Ad-Soyadı: **No:**

Sivas Cumhuriyet Üniversitesi Mühendislik Fakültesi Bilgisayar Mühendisliği Bölümü
2024-2025 Eğitim Öğretim Yılı Bahar Algoritma Analizi ve Tasarımı Course AAD321 Vize Sınavı 2

1- Verilen algoritma:

```
ALGORITHM Mystery(n)
S ← 0
for i ← 1 to n do
       S ← S + i * i * i
return S
```

a. Bu algoritma neyi hesaplıyor?

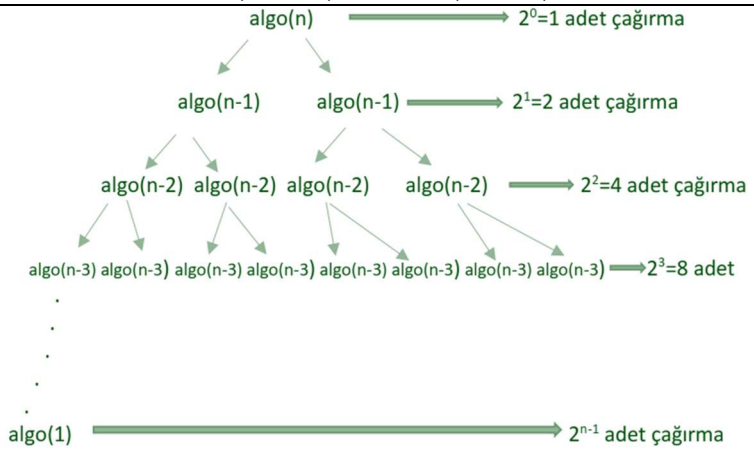
$$S = \sum_{i=1}^{n} i^3 = 1^3 + 2^3 + 3^3 + \ldots + n^3$$

b. Temel işlemi nedir?
S ← S + i * i * i

c. Temel işlem kaç kez yürütülür?
n kez

2- Aldığı bir negatif olmayan tam sayı n Fibonacci dizisinin n'inci terimini hesaplamak için dönen olup olmadığını kontrol eden algoritmanın **C*worst*(*n*)** yani en kötü durum zaman karmaşıklığı nedir?

```
ALGORİTMA fibo(n)
//Input: A non-negative integer n
//Output: The n-th Fibonacci number
if n ≤ 1 then
    return 1
else
    return fibo(n - 1) + fibo(n - 2)
```



Toplam çağırmalar

$2^0+2^1+.....+2^{n-1}=\frac{2^n-1}{2-1}=2^n-1$

Geometrik seri toplamı $= a+ar+ar^2+.....+ar^{n-1}$

$= a(1+r+r^2+.....+r^{n-1})$

$= \frac{ar^n-a}{r-1}$

Toplam 2^n-1 adet fonksiyon yani algoritma çağırılır o halde $T(n)=2^n-1 \in O(2^n)$'dir.

Dikkat edersek algoritmada yaptığımız ufak bir değişiklik ile O değeri n'den 2^n'e çıktı !

En bilinen rekürsif algoritmalardan biri olan Fibonacci'ninde zaman verimliliği $O(2^n)$'dir.

3- Aşağıdaki problemlere belirtilen yöntemleri kullanarak algoritmalar tasarlayın ve sözde kodlarını yazın.

