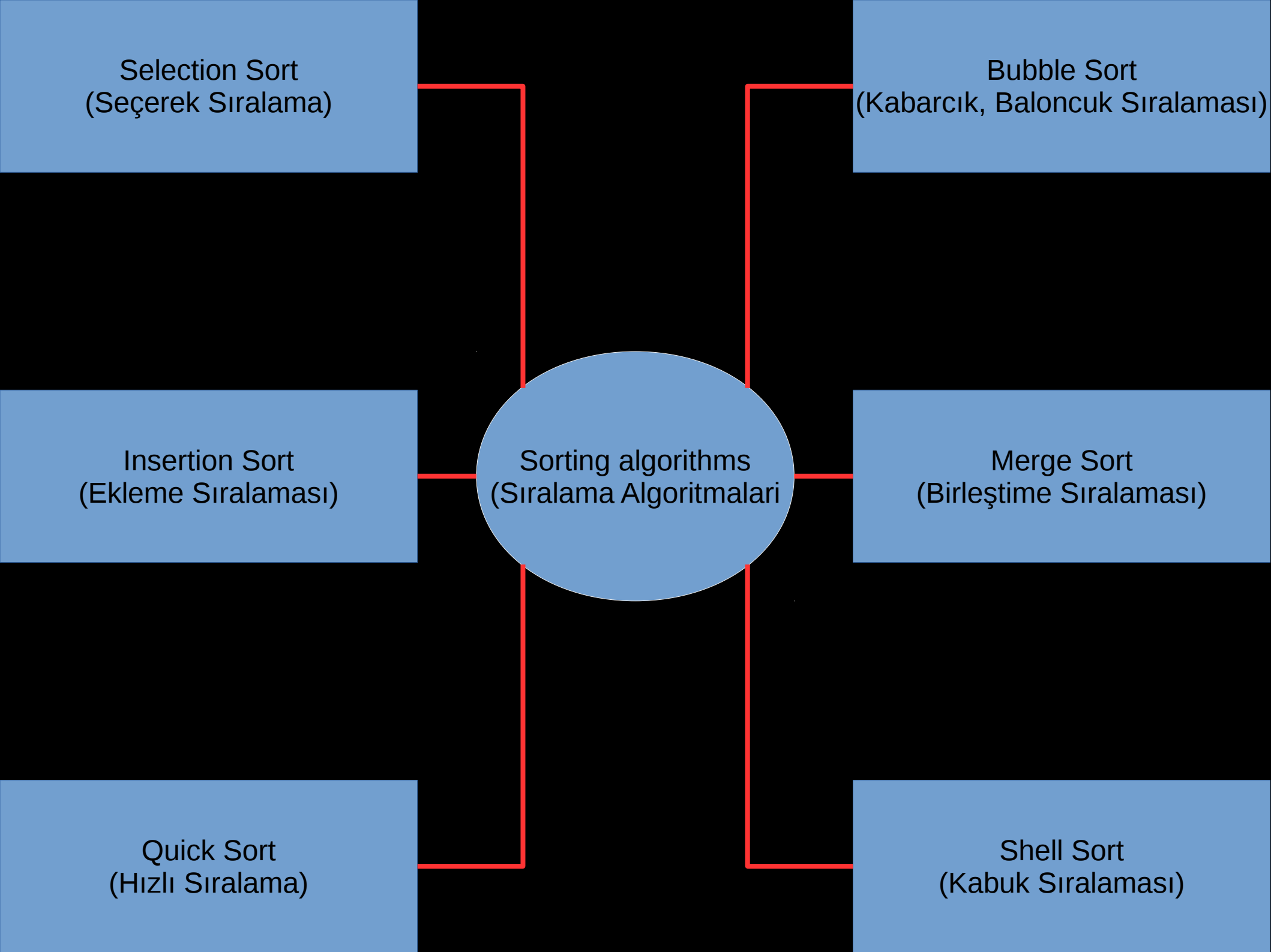
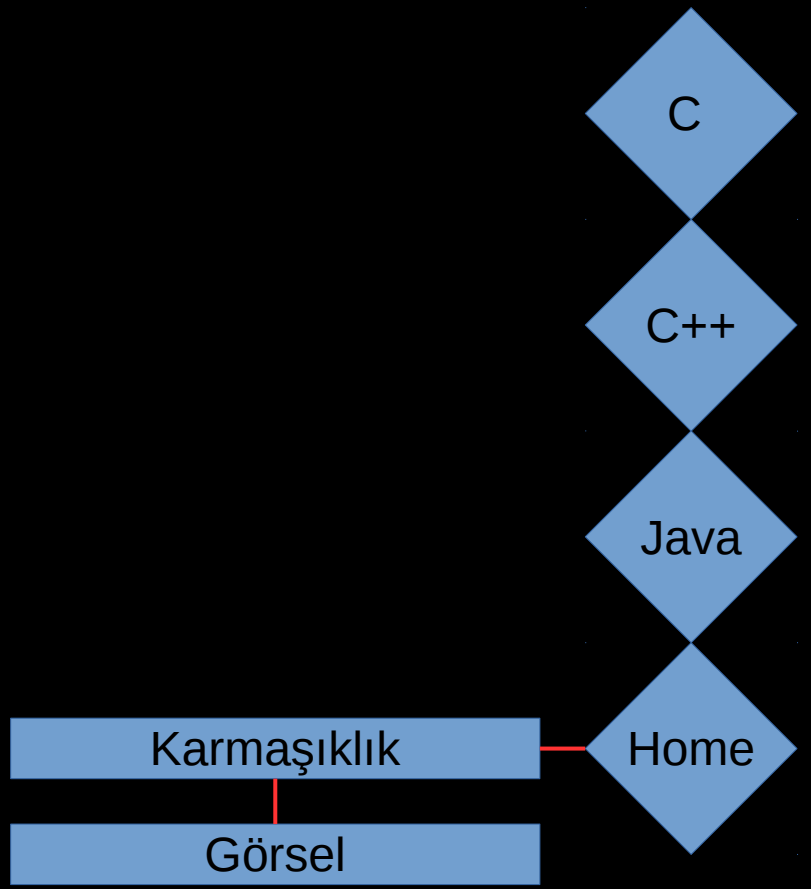
Sıralama Algoritmaları

