8. SIRALAMA VE ARAMA ALGORİTMALARI

Giriş

Sıralama ve arama algoritmaları bilgisayar biliminin en önemli konularının içerisinde sayılabilir. Bu bölümde sıralama ve arama algoritmalarına örnekler verilerek düzensiz (sırasız) durumdaki verilerin nasıl sıralanacağını ve bulunması istenen bir değerin sıralı ve sırasız düzende bulunan veri yapısı içerisinde nasıl aranacağını öğreneceğiz.

Sıralama algoritmaları düzensiz olarak verilmiş olan bir grup sayıyı veya metinsel ifadeyi küçükten büyüğe veya büyükten küçüğe sıralanmasını gerçekleştiren algoritmalardır. Bu amaçla geliştirilmiş çeşitli sıralama algoritmaları bulunmaktadır. Seçerek sıralama (selection sort), Kabarcık Sıralaması (Bubble Sort), Yerleştirmeli Sıralama (Insertion Sort), hızlı sıralama (quick sort), birleştirerek sıralama (merge sort), yığın sıralaması (heap sort) gibi algoritmalar temel sıralama algoritmalarıdır. Bu bölümde konunun daha iyi anlaşılabilmesi için, elemanları sayı türünde veri olan dizilerin sıralanmasına örnekler verilecek ve bu örneklerde Seçerek sıralama (selection sort), Kabarcık Sıralaması (Bubble Sort), Yerleştirmeli Sıralama (Insertion Sort) algoritmaları kullanılacaktır.

Ayrıca, bu ünitede bir veri yapısı içerisinde aranan bir elemanın var olup olmadığının tespiti için kullanılan arama algoritmalarına örnekler verilecektir. Arama algoritmalarının kavranabilmesi için verilen örneklerde iki ayrı temel arama algoritması kullanılacaktır. Kullanılacak bu arama algoritmaları doğrusal arama (sequential search) ve ikili arama (binary search) algoritmalarıdır.

8.1. Seçerek Sıralama (Selection Sort)

Seçerek sıralama algoritması sıralama problemine getirilen en basit çözümdür. Algoritmanın çalışması sırasında dizinin iki bölüme ayrıldığı varsayılır. Bu bölümlerden ilki sıralanmış dizi, ikincisi ise sıralanmamış dizidir. Başlangıçta sıralanmış dizi boş, sıralanmamış dizi ise, dizinin elemanlarının tamamından oluşur.

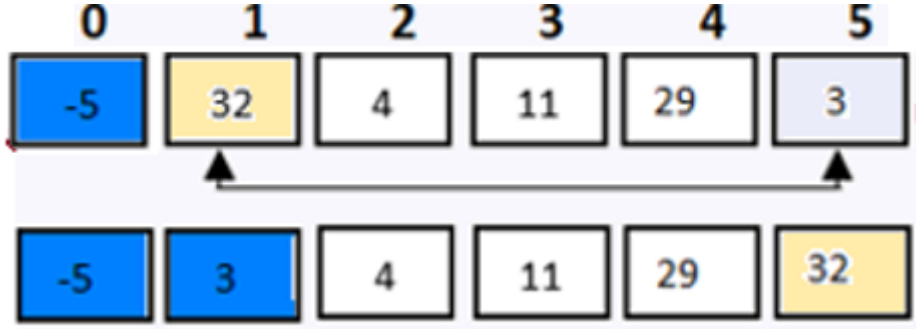
Seçerek sıralama algoritmasının çalıştırılmasına, dizinin ilk elemanının seçilmesi ile başlanır. Seçilen dizi elemanı ile bu elemandan sonra gelen dizi elemanları karşılaştırılır ve en küçük eleman bulunur. Bulunan bu en küçük eleman seçilen eleman ile yer değiştirir. Bu işlemin sonucunda dizinin en küçük elemanı belirlenerek dizinin başına yazılmış olur. Dizinin başına yazılmış olan bu eleman sıralanmış dizinin ilk elemanıdır.

Seçerek sıralama algoritmasının çalışması şekil 8.1'de verilen örnek üzerinde adım adım gösterilmiştir. Şekilden(1. Adım'a bakınız) de görülebileceği gibi başlangıçta dizinin **0**. Elemanı olan **29** seçilmiştir. Sonra, seçilen elemandan sonra gelen dizi elemanları içerisinden en küçük olan eleman bulunmuştur. Örnekte bu eleman **-5**'tir. **-5** dizinin dördüncü elemanıdır.

İlk adımda bulunan en küçük sayı **(-5)**, dizinin **0**'ıncı elemanı **29** ile yer değiştirir ve dizinin elemanlarının sıralaması aşağıdaki gibi olur.

Yukarıdaki işlemden sonra dizinin **1.** Elemanı olan **32** seçilir ve **32**'den sonra gelen dizi elemanları içerisinden en küçük eleman bulunur. Örnekte bu eleman **3**'tür. **3** dizinin **5**. elemanıdır.

Bu adımda bulunan en küçük sayı **(3)** dizinin **5**. elemanı **32** ile yer değiştirir ve dizinin elemanlarının sıralaması aşağıdaki gibi olur.

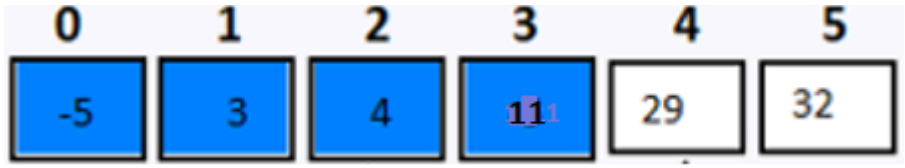


Üçüncü adımda dizinin **2**. elemanı olan **4** seçilir ve **4**'den sonra gelen dizi elemanları içerisinden en küçük eleman araştırılır. Örnekte 4'ten sonra 4'ten küçük eleman yoktur. Dolayısı ile bu adımda dizi elemanları yer değiştirmez. Bu adımdan sonraki dizi elemanlarının sıralı (mavi renkte olan kısım) ve sıralı olmayan kısmı (beyaz renkte olan kısım) aşağıda gösterilmiştir.

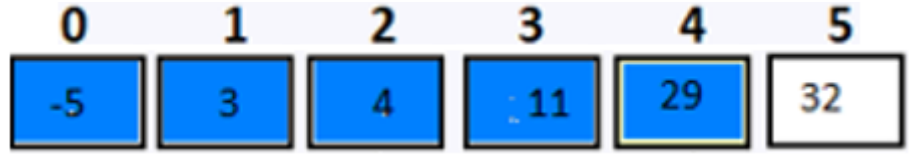
Bundan sonraki dördüncü ve beşinci adımlarda da üçüncü adıma benzer şekilde seçilen elemandan sonraki dizi elemanları seçilen elemandan daha küçük olmadığı için dizi elemanları arasında herhangi bir yer değişikliği işlemi yapılmamıştır.

Aşağıda bu adımlar sonunda dizinin durumu gösterilmiştir.

