulunulan İndeks: `i = 2`

    - Okunan Değer (`gecici`): `arr[2]` yani **5**.
    - `toplam_yol` = 2 + 5 = **7**.
    - Mutasyon: `arr[2] = (5 + 1) % 10 = 6` olur. *(Dikkat: 2. indeks artık 6'yı gösteriyor!)*
    - Yeni İndeks: `i = 5`'e zıplanır.
- **3. Tur:**
    - Bulunulan İndeks: `i = 5`
    - Okunan Değer (`gecici`): `arr[5]` yani **2**.
    - `toplam_yol` = 7 + 2 = **9**.
    - Mutasyon: `arr[5] = (2 + 1) % 10 = 3` olur.
    - Yeni İndeks: `i = 2`'ye **tekrar geri zıplanır**.
- **4. Tur (Kritik Dönemeç):**
    - Bulunulan İndeks: `i = 2`
    - Okunan Değer (`gecici`): `arr[2]` 2. Turda değişmişti! Yeni değer **6**'dır.
    - `toplam_yol` = 9 + 6 = **15**.
    - Mutasyon: `arr[2] = (6 + 1) % 10 = 7` olur.
    - Yeni İndeks: `i = 6`'ya zıplanır.
- **5. Tur:**
    - Bulunulan İndeks: `i = 6`
    - Okunan Değer (`gecici`): `arr[6]` yani **8**.
    - `toplam_yol` = 15 + 8 = **23**.
    - Mutasyon: `arr[6] = (8 + 1) % 10 = 9` olur.
    - Yeni İndeks: `i = 8`'e zıplanır.
- **6. Tur (Sonlanma):**
    - Bulunulan İndeks: `i = 8`
    - Okunan Değer (`gecici`): `arr[8]` yani **0**.
    - `toplam_yol` = 23 + 0 = **23**.
    - Mutasyon: `arr[8] = (0 + 1) % 10 = 1` olur.
    - Yeni İndeks: `i = 0` olur.

**Döngü Kontrolü:** `while (i != 0)` koşuluna bakılır. `i` şu an 0 olduğu için koşul sağlanmaz ve döngü kırılır. Nihai Çıktı: **23**'tür.

**CEVAP**: C

**SORU 47**

```
void islem(int *arr, int n) {
    if (n <= 1) {
        return;
    }
    islem(arr + 1, n - 1);

    arr[0] = arr[0] + arr[1];
}

int main() {
    int dizi[] = {3, 6, 5, 2, 4};
    islem(dizi, 5);

    for(int i = 0; i < 5; i++) {
        printf("%d ", dizi[i]);
    }
    return 0;
}
```

Yukarıdaki program çalıştırıldığında, main fonksiyonundaki döngü ekrana hangi çıktıyı yazdırır?

A) 20 17 11 6 4
B) 9 11 7 6 4
C) 3 9 14 16 20
D) 16 13 7 2 4
E) 3 6 5 2 4

**ÇÖZÜM**

Kodun çalışma mantığını tam olarak anlayabilmek için süreci iki ana aşamaya ayırmalıyız: **Özyinelemeli İniş (Recursive Descent)** ve **Yığının Çözülmesi (Stack Unwinding)**.
Orijinal dizimiz: `{3, 6, 5, 2, 4}`

***Aşama 1: Özyinelemeli İniş (Çağrı Yığınının Oluşturulması)***

Her `islem(arr + 1, n - 1)` çağrısında, bellek adresi bir tam sayı boyutu kadar ileri kaydırılır. Böylece fonksiyonun her yeni kopyası, dizinin bir sonraki elemanını kendi `arr[0]` (başlangıç) noktası olarak kabul eder. Toplama işlemi `arr[0] = arr[0] + arr[1];` satırı özyinelemeli çağrının *altında* olduğu için beklemeye alınır.

- **1. Çağrı (`islem(dizi, 5)`):** `arr`, `dizi[0]` noktasını gösterir (Değeri: 3). `n=5`.
- **2. Çağrı (`islem(dizi+1, 4)`):** `arr`, `dizi[1]` noktasını gösterir (Değeri: 6). `n=4`.
- **3. Çağrı (`islem(dizi+2, 3)`):** `arr`, `dizi[2]` noktasını gösterir (Değeri: 5). `n=3`.

- **4. Çağrı (`islem(dizi+3, 2)`):** `arr`, `dizi[3]` noktasını gösterir (Değeri: 2). `n=2`.
- **5. Çağrı (`islem(dizi+4, 1)`):** `arr`, `dizi[4]` noktasını gösterir (Değeri: 4). `n=1`.
    - *Temel Durum (Base Case):* `n <= 1` koşulu sağlanır. Fonksiyon bellekte hiçbir değişiklik yapmadan anında `return` işlemi ile geri döner.

