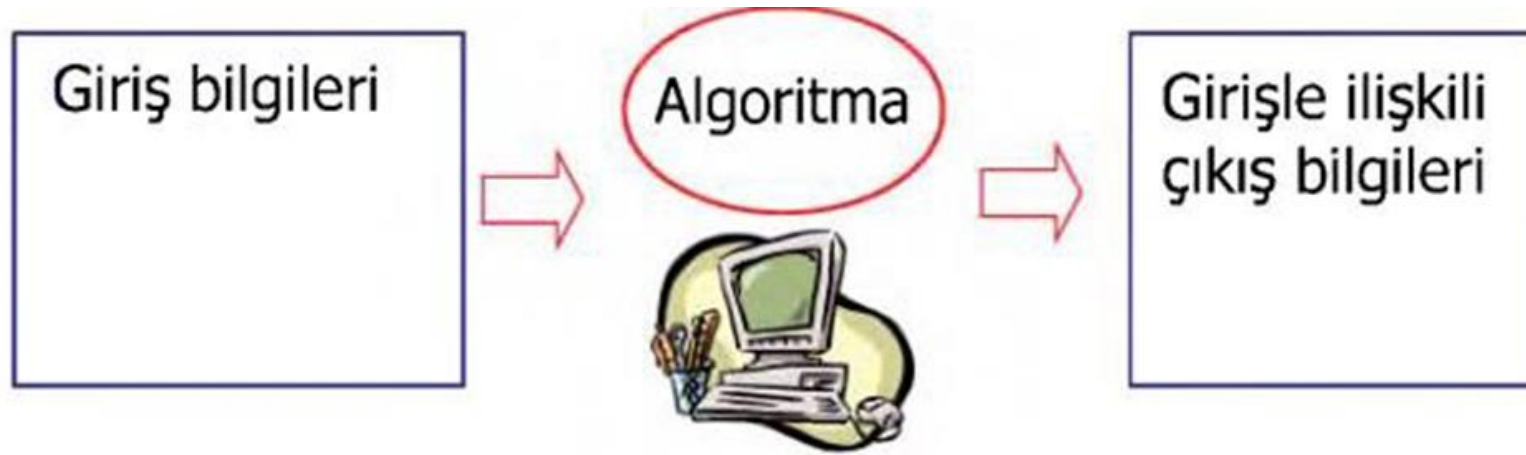




# Algoritma Analizi ve Tasarımı

**TEMEL KAVRAMLAR**

# Algoritmik çözüm



- **Algoritma**, giriş örnekleri üzerindeki işlemleri tanımlar ve uygun çıkış üretir.
- Aynı algoritmik problem için farklı algoritmalar olabilir.
- Fakat en hızlısı ve en az hafıza kullanan algoritma **tercih edilir**.

# Algoritmaların Özellikleri

Bir algoritmanın taşınması gereken ***beş tane temel*** özelliği vardır.

1. Giriş (Input)

2. Belirlilik (Definiteness)

3. Çıkış (Output)

4. Etkililik (Efficiency)

5. Sınırlılık (Boundedness)

# 1. Giriş (Input)

Bir algoritmanın **sıfır** veya **daha fazla giriş değişkeni** vardır.

Giriş değişkenleri algoritma işlemeye başlamadan önce, algoritmaya verilen değerler kümesidir.

Değer kaydetmesi için verilen hafıza bölgesidir.

## 2. Belirlilik (Definiteness)

BİR ALGORİTMANIN  
HER ADIMI İÇİN KESİN  
OLARAK **NE İŞ**  
**YAPACAĞI**  
**BELİRLENMELİDİR.**

**BELİRSİZLİK**  
**OLMAMALIDIR.**

HER DURUM İÇİN  
**HANGİ İŞLEM**  
**GERÇEKLEŞTİRİLECEKSE,**  
O AÇIK OLARAK  
TANIMLANMALIDIR.

### 3. Çıkış (Output)

Her algoritmanın  
**bir veya daha  
fazla çıkış değeri  
vardır.**

Çıkış değerleri ile  
giriş değerleri  
arasında  
**bağıntılar vardır.**

## 4.Etkililik (Efficiency)

Her algoritmanın  
**bir veya daha  
fazla çıkış** değeri  
vardır.

**Çıkış değerleri ile  
giriş değerleri**  
arasında  
bağıntılar vardır.

Olabildiğince hızlı  
çalışmalıdır.

Olabildiğince **az**  
**hafıza**  
**kullanmalıdır.**

## 5. Sınırlılık (Boundedness)

Her algoritma  
*sınırlı sayıda*  
*çalışma* adımı  
sonunda bitmelidir.

Aynı işlemi yapan  
iki algoritmadan en  
az adımda işlemi  
bitiren tercih edilir.



# Algoritmaların Özellikleri

1

Diğer bazı kriterler ise **algoritmanın bilgisayar ortamına aktarılabilme özelliği, basitliği**, vb. gibi özelliklerdir.

2

***İyi algoritmayı belirlemek*** için uygulanan testler veya yapılan işlemler **Algoritma Analizi'** nin konusudur.

# Algoritmaların Yönleri

---

1. Algoritmaları Tasarlama
2. Algoritma ifade edilmesi ve uygulanması
3. Algoritma Analizi (Çözümlemesi)
4. Çözümünüzün yeterince iyi olup olmadığına bakılması
5. Algoritma veya programı doğrulama
6. Algoritmaların test edilmesi

# 1- Algoritmaları Tasarlama

---

- Bulmacaların (puzzle) parçalarını birleştirilmesi gibi yaklaşım güdülebilir.
- Veri yapılarının seçilmesi
- Problemin çözümü için temel yaklaşımlar seçilmesi

En popüler tasarım stratejileri belirlenir:

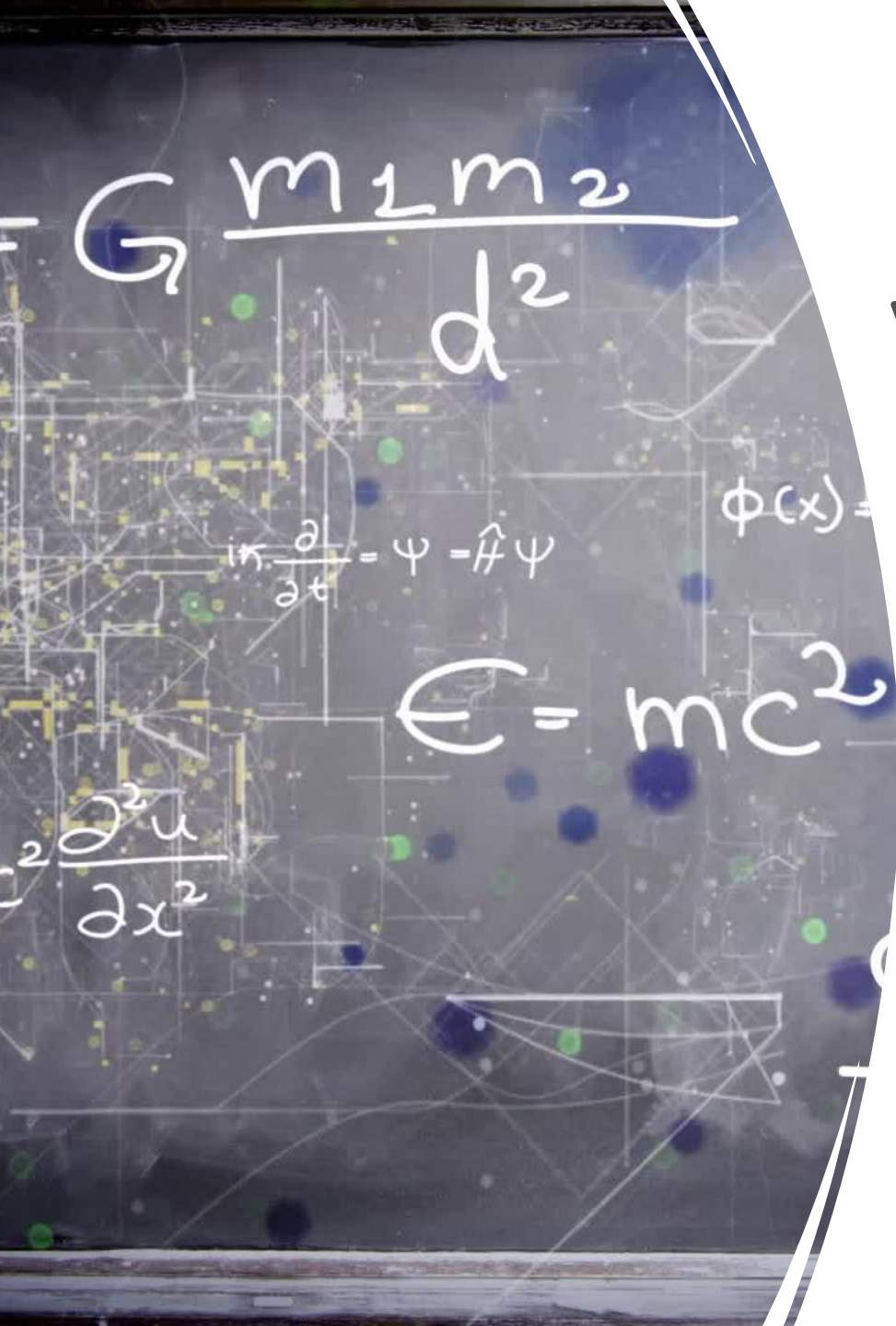
*Böl ve fethet*

Açgözlü

dinamik programlama

özyineleme





## 2- Algoritma ifade edilmesi ve uygulanması

---

Algoritma genellikle aşağıdaki 3 şekilde ifade edilir.

1. Sözel olarak
2. Kaba/Sözde Kod (Pseudo Code)
3. Grafiksel olarak

1. Algoritmanın  
sözel olarak  
ifade edilmesi

Adım 1-Başla

Adım 2-Birinci sayıyı oku ve x ' e kaydet.

Adım 3-İkinci sayıyı oku ve y ' ye kaydet.

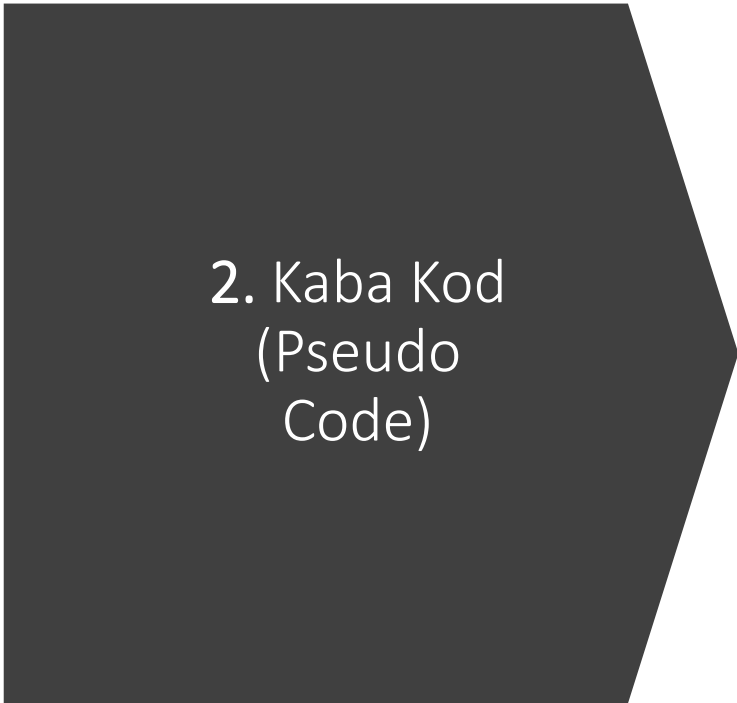
Adım 4-Üçüncü sayıyı oku ve z' ye kaydet.

Adım 5- $top = x + y + z$  işlemini yap.

Adım 6- $ort = top / 3$  işlemini yap.

Adım 7-top ve ort değerlerini ekrana yazdır.