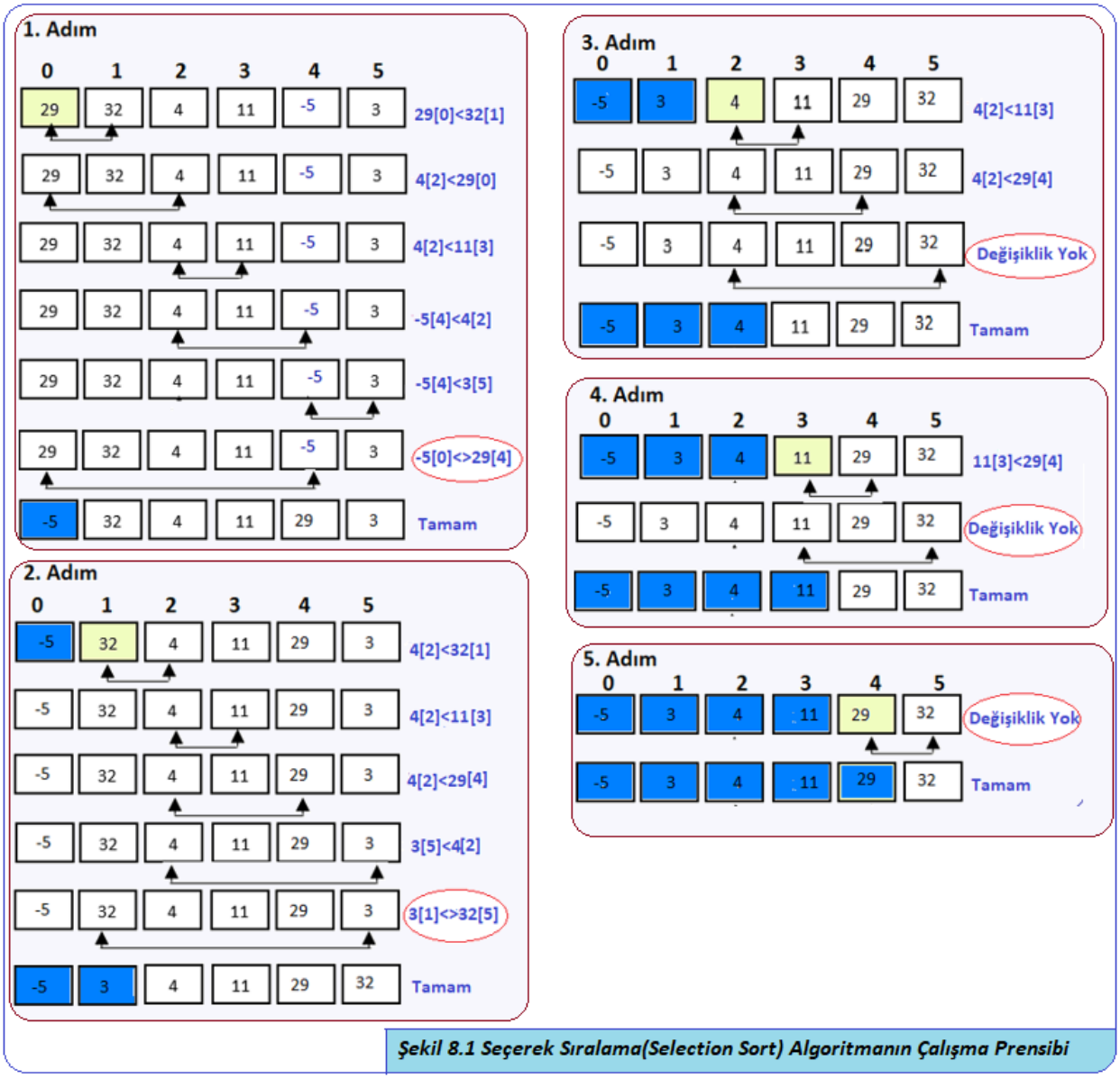
4. Adım sonunda dizi:



5. Adım Sonunda dizi:

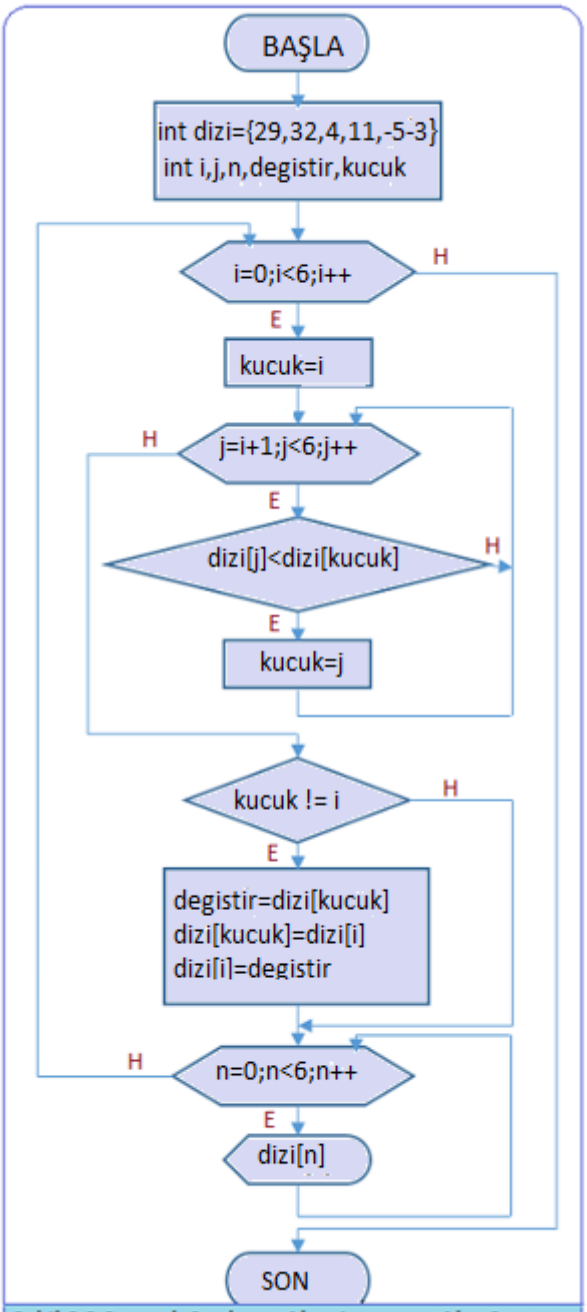


Şekil 8.1'de Seçerek Sıralama (Selection Sort) Algoritmasının çalışma şekli bütün olarak gösterilmiştir.



Şekil 8.1 Seçerek Sıralama(Selection Sort) Algoritmanın Çalışma Prensibi

Şekil 8.2'de Seçerek Sıralama (Selection Sort) algoritmasının akış şeması gösterilmiştir. Akış şeması dikkatli olarak incelendiğinde verilen dizinin elemanlarını sıralayabilmek için algoritmada iç içe çalışan iki adet for döngüsü kullanıldığı görülmektedir. Algoritmanın sonundaki for döngüsünün ise, sıralanmış diziyi ekrana yazdırmak için kullanıldığına dikkat ediniz.



Şekil 8.2 Seçerek Sıralama Algoritmasının Akış Şeması

Aşağıda Seçerek sıralama algoritmasının C++ kodu ve ekran çıktısı verilmiştir.

```cpp
#include <iostream>
using namespace std;
int main() {
    int dizi[6]={29,32,4,11,-5,3};
    int i,j,n,degistir,kucuk;
    for(i=0;i<6;i++){
        kucuk=i;
        for(j=i+1;j<6;j++){
            if(dizi[j]<dizi[kucuk]){
                kucuk=j;
            }
        }
        if(kucuk!=i){
            degistir=dizi[kucuk];
            dizi[kucuk]=dizi[i];
            dizi[i]=degistir;
        }
            for(n=0;n<6;n++)
            cout<<dizi[n]<<" ";
            cout<<endl;
    }
    return 0;
}
```

Program 8.1 Seçerek Sıralama Algoritmasının C++ Kodu

```
-5 32 4 11 29 3
-5 3 4 11 29 32
-5 3 4 11 29 32
-5 3 4 11 29 32
-5 3 4 11 29 32
-5 3 4 11 29 32
```

Program 8.1'in Ekran Çıktısı

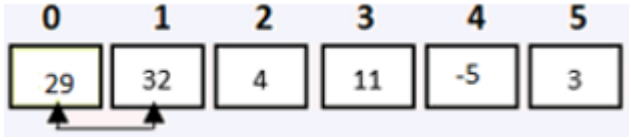
1 **Sıra Sizde!** Yukarıda verilen ve Seçerek sıralama yöntemi ile küçükten büyüğe sıralan dizinin **büyükten küçüğe** sıralanmasını sağlayan algoritmanın akış şemasını çiziniz ve C++ kodunu yazınız?