***Aşama 2: Yığının Çözülmesi (Bekleyen İşlemlerin Yürütülmesi)***

Yığın LIFO (Son Giren İlk Çıkar) prensibiyle geriye doğru çözülmeye başlar. Şimdi beklemeye alınan `arr[0] = arr[0] + arr[1];` işlemleri sırasıyla çalışacaktır. Her adımda dizinin bir önceki hücreye doğru kalıcı olarak değiştiğine dikkat ediniz.

- **4. Çağrıya Dönüş (`n=2` seviyesi):**
    - Burada `arr[0]`, orijinal dizideki `dizi[3]`'e (Değeri: 2) karşılık gelir.
    - `arr[1]`, orijinal dizideki `dizi[4]`'e (Değeri: 4) karşılık gelir.
    - **İşlem:** `2 + 4 = 6`. Bu değer `dizi[3]` konumuna yazılır.
    - *Dizinin anlık durumu:* `{3, 6, 5, 6, 4}`
- **3. Çağrıya Dönüş (`n=3` seviyesi):**
    - Burada `arr[0]`, orijinal dizideki `dizi[2]`'ye (Değeri: 5) karşılık gelir.
    - `arr[1]`, bir alt adımda güncellenen `dizi[3]`'tür (Yeni değeri: 6).
    - **İşlem:** `5 + 6 = 11`. Bu değer `dizi[2]` konumuna yazılır.
    - *Dizinin anlık durumu:* `{3, 6, 11, 6, 4}`
- **2. Çağrıya Dönüş (`n=4` seviyesi):**
    - Burada `arr[0]`, orijinal dizideki `dizi[1]`'e (Değeri: 6) karşılık gelir.
    - `arr[1]`, güncellenen `dizi[2]`'dir (Yeni değeri: 11).
    - **İşlem:** `6 + 11 = 17`. Bu değer `dizi[1]` konumuna yazılır.
    - *Dizinin anlık durumu:* `{3, 17, 11, 6, 4}`
- **1. Çağrıya Dönüş (`n=5` seviyesi - Ana fonksiyon seviyesi):**
    - Burada `arr[0]`, orijinal dizideki `dizi[0]`'dır (Değeri: 3).
    - `arr[1]`, güncellenen `dizi[1]`'dir (Yeni değeri: 17).
    - **İşlem:** `3 + 17 = 20`. Bu değer `dizi[0]` konumuna yazılır.
    - *Dizinin nihai durumu:* `{20, 17, 11, 6, 4}`

**Sonuç:** Tüm çağrılar bellekten silindiğinde, `main` fonksiyonundaki yazdırma döngüsü dizinin son hali olan **20 17 11 6 4** sayılarını ekrana basar.

**CEVAP**: A

**SORU 48**

```
int f1(int x, int y) {
    if (x == 0 || y == 0) {
        return 1;
    }
    return f1(x - 1, y) + f1(x, y - 1) + f1(x - 1, y - 1);
}

int main() {
    printf("%d\n", f1(2, 3));
    return 0;
}
```

Yukarıdaki program çalıştırıldığında, ekrana hangi çıktıyı yazdırır?

A) 10
B) 13
C) 20
D) 25
E) 31

**ÇÖZÜM**

Temel durumumuz aynıdır: `x` veya `y` 0 olduğunda fonksiyon `1` döndürür.

**Adım Adım İzleme (Tablo Yöntemiyle):** Çözüme ulaşmak için önceki değerlerin üzerine inşa etmeliyiz.

1. **`x=1` Sütununun Hesaplanması:**
   - `f1(1, 1)` = `f1(0, 1)` + `f1(1, 0)` + `f1(0, 0)` = 1 + 1 + 1 = **3**
   - `f1(1, 2)` = `f1(0, 2)` + `f1(1, 1)` + `f1(0, 1)` = 1 + 3 + 1 = **5**
   - `f1(1, 3)` = `f1(0, 3)` + `f1(1, 2)` + `f1(0, 2)` = 1 + 5 + 1 = **7**
2. **`x=2` Sütununun Hesaplanması:**
   - `f1(2, 1)` = `f1(1, 1)` + `f1(2, 0)` + `f1(1, 0)` = 3 + 1 + 1 = **5**
   - `f1(2, 2)` = `f1(1, 2)` + `f1(2, 1)` + `f1(1, 1)` = 5 + 5 + 3 = **13**
3. **Ana Çağrının Hesaplanması `f1(2, 3)`:** Artık ihtiyacımız olan tüm alt dallara sahibiz.
   - `f1(2, 3)` = `f1(1, 3)` + `f1(2, 2)` + `f1(1, 2)`
   - `f1(2, 3)` = 7 + 13 + 5 = **25**

**CEVAP**: D

**SORU 49**

```
int f(int n, int c, int a, int b) {
    if (a < b) {
        return c;
    }

    if ((c & 1) == 0) {
        int x = (n & b) ? 1 : 0;

        return f(n, (c << 1) | x, a, b << 1);
    } else {
        int x = (n & a) ? 1 : 0;

        return f(n, (c << 1) | x, a >> 1, b);
    }
}
```

Yukarıdaki program çalıştırıldığında, ekrana hangi çıktıyı yazdırır?