Adım 8-Bitir



## 2. Kaba Kod (Pseudo Code)

Pascal, C/C++, Java, Python veya diğer diller:

- **Kontrol yapıları**  
*if then else, while ve for döngüleri*
- **Atama işlemi :**  $\leftarrow$
- **A dizisinde erişilen eleman:**  $A[i]$
- **Composite tip (record veya object) elemanı:**  
 $A.b$  (veya  $b[A]$ )

## Intersection







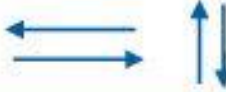


*Input:* Two finite sets  $A, B$

*Output:* A finite set  $C$  such that  $C = A \cap B$

1.  $C \leftarrow \emptyset$
2. **If**  $|A| > |B|$
3.     **Then**  $\text{Swap}(A, B)$
4. **End**
5. **For every**  $x \in A$  **Do**
6.     **If**  $x \in B$
7.         **Then**  $C \leftarrow C \cup \{x\}$
8.     **End**
9. **End**
10. **Return**  $C$

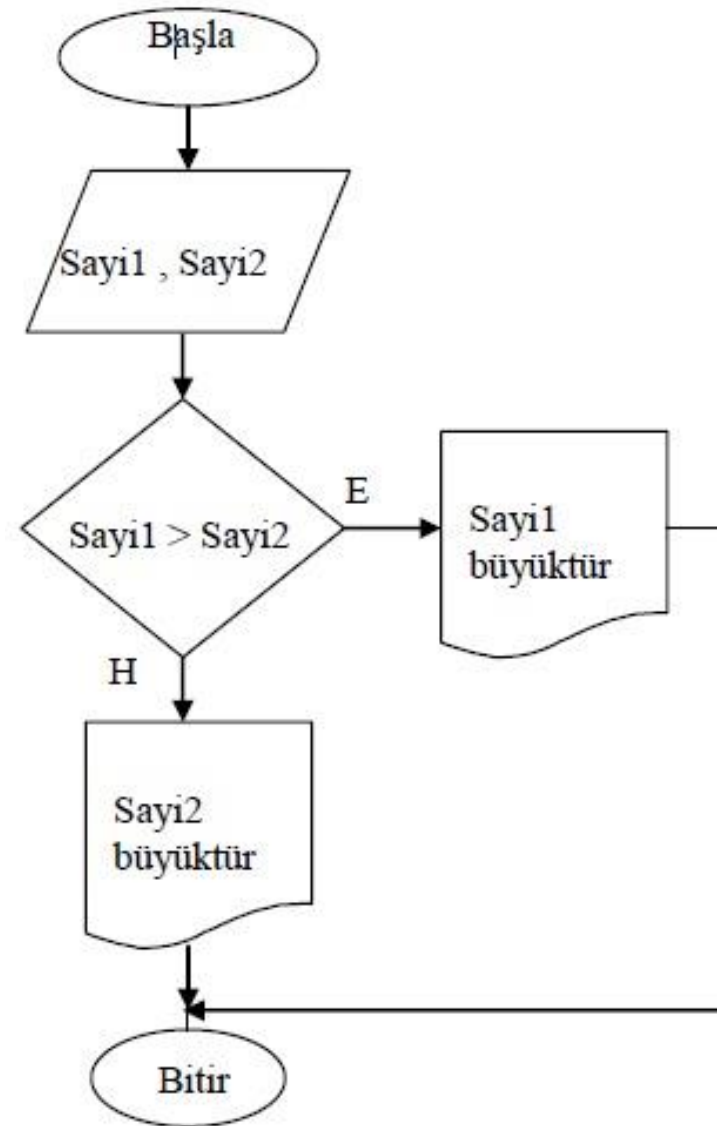
Örnek

### 3. Grafiksel Gösterim (Akış Şeması)

Simge	İşlev
	Başla/Bitir
	Giriş
	Atama/İşlem
	Denetim (Karar)
	Çıkış
	Döngü
	Akış Yönü
	Bağlaç
	Önceden Tanımlı İşlem/Fonksiyon



# Örnek



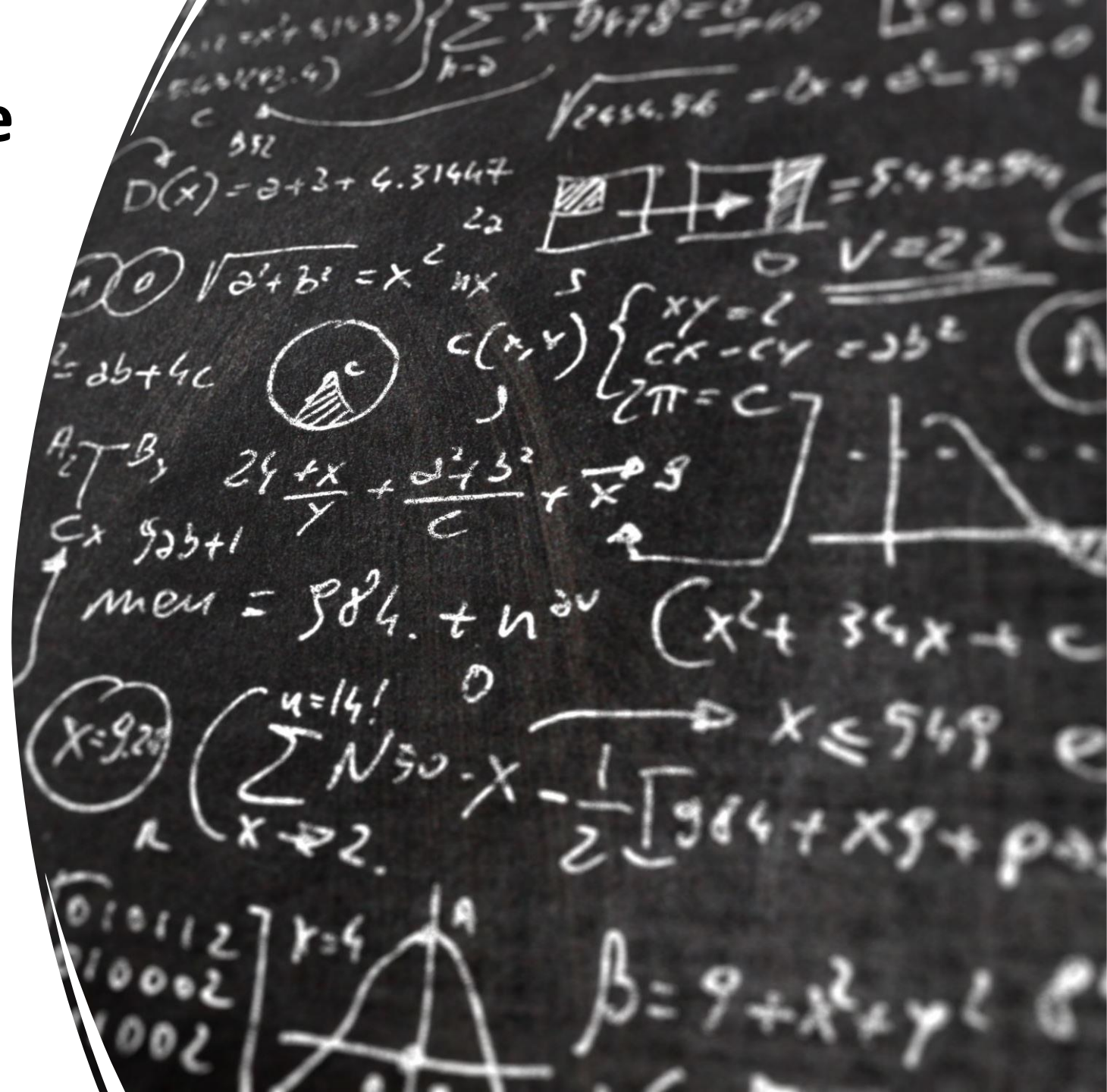
# 3-Algoritma Analizi (Çözümlemesi)

---

Algoritmayı *gerçekte uygulamadan*, **bir algoritmayı çalıştırabilmek için gereken kaynakların** (zaman, alan gibi) **araştırılması demektir.**

## 4. Çözümünüzün yeterince iyi olup olmadığına bakılması

- ❑ Problemi çözmek için alt ve üst sınırlar verilir.
- ❑ Buna göre algoritma analiz edilir.



## 5. Algoritma veya programı doğrulama

---

Algoritmanın verilen *tüm olası girişler* için *hesaplama yaptığını* ve *doğru çıkış ürettiğini* göstermektir.

# 6- Algoritmaların test edilmesi

---

## A. Hata ayıklama (Debugging):

- ❖ Programın örnek veriler üzerinde çalıştırılması sırasında oluşan hataları tespit etme ve onları düzeltme işlemi.

## B. Profil oluşturma (Profiling):

- ❖ Çeşitli veriler üzerinde programın çalıştırılması ve sonuçların hesaplaması için gerekli zamanın (ve alan) ölçülmesi işlemi.

# Algoritma Tasarımı

**Algoritmaları tasarlamada kullanılacak bazı yöntemler :**

---

Özyineleme

---

Böl ve fethet

---

Bilinen probleme indirgeme

---

Dinamik programlama

---

Kaba Seçim veya Haris (Greedy) algoritması

---

Bir veri yapısı icat etme

---

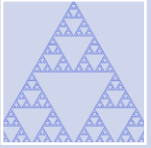
İhtimali (olasılıksal) çözümler

---

Yaklaşım çözümleri



# ❖ Özyineleme



Problemin çözümünün tekrarlı olması durumunda kullanılır.



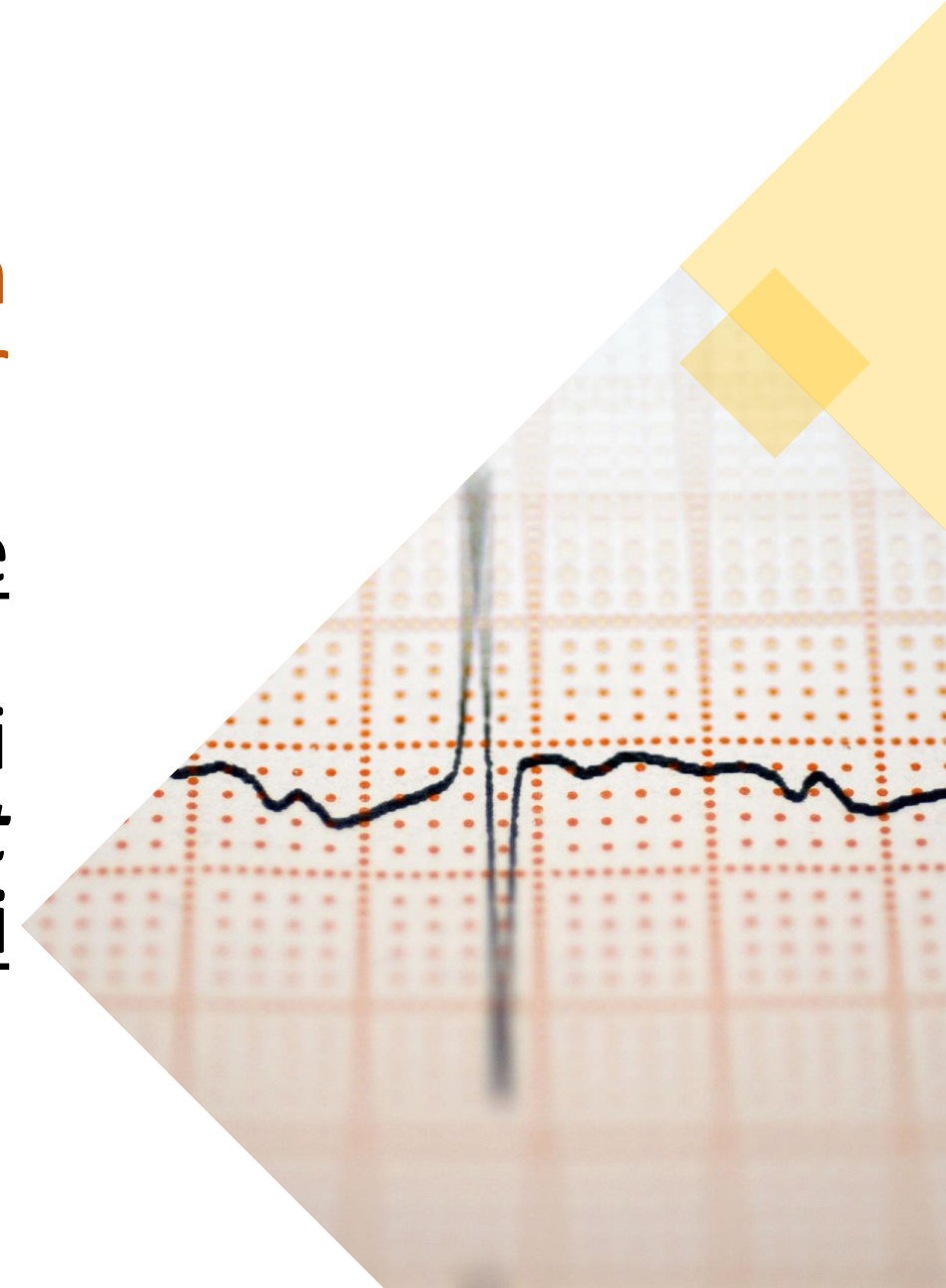
Bilinen *bir* veya *birkaç* çözümden faydalanarak bir sonraki çözümü elde etmede kullanılır.