8.2. Kabarcık Sıralaması (Bubble Sort)

Kabarcık Sıralaması algoritması ilk önce Yer Değiştirme (Exchange Sort) algoritması olarak isimlendirilmiş, daha sonra dizi içerisindeki büyük elemanların algoritmanın her adımında dizi sonuna kadar doğrusal olarak ilerlemesinden dolayı algoritmaya Kabarcık Sıralaması (Bubble Sort) denmiştir.

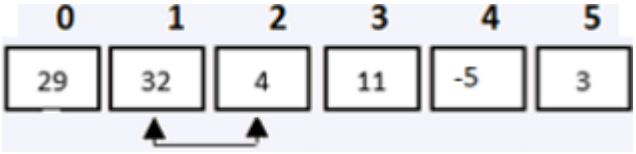
Nabiyev, V. (2016). Algoritmalar. Ankara: Seçkin.

Kabarcık sıralaması algoritmasında dizinin küçükten büyüğe sıralanması aşağıdaki şekilde gerçekleşir; Önce dizinin ilk elemanı (dizinin 0. elemanı) seçilir ve bu eleman kendisinden sonra gelen elemanla (dizinin 1. Elemanı) karşılaştırılır. Karşılaştırma sonunda ilk eleman, kendisinden sonra gelen elemandan büyük ise ilk eleman ile kendisinden sonra gelen eleman yer değiştirir. İlk eleman kendisinden sonra gelen elemandan küçük ise yer değiştirme işlemi yapılmaz.

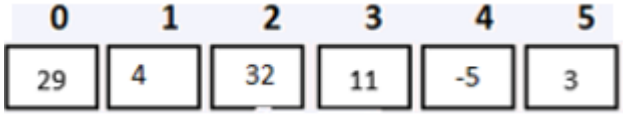


Bu işlemden sonra dizinin birinci elemanı dizinin ikinci elemanı ile karşılaştırılır. Karşılaştırma sonucunda birinci eleman ikinci elemandan büyük ise birinci eleman ile ikinci eleman yer değiştirir. Eğer dizinin birinci elemanı dizinin ikinci elemandan daha küçük ise yer değiştirme işlemi yapılmaz.

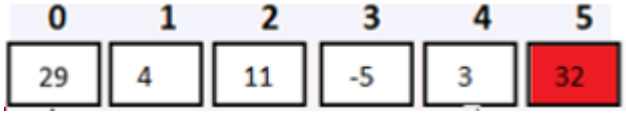
Verilen örnekte dizinin birinci elemanı (32) kendisinden sonra gelen elemandan büyüktür. Dolayısı ile dizinin birinci elemanı ile kendisinden sonra gelen eleman yer değiştirecektir.



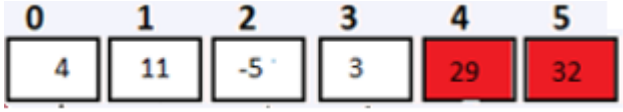
Yer değiştirme işleminin sonunda dizi elemanlarının sıralanışı aşağıdaki gibi olacaktır.



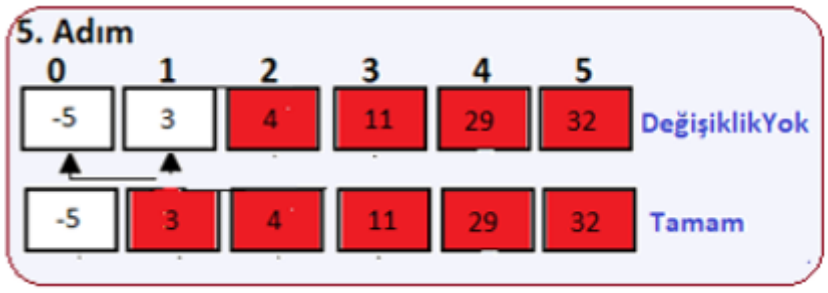
Kabarcık sıralaması yöntemi ile sıralama işleminde, yukarıdaki işlemler dizi sonuna kadar tekrarlanır. 1. Adımın sonunda dizinin en büyük elemanı dizinin sonuna yerleştirilmiş olur (Bakınız Şekil 8.2, 1. Adım). Bir sonraki adımda karşılaştırma işleminin sınırı bir eleman geri çekilir.



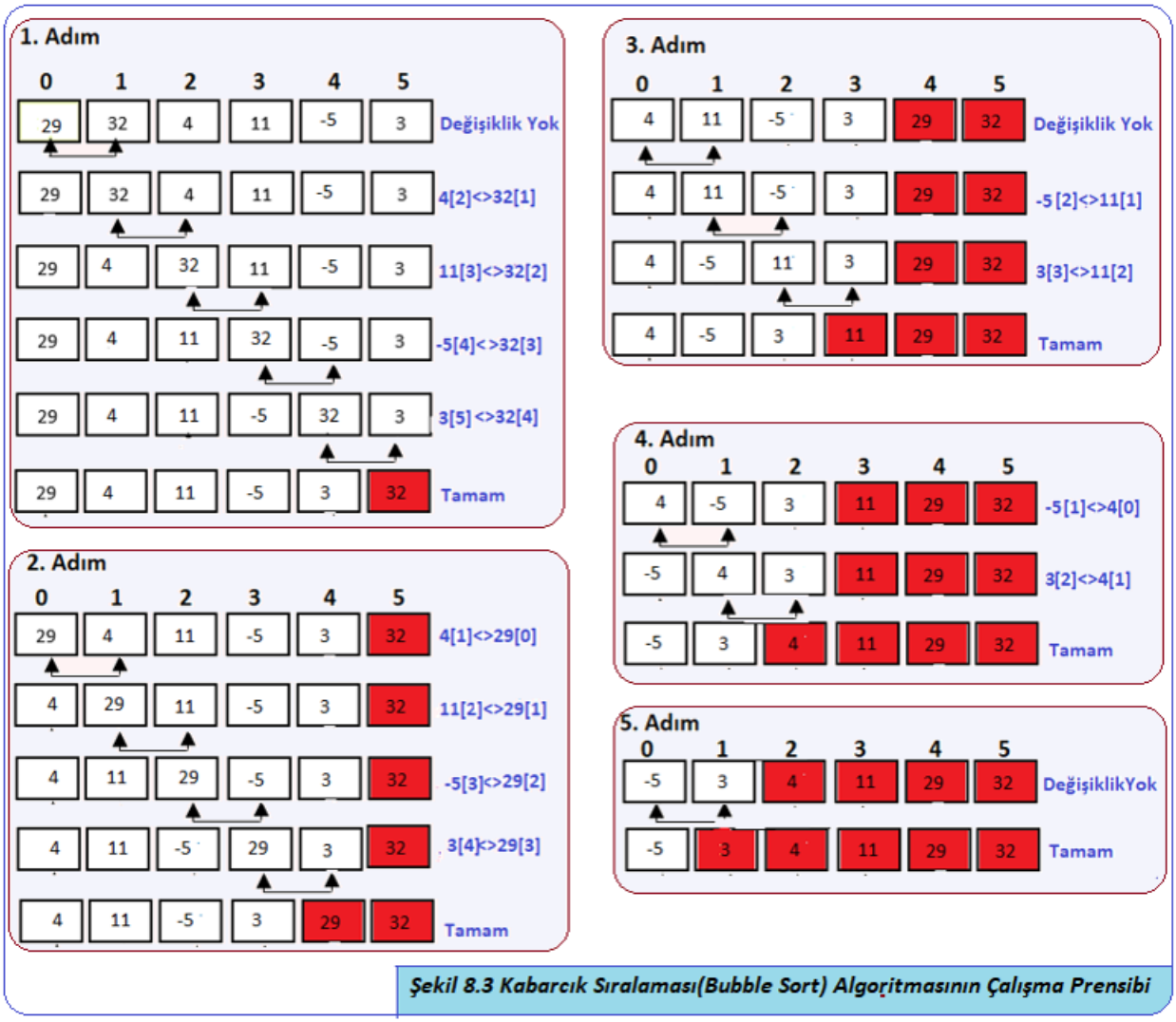
İkinci adımda karşılaştırma işlemine yine dizinin ilk elemanından başlanır ve yukarıda açıklanan kurallara göre ikili karşılaştırmaların sonunda yer değişiklikleri yapılarak dizinin ikinci büyük elemanı 4. elemanın yerine geçer.