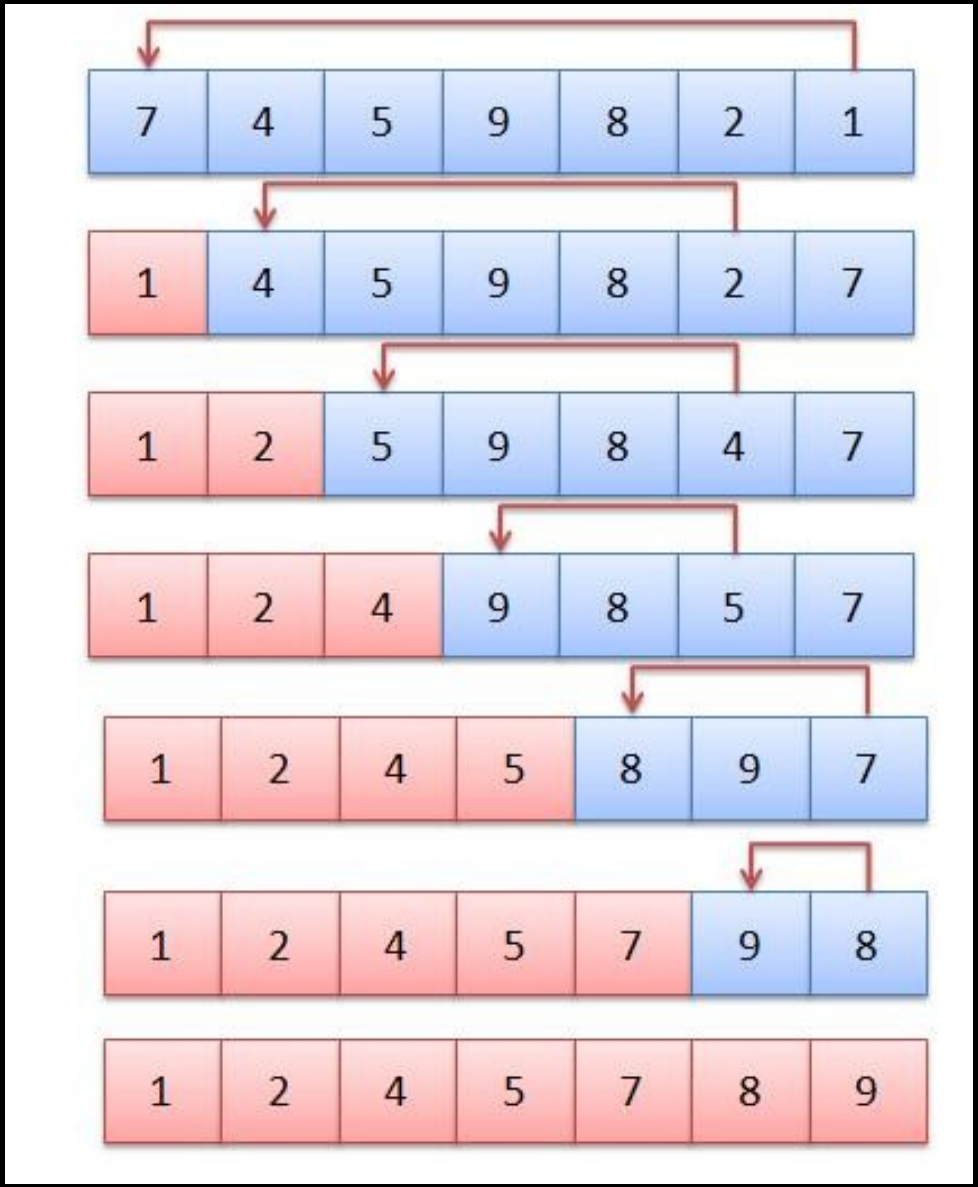

Selection Sort
(Seçerek Sıralama)
Bubble Sort
(Kabarcık, Baloncuk Sıralaması)
Insertion Sort
(Ekleme Sıralaması)
Sorting algorithms
(Sıralama Algoritmalari
Merge Sort
(Birleştime Sıralaması)
Quick Sort
(Hızlı Sıralama)
Shell Sort
(Kabuk Sıralaması)

Selection Sort(Seçerek Sıralama)

```
int selectionSort(int A[],int n){
        int temp;
        int min;

        for(int i=0; i<n-1; i++){
                min=i;
                for(int j=i;j<n;j++){
                    if(A[j]< A[min]){
                        min=j;
                    }
                }
                  temp=A[i];
                  A[i]=A[min];
                  A[min]=temp;
        }

        return 0;
}
```



Selection Sort görsel



Selection Sort karmaşıklık

Bestcase(En iyi durum)	Averagacase(Ortalama durum)	Worstcase(En kötü durum)
n^2	n^2	n^2

C kod

basol@bsl: ~/algoritmalar/selectionSort/cKod

```c
#include <stdio.h>

        int main(){
                int i,j,temp,min;
                int a=6;
                int dizi[a]={6,4,5,1,3,2};

                for(i=0; i<a-1; i++){
                        min=i;
                 for(j=i; j<a; j++){
                        if(dizi[j]<dizi[min]){
                                min=j;
                        }
                                temp=dizi[i];
                                dizi[i]=dizi[min];
                                dizi[min]=temp;
                                }
                }

                for(i=0; i<6; i++){
                        printf("%d\n",dizi[i]);
                }

                return 0;


}
```

basol@bsl: ~/algoritmalar/selectionSort/cKod

```
basol@bsl:~/algoritmalar/selectionSort/cKod$ ./a.out
1
2
3
5
4
6
basol@bsl:~/algoritmalar/selectionSort/cKod$
```

C++ Kod

Uçbirim

```
#include <iostream>
using namespace std;

int selectionSort(int A[],int n){
        int temp;
        int min;

        for(int i=0; i<n-1; i++){
                min=i;
                for(int j=i;j<n;j++){
                  if(A[j]< A[min]){
                        min=j;
                    }
                }
                  temp=A[i];
                  A[i]=A[min];
                  A[min]=temp;
        }

        return 0;
}
        int main(){
                int A[]={6,4,5,1,3,2};
                selectionSort(A,6);
                for(int i=0; i<6; i++)
                        cout<<A[i]<<endl;
        }
```

basol@bsl: ~/algoritmalar/selectionSort/c++Kod

```
basol@bsl:~/algoritmalar/selectionSort/c++Kod$ ./a.out
1
2
3
4
5
6
basol@bsl:~/algoritmalar/selectionSort/c++Kod$
```

Java kod

Uçbirim

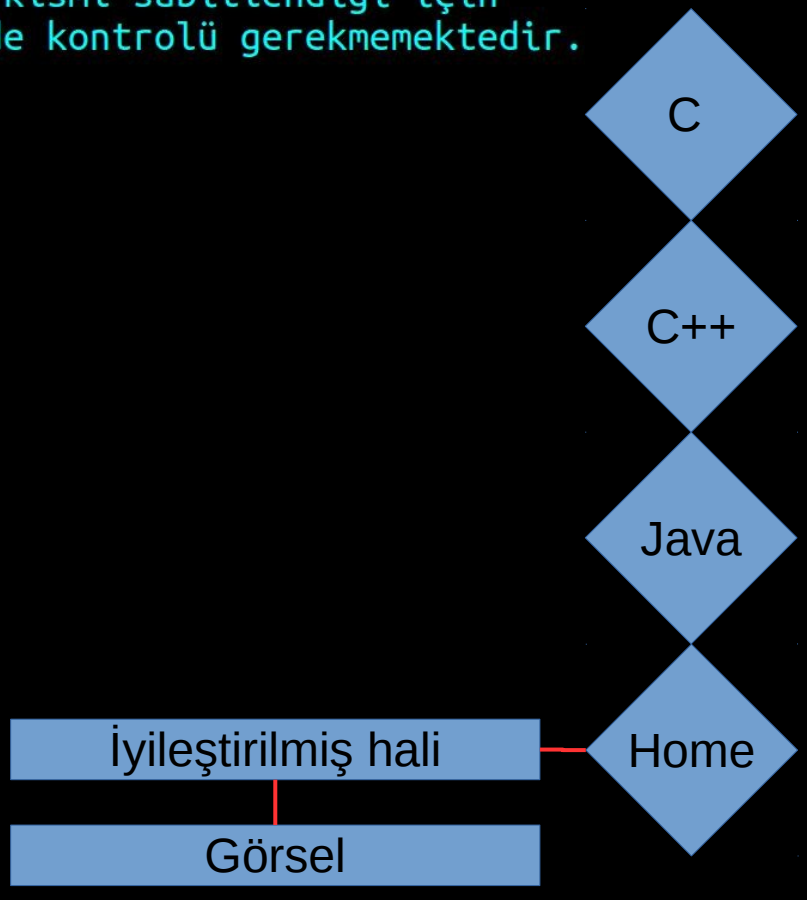
```
public class Siralama{
public static int [] selectionsort(int [] A,int n)
  {
    int tmp;
    int min;

    for(int i=0; i < n-1; i++)
    {
      min=i;

      for(int j=i; j < n; j++)
      {
        if (A[j] < A[min]){

          min=j;
        }

      }
        tmp=A[i];
        A[i]=A[min];
        A[min]=tmp;
    }
    return A;
  }
    public static void main(String args[]){
        int a[] = {6,4,2,3,1,5};
        a = selectionsort(a,6);
        for(int i = 0;i<6;i++)
            System.out.println(a[i]);
    }
}
```

basol@bsl: ~/algoritmalar/selectionSort/javaKod

```
basol@bsl:~/algoritmalar/selectionSort/javaKod$ java Siralama
1
2
3
4
5
6
basol@bsl:~/algoritmalar/selectionSort/javaKod$
```

Bubble Sort(Kabarcık,Baloncuk Sıralaması)

