Nesne Tabanlı Programlama I

Öğr. Gör. Dr. Aysun ALTIKARDEŞ

AMAÇ

- Kabarcık Sıralama (Bubble Sort)
- Araya Yerleştirerek Sıralama (Insertion Sort)
- Seçmeli Sıralama (Selection Sort)
- Hızlı Sıralama (Quick Sort)
- Doğrusal Arama
- İkili Arama

üzerinde önemle durularak, temel programlama bilgilerinin kazandırılması amaçlanmıştır.

Sıralama Algoritmaları
(Sorting Algorithm)

Sıralama, bir grup veriyi artan ya da azalan bir şekilde art arda yerleştirme işlemidir. Sıralama işlemi hem sayısal hem de string türdeki veriler için artan yani 0'dan 9'a veya A'dan Z'ye doğru ya da azalan yani 9'dan 0'a veya Z'den A'ya doğru yapılabilir.

NOT: String ifadelerin sıralanmasında Türkçe karakterler dikkate alınmaz.

Sıralama işlemi için birçok algoritma geliştirilmiştir. Bu algoritmaların her biri farklı performansta olmakla birlikte seçilen veri modeline (dizi, bağlantılı liste gibi) göre de farklılık göstermektedir.

En sık kullanılan algoritmalar

- Kabarcık Sıralama (Bubble Sort)
- Araya Yerleştirerek Sıralama (Insertion Sort)
- Seçmeli Sıralama (Selection Sort)
- Hızlı Sıralama (Quick Sort)
- Birleştirmeli Sırlama (Merge Sort)
- Kümelemeli Sıralama (Heap Sort)
- Kabuk Sıralama (Shell Sort)

Kabarcık Sıralama (Bubble Sort)

Yer değiştirmeli sıralama (Exchange Sort) olarak da bilinir. Dizide yer alan her eleman, sırasıyla kendisinden sonra gelen elemanla karşılaştırılır ve gerekiyorsa yer değiştirilir. Sıralama işlemi yer değiştirme olduğu sürece devam eder.

n elemanlı bir dizi, maksimum n-1 karşılaştırmada sıralanacaktır.

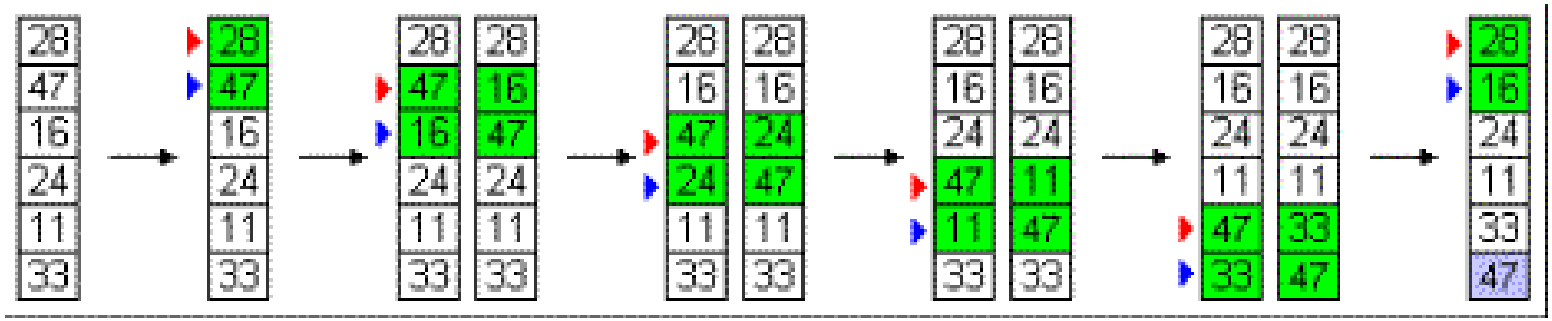
Kabarcık sıralama algoritması kullanılarak dizi elemanlarının artan şekilde sıralanması için gerekli işlem basamakları:

Sıraya konulmamış elemanların her birinin değeri, bir sonraki değerle yeni komşusu ile karşılaştırılır.