i) **Brute Force** yöntemi kullanarak bir diziyi artan sırada sıralayan Bubble Sort algoritmasını yazın.

```
bubbleSort_V1(A[0, ..., n-1])

// Girdi: n uzunluğunda bir A dizisi
// Çıktı: Elemanları küçükten büyüğe sıralanmış A dizisi

while true:
    sıralı_ardışık ← 0
    // Bu, önce küçük sonra büyük olan sıralı ikilileri sayacak

    for i = 0'dan n-2'ye:
        if A[i+1] < A[i]:  // Sıralanmamış ikili tespiti
            temp ← A[i+1]
            A[i+1] ← A[i]
            A[i] ← temp
        else:
            sıralı_ardışık ← sıralı_ardışık + 1

    if sıralı_ardışık == n-1:  // Sıralanmış ise tüm ardışık ikililer
        break  // while'dan çıkmak için

return A
```

```
bubbleSort_V2(A[0, ..., n-1])

// Orjinal bubble sort
// Girdi: n uzunluğunda bir A dizisi
// Çıktı: Elemanları küçükten büyüğe sıralanmış A dizisi

for i = 0'dan n-2'ye:
    for j = 0'dan n-2-i'ye:
        if A[j+1] < A[j]:
            temp ← A[j+1]
            A[j+1] ← A[j]
            A[j] ← temp
```

ii) **Recursive** Bir dizideki elemanların sırasını bir şekilde tersine çevirmek için bir algoritma yazın.

```
ALGORITHM ReverseArrayRecursive(arr, start, end)
//Input: arr: An array of numbers to be reversed.
//start: The starting index of the array (initially 0 for full reversal).
//end: The ending index of the array (initially length(arr) - 1 for full
reversal).
//Output:The array arr with elements reversed in place.
    if start ≥ end then
        return
    swap(arr[start], arr[end])
    ReverseArrayRecursive(arr, start + 1, end - 1)
```

Örneğin:

Girdi: ReverseArrayRecursive([1, 2, 3, 4, 5], 0, 4)

Adımlar:

0 ve 4. indislerdeki elemanları değiştir: [5, 2, 3, 4, 1]

1 ve 3. indislerdeki elemanları değiştir: [5, 4, 3, 2, 1]

2 ve 2. indis aynı olduğu için değişim yapılmaz (temel durum sağlandı).

Çıktı: [5, 4, 3, 2, 1]

4- t(n) ϵ O(g(n)) olsun. Bu durumda her k reel sayısı için k.t(n) ϵ O(g(n)) olur. Yani bir fonksiyonu sabit bir değerle çarpmak onun büyüme oranını değiştirmez. Daha önce en fazla g(n) kadar büyüyen bir fonksiyon, bir katsayı ile çarpılırsa yine en fazla g(n) kadar büyür.

Soru: bu teorim ispatlayınız.

Kanıt:

t(n) ϵ O(g(n)) ise bir n_0>0 ve c>0 vardır. Öyle ki ∀ n≥n_0 için t(n) ≤c.g(n)

- K negatif ise k.t(n) ≤t(n) ≤c.g(n) olacağından k.t(n) ϵ O(g(n)) olur.
- K pozitif olsun. Bu durumda yukarıdaki eşitsizlik k ile çarpıldığında yönü değişmez. k.t(n) ≤k.c.g(n) olup k.c yeni c değerimiz olur. Buna c diyelim. O halde k.t(n) ≤c.g(n) olur.
- Bu yüzden k.t(n) ϵ O(g(n)) olur.

5- Brute Force Algoritması Nedir?