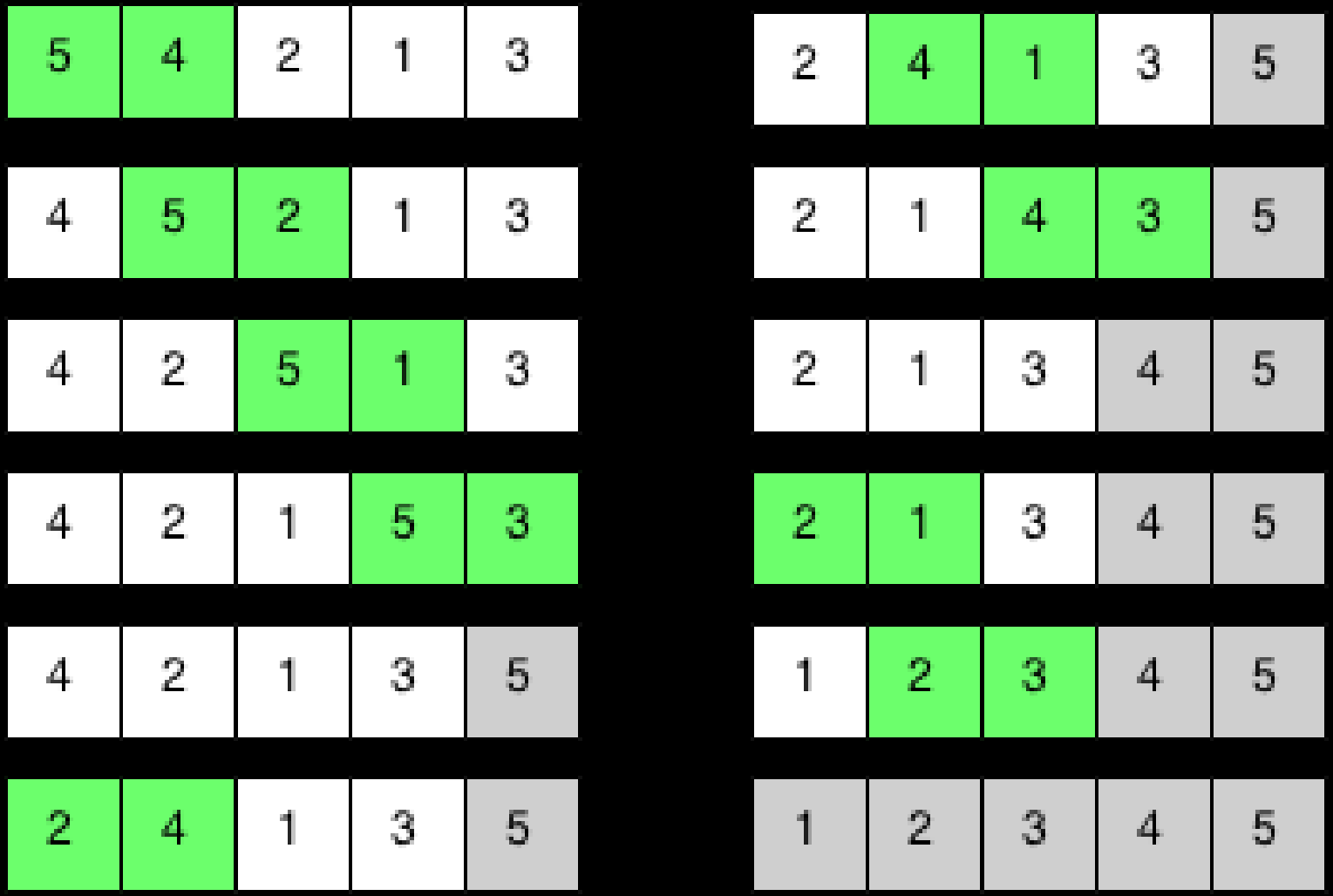
```
void bubblesort(int A[])
{

     int tmp;

    for(int i=0; i<A.length; i++)
    {
        for(int j=A.length-1 ; j>i;j--) //i'ye kadar olan kısmı sabitlendiği için
                                        //tekrar geçişlerde kontrolü gerekmemektedir.
        {
        if(A[j-1]>A[j])
        {
          tmp=A[j-1];
          A[j-1]=A[j];
          A[j]=tmp;
        }
      }
    }
}               //bestcase: n'kare
                //averagacase: n'kare
                //worstcase: n'kare
```



Bubble Sort iyileştirilmiş

```
void bubblesort(int A[])
  {
     int tmp;

    for(int i=0; i<A.length; i++)
    {
    int sirali=1;
      for(int j=A.length-1 ; j>0;j--)
      {
        if(A[j-1]>A[j]) // buraya girmiyorsak dizi sıralı demektir
        {
          sirali=0;
          tmp=A[j-1];
          A[j-1]=A[j];
          A[j]=tmp;
        }
      }
      if(sirali)//şayet dizinin üstünden geçtiğimiz halde
                //hiç bir değer yer değiştirmiyorsa
                // dizi sıralıdır döngüden çıkılabilir
         break;
    }
  }             //bestcase: n (siralami kontrolu yapildigi icin)
                //averagacase: n'kare
                //worstcase: n'kare
~
```

Bubble Sort görsel



Bubble Sort

Home

C kod

```c
#include <stdio.h>
#include <conio.h>
int i,j,tasinan,N;
int A[100];

void main(void){
        printf("dizi eleman sayisini giriniz: ");
        scanf("%d",&N);
        for(i=1; i<=N; i++){
                printf("A[%d]: ",i);
                scanf("%d",&A[i]);
                printf("\n");
        }
        for(i=1; i<=N; i++){
                for(j=1;j<N; j++){
                        if(A[j+1]<A[j]){
                                tasinan=A[j];
                                A[j]=A[j+1];
                                A[j+1]=tasinan;
                        }
        }

}
        for(i=1; i<=N; i++){
                printf("%d ",A[i]);
        }
        printf("\n");
}
```

```
basol@bsl: ~/algoritmalar/bubbleSort/cKod
basol@bsl:~/algoritmalar/bubbleSort/cKod$ gcc buuble.c
basol@bsl:~/algoritmalar/bubbleSort/cKod$ ./a.out
dizi eleman sayisini giriniz: 5
A[1]: 40

A[2]: 50

A[3]: 30

A[4]: 20

A[5]: 10

10 20 30 40 50
basol@bsl:~/algoritmalar/bubbleSort/cKod$
```

C++ kod

Uçbirim

```
#include <iostream>
#include <string>

using namespace std;

int bubbleSort(int dizi[]){
        int temp;

        for(int i=0; i<6/*dizi.length*/; i++){
                int sirali=1;
                        for(int j=5/*dizi.length-1*/; j>i; j--){
                          if(dizi[j-1]>dizi[j]){
                                sirali=0;
                                temp=dizi[j-1];
                                dizi[j-1]=dizi[j];
                                dizi[j]=temp;
                                }
                        }
                        if(sirali)
                                break;
                        }
                }
int main(){
        int dizi[]={6,4,5,3,1,2};
        bubbleSort(dizi);
                for(int i=0; i<6/*dizi.length*/; i++)
                        cout<<dizi[i]<<endl;

}
```

Java kod

basol@bsl: ~/algoritmalar/bubbleSort/javaKod

```
class siralama{
  public void bubblesort(int [] A)
  {
     int tmp;

    for(int i=0; i<A.length; i++)
    {
    boolean sirali=true;
      for(int j=A.length-1 ; j>i;j--)
      {

        if(A[j-1]>A[j]) //şayet buraya girmiyorsak dizi sıralı demektir
        {
          sirali=false;
          tmp=A[j-1];
          A[j-1]=A[j];
          A[j]=tmp;
        }
      }
      if(sirali)
              //dizi sirali cikilabiir
         break;
    }
  }
}
```

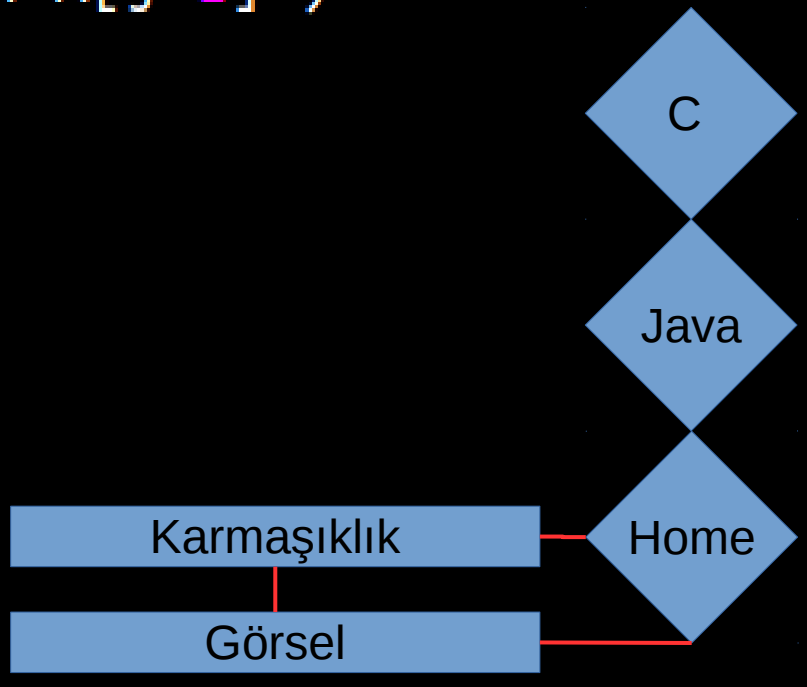
1,1 Tümü

```
public class kabarciksiralama {
    public static void main(String args[]){
                    int [] x = {6,4,5,3,1,2};
                    siralama s = new siralama();
            s.bubblesort(x);
            for(int i : x){
                System.out.println(i);
            }
    }
```