Eğer karşılaştırılan değer komşu elemandan daha büyükse, komşusu ile yer değiştirlir. Böylece sıralı olmayan elemanlar her tarandığında, sadece yan yana bulunan iki eleman arasında sıralama yapılmış olur. Dizinin başından sonuna kadar tüm elemanlar bir kez işleme tabi tutulduğunda dizinin son elemanı en büyük eleman haline gelecektir.(Yani ilk geçişte, dizi içerisindeki en büyük eleman en sona gider.)

Tüm elemanlar sıralana kadar, sıralanmamış elemanların taranma işlemi tekrarlanır.

Kabarcık Sıralama (Bubble Sort)



Java Programlama Dilinde Kodlanması

```
for(i=0; i<n-1; i++){
        for(k=0; k<n-1-i; k++)
                if(a[k+1]<a[k]){
                        Bos=a[k];
                        a[k]=a[k+1];
                        a[k+1]=Bos;
                }
}
```

Kabarcık Sıralama (Bubble Sort)

Örnek: Bilgisayarın 0 ile 100 arasında rastgele ürettiği n adet sayıyı küçükten büyüğe doğru kabarcık sıralama (bubble sort) algoritmasını kullanarak sıralayan programı yazınız.

CEVAP:

```
import java.util.Scanner;
public class Kabarcik {
public static void main(String[] args) {
Scanner tara=new Scanner(System.in);
int i,k,Bos,n;
System.out.println("Dizi boyutunu giriniz");
n=tara.nextInt();
int a[]=new int [n];
System.out.println("a dizisinin sıralanmamış elemanları");
for (i=0;i<n;i++){
                a[i]=(1+(int)(Math.random()*100));
                System.out.print(a[i]+"\t");
}
for(i=0; i<n-1; i++){
                for(k=0; k<n-1-i; k++)
                                if(a[k+1]<a[k]){
                                                Bos=a[k];
                                                a[k]=a[k+1];
                                                a[k+1]=Bos;
                                }
}
System.out.println();
System.out.println("a dizisinin sıralı elemanları");
for (i=0;i<n;i++){
                System.out.print(a[i]+"\t");
}
}}
```

Ekran Çıktısı

```
Dizi boyutunu giriniz
5
a dizisinin sıralanmamış elemanları
58      86      52      18      26
a dizisinin sıralı elemanları
18      26      52      58      86
```

Araya Yerleştirerek Sıralama (Insertion Sort) Algoritması

Araya yerleştirerek/sokarak sıralama algoritmasında sıralanacak dizisinin ilk elemanı dışındaki tüm elemanlar sırasıyla incelenir. Sıralanacak dizi iki parça gibi düşünülür.

Sıralı olan kısım ve

Henüz sıralanmış olan kısım.

Sıralanmamış kısımdan elemanlar alınarak sıralı kısımda uygunyerde uygun yerde araya sokulur. Şöyle ki, dizinin ikinci elemanı alındığında, dizi 2 elemanlı gibi kendi içinde sıralanır; 3. eleman alındığında dizi 3 elemanlı gibi kendi içinde sıralanır ve benzer şekilde dizinin henüz sıralanmamış tarafından yeni bir eleman alınıp, sıralı olan tarafta araya yerleştirilir. Tüm elemanlar sıralanana kadar bu işlem devam eder.