Elde edilen **çözüm** ile **önceki çözümlerin birkaçının kullanılması** ile **bir sonraki çözümün elde edilmesi** ile problemi çözme işlemine **özyineleme yöntemi** denir.

## ❖ Böl ve fethet

- Kompleks problemlerin bir bütün olarak çözümleri çok zor olabilmektedir.
- Bu problemler alt problemlere bölünürler.
- Bu bölünme işleminin yapılabilmesi için *alt problemlerin bir üst seviyedeki problem* ile aynı özelliği sağlamalıdır.

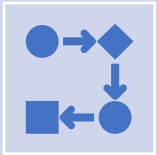




# ❖ Dinamik programlama



**Böl ve yönet** yöntemine benzer olarak alt problemlerin çözümlerini birleştirerek çözüme gitme mantığına sahiptir.



**Alt problem tekrarı varsa**, bunlardan **bir tanesi çözülür** ve bu çözüm diğer tekrarlarda kullanılır.

# ❖ Kaba Seçim veya Haris (Greedy) algoritması



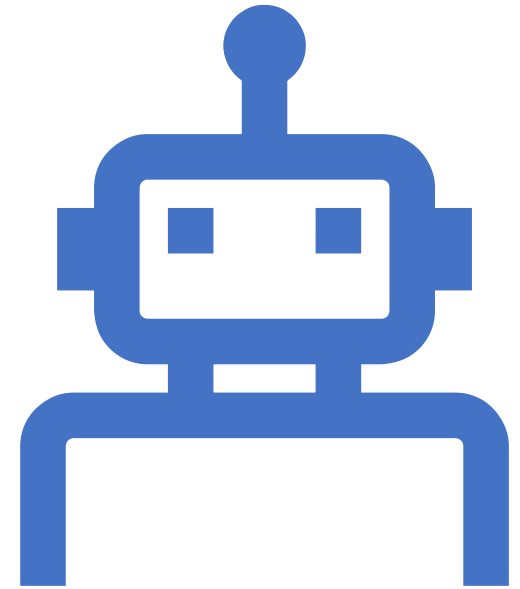
Optimizasyon problemlerinin çözümü için **yerel optimumların seçilmesi ilkesinden yola çıkar.**



**Veriyi belli bir kriter gere düzenledikten sonra ilk veri her zaman optimum çözüme götürür mantığına sahiptir.**

## ❖ Bir veri yapısı icat etme

O ana kadar var olamayan bir veri yapısının icat edilmesi ile problemin çözülmesine **veri yapısı icat etme yöntemi** denir.



## ❖ Bilinen probleme indirgeme

Kompleks olan bir problemin çözümünü yapmak için çözümü **bilinen bir veya birden fazla başka probleme** dönüştürüp şekilde problemi çözme yöntemidir.



## ❖ İhtimali (olasılıksal) çözümler

---

Bazı durumlarda etkili bir şekilde  
problem çözümü yapılabilme  
içim olasılıktan faydalanırız.

8687934714 456498798358687979835868 9347057647597 475456498218847 97621884 798 58  
4 5868797983588 934705766218847597 4754 6498 1884759 14754 6498 983 868 97983586879  
59 4 5456498218847545649879835868797983 868793470 7662 884 579835868793470 7662 8  
59798 586879347058687934705766218847597147545649821884 59714 456498798358687979835  
4 54564987983586879798358687934705766218847579835868 934705 66 1884 597 4 4 64982  
576621884 5 983586879347057662 884759714 54564982 884759 147 45649879835868 9798358  
868 934 14 5456498798358687979835868 934705764759714 4564982 88475976 1884 5798 58  
735868 9 983588793470576621884759 147545649821884759714 5456498798 586879 983586879  
2597 4 54564982188475456498 983586879798 5868 9 470 766 1884 7983586879347057662 8  
9 98358687934 0 868793470576621884 59 14 54564982 884 59 4754 6498 98 5868 9 9835  
14 5456498798358687979835868 93470 66 18847 798 58687934 05766 8847 9714754564982

## ❖ Yaklaşım çözümleri

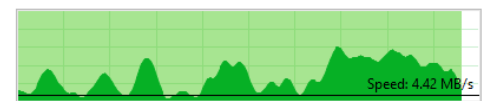
Çözümü deterministik **Turing makinesi** ile yapılamayan yani karmaşık hesaplamaların belirli bir yöntem ile çözülemediği bu problemlerin bir kısmına bazı kriterler uygulayarak yaklaşım mantığı ile çözüm üretilebilmektedir.



# Algoritma Analizi

- ❑ *Algoritma analizi, bilgisayar programının performansı (başarım) ve kaynak kullanımı konusunda teorik çalışmalardır.*
- ❑ Bir başka ifadeyle, *algoritmanın icra edilmesi sırasında duyacağı **kaynak miktarının tahmin edilmesi** denilebilir.*
  - Kaynak denildiğinde, ***bellek, iletişim bant genişliği, mantık kapıları*** akla gelebilir.
  - En önemli kaynak algoritmanın icra edilebilmesi için **gerekli olan zamanın tahmin edilmesidir.**

Copying 30,379 items from Desktop to Copy\_test  
95% complete



Name: Untitled Game (v02).fm  
Time remaining: About 30 seconds  
Items remaining: 1,649 (154 MB)

# Algoritma Analizi

- ❑ Algoritma analizi, farklı çözüm yöntemlerinin verimliliğini analiz eder.
- ❑ Algoritmanın performansı yani başarımı oldukça önemlidir.

Bunun birkaç nedeni vardır.

- Başarım (performans) genelde yapılabilir olanla imkansızın arasındaki çizgiyi tanımlar.
- **Program davranışlarını açıklamak için ortak dil oluşturur.**
- Başarım bilgi işleme'nin değerini gösteren birimdir.
- Program başarımından alınan dersler diğer bilgi işleme kaynaklarına genellenebilir.



# Performanstan daha önemli ne(ler) vardır ?

Modülerlik

Doğruluk

Bakım  
kolaylığı

İşlevsellik

Sağlamlık

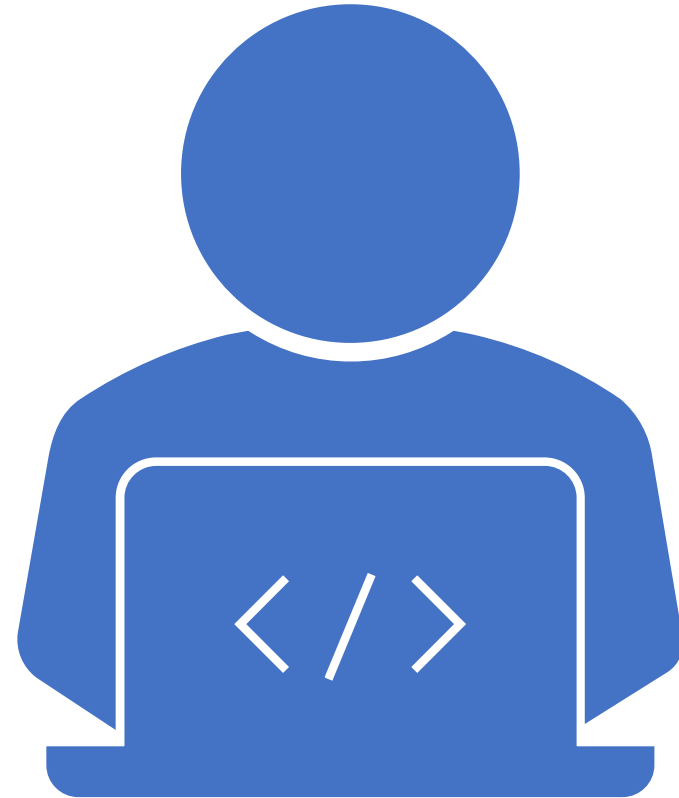
# Performanstan daha önemli ne vardır ?



# Algoritmik Performans

Algoritmik performansın iki yönü vardır:

1. Zaman (Time)
2. Alan (Space)



# 1. Zaman (Time)

Yönergeler veya talimatlar zaman alabilir.

- ❖ Algoritma ne kadar hızlı bir performans gösteriyor?
- ❖ Algoritmanın çalışma zamanını (runtime) ne etkiler?
- ❖ Bir algoritma için gerekli olan zaman nasıl tahmin edilir?
- ❖ Gerekli olan zaman nasıl azaltılabilir?



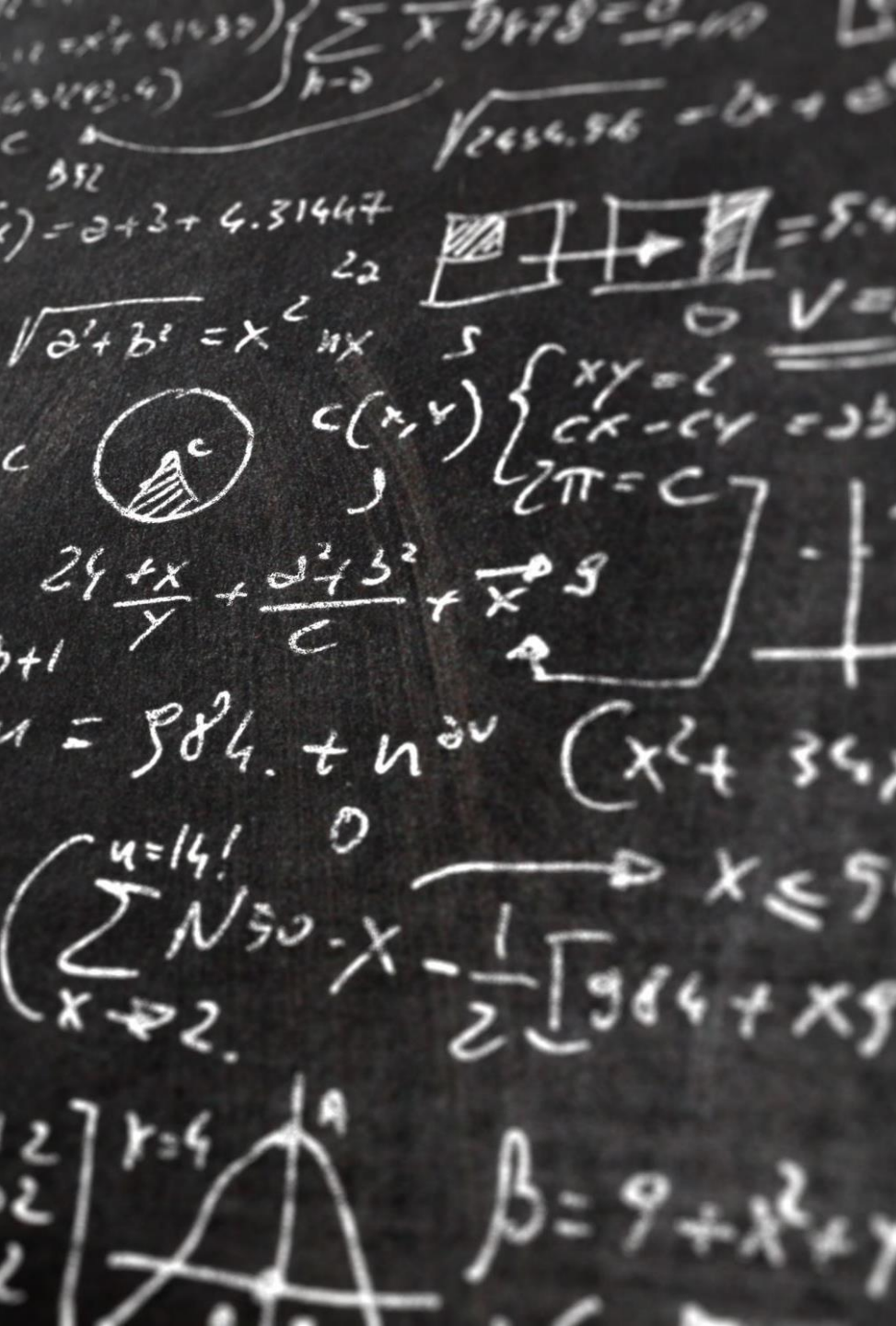
## 2. Alan (Space)

Veri yapıları yer kaplar.

- ❖ Ne tür veri yapıları kullanılabilir?
- ❖ Veri yapılarının seçimi çalışma zamanını nasıl etkiler?



**DATA STRUCTURE**



# Algoritma Analizi

---

- Bir algoritmanın analizinin yapılabilmesi için **matematiksel bilgilere** (temel olasılık, kümeler, cebir, v.b.) ihtiyaç duyulur.
- Ayrıca **bazı terimlerin formül olarak ifade edilmesi** gereklidir.
- **Çünkü her giriş için algoritmanın davranışı farklı olabilir.**

# Algoritma Analizi

---

Benzer problemi  
çözmek için iki  
algoritmanın zaman  
verimliliğini nasıl  
karşılaştırabiliriz?