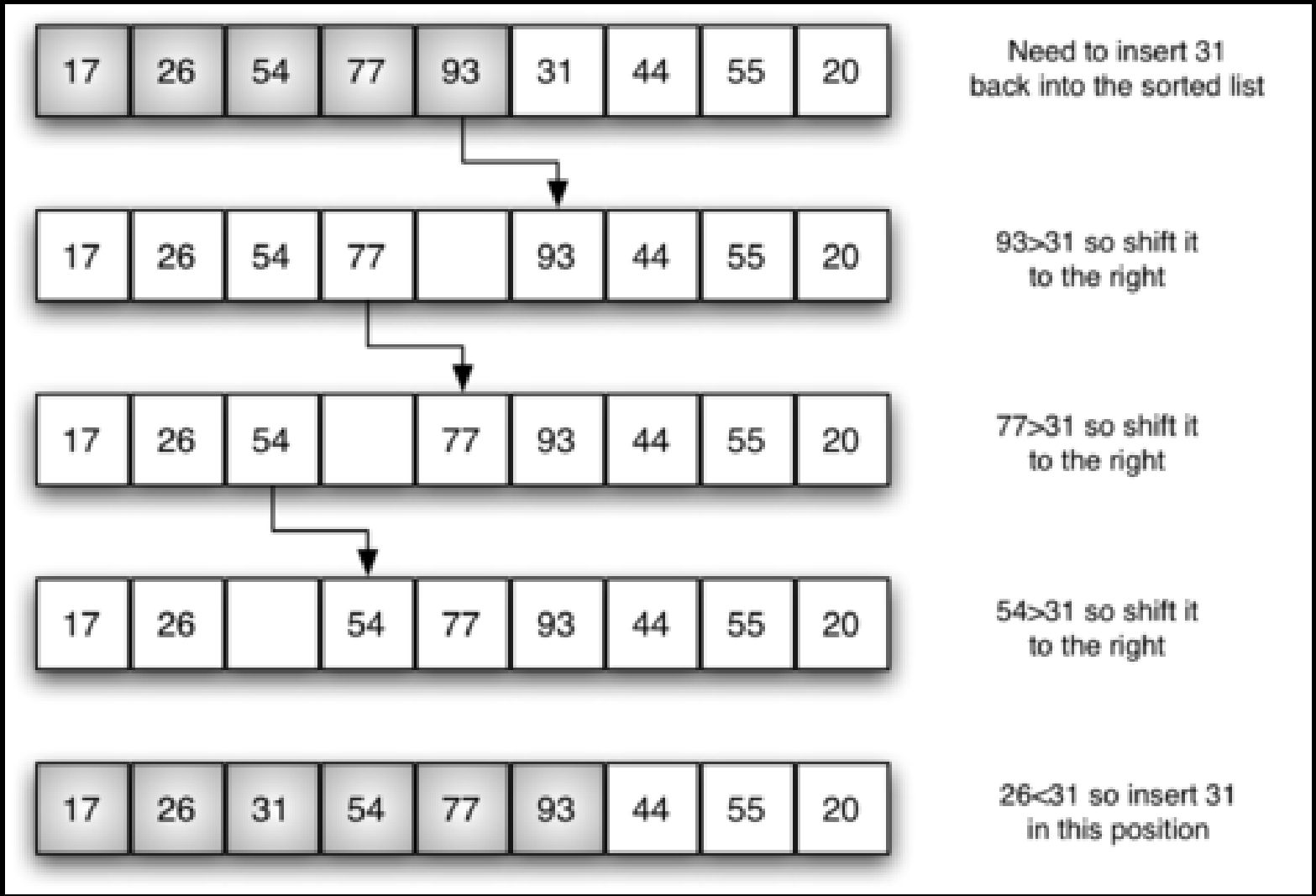
Insertion Sort (Ekleme Sıralaması)

basol@bsl: ~/algoritmalar/insertionSort

```
void insertionSort()
{
        int i,j,tasinan;
        for(i=1; i<=N-1; i++)
        {
                tasinan=a[i];
                j=i;
                while( j>0 && tasinan<A[j-1] )
                {
                        a[j]=a[j-1];
                        j--;
                }
                a[j]=tasinan;
        }
}
```



Insertion Sort görsel



C kod

```c
#include <stdio.h>
#include <conio.h>
int i,j,tasinan,N;
int A[100];

void main(void){
        printf("dizi eleman sayisini giriniz: ");
        scanf("%d",&N);
        for(i=1; i<=N; i++){
                printf("A[%d]: ",i);
                scanf("%d",&A[i]);
                printf("\n");
        }
        for(i=2; i<=N; i++){
                tasinan=A[i];
                j=i;
                while(j>1 && tasinan<A[j-1]){
                        A[j]=A[j-1];
                        j--;
                }
                A[j]=tasinan;
        }
        for(i=1; i<=N; i++){
                printf("%d ",A[i]);
        }
        printf("\n");
}
```

```
basol@bsl: ~/algoritmalar/insertionSort/cKod
basol@bsl:~/algoritmalar/insertionSort/cKod$ gcc insertion.c
basol@bsl:~/algoritmalar/insertionSort/cKod$ ./a.out
dizi eleman sayisini giriniz: 5
A[1]: 45

A[2]: 35

A[3]: 15

A[4]: 25

A[5]: 5

5 15 25 35 45
basol@bsl:~/algoritmalar/insertionSort/cKod$
```

Java kod

```
public class InsertionSort{
 public static void main(String a[]){
        int i;
        int array[] = {12,9,4,99,120,1,3,10};
 System.out.println("Sıralamadan önceki durum:");
        for(i = 0; i < array.length; i++)
                System.out.print( array[i]+" ");
                System.out.println();
                insertion_srt(array, array.length);
                System.out.print("Sıralamadan sonraki durum:\n");
        for(i = 0; i <array.length; i++)
                System.out.print(array[i]+" ");
                System.out.println();
 }
 public static void insertion_srt(int array[], int n){
        for (int i = 1; i < n; i++){
                int j = i;
                int tasinan = array[i];
                while ((j > 0) && (array[j-1] > tasinan)){
                array[j] = array[j-1];
                 j--;
        }
        array[j] = tasinan;
 }
 }}
```

```
basol@bsl: ~/algoritmalar/insertionSort/javaKod
basol@bsl:~/algoritmalar/insertionSort/javaKod$ javac InsertionSort.java
basol@bsl:~/algoritmalar/insertionSort/javaKod$ java InsertionSort
Sıralamadan önceki durum:
12 9 4 99 120 1 3 10
Sıralamadan sonraki durum:
1 3 4 9 10 12 99 120
lar/insertionSort/javaKod$
```