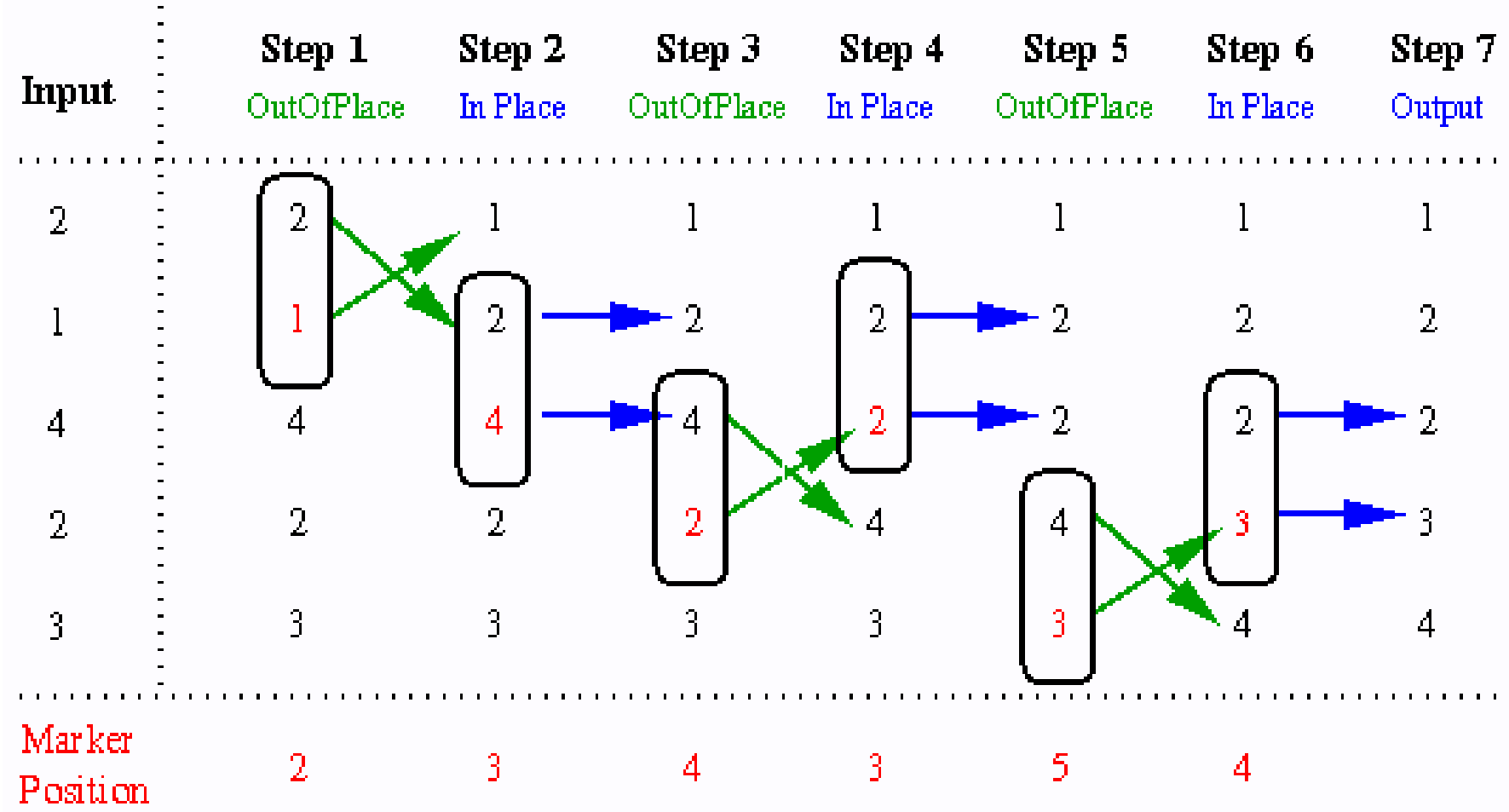
Araya Yerleştirerek Sıralama (Insertion Sort) Algoritması

Araya yerleştirerek sıralama algoritmasını kullanarak dizi elemanlarını artan şekilde sıralamak için gerekli işlem basamakları;

Sıraya konulmamış elemanların ilkini al.

Bu elemanı, ilk olarak kendinden önceki eleman (sıraya konulmuş son eleman) ile karşılaştır. Eğer kendinden önceki elemanın değeri daha büyükse onunla yer değiştir. Bu işleme, kendisinden önceki elemanının değerini daha küçük bulana kadar devam et.

Tüm elemanlar sıraya konulana dek, yukarıdaki iki adımı tekrarla.