Yukarıda açıklanan işlemler, bu örnek için üçüncü adımda, dördüncü ve beşinci adımda tekrarlanarak beşinci adımın sonunda başlangıçta sırasız olarak verilen dizi küçükten büyüğe sıralanmış olacaktır.



Şekil 8.2'de Kabarcık Sıralamasının (Bubble Sort) Algoritmasının çalışma şekli bütün olarak gösterilmiştir.



Şekil 8.3 Kabarcık Sıralaması(Bubble Sort) Algoritmasının Çalışma Prensibi

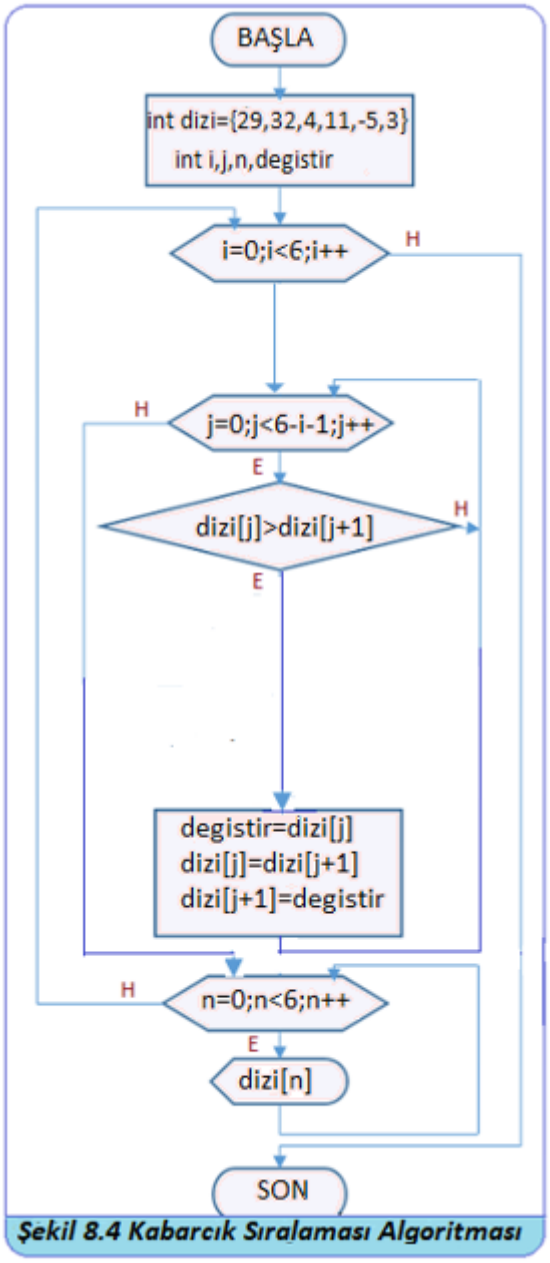
```cpp
#include <iostream>
using namespace std;
int main() {
    int dizi[6]={29,32,4,11,-5,3};
    int i,j,n,degistir;
    for(i=0;i<6;i++){
        for(j=0;j<6-i-1;j++)
        {
            if(dizi[j]>dizi[j+1])
            {
                degistir=dizi[j];
                dizi[j]=dizi[j+1];
                dizi[j+1]=degistir;
            }
        }
        for(n=0;n<6;n++)
            cout<<dizi[n]<<" ";
            cout<<endl;
    }
    return 0;
}
```

Program 8.2 Kabarcık Sıralaması Algoritmasının C++ Kodu

```
29 4 11 -5 3 32
4 11 -5 3 29 32
4 -5 3 11 29 32
-5 3 4 11 29 32
-5 3 4 11 29 32
-5 3 4 11 29 32
```

Program 8.2'nin Ekran Çıktısı

Yukarıda Kabarcık sıralamasının C++ kodu ve kodun ekran çıktısı verilmiştir. Aşağıda ise Kabarcık sıralamasının akış şeması verilmiştir. Akış şeması incelendiğinde kabarcık sıralamasının da seçerek sıralamada olduğu gibi iç içe çalışan iki adat for döngüsü ile gerçekleştirildiği görülmektedir. Akış şemasında sonda yer alan for döngüsünün sıralanmış diziyi ekrana yazdırmak için kullanıldığına dikkat ediniz.



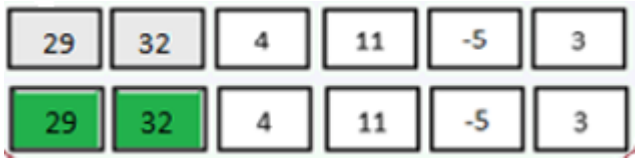
Şekil 8.4 Kabarcık Sıralaması Algoritması

2 **Sıra Sizde!** Yukarıda verilen ve kabarcık sıralaması yöntemi ile küçükten büyüğe sıralan dizinin, aynı yöntemle **büyükten küçüğe** sıralanmasını sağlayan algoritmanın akış şemasını çiziniz ve C++ kodunu yazınız?

8.3. Yerleştirmeli Sıralama (Insertion Sort)

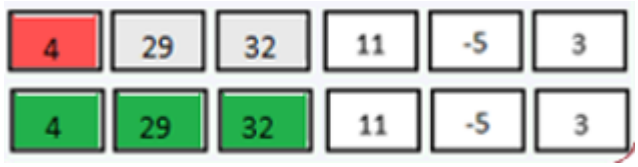
Yerleştirmeli sıralama algoritmasının çalıştırılmasına dizinin ikinci elemanı seçilerek başlanır. Seçilen ikinci eleman kendinden önceki elemanlarla karşılaştırılır ve dizinin büyük elemanları sağa kaydırılır. Bu kaydırma sonucunda açılan uygun pozisyona o anda sıralanmakta olan eleman yerleştirilir. Şekil 8.5'te 6 elemanlı sıralı olmayan bir dizi verilmiştir ve bu dizinin elemanlarının yerleştirmeli sıralama yöntemi ile sıralanması adım adım gösterilmiştir.

İlk adımda dizinin 1. elemanı (32) seçilmiş ve kendisinden gelen önceki elemanla (29) karşılaştırılmış 29 32'den küçük olduğu için dizi elemanlarının yerlerinde herhangi bir değişiklik yapılmamıştır.