## □ Basit yaklaşım:

- Bir programlama dilinde bu algoritmaların uygulanması
- Uygulamaların çalışma zamanlarının karşılaştırılması





# Algoritmalar yerine programların karşılaştırılmasındaki zorluklar

---

## ❑ Programın kullanabileceği veri nedir?

Analiz yöntemi **veriye** bağımlı olmamalıdır.  
Çalışma zamanı **verinin** **büyüklüğü** ile değişebilir.

## ❑ Hangi bilgisayarı kullanmak gerekir?

Belirli bir bilgisayara bağımlı olmadan karşılaştırılmalıdır.



# Algoritma nasıl kodlanmalıdır?



**Çalışma zamanını karşılaştırmak, uygulamaları karşılaştırmak anlamına gelir.**



**Uygulamalar, programlama tarzına duyarlı olduğundan karşılaştıramayız.**



**Programlama tarzı çok verimli bir algoritmanın çalışma zamanını bile etkileyebilir.**



**Programları karşılaştırmak, bir algoritmanın kesin ölçümü için uygun değildir.**

# Algoritmaları Karşılaştırmak



Algoritma analizi, özel uygulamalardan, bilgisayarlardan veya veriden bağımsızdır.



Algoritma analizi, ***tasarlanan program veya fonksiyonun*** belirli bir işleme göre matematiksel ifadesini bulmaya dayanır.



Algoritmaları analizinde ***ilk olarak, algoritmanın etkinliğini değerlendirmek için*** belirli bir çözümde anlamlı olan işlemlerin kaç adet olduğu sayılır.



Daha sonra büyüme fonksiyonları kullanılarak **algoritmanın verimliliği ifade edilir**.

# Algoritmaların Analizi

Problem	$n$ elemanlı giriş	Temel işlem
Bir listede arama	liste $n$ elemanlı	karşılaştırma
Bir listede sıralama	liste $n$ elemanlı	karşılaştırma
İki matrisi çarpma	$n \times n$ boyutlu iki matris	çarpma
Bir ağaçta dolaşma	$n$ düğümlü ağaç	Bir düğüme erişme
Hanoi kulesi	$n$ disk	Bir diski taşıma



**Not:** Temel işlem tanımlanarak bir algoritmanın karmaşıklığını ölçebilir.



Giriş büyüklüğü  $n$  için bu temel işlemi, algoritmanın kaç kez gerçekleştirdiği sayılabilir.

# Algoritmaların Analizi

- ❑ Eğer problemin boyutu çok küçük ise algoritmanın verimliliğini muhtemelen ihmal edebiliriz.
- ❑ Algoritmanın **zaman** ve **bellek** gereksinimleri arasındaki ağırlığı dengelemek zorundayız.
  - Örneğin *dizi tabanlı listelerde* geri alma işlemleri  $O(1)$ 'dir.
  - *Bağlı listelerde* geri alma işlemi ise  $O(n)$ 'dir.

Fakat eleman **ekleme** ve **silme** işlemleri *bağlı liste uygulamalarında çok daha kolaydır.*

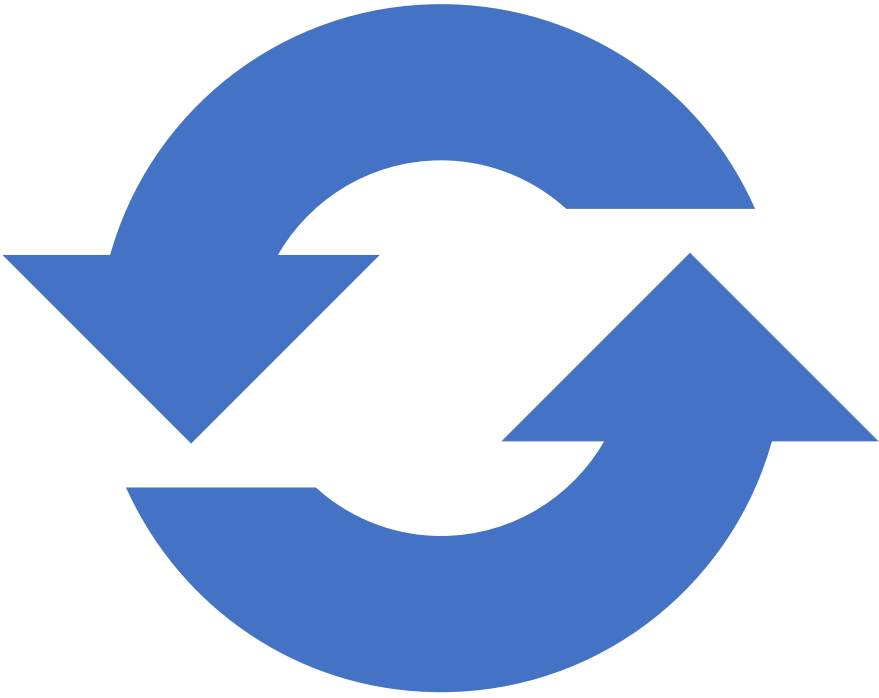
# Çalışma zamanı veya koşma süresi (running time)

- ❑ ' $n$ ' boyutlu bir problemin algoritmasını çalıştırmak için gerekli zamandır ve  $T(n)$  ile gösterilir.
    - ***Bir programın veya algoritmanın işlevini yerine getirebilmesi için,***
      - ***döngü sayısı***
      - ***aritmetik işlem sayısı***
      - ***atama sayısı***
- gibi işlevlerden kaç adet yürütülmesini gösteren bir bağıntıdır.*

# Örnek

	<u>Cost</u>	<u>Times</u>
if (n < 0)	c1	1
absval = -n;	c2	1
else		
absval = n;	c3	1

Toplam maliyet  $\leq c1 + \max(c2, c3)$



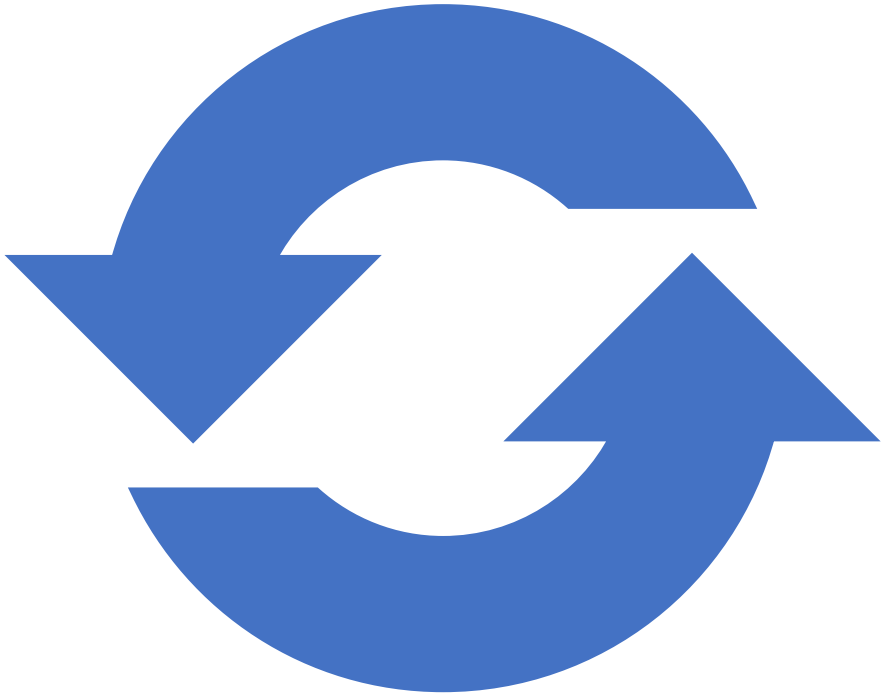
# Tahmin için Genel Kurallar :

## ❑ Döngüler (Loops)

Bir döngünün çalışma zamanı **en çok döngü içindeki deyimlerin** çalışma zamanının iterasyon sayısıyla çarpılması kadardır.

## ❑ İç içe döngüler (Nested Loops)

Grubunun içindeki *deyimin toplam çalışma zamanı*, *deyimlerin çalışma sürelerinin bütün döngülerin boyutlarının çarpımı* kadardır.  
*Analiz içten dışa doğru yapılır.*



## Tahmin için Genel Kurallar :

### ☐ Ardışık deyimler

- Her deyim zamanı birbirine eklenir.

### ☐ if/else

- En kötü çalışma zamanı:

Test zamanına **then** veya **else** kısmındaki çalışma zamanının hangisi *büyükse* o kısım eklenir.



- **Örnek:** Basit bir döngü

- `i = 1;`
- `sum = 0;`
- `while (i <= n) {`
- `i = i + 1;`
- `sum = sum + i;`
- `}`

Maliyet

`c1`

`c2`

`c3`

`c4`

`c5`

Tekrar

1

1

$n+1$

$n$

$n$

- Toplam maliyet= $c1 + c2 + (n+1)*c3 + n*c4 + n*c5 = 3n+3$
- $T(n)=3n+3$
- Bu algoritma için gerekli zaman  **$n$**  ile doğru orantılıdır.

## ○ Örnek: İç içe döngü

	Maliyet	Tekrar
○ <code>i=1;</code>	$c_1$	1
○ <code>sum = 0;</code>	$c_2$	1
○ <code>while (i &lt;= n) {</code>	$c_3$	$n+1$
○ <code>  j=1;</code>	$c_4$	$n$
○ <code>  while (j &lt;= n) {</code>	$c_5$	$n*(n+1)$
○ <code>    sum = sum + i;</code>	$c_6$	$n*n$
○ <code>    j = j + 1;</code>	$c_7$	$n*n$
○ <code>  }</code>		
○ <code>  i = i + 1;</code>	$c_8$	$n$
○ <code>}</code>		
○ Toplam maliyet= $c_1 + c_2 + (n+1)*c_3 + n*c_4 + n*(n+1)*c_5 + n*n*c_6 + n*n*c_7 + n*c_8$		
○ Bu algoritma için gerekli zaman $n^2$ ile doğru orantılıdır.		

# Kaynakça

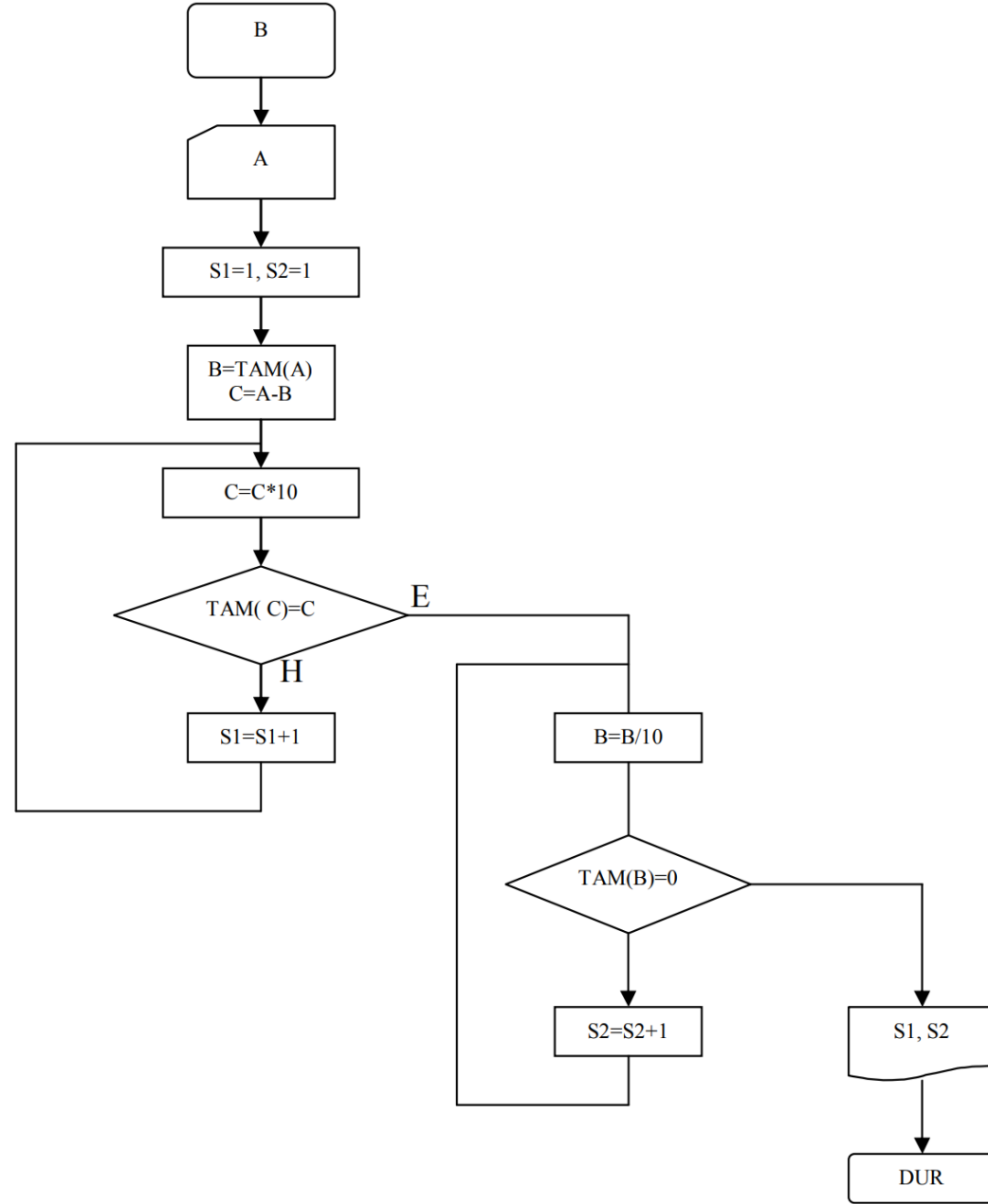
- ▶ Algoritmalar : Prof. Dr. Vasif NABİYEV, Seçkin Yayıncılık
- ▶ Algoritmalara Giriş : Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest and Clifford Stein, Palme YAYINCILIK
- ▶ Algoritmalar : Robert Sedgewick , Kevin Wayne, Nobel Akademik Yayıncılık
- ▶ M.Ali Akcayol, Gazi Üniversitesi, Algoritma Analizi Ders Notları
- ▶ Doç. Dr. Erkan TANYILDIZI, Fırat Üniversitesi, Algoritma Analizi Ders Notları