A) 14
B) 19
C) 22
D) 25
E) 28

## ÇÖZÜM

Sayı ve maskelerin ikilik (binary) düzlemde ele alınması gerekir.

- **Sayı (n):** 25 **11001** (5 bitlik bir sayı)
- **Sol Maske (a):** 16 **10000** (En soldaki biti okur)
- **Sağ Maske (b):** 1 **00001** (En sağdaki biti okur)
- **Biriktirici (c):** 0

**Adım Adım İzleme:**

**1. Çağrı (`c = 0`, Çift):**

- `c` çift olduğu için sağ maskeden (`b=1`) okuma yapılır.
- 25'in (**11001**) en sağındaki bit **1**'dir. `x = 1`.
- `c`, 1 sola kaydırılır ve x eklenir: `c = (0 << 1) | 1 = 1`.
- Sağ maske (`b`) sola kayar: `b` artık 2 (00010) olur.
- *Yeni Çağrı:* `f(25, 1, 16, 2)`

**2. Çağrı (`c = 1`, Tek):**

- `c` tek olduğu için sol maskeden (`a=16`) okuma yapılır.
- 25'in **11001** en soldaki (5.) biti **1**'dir. `x = 1`.
- `c`, 1 sola kaydırılır ve x eklenir: `c = (1 << 1) | 1 = 3`. (İkilik: 11)
- Sol maske (`a`) sağa kayar: `a` artık 8 (01000) olur.
- *Yeni Çağrı:* `f(25, 3, 8, 2)`

**3. Çağrı (`c = 3`, Tek):**

- `c` tek olduğu için sol maskeden (`a=8`) okuma yapılır.
- 25'in **11001** soldan ikinci (4.) biti **1**'dir. `x = 1`.
- `c = (3 << 1) | 1 = 7`. (İkilik: 111)
- Sol maske (`a`) sağa kayar: `a` artık 4 (00100) olur.
- *Yeni Çağrı:* `f(25, 7, 4, 2)`

**4. Çağrı (`c = 7`, Tek):**

- `c` tek olduğu için sol maskeden (`a=4`) okuma yapılır.
- 25'in **11001** ortadaki (3.) biti **0**'dır. `x = 0`.
- `c = (7 << 1) | 0 = 14`. (İkilik: 1110)
- Sol maske (`a`) sağa kayar: `a` artık 2 (00010) olur.
- *Yeni Çağrı:* `f(25, 14, 2, 2)`

**5. Çağrı (`c = 14`, Çift):**

- Burada maskeler çakıştı (`a=2, b=2`). Temel duruma henüz ulaşılmadı çünkü `2 < 2` koşulu sağlanmaz! Bu çok kritik bir adımdır.
- `c` çift olduğu için sağ maskeden (`b=2`) okuma yapılır.
- 25'in **11001** sağdan ikinci (2.) biti **0**'dır. `x = 0`.
- `c = (14 << 1) | 0 = 28`. (İkilik: 11100)
- Sağ maske (`b`) sola kayar: `b` artık 4 (00100) olur.
- *Yeni Çağrı:* `f(25, 28, 2, 4)`

**6. Çağrı (Sonlanma):**

- `a < b` koşulu `(2 < 4)` test edilir. Koşul **Doğru**'dur.
- Program özyinelemeyi durdurur ve o anki `c` değerini döndürür.
- **Sonuç: 28**

**CEVAP**: E

**SORU 50**

```
void degistir(int x, int y){
    x = x op1 y;
    y = x op2 y;
    x = x op3 y;
    printf("%d %d\n", x, y);
}
```

Yukarıdaki fonksiyon, harici bir geçici değişken kullanmadan x ve y parametrelerinin değerlerini kendi aralarında değiştirmeyi (swap) amaçlamaktadır. Gönderilecek x ve y değerlerinin sadece rakamlar olduğu bilinmektedir. Buna göre, fonksiyonun bu değiştirme işlemini her koşulda doğru bir şekilde gerçekleştirmesi için koddaki op1, op2 ve op3 yerlerine sırasıyla aşağıdaki operatör gruplarından hangisi getirilmelidir?

