YILDIZ TEKNİK ÜNİVERSİTESİ
ELEKTRİK-ELEKTRONİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ
2014-2015 ÖĞRETİM YILI GÜZ YARIYILI

VERİ YAPILARI VE ALGORİTMALAR ÖDEV-7 /LABORATUVAR-4 (BLM-2512/ GRUP:1)

Hazırlanan Anabilim Dalı
Bilgisayar Bilimleri Anabilim Dalı

KONU: MERGE SORT VE QUİCK SORT ALGORİTMALARININ TASARLANMASI VE BİR PROGRAMLAMA DİLİ ÜZERİNDE GERÇEKLEŞTİRİLMESİ

Hazırlayan

Mert Sevil
09013057
Bilgisayar Mühendisliği Lisans Programı

Öğretim Üyesi

Prof. M. Yahya KARSLIGİL

İSTANBUL, 2014

1. İçindekiler

2. Konu ile ilgili teorik bilginin aktarılması

2.1 Merge Sort Algoritması

Birleşmeli Sıralama (*Merge Sort*), bilgisayar bilimlerinde $\mathcal{O}(n\ \log(n))$ derecesinde karmaşıklığa sahip bir sıralama algoritmasıdır. Girdi olarak aldığı diziyi en küçük hale gelene kadar ikili gruplara böler ve karşılaştırma yöntemi kullanarak diziyi sıralar. [1]

Algoritmanın çalışması kavramsal olarak şöyledir:

1. Sıralı olmayan listeyi ortadan eşit olarak iki alt listeye ayırır.
2. Alt listeleri kendi içinde sıralar.
3. Sıralı iki alt listeyi tek bir sıralı liste olacak şekilde birleştirir.

Bu algoritma John von Neumann tarafından 1945 yılında bulunmuştur. [1]

Merge Sort algoritması ile Heap Sort algoritması aynı zaman aralığına sahip olmalarına rağmen Heap Sort algoritması bellekte Merge Sort algoritmasına göre daha az yer tutar ve bu algoritmalar gerçeklendiğinde Heap Sort algoritması daha hızlı çalışır. Quick Sort algoritması da bellek tabanlı dizilerde Merge Sort'a göre daha hızlı çalışmaktadır. Ancak bağlı liste sıralamasında seçilebilecek en performanslı algoritma Merge Sort algoritmasıdır. Çünkü bağlı listelerin yapısı gereği mergesort bellekte fazladan sadece 1 birim yer tutar ve bağlı listelerin yavaş ve rastgele erişim performansı nedeniyle quicksort gibi diğer algoritmaların çalışma performansı düşer, Heap Sort gibi algoritmalar için ise imkansızdır.[2]

2.1 Quick Sort Algoritması

Hızlı Sıralama (İngilizcesi: Quicksort) günümüzde yaygın olarak kullanılan bir sıralama algoritmasıdır. Hızlı sıralama algoritması *n* adet sayıyı, ortalama bir durumda, $\mathcal{O}(n\ log(n))$karmaşıklığıyla, en kötü durumda ise $\mathcal{O}(n^2)$karmaşıklığıyla sıralar. Algoritmanın karmaşıklığı aynı zamanda yapılan karşılaştırma sayısına eşittir. [3]

Hızlı sıralama algoritması, sıralanacak bir sayı dizisini daha küçük iki parçaya ayırıp oluşan bu küçük parçaların kendi içinde sıralanması mantığıyla çalışır.

Algoritmanın adımları aşağıdaki gibidir:

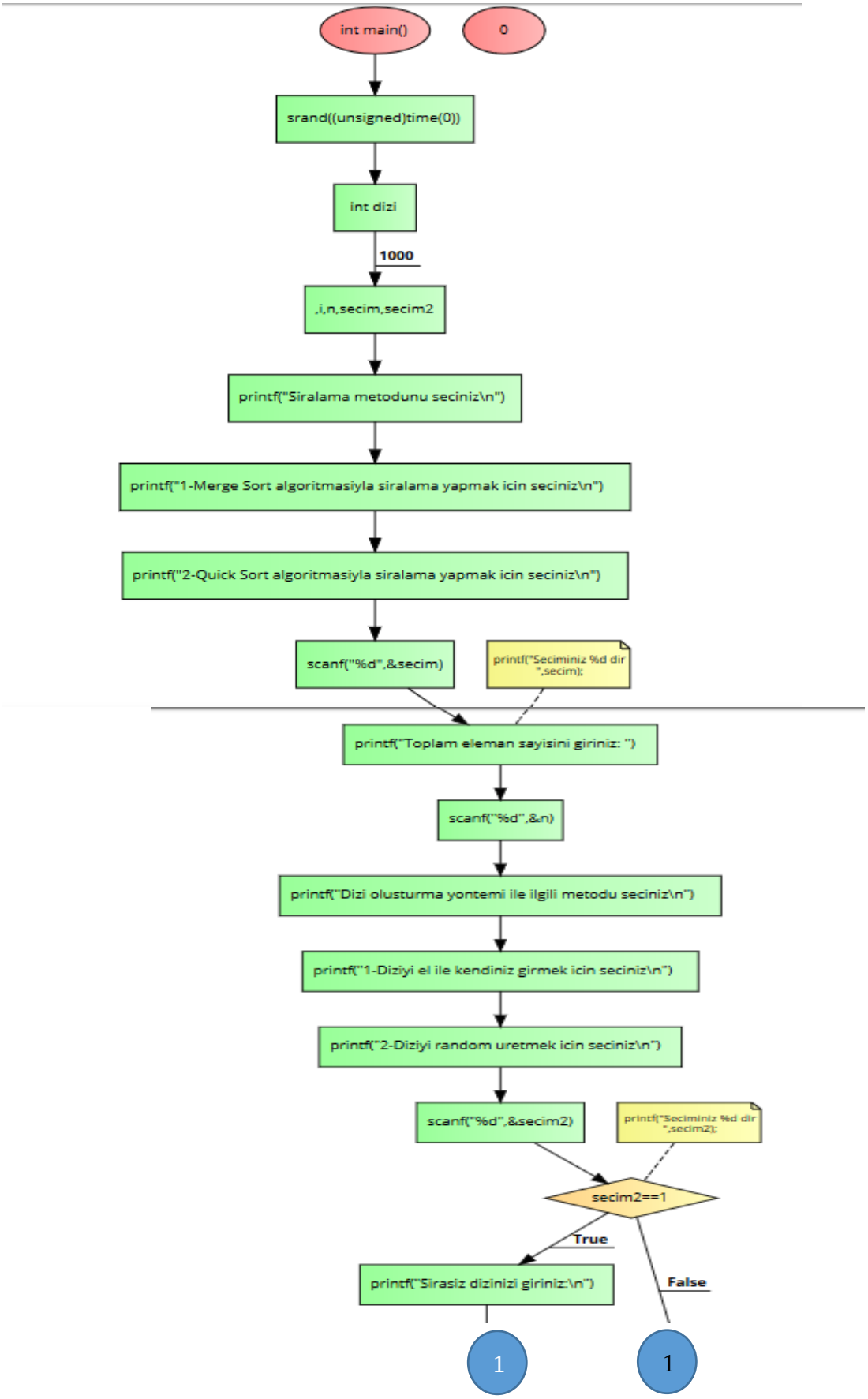
1. Sayı dizisinden herhangi bir sayıyı pivot eleman olarak seç.
2. Sayı dizisini pivottan küçük olan tüm sayılar pivotun önüne, pivottan büyük olan tüm sayılar pivotun arkasına gelecek biçimde düzenle (pivota eşit olan sayılar her iki yana da geçebilir). Bu bölümlendirme işleminden sonra eleman sıralanmış son dizide olması gerektiği yere gelir. Algoritmanın bu aşamasına bölümlendirme aşaması denir.
3. Pivotun sol ve sağ yanında olmak üzere oluşan iki ayrı küçük sayı dizisi, hızlı sıralama algoritması bu küçük parçalar üzerinde yeniden özyineli olarak çağrılarak sıralanır.