# UYGULAMALAR

**1. Rasgele girilen bir rasyonel sayısının ondalıklı kısmının ve tam kısmının hane sayısını bulan algoritma akış şemasını çiziniz.**

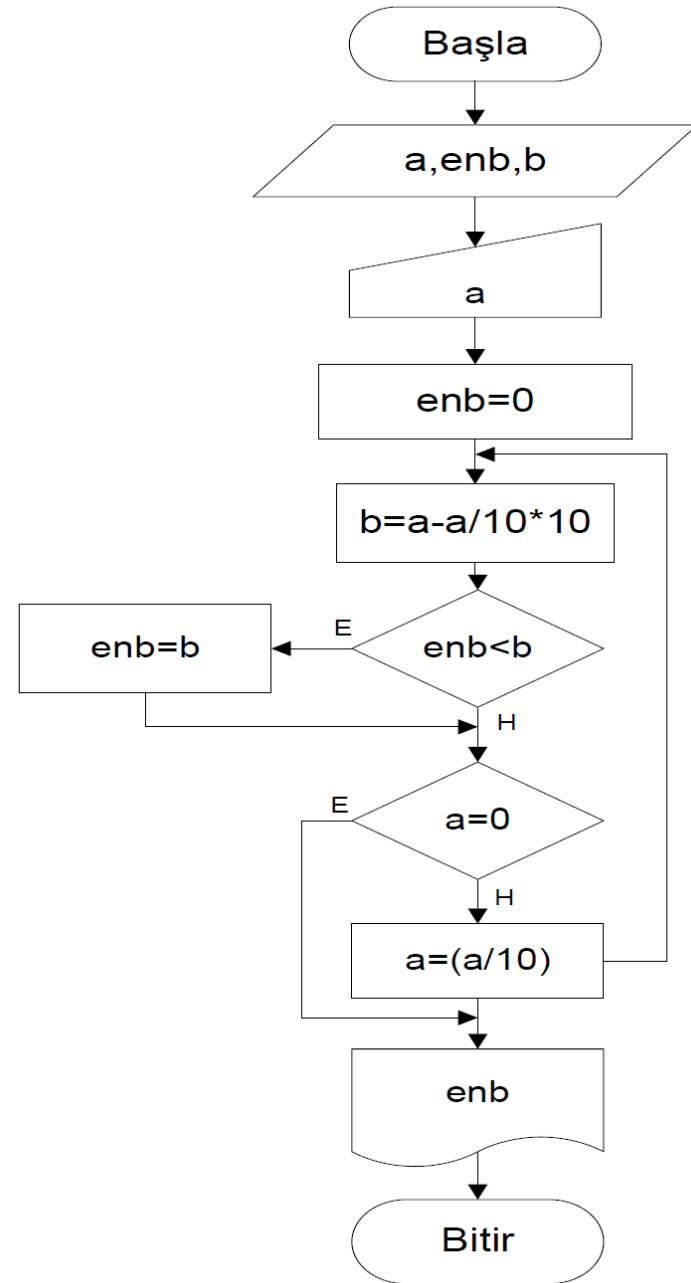
*Not: Algoritmada bir ondalıklı sayının tam kısmını hesaplayan fonksiyon TAM fonksiyonu olarak alınabilir.  
Örneğin  $TAM(4.7)=4$  olarak hesaplar.*

Çözüm:



**2. Girilen bir tam sayının hanelerindeki en büyük sayıyı bulan algoritma ve akış diyagramını çiziniz.**

## Çözüm:



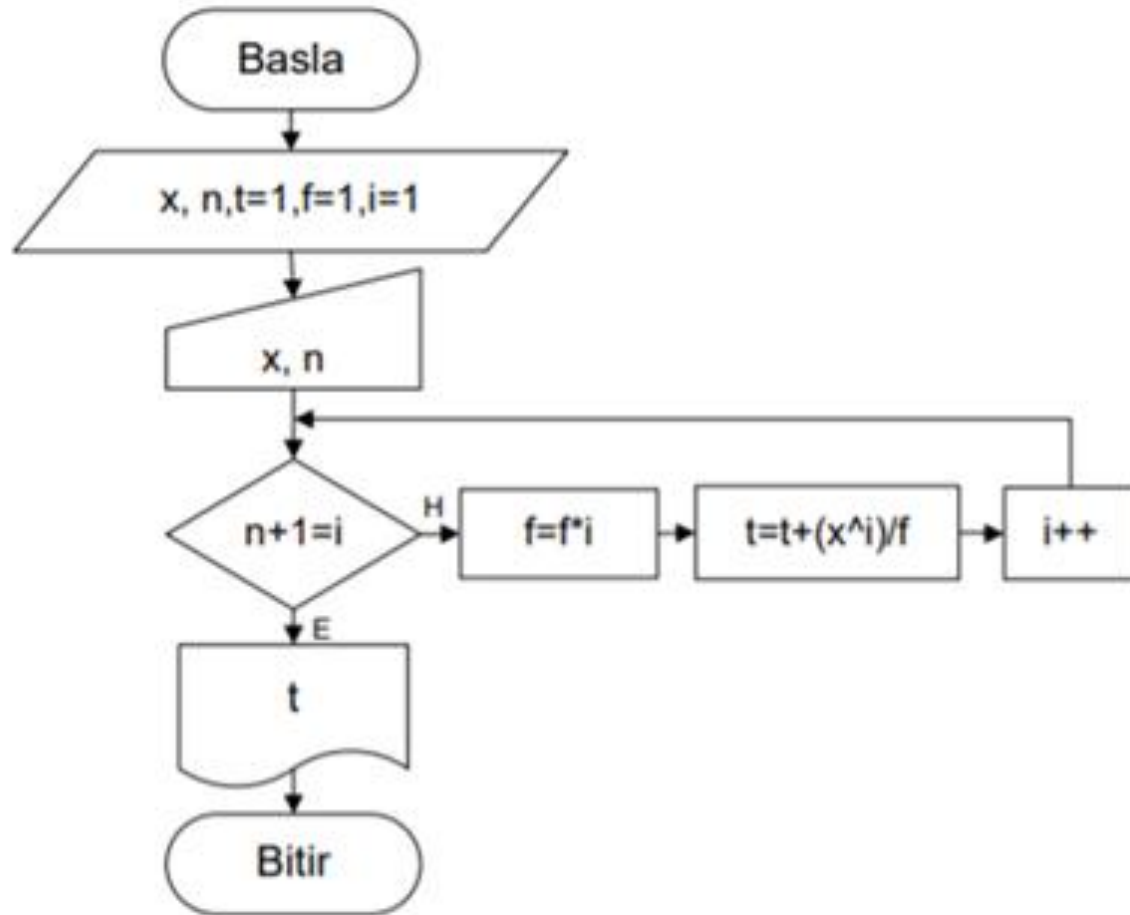


**3.  $e^x$  fonksiyonunun seri açılımı şu şekildedir :**

$$e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \dots + \frac{x^n}{N!} = \sum_{k=0}^N \frac{x^k}{k!}$$

Buna klavyeden  **$x$**  ve  **$N$**  değerleri girildikten sonra  **$e^x$**  fonksiyonun değerini hesaplayan algoritmanın **yalnızca akış şemasını çiziniz**

# Çözüm

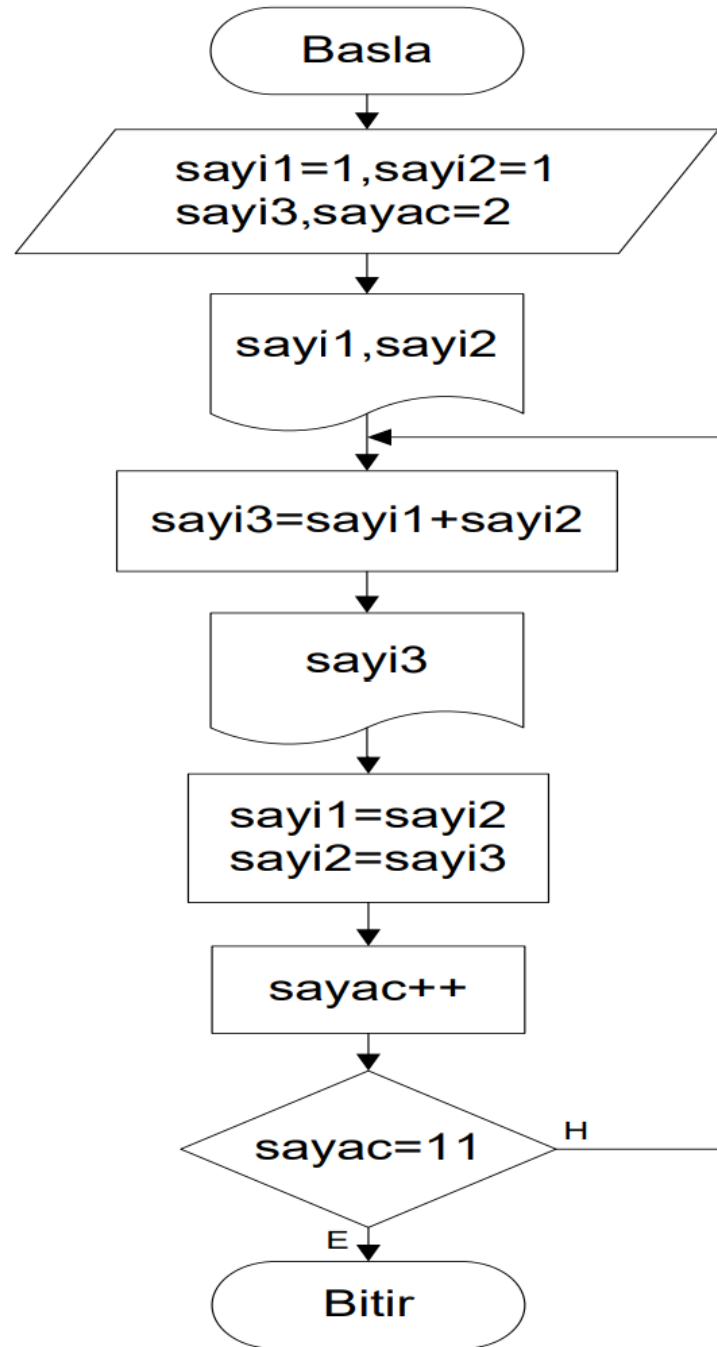


## 4. Fibonacci serisinin ilk 10 terimini ekrana basan algoritma akış şemasını çiziniz.

Algoritma Testi:

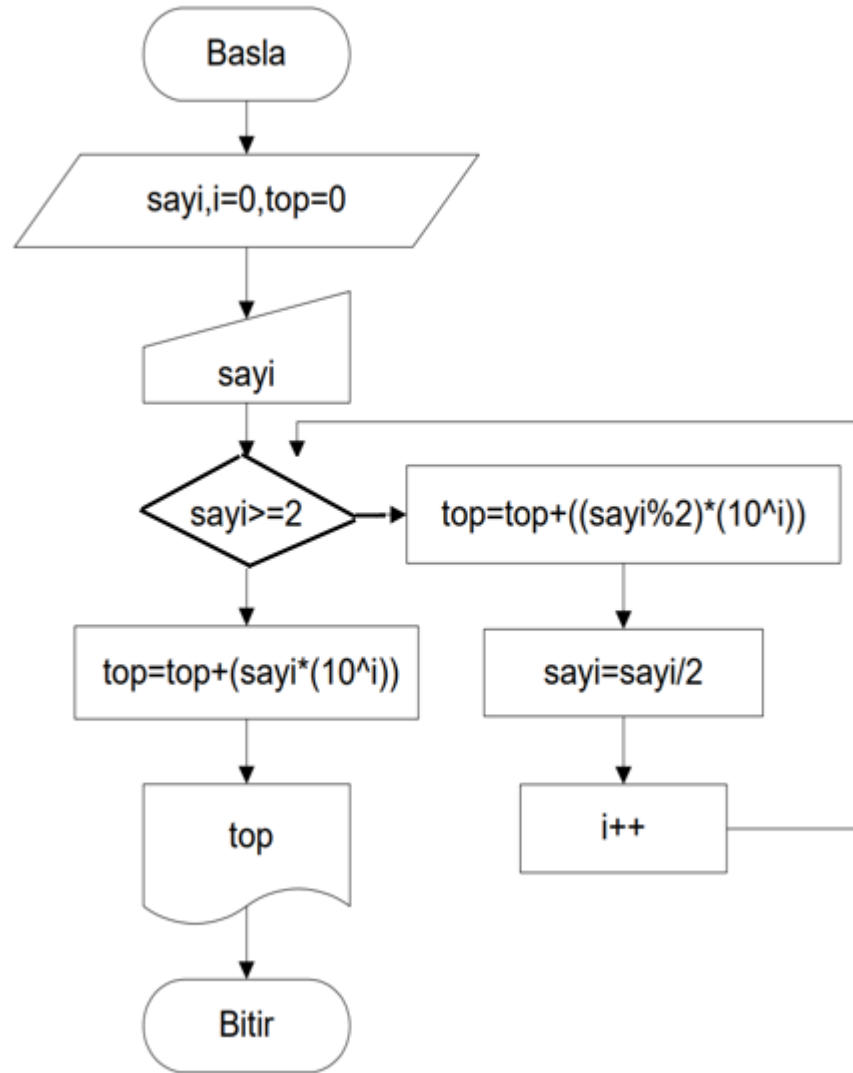
	Sayı1	Sayı2	Sayı3	Sayac
1	1	1	2	2
2	1	2	3	3
3	2	3	5	4
4	3	5	8	5
5	5	8	13	6
6	8	13	21	7
7	13	21	34	8
Sonuç	21	34	55	9

Çözüm:



**5. Girilen decimal (onluk) bir sayının binary (ikilik) bir sayıya dönüştüren programın algoritma akış diyagramını çiziniz.**

# Çözüm:



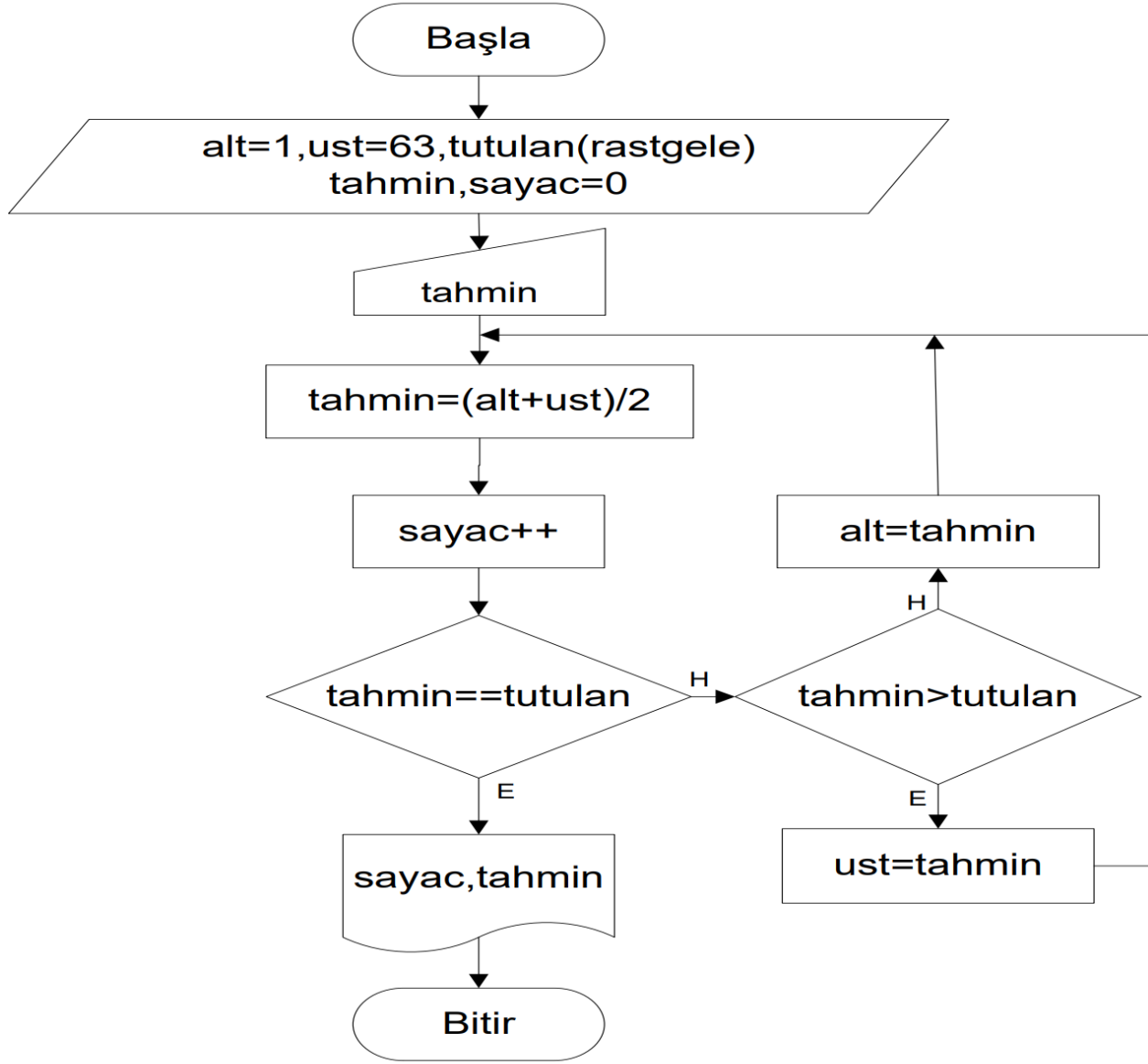
## Algoritma Testi:

sayi	i	(Sayi %2 )*10 <sup>i</sup>	Top
5	0	1*1	1
2	1	0*10	1
1	2	1*100	101

Not( sayi 1 olunca işlem 1 kereye mahsus (sayi\*10<sup>i</sup>) yapıcak ve biticek)

**6. 1'den 63'e kadar olan sayılar arasında istenilen sayıyı maksimum 6 seferde bulan programın algoritma akış şemasını çiziniz.**

# Çözüm:



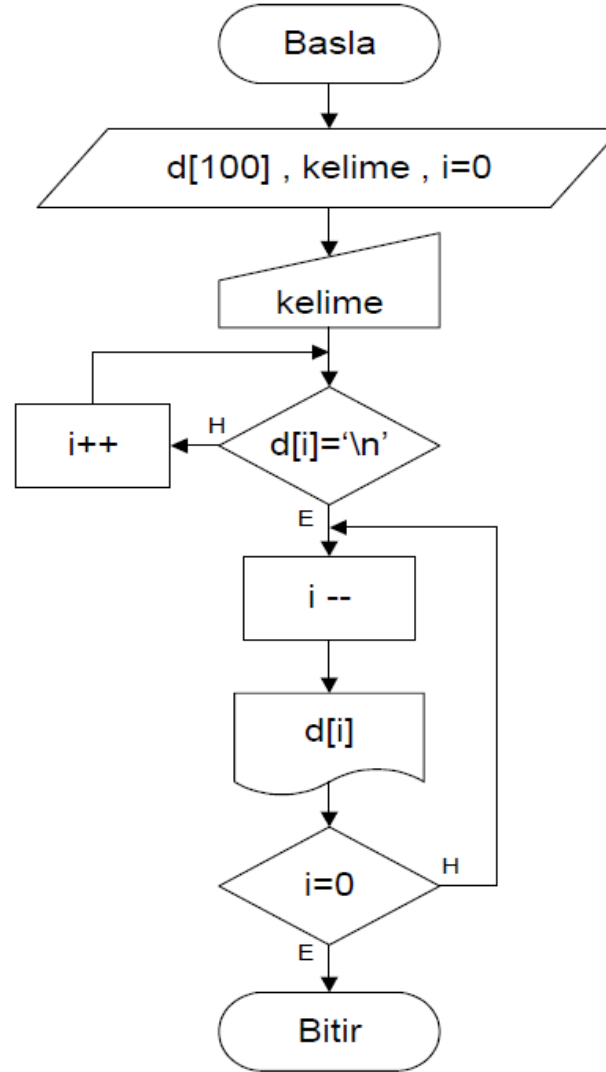
## Algoritma test:

Sayac	Alt	Ust	Sayı(tutulan)	Tahmin	Değerlendirme
1	1	63	7	32>7	(1+63)/2
2	1	32(>7)	7	16>7	(1+32)/2
3	1	16(>7)	7	8>7	(1+16)/2
4	1	8(>7)	7	4<7	(1+8)/2
5	4(<7)	8	7	6<7	(4+8)/2
6	6(<7)	8	7	7=7	(6+8)/2



**7. Girilen kelimeyi tersten yazdıran programın algoritmasını ve akış diyagramını çiziniz.**

# Çözüm:



**8. Tersinden ve düzünden okunuşu aynı olan yazılara polindrom denir. Bir yazının polindrom olup olmadığını bulan programın akış diyagramını çiziniz.**

*Örneğin KÜTÜK polindrom iken BİLGİSAYAR polindrom değildir.*

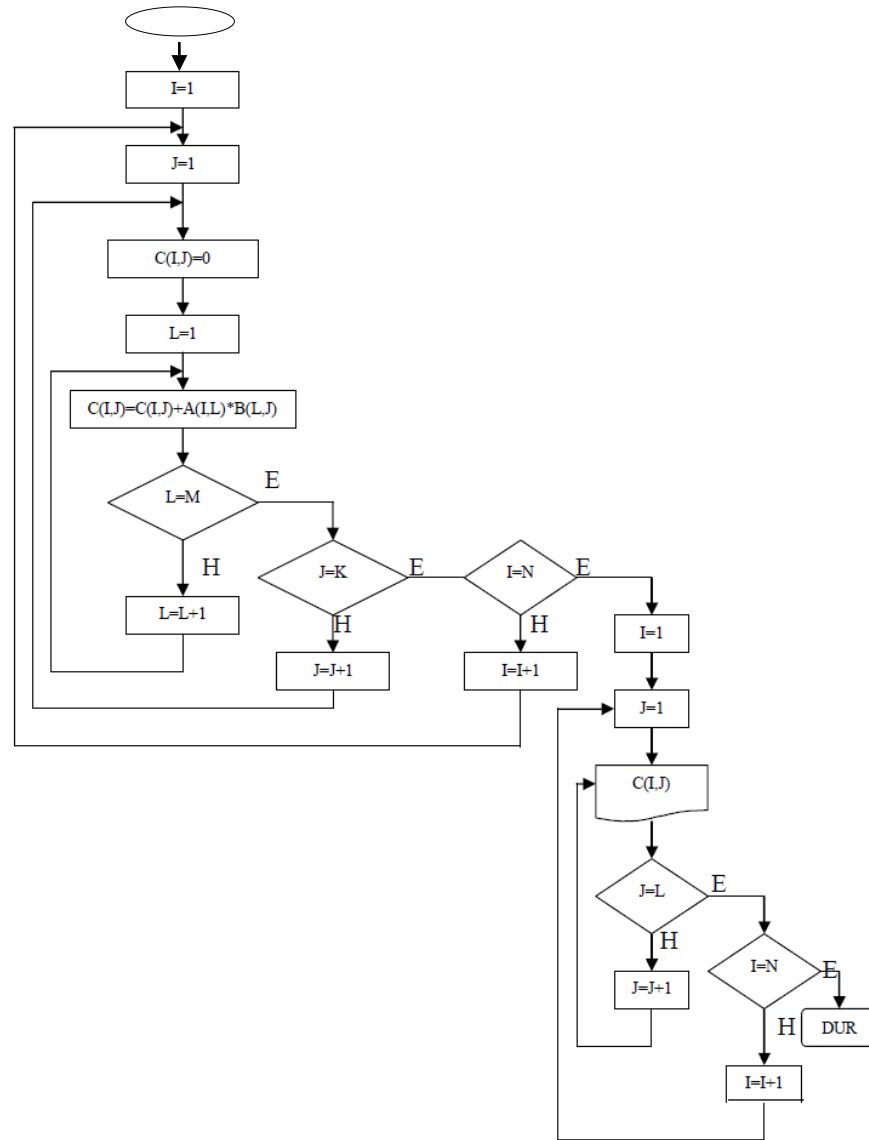


## 9. NxM ve MxK' lik iki matrisin çarpımını bulan algoritma ve akış şemasının oluşturulması.

Örneğin  $C=A*B$

$$A = \begin{pmatrix} 1 & 2 & -1 \\ 3 & 1 & 0 \\ 2 & 4 & 5 \end{pmatrix} \text{ ve } B = \begin{pmatrix} 2 & 1 \\ 0 & 3 \\ 4 & 2 \end{pmatrix}$$
$$C = \begin{pmatrix} -2 & 5 \\ 6 & 6 \\ 24 & 24 \end{pmatrix}$$