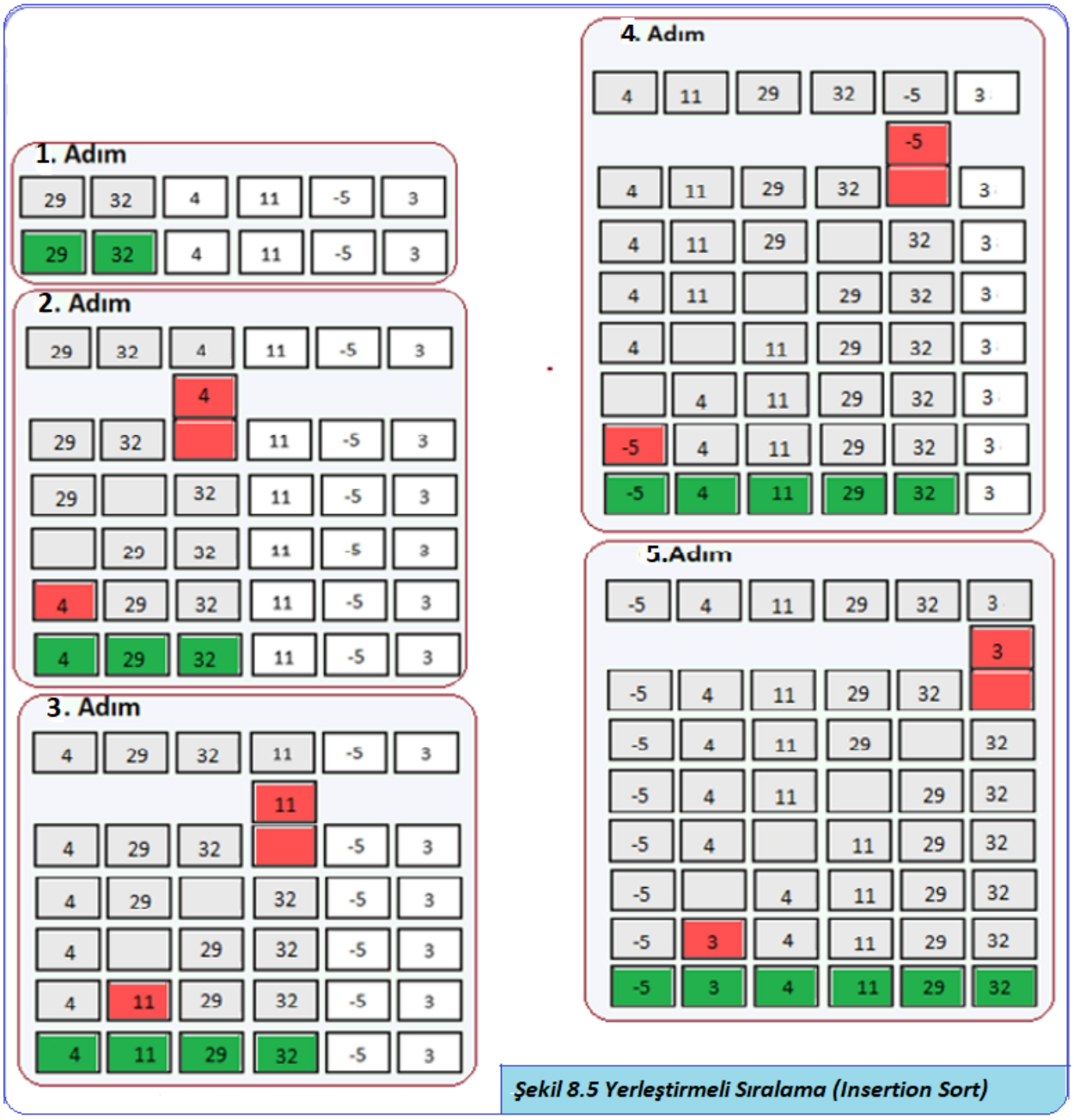
İkinci adımda dizinin 2. Elemanı (4) seçilmiş ve kendisinden önce gelen dizinin 1. ve 0. elemanları ile karşılaştırılmıştır.

Karşılaştırma sonucunda önce dizinin 1. elemanı olan 32, dizinin ikinci elemanı olan 4'ün yerine, sonra da dizinin 0. elemanı olan 29, dizinini birinci elemanı olan 32'nin yerine gelmiştir. Seçilmiş olan dizi elemanı bu durumda dizinin 0. elemanı olmuştur. İkinci adımın sonunda dizinin sıralaması aşağıdaki gibi olmuştur.

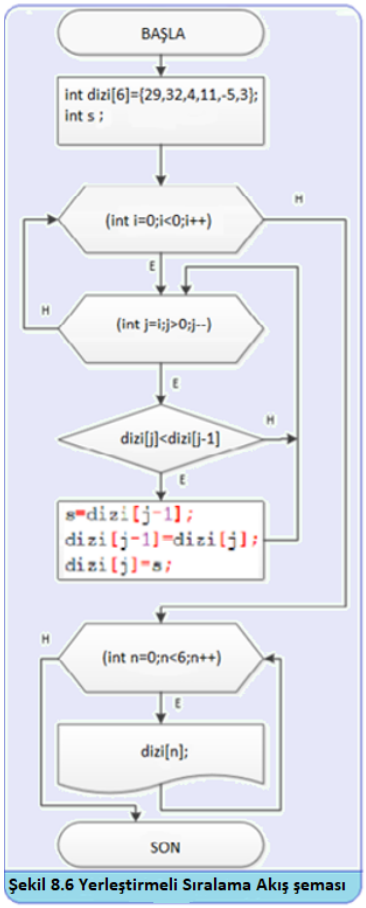


Bu işlemler aynı kurallarla üçüncü, dördüncü ve beşinci adımlarda sürdürülerek beşinci adımın sonunda dizi küçükten büyüğe sıralanmıştır.

Şekil 8.3'de Yerleştirmeli Sıralama (Insertion Sort) Algoritmasının çalışma şekli bütün olarak gösterilmiştir.



Şekil 8.5 Yerleştirmeli Sıralama (Insertion Sort)



Şekil 8.6 Yerleştirmeli Sıralama Akış şeması

```
#include <iostream>
using namespace std;
int main() {
    int dizi[6]={29,32,4,11,-5,3},s,n;
    for(n=0;n<6;n++)
        cout<<dizi[n]<<" ";
        cout<<endl;
    for(int i=1;i<6;i++){
        for(int j=i;j>0;j--){
            if(dizi[j]<dizi[j-1])
            {
                s=dizi[j-1];
                dizi[j-1]=dizi[j];
                dizi[j]=s;
            }
        /*  else
            {
                break;
            } */
        }
        for(n=0;n<6;n++)
            cout<<dizi[n]<<" ";
            cout<<endl;
    }
    return 0;
}
```

Program 8.3 Yerleştirmeli Sıralama C++ Kodu

```
29 32 4 11 -5 3
29 32 4 11 -5 3
4 29 32 11 -5 3
4 11 29 32 -5 3
-5 4 11 29 32 3
-5 3 4 11 29 32
```

Program 8.3'ün Ekran Çıktısı

Şekil 8.6'de Yerleştirmeli Sıralama (Insertion Sort) algoritmasının akış şeması gösterilmiştir. Akış şeması incelendiğinde seçerek sıralama ve kabarcık sıralamasında olduğu gibi yerleştirmeli sıralamada da verilen dizinin elemanlarını sıralayabilmek için algoritmada iç içe çalışan iki adet **for** döngüsü kullanılmıştır. Algoritmanın sonundaki **for** döngüsün bu örnekte de, sıralanmış diziyi ekrana yazdırmak için kullanılmıştır.

Program 8.3 Yerleştireli sıralama algoritmasının C++ koduna Örnek olarak verilmiştir.