Insertion Sort

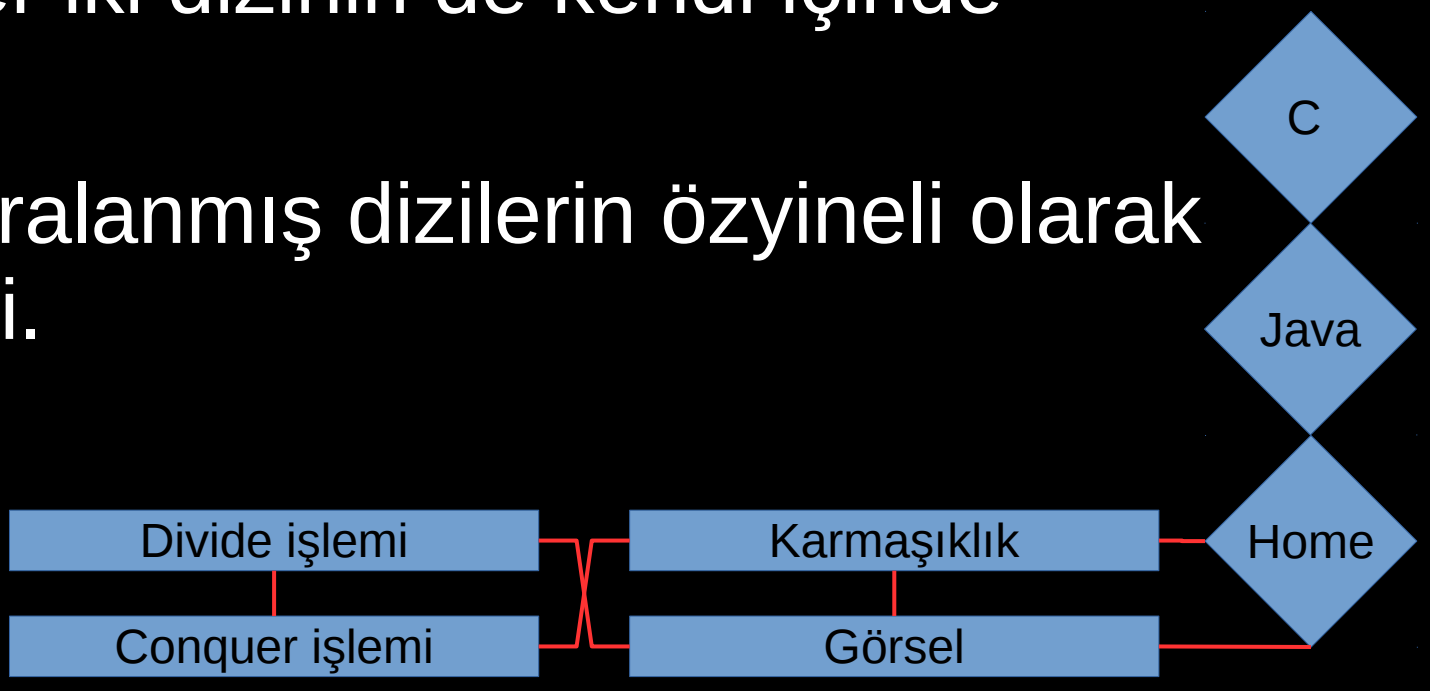
Home

Insertion Sort karmaşıklık

Bestcase(En iyi durum)	Averagacase(Ortalama durum)	Worstcase(En kötü durum)
n	n^2	n^2

Merge Sort(Birleştirme Sıralaması)

- Divide and Conquer(Parçala ve Fethet) yaklaşımı vardır.
- Divide: n elemanlı dizinin n/2 elemanlı iki diziye bölünmesi
- Conquer: Her iki dizinin de kendi içinde sıralanması.
- Combine: Sıralanmış dizilerin özyineli olarak birleştirilmesi.



Divide (mergeSort fonksiyonu)

basol@bsl: ~/algoritmalar/mergeSort

```
public int [] mergesort(int [] m){
     int x=0;
     int y=0;
     int middle=m.length/2;
     int left[] =new int [middle];
     int right[] =new int [middle];
     int result[] =new int[(m.length)];

     if(m.length<= 1){
       return m;
     }

     for(int i=0; i<middle; i++){//sol dizi
       left[x]=m[i];
       x++;
     }
     for(int i=middle; i<m.length; i++){//sag dizi
       right[y]=m[i];
       y++;
     }                    //recursive
     left=mergesort(left);// sol diziyi tekrar mergeSort fonksiyonuna gönderdi
     right=mergesort(right);// sag diziyi tekrar mergeSort fonksiyonuna gönderdi.
     result=merge(left,right);//iki parcayı birlestiren merge fonksiyonu cagrildi
     return result;
   }
```

Conquer (merge fonksiyonu)

basol@bsl: ~/algoritmalar/mergeSort

```
public int [] merge(int []left,int []right) {
   int result[] =new int [left.length + right.length];
     int x=0;
     int y=0;
     int k=0;
// 3 durum var
   while(left.length>x && right.length>y){// Her iki dizide de eleman varsa:
     if(left[x] <= right[y]){ // iki dizideki en bastaki elemanlari karsilastir
                              // kucuk olani sonuc dizisine at.
       result[k]=left[x];
       x++;
       k++;
     }

     else{
        result[k]=right[y];
        y++;
        k++;

      }
   }
// dizilerden birinde eleman kalmamissa
   if(left.length>x){ //sag dizi eleman yok
     while(x < left.length){

     result[k]=left[x]; // elmanlari direk result dizisine at.
     x++;
     k++;

     }
   }

   if(right.length>y){ // sol dizide elaman yoksa
     while(y < right.length){

     result[k]=right[y];// elamanlari direk result dizisine at
     y++;
     k++;
     }
```

C kod

```c
void main(void){
        oku();
        bol(1,n);
        yaz();
}
void oku(void){
        int i;
        printf("dizinin eleman sayisini giriniz: ");
        scanf("%d",&n);
        for(i=1; i<=n; i++){
                printf("dizi elemanini giriniz: ");
                scanf("%d",&A[i]);
        }
}
void bol(int alt,int ust){
        int orta,alts,usts;
        if(alt<ust){
                alts=alt;
                usts=ust;
                orta=(alts+usts)/2;
                bol(alts,orta);
                birles(alts,orta,usts);
        }
}
void birles(int alts,int orta,int usts){
        int i,ass,usb,g;
        ass=orta;
        usb=orta+1;
        while((alts<=ass) && (usb<=usts)){
                if(A[alts]<A[usb])
                        alts++;
                else{
                        g=A[usb];
                        for(i=usb-1; i>=alts; i--)
                                A[i+1]=A[i];
                        A[alts]=g;
                        alts++;
                        ass++;
                        usb++;
                }
        }
}
void yaz(void){
        int i;
        for(i=1;i<=n;i++)
                printf("%d ",A[i]);
                printf("\n");
```

```
basol@bsl: ~/algoritmalar/mergeSort
basol@bsl:~/algoritmalar/mergeSort$ ./a.out
dizinin eleman sayisini giriniz: 5
dizi elemanini giriniz: 50
dizi elemanini giriniz: 30
dizi elemanini giriniz: 40
dizi elemanini giriniz: 10
dizi elemanini giriniz: 20
10 20 30 40 50
basol@bsl:~/algoritmalar/mergeSort$
```