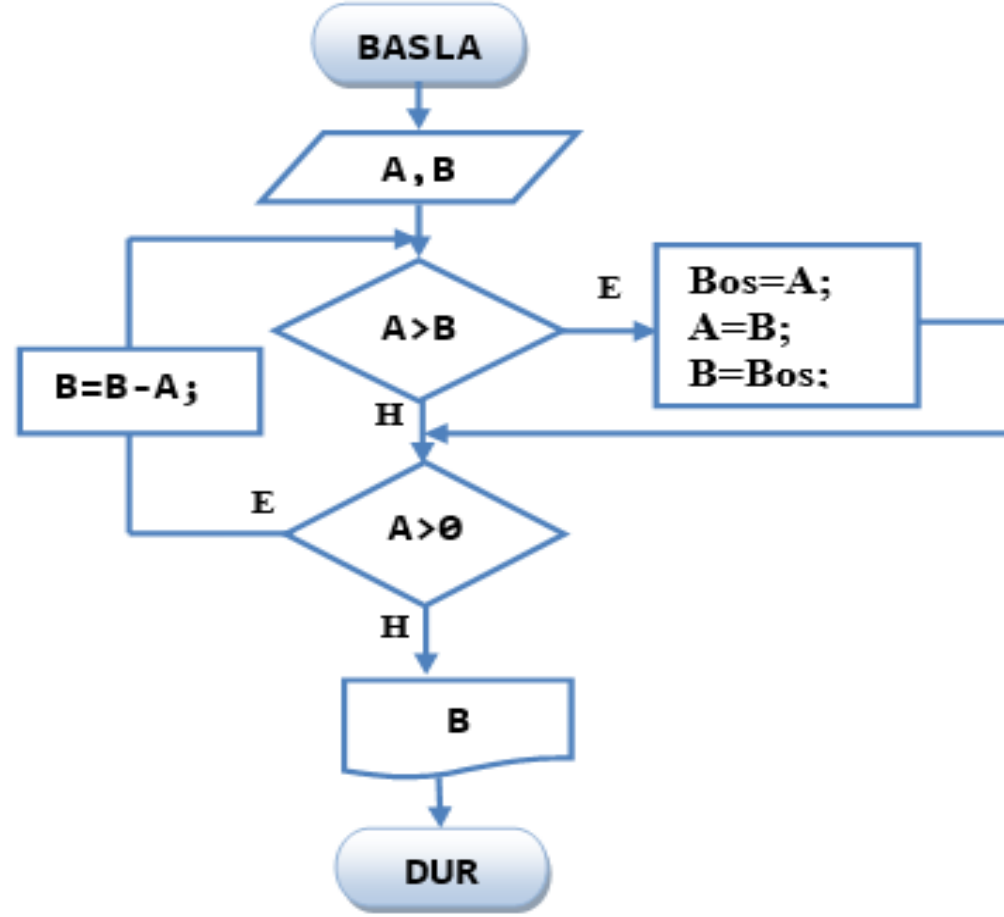
## Çözüm:



**10. Girilen iki sayının en büyük ortak bölenlerini (OBEB) bulan programın akış şeması aşağıda verilmiştir.**

# Çözüm:

	A	B	Bos	Koşul	İşlem (B=B-A)	Ekran
1. adım	12	15				
2. adım	12	15		12>15		
3. adım	12	3		12>0	B=15-12	
4. adım (yer değiştirme)	3	12	12	12>3		
5. adım		9		3>0	B=12-3	
6. adım		6		3>9, 3>0	B=9-3	
7. adım		3		3>6, 3>0	B=6-3	
8. adım		0		3>3, 3>0	B=3-3	
9. adım(yer değiştirme)	0	3	3	3>0		
10. adım				0>0		3





## ❖ Girilen iki sayının en büyük ortak bölenlerini (OBEB)ÖKLİD Algoritması ile çözünüz?

EBOB (600,136)=

$$600 = 4 \cdot 136 + 56$$

$$136 = 2 \cdot 56 + 24$$

$$56 = 2 \cdot 24 + \textcircled{8} \text{ EBOB}$$

$$24 = 3 \cdot 8 + 0$$

1. Eğer  $A=0$  ise,  $\text{OBEB}(0,B)=B$  olacağı için  $\text{OBEB}(A,B)=B$  olur ve bu noktada durabiliriz.
2. Eğer  $B=0$  ise,  $\text{OBEB}(A,0)=A$  olacağı için  $\text{OBEB}(A,B)=A$  olur ve bu noktada durabiliriz.
3. Eğer  $A = B \cdot Q + R$  ve  $B \neq 0$  ise bu durumda  
•  $\text{OBEB}(A,B) = \text{OBEB}(B,R)$  olur. Burada  $Q$  bir tam sayıdır,  $R$  0 ve  $B-1$  arasında bir tam sayıdır.

• İlk iki özellik, iki sayıdan birinin 0 olması durumunda OBEB'i bulmamızı sağlar.


• Üçüncü özellik ise, daha büyük ve zor sayıları alıp, daha küçük, basit sayılara indirgeyerek problemi çözmemizi sağlar.

# 11. Verilen pay ve paydası girilen bir kesrin sürekli (Zincir) kesir karşılığını hesaplayan algoritma akış şemasını yazınız?

$$a_0 + \frac{1}{a_1 + \frac{1}{a_2 + \frac{1}{a_3 + \ddots}}} = [a_0; a_1, a_2, a_3, \dots]$$

Örneğin  $\frac{72}{19}$  kesrinin karşılığı:

- $72 = 3 \times 19 + 15$
- $19 = 1 \times 15 + 4$
- $15 = 3 \times 4 + 3$
- $4 = 1 \times 3 + 1$
- $3 = 3 \times 1 + 0$


$$\frac{72}{19} = \underline{[3; 1, 3, 1, 3]} = 3 + \frac{1}{1 + \frac{1}{3 + \frac{1}{1 + \frac{1}{3}}}}$$

Bu bölme işlemi, aslında Öklid algoritmasıdır.

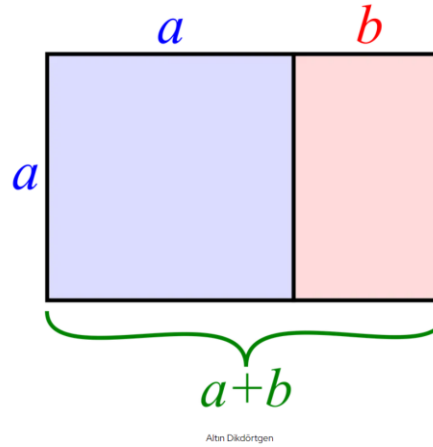
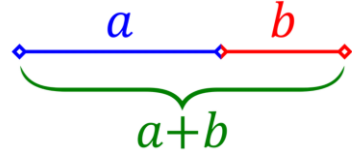
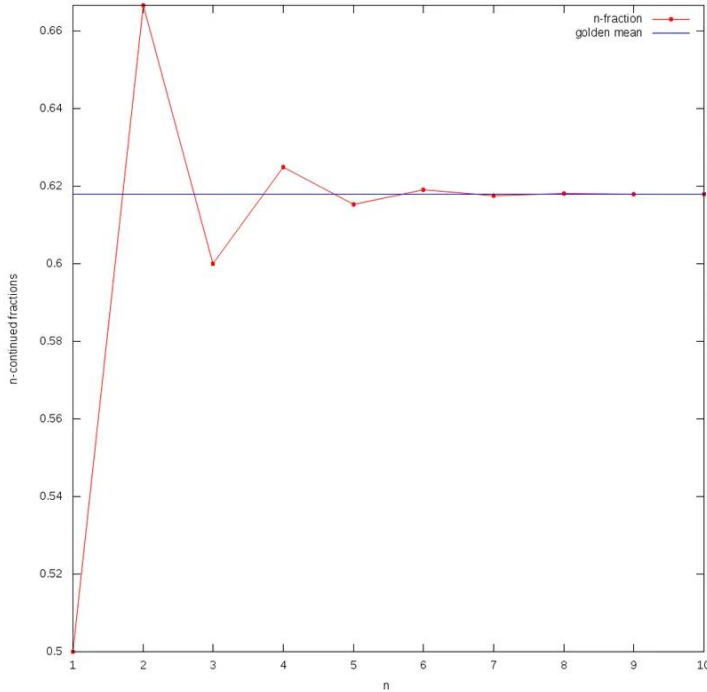
11. En temel sürekli kesir olan  $q=1 + \frac{1}{1+\frac{1}{1+\frac{1}{1+\frac{1}{1+\dots}}}}$  altın oranı 1.618034 değerini vermektedir.

E hata değeri ile bu kesrin hesaplanmasını sağlayan algoritmayı çiziniz.

❖ **HATIRLATMA:**  
**Altın oranı :**

$$\frac{a+b}{a} = \frac{a}{b} = \varphi = 1.618034$$

Finite continued fraction approximation for golden mean



$$0 + 1 = 1$$

$$1 + 1 = 2$$

$$1 + 2 = 3$$

$$2 + 3 = 5$$

$$3 + 5 = 8$$

$$5 + 8 = 13$$

$$8 + 13 = 21$$

$$13 + 21 = 55$$

$$21 + 34 = 55$$

$$34 + 55 = 89$$

$$55 + 89 = 144$$

$$89 + 144 = 233$$

$$144 + 233 = 377$$

$$233 + 377 = 610$$

$$377 + 610 = 987$$

...

$$2/1 = 2$$

$$3/2 = 1.5$$

$$5/3 = 1.6666666666...$$

$$8/5 = 1.6$$

$$13/8 = 1.625$$

$$21/13 = 1.61538461538...$$

$$34/21 = 1.61904761905...$$

$$55/34 = 1.61764705882...$$

$$89/55 = 1.61818181818...$$

$$144/89 = 1.61797752809...$$

$$233/144 = 1.61805555556...$$

$$377/233 = 1.61802575107...$$

$$610/377 = 1.61803713528...$$

$$987/610 = 1.61803278689...$$

...