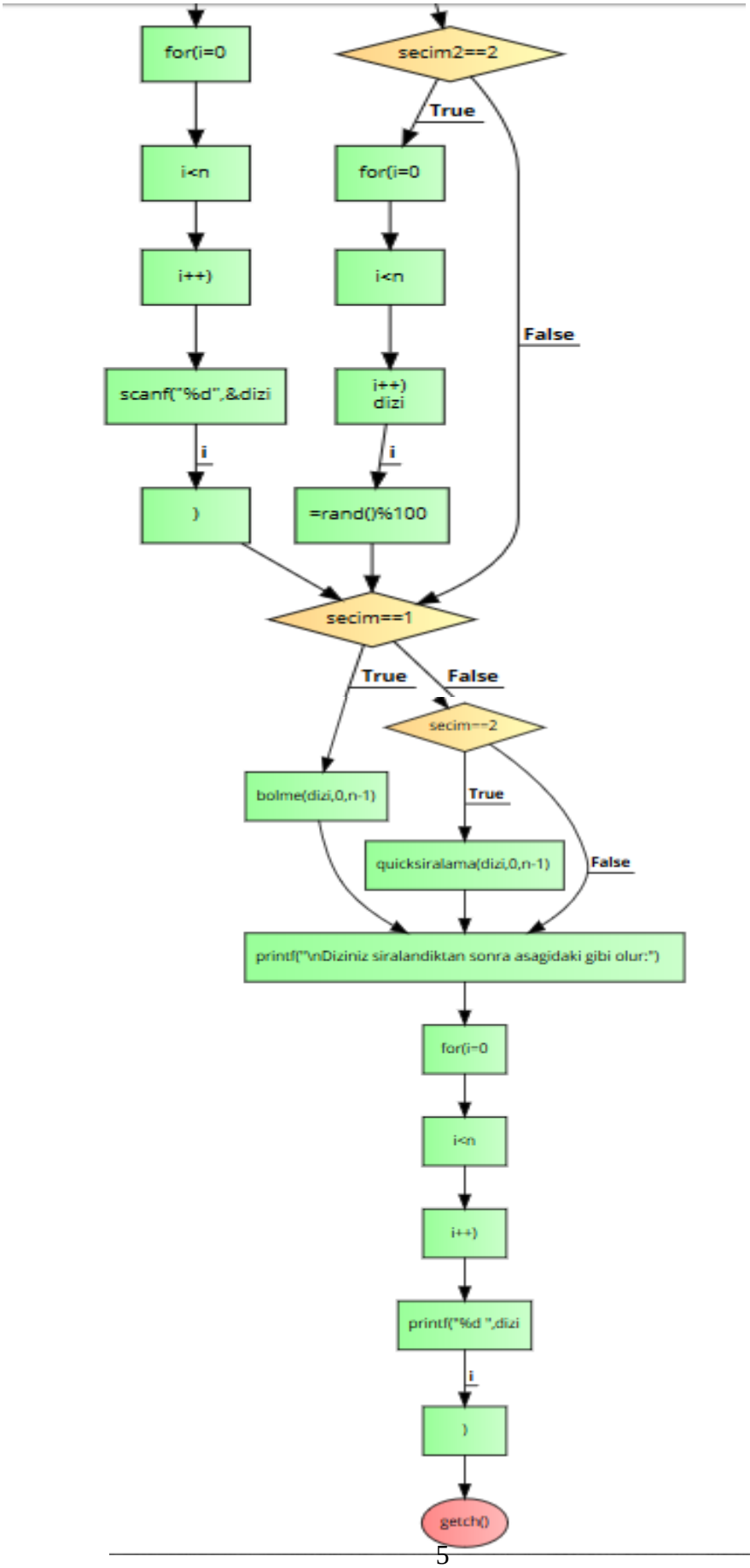
Algoritma içinde sayı kalmayan (eleman sayısı sıfır olan) bir alt diziye ulaştığında bu dizinin sıralı olduğunu varsayar.[3]

Verinin hafızada sıralı tutulması için geliştirilen sıralama algoritmalarından (sorting algorithms) bir tanesidir. Basitçe sıralanacak olan dizideki orta noktada (mean) bulunan bir sayıyı seçerek diğer bütün sayıları bu orta sayıdan büyük veya küçük diye sınıflayarak sıralama yapmayı hedeflemektedir. Bu açıdan bir parçala fethet (divide and conquere) yaklaşımıdır. Ayrıca bu seçilen orta noktaya eksen (pivot) adı da verilir çünkü bütün diğer sayılar bu sayının ekseninde sıralanacaktır. [4]