Java kod

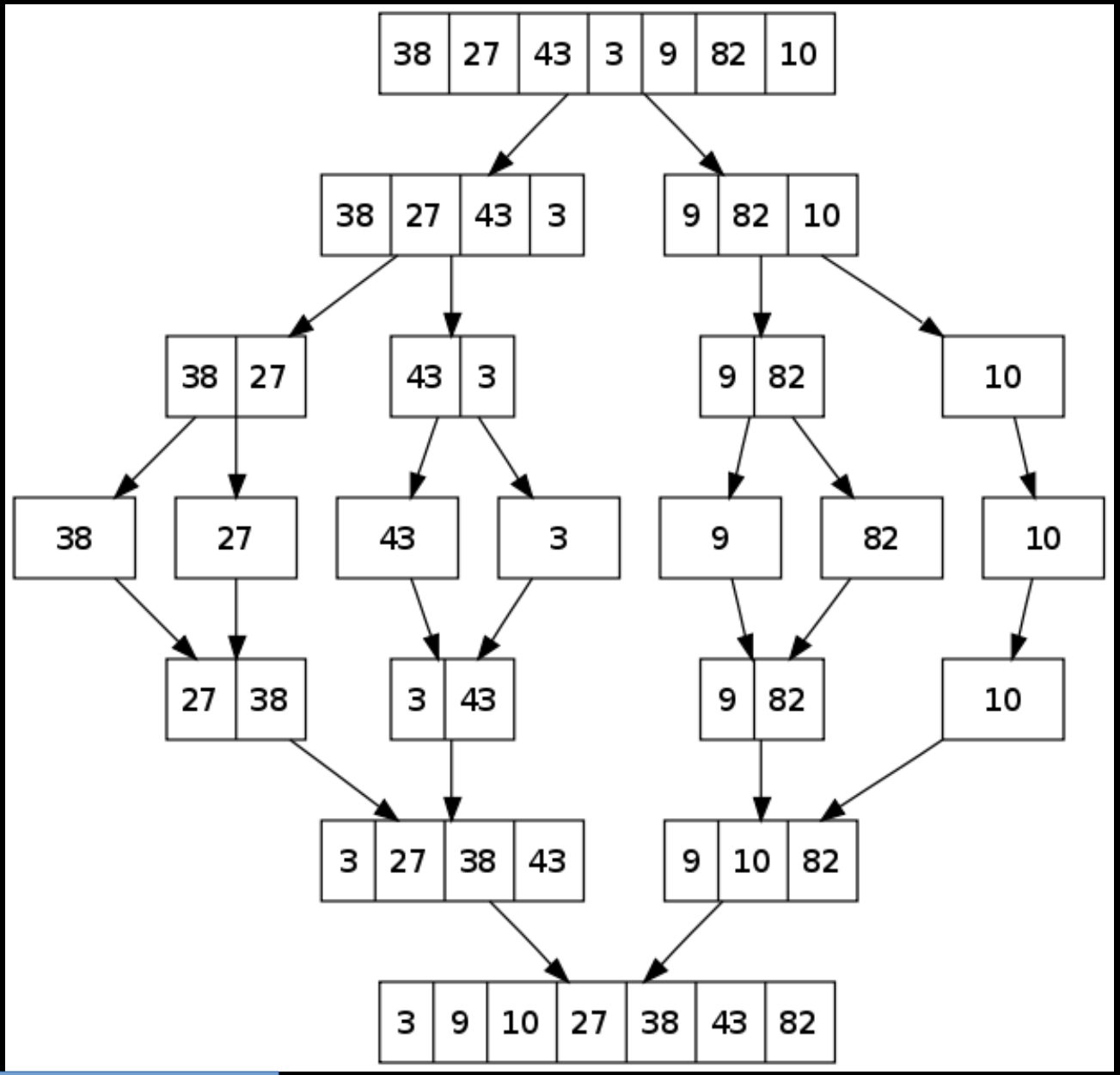
```java
public class MergeSort {

    private int[] list;

   // siralancak listeyi alan inşa fonksiyonu
    public MergeSort(int[] listToSort) {
        list = listToSort;
    }

   // listeyi döndüren kapsülleme fonksiyonu
    public int[] getList() {
        return list;
    }

   // dışarıdan çağırılan sıralama fonksiyonu
    public void sort() {
        list = sort(list);
    }

  // Özyineli olarak çalışan ve her parça için kullanılan sıralama fonksiyonu
    private int[] sort(int[] whole) {
        if (whole.length == 1) {
            return whole;
        }
        else {
           // diziyi ikiye bölüyoruz ve solu oluşturuyoruz
            int[] left = new int[whole.length/2];
            System.arraycopy(whole, 0, left, 0, left.length);

            //dizinin sağını oluşturuyoruz ancak tek sayı ihtimali var
            int[] right = new int[whole.length-left.length];
            System.arraycopy(whole, left.length, right, 0, right.length);

            // her iki tarafı ayrı ayrı sıralıyoruz
            left = sort(left);
            right = sort(right);

            // Sıralanmış dizileri birleştiriyoruz
            merge(left, right, whole);

            return whole;
        }
    }

    // birleştirme fonksiyonu
    private void merge(int[] left, int[] right, int[] result) {
        int x = 0;
        int y = 0;
        int k = 0;
```

```
basol@bsl: ~/algoritmalar/quickSort/java
basol@bsl:~/algoritmalar/quickSort/java$ javac MergeSort.java
basol@bsl:~/algoritmalar/quickSort/java$ java MergeSort
ilk hali:
50
30
70
80
20
10
40
60
100
90
siralanmis hali:
10
20
30
40
50
60
70
80
90
100
basol@bsl:~/algoritmalar/quickSort/java$
```

Merge Sort karmaşıklık

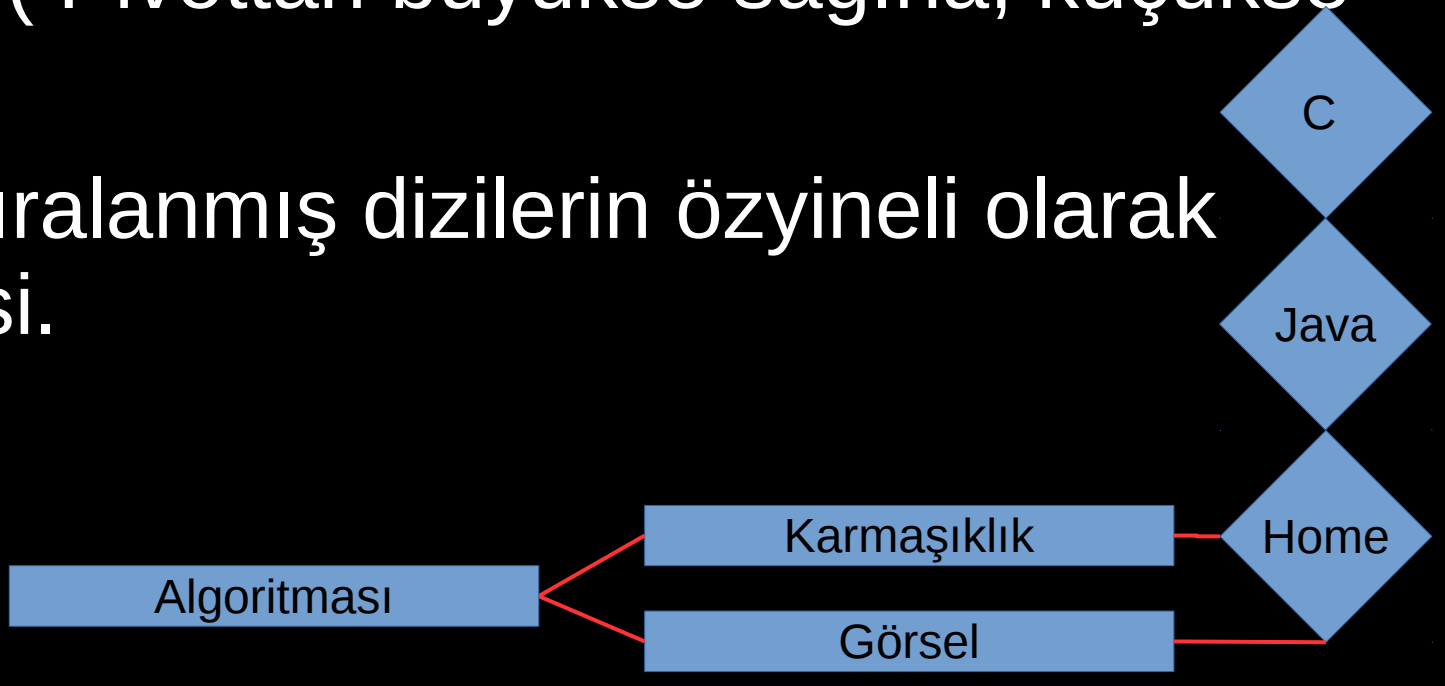
Bestcase(En iyi durum)	Averagacase(Ortalama durum)	Worstcase(En kötü durum)
$n * \log_2 n$	$n * \log_2 n$	$n * \log_2 n$

Merge Sort görsel



Quick Sort (Hızlı Sıralama)

- Divide and Conquer(Parçala ve Fethet)
- Divide: n elemanlı dizinin n/2 elemanlı iki diziye bölünmesi
- Conquer: Her iki dizinin de pivot elemana göre sıralanması.(Pivottan büyükse sağına, küçükse soluna)
- Combine: Sıralanmış dizilerin özyineli olarak birleştirilmesi.