Input
Step 1
OutOfPlace
Step 2
In Place
Step 3
OutOfPlace
Step 4
In Place
Step 5
OutOfPlace
Step 6
In Place
Step 7
Output
2
1
4
2
3
Marker Position
2
3
4
3
5
4

Araya Yerleştirerek Sıralama (Insertion Sort) Algoritması

Java Programlama Dilinde Kodlanması

```
for (i=1; i<n; i++){
        Bos=a[i];
        while(i>0 &&a[a-1]>Bos){
                a[i]=a[i-1];
                --i;
        }
        a[i]=Bos;
}
```

NOT: while(i>0 &&a[a-1]>Bos) satırı *while(a[a-1]>Bos && i>0)* şeklinde yazılırsa program dizisinin sınırları aşıldığı gerekçesi ile duracaktır. Bunun nedenlerinden biri **&&** işaretinin simetrik bir operatör olmamasıdır.

Araya Yerleştirerek Sıralama (Insertion Sort) Algoritması

Örnek: Bilgisayarın 0 ile 100 arasında rastgele ürettiği n adet sayıyı küçükten büyüğe doğru araya yerleştirerek sıralama (insertion sort) algoritmasını kullanarak sıralayan programı yazınız.

CEVAP:

```
import java.util.Scanner;
public class Insertion {
public static void main(String[] args) {
Scanner tara=new Scanner(System.in);
int i,Bos,n;
System.out.println("Dizi boyutunu giriniz");
n=tara.nextInt();
int a[]=new int [n];
System.out.println("a dizisinin sıralanmamış elemanları");
for (i=0;i<n;i++){
                a[i]=(1+(int)(Math.random()*100));
                System.out.print(a[i]+"\t");
}
for(i=1; i<n; i++){
                Bos=a[i];
                while(i>0 && a[i-1]>Bos){
                                a[i]=a[i-1];
                                --i;
                }
                a[i]=Bos;
}
System.out.println();
System.out.println("a dizisinin sıralı elemanları");
for (i=0; i<n; i++){
                System.out.print(a[i]+"\t");
}
}}
```

Ekran Çıktısı

```
Dizi boyutunu giriniz
5
a dizisinin sıralanmamış elemanları
83    52    19    68    31
a dizisinin sıralı elemanları
19    31    52    68    83
```

Seçmeli Sıralama (Selection Sort) Algoritması

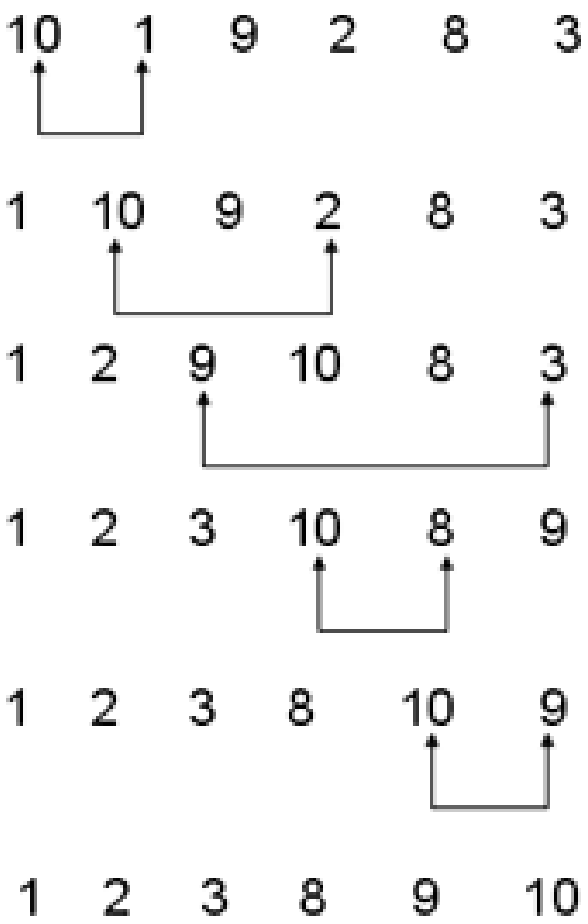
Seçmeli sıralamasını kullanarak dizi elemanlarını artan şekilde sıralamak için gerekli işlem basamakları:

Sıraya konulmamış elemanlar içindeki en küçük değerdeki elemanı bul.

Bulunan bu elemanı, sıraya konulmamış ilk eleman ile yer değiştir.

Sıraya konulmamış ilk elemanın yeri dizinin sonu olana kadar bu işlemi tekrarla.