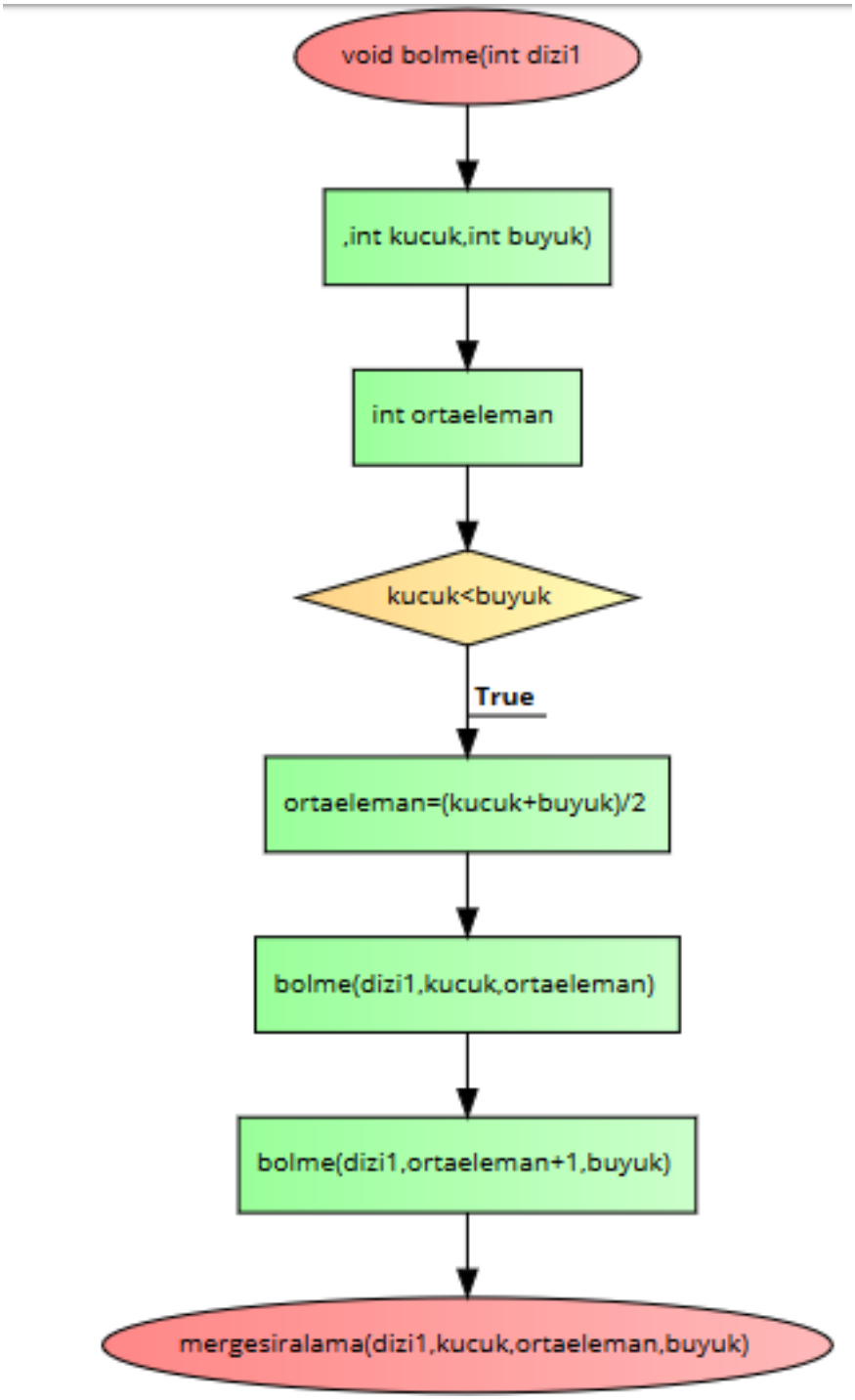
3. Algoritmanın aktarılması

Algoritma üç ana bölümden oluşmaktadır. İnt main kısmında kullanıcı arayüzüde dahil olmak üzere dizinin tanımlanması ve yapılacak işlemlere ait seçimler gerçekleştirilir.