\* Fibonacci sayılarının 12. teriminden itibaren **ardışık bölümler (bir önceki ile bir sonraki terimin bölünmesi= altın orana yaklaşmaktadır.**

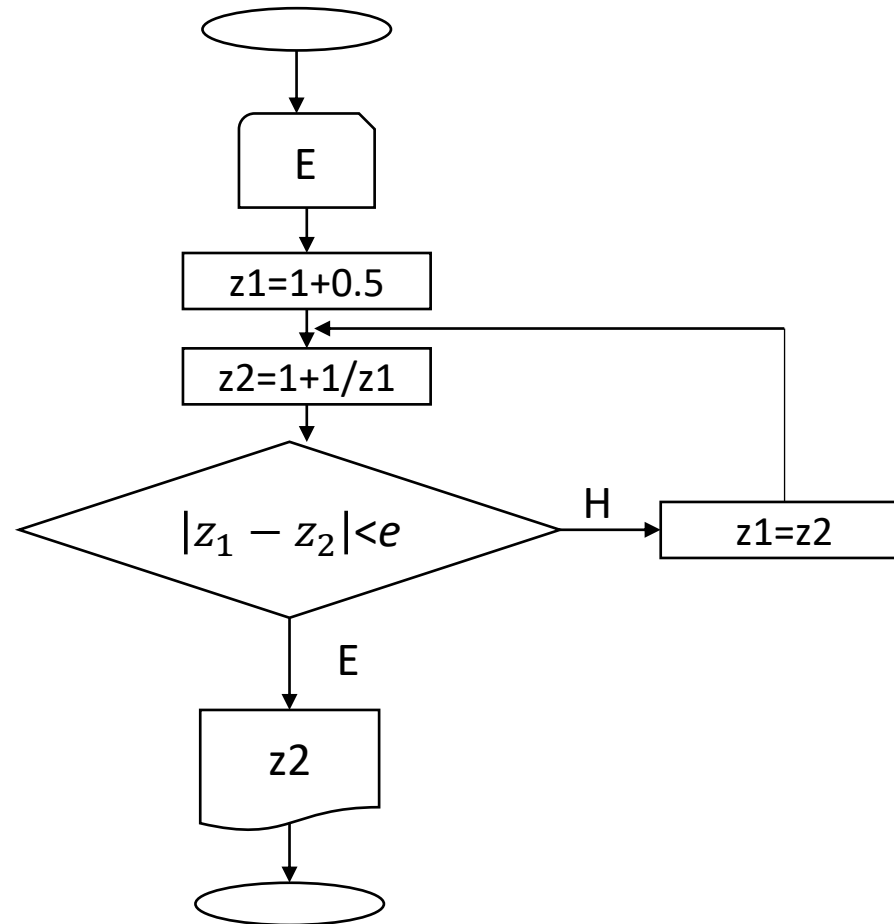
Fibonacci Serisi, hızlı bir şekilde altın orana yakınsamaktadır. Burada oranlar, küçük sayının büyük sayıya oranı şeklinde verildiği için 0.618... sayısına yakınsamaktadır. Tam tersi de aynı sonucu verecektir.

# Çözüm:

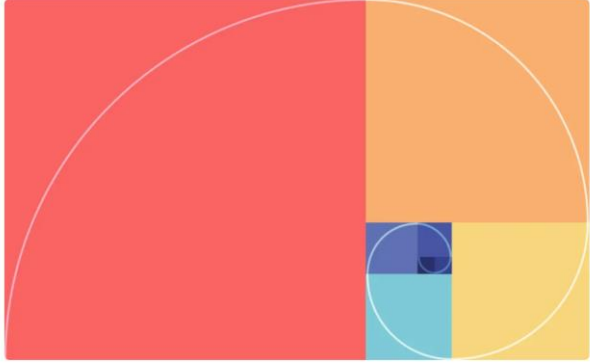
$$q = 1 + \frac{1}{1 + \frac{1}{1 + \frac{1}{z_1 \left( 1 + \frac{1}{1 + \dots} \right)}}}$$

z2

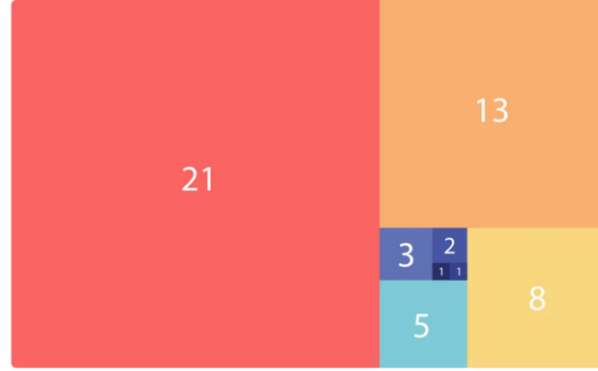
Sürekli hesaplamalarda iki değer arasındaki hata,  $e$ 'den küçük oluncaya kadar devam edeceğinden herhangi bir payda kısmına sabit bir değer verebiliriz. Burada 0.5 değeri seçilmiştir.



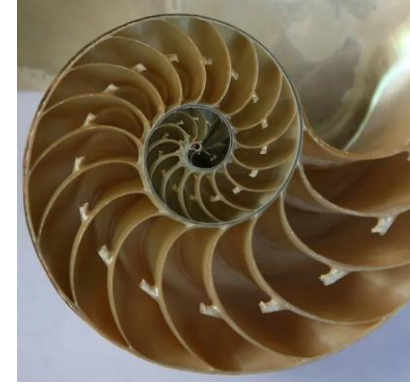
# Altın oran'ın örnekleri



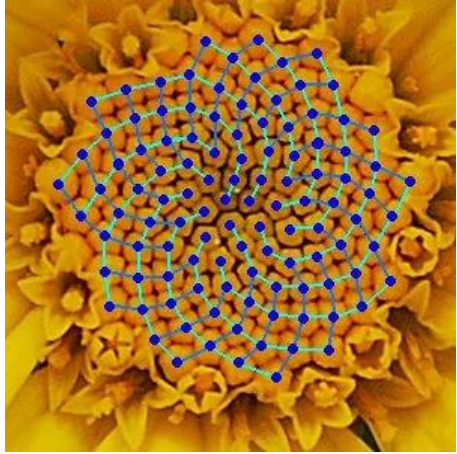
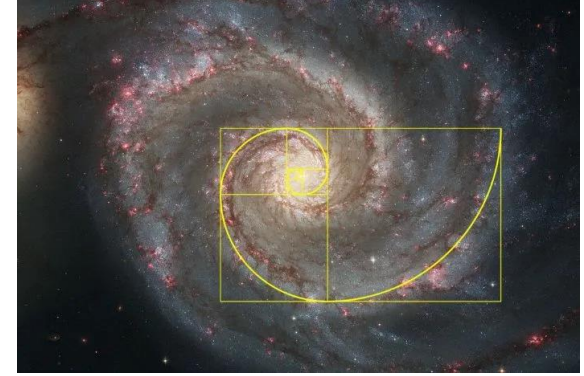
Ortaya "altın spiral" çizgi çıkar.



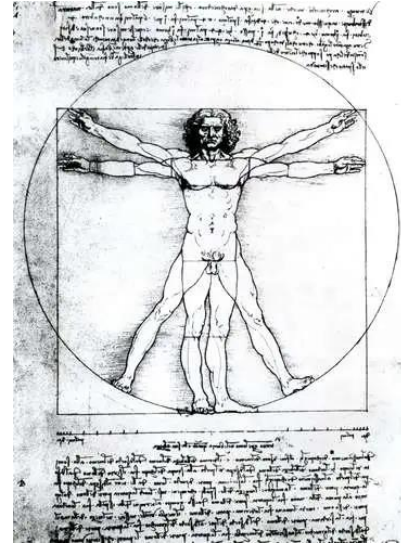
Fibonacci sayılarına göre oranlanmış dikdörtgen.



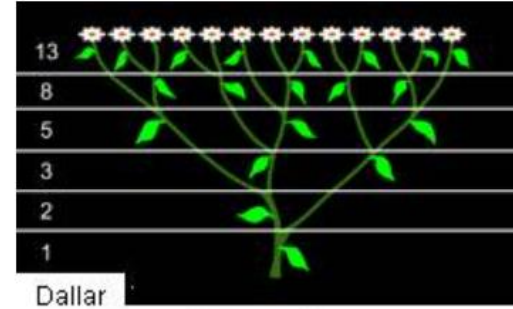
Fibonacci spirali ile görülen bir deniz kabuğu.



Tohumların merkezden başlayarak yaptığı spiral dağılım 1,618 değerine yakındır.

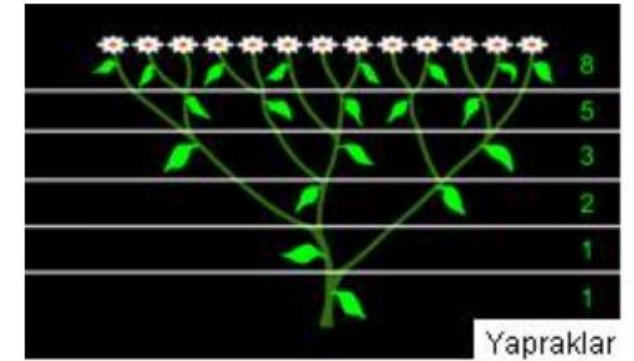


Kendi vücudunuza. **Omuzdan parmak ucuna** kadar ölçmek ve ardından bu sayıyı **dirsekten parmak ucuna** kadar olan uzunluğa bölmek altın oranı verir. Benzer şekilde, **baştan ayağa** kadar ölçüm yapmak ve bunu **göbek deliğinden ayağa** kadarki uzunluğa bölmek 1,618'i verir



Dallar

Dal sayıları



Yapraklar

Yaprak sayıları

❑ Sürekli kesirlere başka örnekler vermek gerekirse **bir robot tasarımında dişlilerin açı hızlarının  $a$**  olacak şekilde birleştirilmesi gösterilebilir.

**Tekerleklerin açısal hızları,** dişler sayısı ile ters orantılı olacaktır.

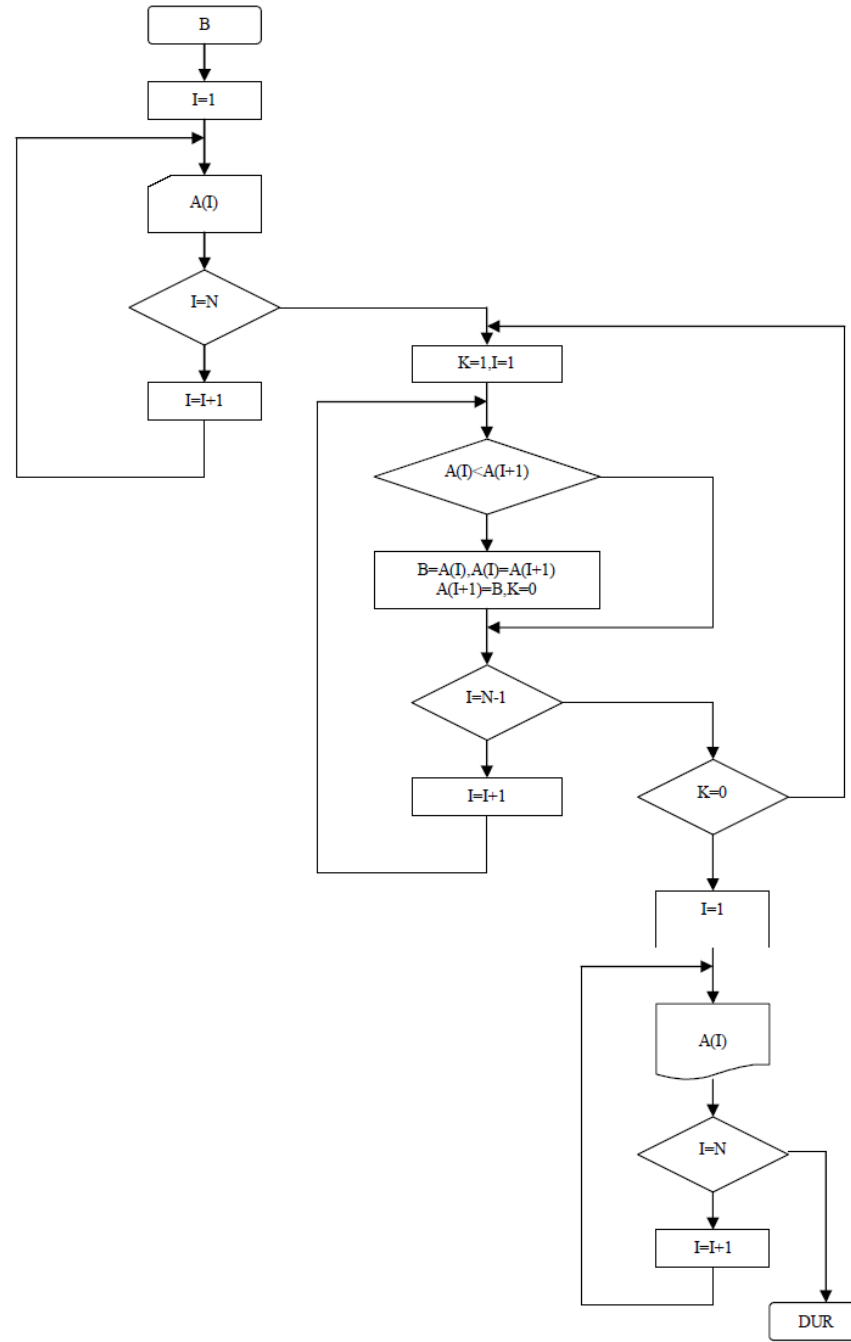
Diş sayıları tam sayıdır.

$a$  ise irrasyonel sayı olacaktır.

❑ Bir başka örnek ise **yılın günlerinin daha dakika cinsinden** gösterilmeside sürekli kesirlerin kullanımından faydalanabiliriz.

**10. N elemandan oluşan bir A sayı dizisinin küçükten büyüğe doğru sıralamasını yapan algoritma akış şemasının oluşturulması.**

# Çözüm:





**11. Girilen bir sayının Roma rakamı ile ifade eden algoritmayı geliştiriniz.**