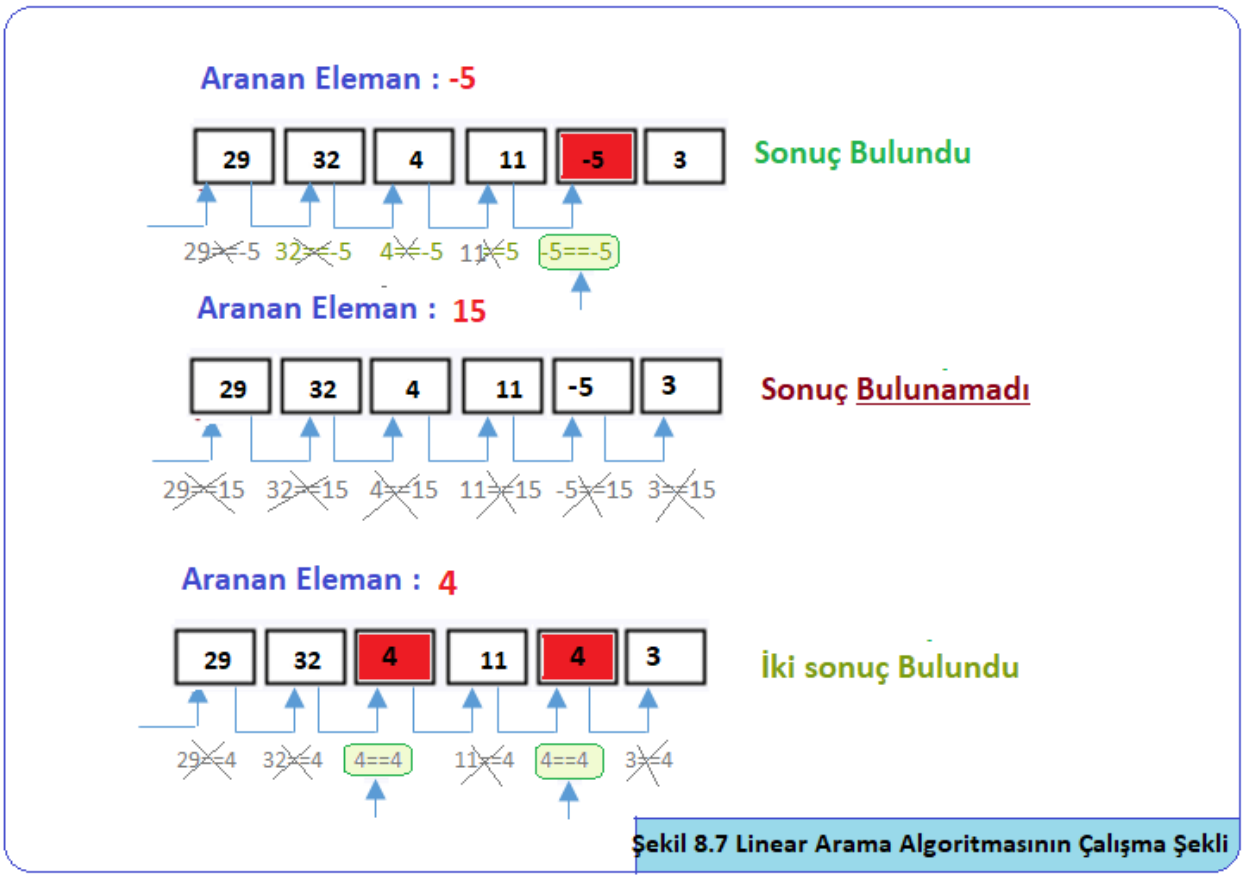
3 **Sıra Sizde!** Yukarıda verilen ve yerleştirmeli sıralama (Insertion Sort) yöntemi ile küçükten büyüğe sıralan dizinin, aynı yöntemle **büyükten küçüğe** sıralanmasını sağlayan algoritmanın akış şemasını çiziniz ve C++ kodunu yazınız?

8.4. Doğrusal Arama (Sequential Search)

Lineer arama olarak da isimlendirilen bu algoritma en temel arama algoritması olarak bilinir. Doğrusal arama algoritması sıralı veya sıralı olmayan diziler üzerinde arama yapmak için kullanılabilir. Doğrusal aramada aranan eleman, sırayla dizinin elemanları ile karşılaştırılır. Aranan elemanın değerinin dizi elemanlarından birine eşit olduğu noktada algoritmanın çalışması sonlandırılır. Bazı durumlarda aranan eleman dizi

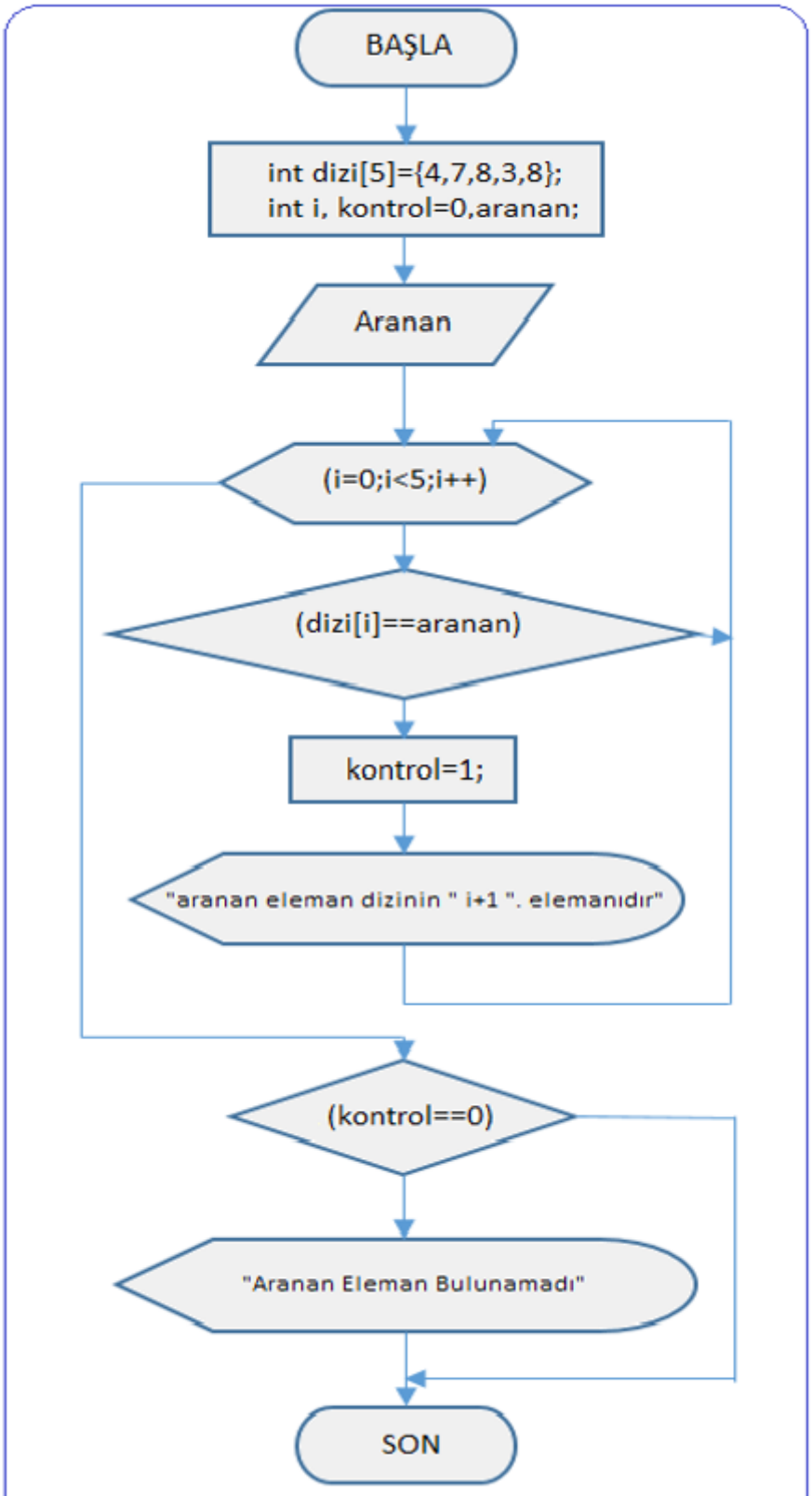
elemanları arasında bulunmayabilir. Bu durumda karşılaştırmalar dizinin ilk elemanından son elemana kadar sürdürülür.