Seçmeli Sıralama
(Selection Sort) Algoritması



Seçmeli Sıralama
(Selection Sort) Algoritması

Java Programlama Dilinde Kodlanması

```
for (i=0; i<n-1; i++){
        enk=i;
        for(k=i+1; k<n; k++){
                if(a[enk]>a[k]){
                        enk=k;
                }
        }
        Bos=a[i];
        a[i]=a[enk];
        a[enk]=Bos;
}
```

Seçmeli Sıralama
(Selection Sort) Algoritması

Örnek: Bilgisayarın 0 ile 100 arasında rastgele ürettiği n adet sayıyı küçükten büyüğe doğru seçmeli sıralama (selection sort) algoritmasını kullanarak sıralayan programı yazınız.

CEVAP:

```
import java.util.Scanner;
public class Secmeli {
public static void main(String[] args) {
Scanner tara=new Scanner(System.in);
int i,k,Bos,n,enk;
System.out.println("Dizi boyutunu giriniz");
n=tara.nextInt();
int a[]=new int [n];
System.out.println("a dizisinin sıralanmamış elemanları");
for (i=0;i<n;i++){
                a[i]=(1+(int)(Math.random()*100));
                System.out.print(a[i]+"\t");}
for (i=0; i<n-1; i++){
                enk=i;
                for(k=i+1; k<n; k++){
                                if(a[enk]>a[k]){
                                                enk=k;}
                }
                Bos=a[i];
                a[i]=a[enk];
                a[enk]=Bos;}
System.out.println();
System.out.println("a dizisinin sıralı elemanları");
for (i=0; i<n; i++){
                System.out.print(a[i]+"\t");}
}}
```

Ekran Çıktısı

```
Dizi boyutunu giriniz
5
a dizisinin sıralanmamış elemanları
83    52    19    68    31
a dizisinin sıralı elemanları
19    31    52    68    83
```

Hızlı Sıralama (Quick Sort) Algoritması

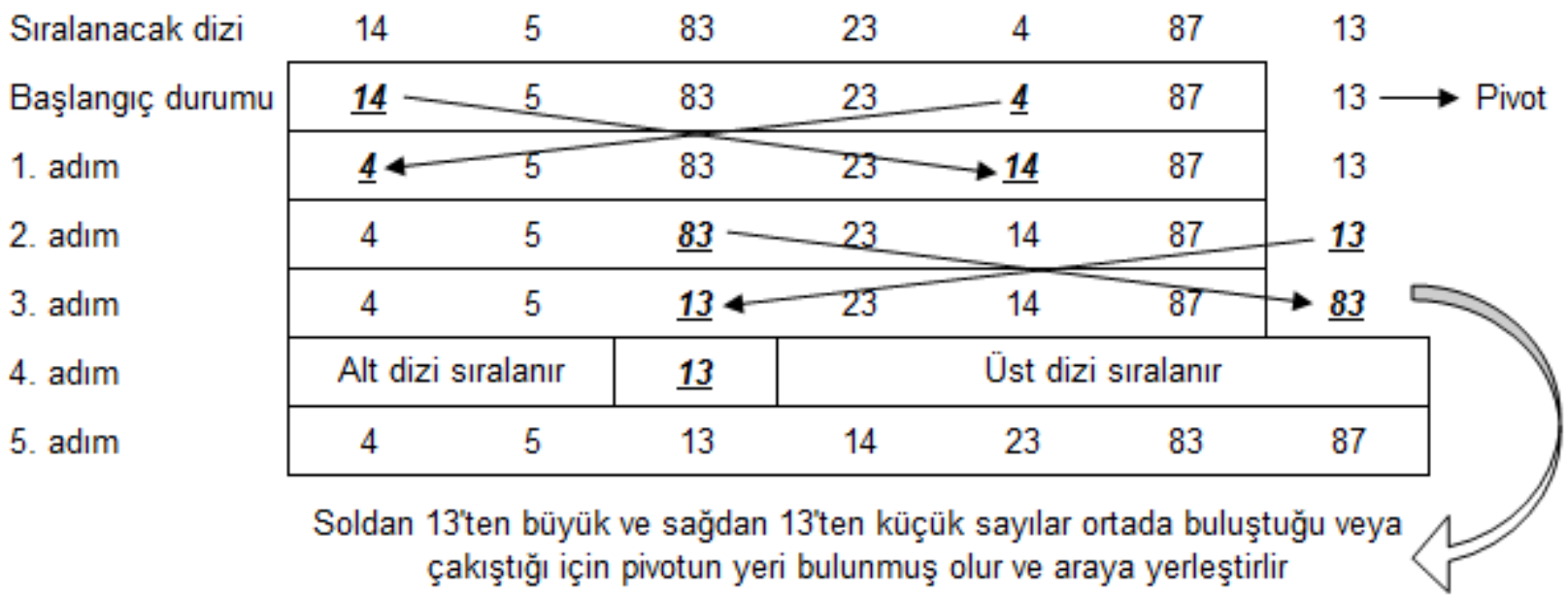
Hızlı sıralama (Quick Sort) algoritmasını kullanarak dizi elemanlarını artan şekilde sıralamak için gerekli işlem basamakları:

İlk olarak sıralanacak diziyi ikiye bölmek için Pivot (karşılaştırma değeri) seçilir. Pivot genellikle verilen dizinin ilk elemanı ya da son elemanı olabilir. Dizide pivottan büyük elemanlar pivotun sağına(üst), pivottan küçük elemanlar ise pivotun soluna (alt) konur. Pivot ise oluşan bu iki kümenin ortasına (orta) yerleşir. Böylece verilen dizi birbirinden bağımsız olarak iki alt diziye ayrılmış olur.

Hızlı sıralama algoritması bağımsız bu iki alt dizi (ust ve alt) içerisinde de recursive (özyineleme) olarak çağrılır ve bu diziler kendi içerisinde 1. adım tekrarlanarak ikiye ayrılır. Bu işlemler diziler parçalanamayacak duruma gelinceye kadar tekrarlanır.

Alt	Orta	Ust
Pivot elemanından küçük elemanlar	Pivot elemanı	Pivot elemanından büyük elemanlar

Hızlı Sıralama (Quick Sort) Algoritması



Hızlı Sıralama (Quick Sort) Algoritması

Sıralanacak dizinin son sayısını (13) pivot (karşılaştırma) elemanı olarak seçtik bu eleman daha sonraki arama ve yer değiştirme işlemlerine tabi olmaz.

Sol başta pivot elemanından örneğimizde 13'den büyük olan ilk sayı bulunur ve sağ baştan 13'den küçük ilk sayı bulunur ve bu iki sayı yer değiştirilir.

Soldan pivot elemanından büyük ve sağdan pivot elemanından küçük sayılar ortada buluşana kadar 2. Adım tekrarlanır.

3.adımda soldan 13'den büyük ve sağdan 13'den küçük sayılar ortada buluştuğu için 13 ile 83 yer değiştirir ve pivotun solundaki alt dizi ve sağındaki üst dizi kendi içinde aynı teknikle sıralanır.

Arama Algoritmaları
(Searching Algorithm)

Sıralı ya da sırasız listedeki bir elemanın yerinin bulunması işlemine arama denir.

Bir listede ya da dizi içerisinde aranan ifadeye anahtar denir.

En temel arama algoritmaları doğrusal arama (linear search) ve ikili arama (binary search) algoritmalarıdır. Genelde küçük boyutlu ve sırasız verilerin aranmasında doğrusal arama, büyük boyutlu ve sıralı verilerin aranmasında ikili arama algoritmaları tercih edilir.

Doğrusal Arama
(Linear Search)

Doğrusal arama sıralı ya da ardışık arama olarak da isimlendirilir ve bilinen en basit arama algoritmasıdır. Kayıt sayısının az olduğu veri gruplarında arama yaparken doğrusal arama algoritması kullanılabilir.

Arama işlemine genelde dizinin başındaki elemanla başlanır, aranan bulununcaya kadar ya da listede eleman kalmayıncaya kadar devam edilir.