Quick Sort algoritma

```
void quickSort(int arr[], int left, int right) {
      int i = left, j = right;
      int tmp;
      int pivot = arr[(left + right) / 2];
      // pivot'u istedigimiz eleman secebiliriz.
      /* partition islemi */
      while (i <= j) {
            while (arr[i] < pivot)
                  i++;
            while (arr[j] > pivot)
                  j--;
            if (i <= j) {
                  tmp = arr[i];
                  arr[i] = arr[j];
                  arr[j] = tmp;
                  i++;
                  j--;
            }
      };

      /* recursive */
      if (left < j)
            quickSort(arr, left, j);
      if (i < right)
            quickSort(arr, i, right);
```

C kod

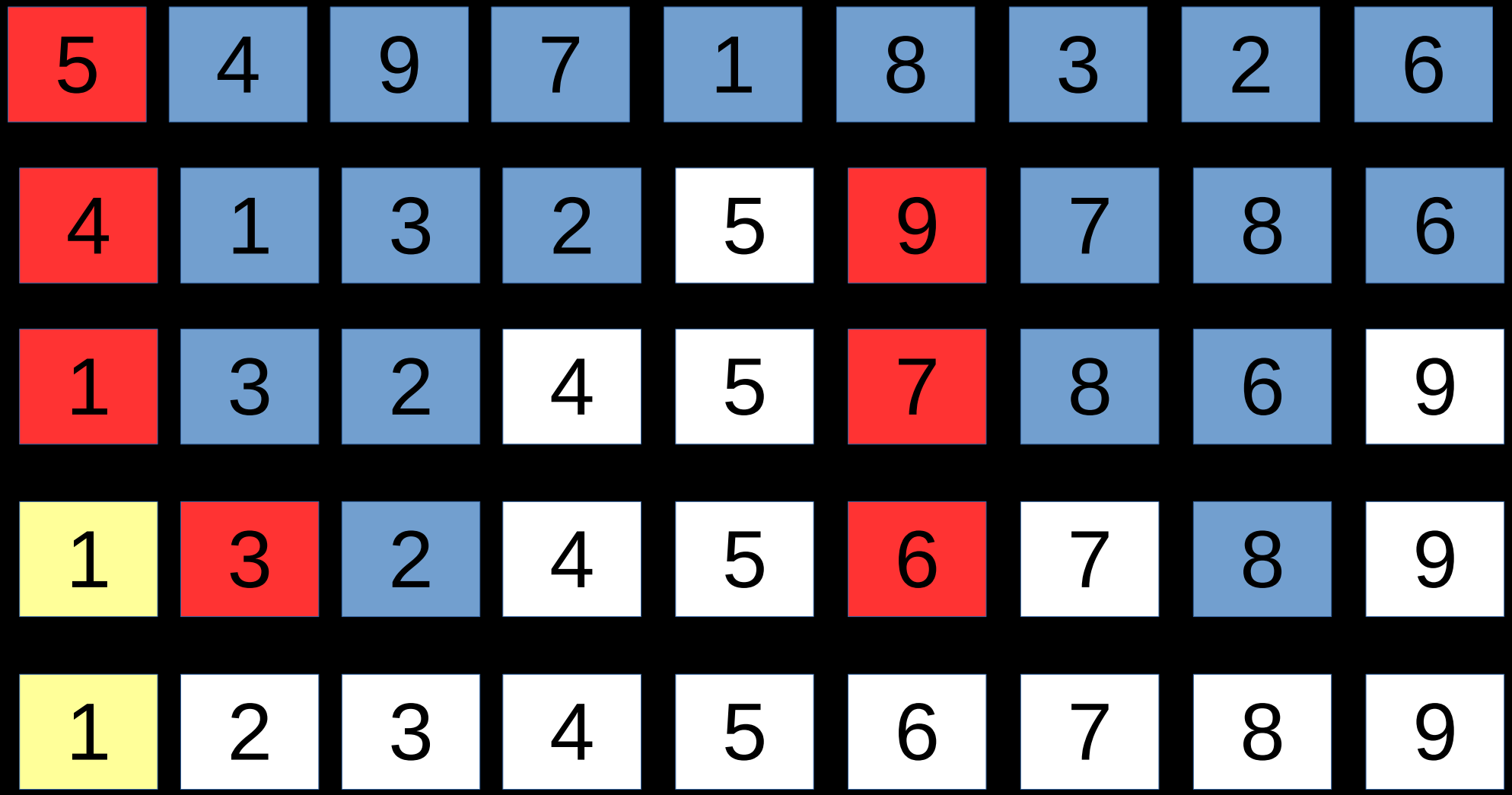
```c
#include <stdio.h>
int N=10;
int a[10]={50,70,30,40,10,80,30,100,20,90};

void degis(int x[],int i, int j){
        int temp;
        temp=x[i];
        x[i]=x[j];
        x[j]=temp;
}

void quickSort(int a[], int sol,int sag){
        int i,j,v;
        if(sag>sol){
                v=a[sag];
                i=sol-1;
                j=sag;
                while(i<j){
                        while(a[++i] < v);
                        while(a[--j] > v);
                                if(i<j)
                                degis(a,i,j);
                }
            degis(a,i,sag);
            quickSort(a,sol,i-1);
            quickSort(a,i+1,sag);
        }
}
void main(){
        int i;
        printf("\nDizinin ilk hali:\n");
        for(i=0; i<N; i++)
                printf("%d ",a[i]);

        quickSort(a,0,N);

        printf("\nSiralanmis hali:\n");
        for(i=1; i<=N; i++)
                printf("%d ",a[i]);
}
```

```
basol@bsl: ~/algoritmalar/quickSort/cKod
basol@bsl:~/algoritmalar/quickSort/cKod$ ./a.out

Dizinin ilk hali:
50 70 30 40 10 80 30 100 20 90
Siralanmis hali:
10 20 30 30 40 50 70 80 90 100 basol@bsl:~/algori
```

Java kod

```java
int partition(int arr[], int left, int right)
{
      int i = left, j = right;
      int tmp;
      int pivot = arr[(left + right) / 2];

      while (i <= j) {
            while (arr[i] < pivot)
                  i++;
            while (arr[j] > pivot)
                  j--;
            if (i <= j) {
                  tmp = arr[i];
                  arr[i] = arr[j];
                  arr[j] = tmp;
                  i++;
                  j--;
            }
      };

      return i;
}

void quickSort(int arr[], int left, int right) {
      int index = partition(arr, left, right);
      if (left < index - 1)
            quickSort(arr, left, index - 1);
      if (index < right)
            quickSort(arr, index, right);
```

Quick Sort görsel



Quick Sort

Home

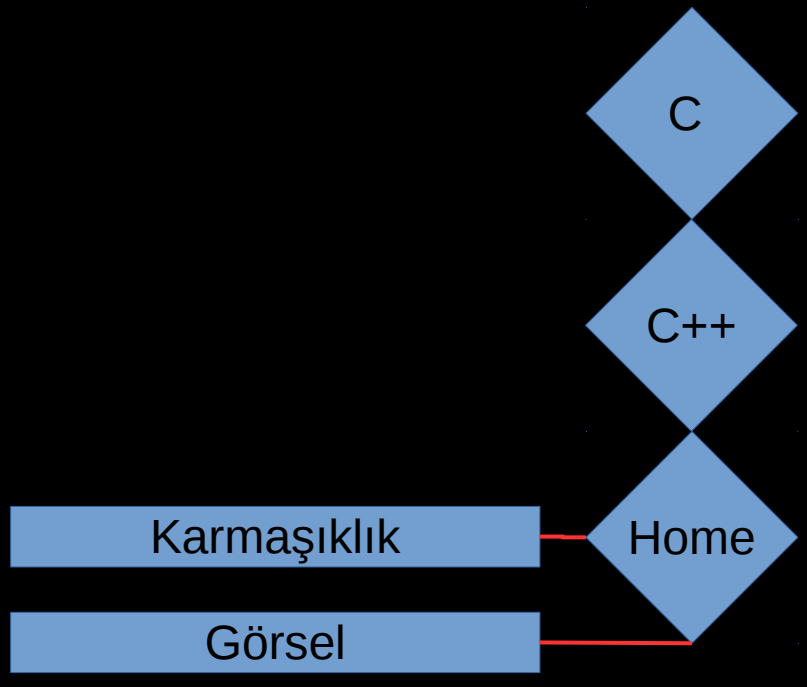
Quick Sort Karmaşıklık

Bestcase(En iyi durum)	Averagacase(Ortalama durum)	Worstcase(En kötü durum)
$n * \log_2 n$	$n * \log_2 n$	n^2

Shell Sort(Kabuk Sıralaması)