Üzerinde doğrusal arama yapılan dizinin elemanları arasında birden fazla aynı değere sahip eleman varsa ve bunların tamamının konumunun bulunması amaçlanıyorsa bu durumda da karşılaştırma işlemi dizinin ilk elemanından son elemana kadar sürdürülür. Şekil 8.7'de doğrusal Aramanın çalışma şekli gösterilmiştir.



Şekil 8.7 Linear Arama Algoritmasının Çalışma Şekli

Şekil 8.8'de doğrusal aramanın akış şeması, program 8.4'te doğrusal aramanın C++ kodu verilmiştir.



Şekil 8.8 ***Lineer Arama Algoritmasının Akış Şeması***

```cpp
#include <iostream>
#include <locale.h>
using namespace std;
int main() {
    setlocale(LC_ALL,"Turkish");
     int dizi[5]={4,7,8,3,8},i,kontrol=0,aranan;
    cout<<" Aranan Sayıyı Giriniz : ";
    cin>>aranan;
    for(i=0;i<5;i++)
    {
        if(dizi[i]==aranan){
            cout<<"aranan eleman dizide indisi "<<i<<" olan elemandır"<<endl;
            kontrol=1;
            // break;
                }
    }
    if(kontrol==0)
    cout<<"aranan eleman bulunamadı"<<endl;
    return 0;
}
```

Program 8.4 Lineer Arama Algoritmasının C++ kodu

```
 Aranan Sayıyı Giriniz : 3
aranan eleman dizide indisi 3 olan elemandır

--------------------------------
Process exited after 7.068 seconds with return value 0
Press any key to continue
```

Program 8.4'ün Ekran Çıktısı

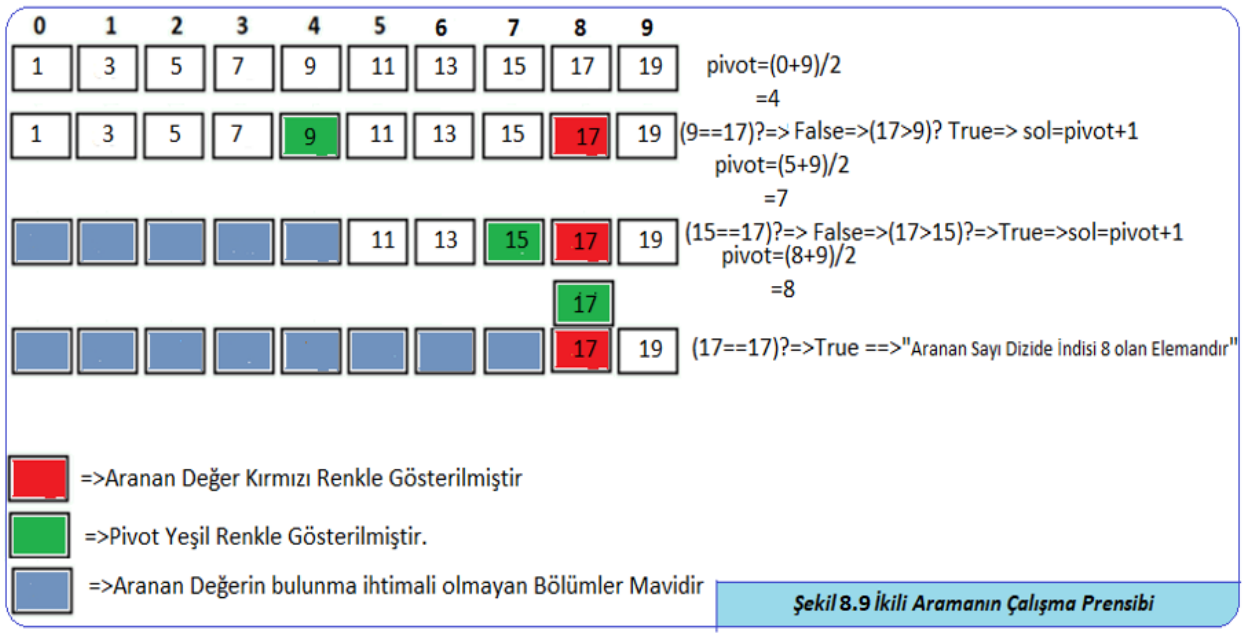
8.5. İkili Arama (Binary Search)

İkili arama algoritması doğrusal arama algoritmasından farklı olarak sadece sıralı diziler üzerinde arama yapan bir algoritmadır. İkili arama parçala fethet prensibi ile çalışır. Sıralı bir dizi üzerinde ikili arama yöntemi ile arama yaparken dizi önce ortadan ikiye bölünür ve aranan değerin ortadaki bu sayıya eşit olup olmadığına bakılır. Eğer bu karşılaştırma sonunda arana değer dizinin ortasındaki değere eşit ise aranan sayı bulunmuştur, arama sonlandırılır. Eşit değil ise aranan değerin ortadaki bu sayıdan büyük veya küçük mü olduğu araştırılır.

Eğer Aranan değer ortadaki sayıdan büyük ise arama ortadaki değerin sağında yapılır. Bunun için ortadaki değerin sağında kalan kısım için yeni bir orta değer bulunur ve arama için ilk adımda açıklanan işlemler tekrarlanır.

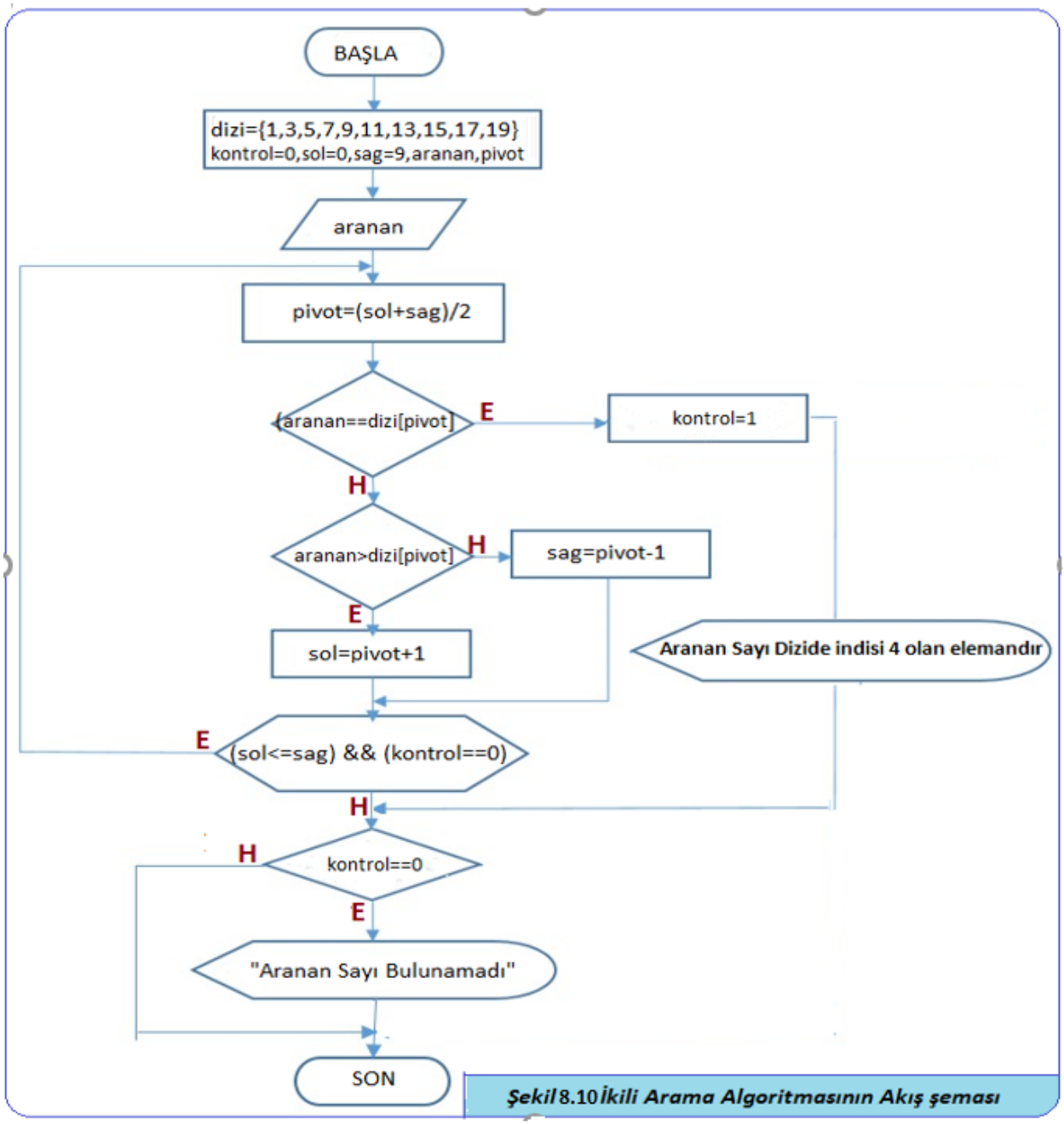
Eğer Aranan değer ortadaki sayıdan küçük ise arama ortadaki değerin solunda yapılır. Bunun için ortadaki değerin sağında kalan kısım için yeni bir orta değer bulunur ve arama için ilk adımda açıklanan işlemler tekrarlanır.