– Bir problemi çözmek için en basit yaklaşım

– Genellikle problemin tanımına ve konseptine bağımlıdır

– Genellikle uygulaması en basit çözümdür

Veya

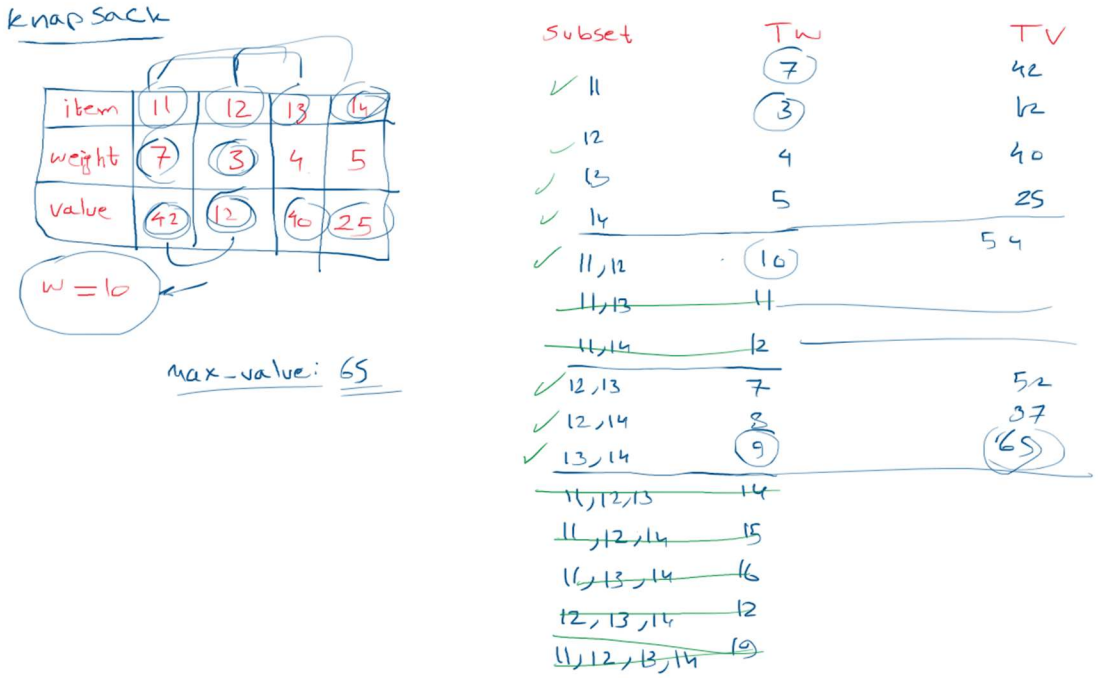
Bir problem için çözüm bulmak amacıyla tüm olası çözümleri denemek gerekir.

6- Bir sırt çantası probleminde, maksimum taşıma kapasitesi W=10 kg'dır.

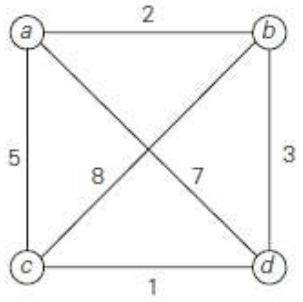
Aşağıdaki tabloda verilen her bir öğenin ağırlığı ve değeri bulunmaktadır.

Item (Öğeler)	11	12	13	14
Wight (Ağırlıklar)	7	3	4	5
Value (Değerler)	42	12	40	25

Hangi öğeleri seçerek sırt çantasını en fazla toplam değerle doldurabilirsiniz? Toplam değer maksimum olacak şekilde bir seçim yapınız.



7- Bir satıcı, aşağıda verilen ağırlıklı grafa göre dört farklı şehri dolaşıp başlangıç noktasına geri dönmek istemektedir. Satıcının toplam yol uzunluğunu en aza indirmesi gerekmektedir. En kısa yolu (optimal turu) belirleyiniz.



Tour	Length	
$a \rightarrow b \rightarrow c \rightarrow d \rightarrow a$	$l = 2 + 8 + 1 + 7 = 18$	
$a \rightarrow b \rightarrow d \rightarrow c \rightarrow a$	$l = 2 + 3 + 1 + 5 = 11$	optimal
$a \rightarrow c \rightarrow b \rightarrow d \rightarrow a$	$l = 5 + 8 + 3 + 7 = 23$	
$a \rightarrow c \rightarrow d \rightarrow b \rightarrow a$	$l = 5 + 1 + 3 + 2 = 11$	optimal
$a \rightarrow d \rightarrow b \rightarrow c \rightarrow a$	$l = 7 + 3 + 8 + 5 = 23$	
$a \rightarrow d \rightarrow c \rightarrow b \rightarrow a$	$l = 7 + 1 + 8 + 2 = 18$	

Yukarıda **bu sayfa secdece** listelenen sorulardan istediğiniz 3 tanesini cevaplayınız.

Süre 55dk Başarılar dilerim...

Eng: Abdulrahman Hamdi