I. +, -, -
II. ^, ^, ^
III. *, /, *
IV. -, +, -

A) I ve II
B) I ve III
C) II ve III
D) I, II ve III
E) I, II ve IV

## ÇÖZÜM

Soruda bizden istenen, fonksiyonun sonundaki `printf` komutunun ekrana sırasıyla ilk `y` ve ilk `x` değerlerini basmasını sağlayacak operatörleri bulmaktır.

**I. Öncül: `+, -, -` (Doğru)** Bu yöntem, toplam üzerinden geriye doğru çıkarma yaparak değerleri ayrıştırma prensibine dayanır.

- **Adım 1:** `x = x + y;` (Artık `x` değişkeni, her iki sayının toplamını tutan bir taşıyıcıdır.)
- **Adım 2:** `y = x - y;` (Toplamdan `y` çıkarsa geriye ilk `x` kalır. Artık `y`'nin içinde ilk `x` değeri güvenle saklanmaktadır.)
- **Adım 3:** `x = x - y;` (Toplamdan yeni `y` yani ilk `x` çıkarıldığında, geriye ilk `y` değeri kalır. Bu da `x`'in içine yazılır.)

**Profesörün Notu:** Soru metnindeki "sadece rakamlar" ifadesi bu adım için çok kritiktir. Eğer sayılar bilgisayarın `int` kapasitesini aşacak kadar büyük olsaydı, `x + y` adımında **Tamsayı**

**Taşması (Integer Overflow)** yaşanabilirdi. Ancak rakamların toplamı en fazla 18 olabileceği için bu algoritma C dilinde kusursuz çalışır.

**II. Öncül: `^, ^, ^` (Doğru)** Bu yöntem, bilgisayar bilimleri literatüründe **XOR Swap** (Dışlayıcı VEYA Takası) olarak bilinen, taşma riski barındırmayan endüstri standardı bir tekniktir. XOR operatörü, aynı bitleri 0, farklı bitleri 1 yapar.

- **Adım 1:** `x = x ^ y;` (Artık `x`, iki sayının bit düzeyindeki fark maskesini tutar.)
- **Adım 2:** `y = x ^ y;` (Fark maskesi ile `y` tekrar XOR işlemine girdiğinde, matematiksel olarak `y`'ler birbirini sıfırlar ve geriye ilk `x` kalır.)
- **Adım 3:** `x = x ^ y;` (Fark maskesi ile yeni `y` yani ilk `x` XOR işlemine girdiğinde, bu kez `x`'ler birbirini sıfırlar ve geriye ilk `y` kalır.)

**III. Öncül: `*, /, *` (Yanlış)** Bu öncül, dikkatsiz öğrencileri avlamak için tasarlanmış çift kusurlu bir çeldiricidir.

- **Adım 1:** `x = x * y;` (x, çarpımı tutar.)
- **Adım 2:** `y = x / y;` (Çarpım `y`'ye bölünürse geriye ilk `x` kalır. Buraya kadar sorun yok gibi görünür.)
- **Adım 3:** `x = x * y;` (Hata burada başlar. Son işlemin de bölme `/` olması gerekirken çarpma `*` olarak verilmiştir. `x = (x * y) * x` şeklinde bir işlem gerçekleşir ve takas bozulur.)

**Kritik Hata (Sıfıra Bölme):** Rakamlar kümesi 0'ı da kapsar. Eğer parametre olarak `y = 0` gelirse, 2. adımdaki `x / 0` işlemi programın anında çökmesine (**Division by Zero** - Çalışma Zamanı Hatası) sebep olur.

**IV. Öncül: `-, +, -` (Yanlış)** Bu öncül, ilk öncülün simetrisi gibi görünse de işaret hatasına yol açar.

- **Adım 1:** `x = x - y;` (x, iki sayının farkını tutar.)
- **Adım 2:** `y = x + y;` (Fark ile `y` toplanırsa: `x - y + y = x`. Artık `y`'nin içinde ilk `x` vardır.)
- **Adım 3:** `x = x - y;` (Farktan yeni `y` yani ilk `x` çıkarılırsa: `x - y - x = -y` sonucu elde edilir.) Bu işlemin sonunda `x` değişkeni, `y`'nin kendisini değil, **negatif (eksi) işaretli halini** alır. Takas işlemi hatalı sonuçlanır.

**Nihai Sonuç:** Tüm algoritmik olasılıklar değerlendirildiğinde, sadece I ve II numaralı öncüllerin takas işlemini her koşulda başarıyla tamamladığı kanıtlanmıştır.

**CEVAP**: A