Arama işlemi, dizi üzerinde aranan değer bulunana kadar yukarıda açıklanan prensiplerle sürdürülür. Şekil 8.9'da ikili arama algoritmasının çalışma prensibi gösterilmiştir.



Şekil **8.9** *İkili Aramanın Çalışma Prensibi*

Şekil 8.10'da ikili arama algoritmasının akış şeması verilmiştir.



Şekil **8.10** *İkili Arama Algoritmasının Akış şeması*

İkili arama algoritmasının C++ kodu ve C++ kodunun ekran çıktısı aşağıda verilmiştir.

```
#include <iostream>
#include <locale.h>
using namespace std;
int main ( )
{
    setlocale(LC_ALL,"Turkish");
    int dizi[10]= {1,3,5,7,9,11,13,15,17,19};
    int aranan,kontrol=0,sol=0,sag=9,pivot;
    cout<<"Aranan sayiyi girin: ";
    cin>>aranan;
    do {
        pivot=(sol+sag)/2;
        if(aranan ==dizi[pivot]){
            kontrol=1;
            cout<<"aranan eleman dizide indisi ";
            cout<<pivot<<" olan elemandır"<<endl;
            break;
        }
        else if(aranan>dizi[pivot]){
            sol=pivot+1;}
        else{
            sag=pivot-1;}
    }while ((sol<=sag) && (kontrol==0));
        if (kontrol==0)
        cout<<"Aranan sayı bulunamadı!";
    return 0;
}
```

Program 8.5 İkili Arama Algoritmasının Akış Şeması

Bölüm Özeti

Bu bölümde ilk önce sıralama algoritmalarını ele aldık ve konuyu üç temel sıralama algoritması üzerinden açıkladık. Sırala algoritmalarının nasıl çalıştığını açıklarken önce algoritmanın çalışma prensibini şekil üzerinde gösterdik. Ele alınan sıralama algoritmasının çalışma mantığını öğrenmiş olmak bize hem algoritmanın akış şemasını çizerken hem de C++ kodunu yazarken önemli kolaylıklar sağladı.

Sıralama ve Arama Algoritmaları bölümünde ele alınan ilk sıralama algoritması Seçerek Sıralama algoritması oldu. Seçerek sıralama algoritmasının çalışma prensibinin ilk önce dizinin ilk elemanının ele alınması ve bu elemandan sonra geriye kalan dizi elemanları içerisinden en küçük elemanın bulunup ele alınan elemanla en küçük elemanın yer değiştirmesi, daha sonra dizinin ikinci elemanın ele alınıp işlemlerin bu şekilde sürdürülmesi şeklinde olduğu görüldü.

Ele alınan ikinci sıralama algoritması, Kabarcık sıralaması oldu. Kabarcık sıralaması algoritmasının çalışma prensibinin dizi elemanlarından komşu iki elemanın karşılaştırılması, karşılaştırma sonucunda elemanların uygun yerlerine yerleştirilmesi ve bu işlemin dizi elemanlarının sıralanması tamamlanıncaya kadar sürdürülmesine dayandığı gösterildi.

Sıralama algoritmalarında ele alınan üçüncü ve son algoritma yerleştirmeli sıralama algoritması oldu. Bu algoritmanın çalışma prensibinin ise, dizinin ilk elemanın yerinde bırakılması, dizinin ikinci elemanın seçilmesi, seçilen ikinci elemanın kendinden önceki elemanlarla karşılaştırılması, dizinin büyük elemanlarının sağa kaydırılması, daha sonra dizinin üçüncü sıradaki elemanının seçilmesi ve seçilen elemanın kendinden önce gelen elemanlarla karşılaştırılarak işlemlerin dizi sıralanıncaya kadar sürdürülmesi şeklinde olduğu açıklandı.

Sıralama algoritmalarından sonra bu bölümde ele alınan ikinci önemli ana başlık Arama algoritmaları konusu oldu. Arama algoritmalarından doğrusal arama ve ikili arama algoritmaları ele alınarak diziler üzerinde arama işlemlerinin nasıl yapılacağı bu bölümde anlatılan ikinci önemli konuydu.

Doğrusal arama algoritmasının, aranan elemanın dizinin ilk elemanından başlanarak sırayla karşılaştırılması ve aranan elemana eşit dizi elemanı bulunduğunda aramanın sonlandırılması prensibine dayandığı şekil üzerinde anlatıldı, algoritmanın akış şeması çizildi ve C++ kodu yazıldı.

Bu bölümün son konusu olarak ikili arama algoritması ele alındı. İkili arama algoritması sıralı veriler üzerinde arama yapan bir algoritmadır. Algoritmanın çalışma prensibi, ilk önce dizinin ortasındaki elemanın aranan eleman olup olmadığına bakılır. Eğer aranan eleman dizinin ortasında ise arama işlemi sonlandırılır. Eğer aranan eleman ortadaki eleman değilse aranan veri, ya ortadaki elemanın solunda ya da ortadaki elemanın sağında olabilir. Dolayısıyla bu bilgi kullanılarak dizinin yarısı arama işleminden çıkartılır. Dizinin kalan kısmında yeni bir orta değer hesaplanır ve işlemler bu şekilde, aranan eleman bulunana kadar sürdürülür.

H. Burak Tungut, Algoritma ve Programlama Mantığı, KODLAB Yayın Dağıtım Yazılım ve Eğitim Hizmetleri San. ve Tic. Ltd. Şti, 2019.

Duygu Arbatlı Yağcı, Nesne Yönelimli C++ Programlama Kılavuzu, Alfa Basım Yayım Dağıtım Ltd. Şti, 2016.

G. Murat Taşbaşı, C programlama, Altaş yayıncılık ve Elektronik Tic. Ltd. Şti. 2005.

Muhammed Mastar, Süha Eriş, C++, KODLAB Yayın Dağıtım Yazılım ve Eğitim Hizmetleri San. ve Tic. Ltd. Şti, 2012.

Sabahat Karaman, Pratik C++ Programlama, Pusula Yayınevi,2004

Dr. Rifat Çölkesen, Veri Yapıları ve Algoritmalar, Papatya,2002

Prof. Dr. Vasif Nabiyev, Teoriden Uygulamalara Algoritmalar, Seçkin,2007