Doğrusal Arama
(Linear Search)

Java Dilinde Kodlanması:

```
int x;
do{
        if (x==a[i]){
            System.out.println(x + " dizisinin " + i + ".eleman olarak bulundu");
            System.exit(0);
        }
        else{
            i=i+1;
        }
}while (i<n);
System.out.println(x +  "sayısı listede yoktur");
```

Örnek: Bir grup sayı içerisinden aranan sayının olup olmadığını bulan programı yazınız.

Çözüm:

```
import java.util.Scanner;
public class Search {
public static void main(String[] args) {
Scanner tara=new Scanner(System.in);
int a[]={6,8,3,7,5,6,1,4};
System.out.println("Aranan sayıyı giriniz");
int aranan = tara.nextInt();
boolean durum=false;
for (int x:a)
        {if (x==aranan)
                {durum=true; break;}
}
if(durum)
        System.out.println("Aranan bulundu!");
else
        System.out.println("Aranan bulunamadı!");
}}
```

Doğrusal Arama
(Linear Search)

Örnek: Bilgisayarın rastgele ürettiği 1 ile 100 arasındaki sayılardan oluşan n elemanlı bir A dizisi içerisinde, aranan sayının, dizinin kaçıncı sırasında olduğunu bulan programı doğrusal arama algoritmasını kullanarak yazınız.

Çözüm:

```
import java.util.Scanner;
public class Dogrusalarama {
public static void main(String[] args) {
Scanner tara=new Scanner(System.in);
int i,n,aranan,sayac=0;
System.out.println("Dizi boyutunu giriniz");
n=tara.nextInt();
int a[]=new int [n];
System.out.println("a dizisinin elemanları");
for (i=0;i<n;i++){
            a[i]=(1+(int)(Math.random()*100));
            System.out.print(a[i]+"\t");}
System.out.println();
System.out.println("Aradığınız sayıyı giriniz");
aranan=tara.nextInt();
do
            {if (aranan==a[sayac])
                        {System.out.println(aranan + " dizisinin " + sayac + ".elemanı olarak bulundu");
                        System.exit(0);}
            else
            sayac=sayac+1;
}while (sayac<n);
System.out.println(aranan + " sayısı listede yoktur");
}}
```

İkili Arama
(Binary Search)

İkili arama algoritması, sayısal ya da string sıralı haldeki veriler üzerinde böl ve yönet tekniği ile çalışan bir algoritmadır.

İkili algoritmasının gerçekleştirilebilmesi için verilerin önceden, artan ya da azalan bir şekilde sıralanması gerekir.

İkili arama algoritmasında, sıralı dizi ortadan ikiye ayrılarak aranan veri bu alt dizilerde aranır. Arama işlemi alt dizilerde tekrarlanarak aranan veri bulunmaya çalışılır.

İkili Arama
(Binary Search)

Java Dilinde Kodlanması:

```
public int ikiliarama (int dizi[], int aranan){
int alt=0;
int ust=dizi.length-1;
while (alt<=ust){
        int orta=(alt+ust)/2;
        if (dizi [orta]==aranan)
                return orta;
        else if (dizi[orta]<aranan)
                alt=orta+1;
        else
                ust=orta-1;
}
return -1;
}
```

İkili Arama
(Binary Search)

Örnek: 0'dan 30'a kadar olan çift sayıların tutulduğu bir A dizi içerisinde, aranan sayının dizinin kaçıncı sırasında olduğunu bulan programı ikili arama algoritmasını kullanarak yazınız.

Çözüm:

```
class Ikiliara{
static int ara(int dizi[ ], int aranan){
int alt=0;
int ust=dizi.length-1;
while (alt<=ust){
                int orta=(alt+ust)/2;
                if (dizi [orta]==aranan)
                                return orta;
                else if (dizi[orta]<aranan)
                                alt=orta+1;
                else
                                ust=orta-1;
}
return -1;}

public static void main(String[] args) {
int a[]= {0,2,4,6,8,10,12,14,16,18,20,22,24,26,28,30};
Scanner tara=new Scanner (System.in);
int i,x;
System.out.println("Aradığınız sayıyı giriniz");
x=tara.nextInt();
i=Ikiliara.ara(a,x);
if(i==-1)
                System.out.println("Aradığınız sayı dizide yoktur");
else
                System.out.println(x + " dizisinin " + i + " .elemanıdır.");
}}
```