- Herhangi bir sıralama algoritması üzerinde çalışır.

```
void shell_sort (int *p, int size)
{
   int   i, j, k, temp;   // k: atlama miktari
   for (k = size; k > 1; ) {
      k = (k < 5) ? 1 : ((k * 5 - 1) / 11); // 5/11 oraninda kuculen atlama
                  /*bubble sort kullandik*/
         for (i = k - 1; ++i < size; ) {
         temp = p[i];
         for (j = i; p[j - k] > temp; ) {
            p[j] = p[j - k];
            if ((j -= k) < k)
               break;
         }
         p[j] = temp;
      } /* */
   }
}
```



C kod

```c
#include <stdio.h>
int N=10;
int a[10]={50,70,30,40,10,80,30,100,20,90};

void degis(int x[],int i, int j){
        int temp;
        temp=x[i];
        x[i]=x[j];
        x[j]=temp;
}

void shellSort(){
        int orta,i,j,k;
        orta= N/2;
        while(orta>0){
                for(i=orta; i<N; i++){
                        j=i-orta;
                        while(j>=0){
                                k=j+orta;
                                if(a[j]<a[k])
                                        j=-1;
                                else{
                                        degis(a,j,k);
                                        j=j-orta;
                                }
                        }
                }
                if(orta==0)
                        break;

                orta=orta/2;
         }
}
void main(){
        int i;
        printf("\nDizinin ilk hali:\n");
        for(i=0; i<N; i++)
                printf("%d ",a[i]);

        shellSort();

        printf("\nSiralanmis hali;\n");
        for(i=0; i<N; i++)
                printf("%d ",a[i]);
}
```

```
basol@bsl: ~/algoritmalar/shellSort
basol@bsl:~/algoritmalar/shellSort$ gcc shell.c
basol@bsl:~/algoritmalar/shellSort$ ./a.out

Dizinin ilk hali:
50 70 30 40 10 80 30 100 20 90
Siralanmis hali;
10 20 30 30 40 50 70 80 90 100 basol@bsl:~/algorit
```

Shell Sort

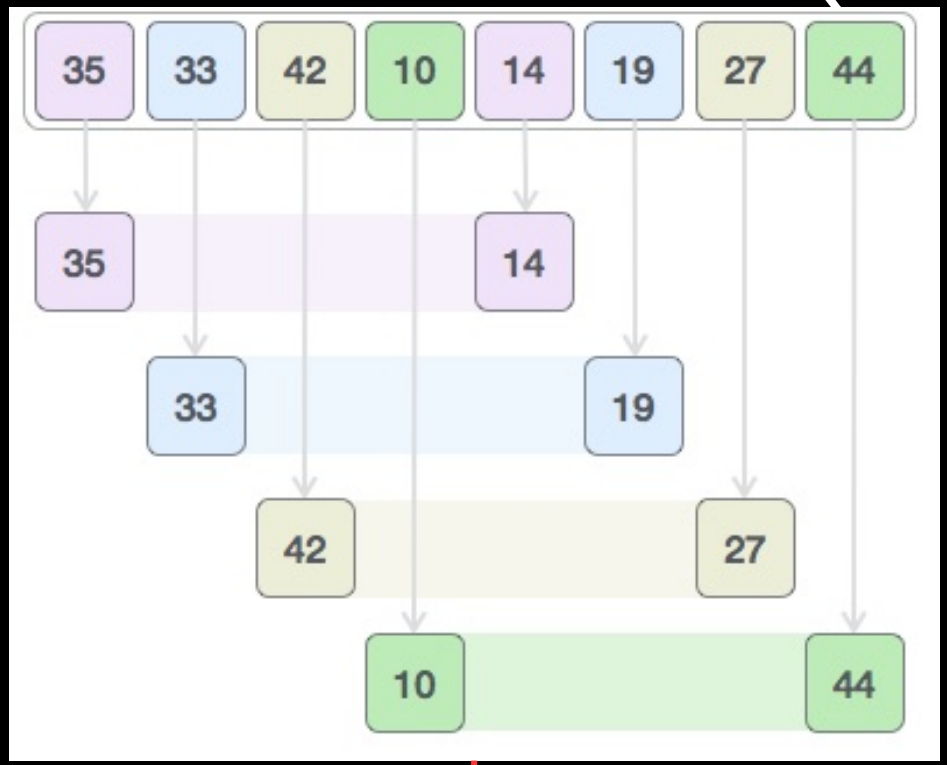
Home

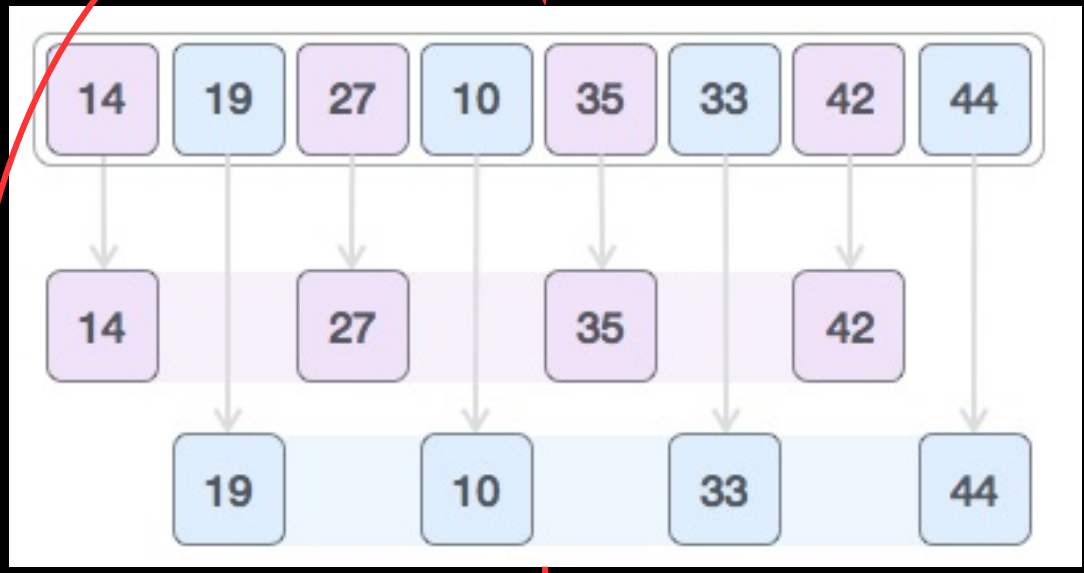
C++ kod

```cpp
#include <iostream>
#include <stdio.h>
using namespace std;
void shell_sort (int *p, int size)
{
   int   i, j, k, temp;   // k: atlama miktari
   for (k = size; k > 1; ) {
      k = (k < 5) ? 1 : ((k * 5 - 1) / 11);
                /*bubble sort kullandik*/
         for (i = k - 1; ++i < size; ) {
         temp = p[i];
         for (j = i; p[j - k] > temp; ) {
            p[j] = p[j - k];
            if ((j -= k) < k)
               break;
         }
         p[j] = temp;
      } /* */
   }
}

int main(){
        int A=11;
        int a[]={11,10,9,8,7,6,5,4,3,2,1};
        shell_sort(a,A);
        for(int i=0; i<A; i++)
                printf("%d ",a[i]);
}
```

```
basol@bsl: ~/algoritmalar/shellSort/C++
basol@bsl:~/algoritmalar/shellSort/C++$ ./a.out
1 2 3 4 5 6 7 8 9 10 11 basol@bsl:~/algoritmalar/shellSort/C++$
```

Shell Sort(Kabuk Sıralaması)





Artık herhangi bir algoritma ile devamını sıralıyabiliriz.

Shell Sort

Home

Shell Sort karmaşıklık

Bestcase(En iyi durum	Averagacase(Ortalama durum)	Worstcase(En kötü durum)
$n^{3/2}$	$n^{3/2}$	$n^{3/2}$

Karşılaştırma

	Bestcase(En iyi durum)	Averagacase(Ortalama durum)	Worstcase(En kötü durum)
Selection Sort	n^2	n^2	n^2
Bubble Sort	n	n^2	n^2
Insertion Sort	n	n^2	n^2
Merge Sort	$n * \log_2 n$	$n * \log_2 n$	$n * \log_2 n$
Quick Sort	$n * \log_2 n$	$n * \log_2 n$	$n * \log_2 n$
Shell Sort	$n^{3/2}$	$n^{3/2}$	$n^{3/2}$

Home

YUSUF BAŞOL