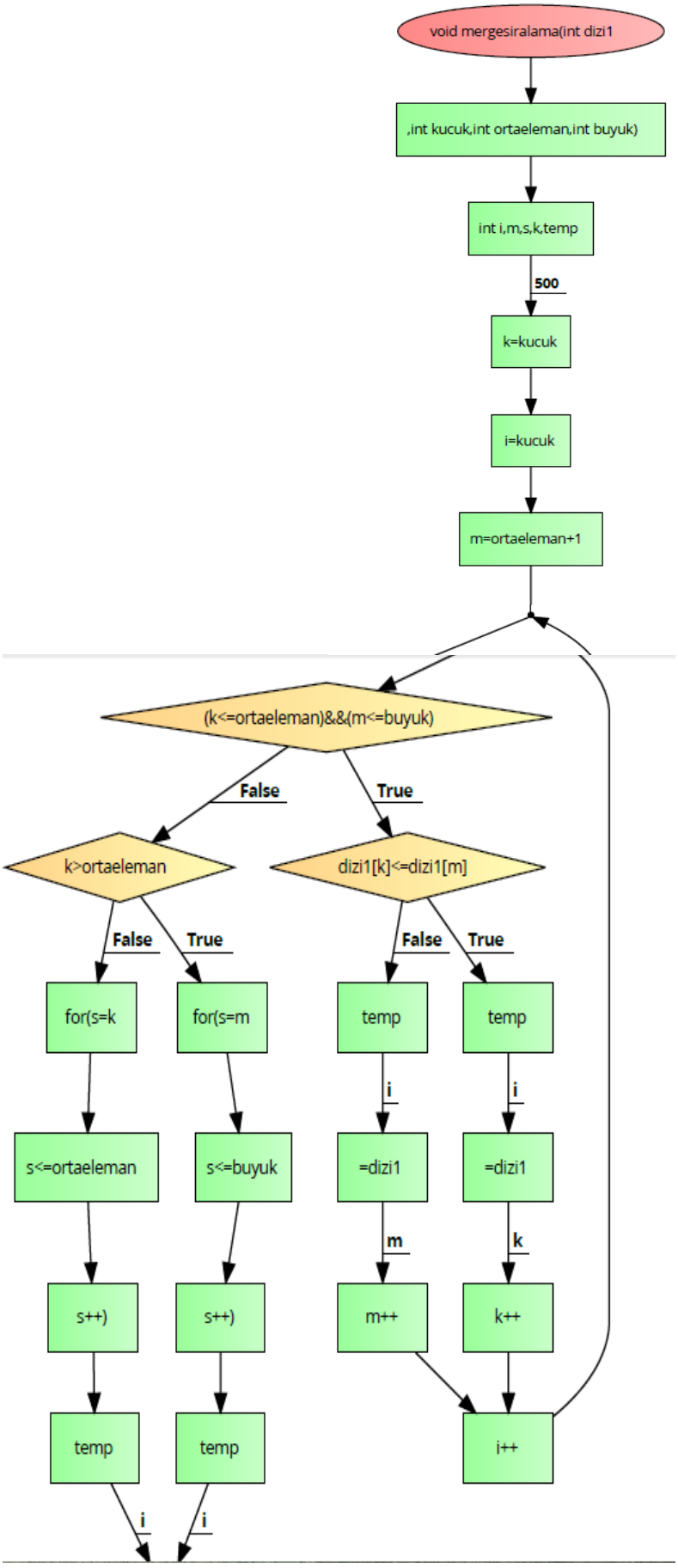

int main()
0
srand((unsigned)time(0))
int dizi
1000
,i,n,secim,secim2
printf("Siralama metodunu seciniz\n")
printf("1-Merge Sort algoritmasiyla siralama yapmak icin seciniz\n")
printf("2-Quick Sort algoritmasiyla siralama yapmak icin seciniz\n")
scanf("%d",&secim)
printf("Seciminiz %d dir ",secim);
printf("Toplam eleman sayisini giriniz: ")
scanf("%d",&n)
printf("Dizi olusturma yontemi ile ilgili metodu seciniz\n")
printf("1-Diziyi el ile kendiniz girmek icin seciniz\n")
printf("2-Diziyi random uretmek icin seciniz\n")
scanf("%d",&secim2)
printf("Seciminiz %d dir ",secim2);
secim2==1
True
False
printf("Sirasiz dizinizi giriniz:\n")
1
1

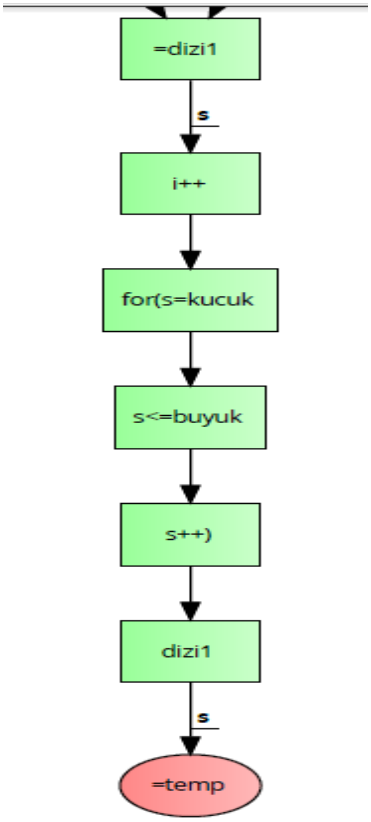

for(i=0
i<n
i++)
scanf("%d",&dizi
i
)
secim2==2
True
for(i=0
i<n
i++)
dizi
i
=rand()%100
False
secim==1
True
False
secim==2
bolme(dizi,0,n-1)
True
quicksiralama(dizi,0,n-1)
False
printf("\nDiziniz siralandiktan sonra asagidaki gibi olur:")
for(i=0
i<n
i++)
printf("%d ",dizi
i
)
getch()

Aşağıdaki akış void bölme ismindeki fonksiyona ait akıştır. Bu akış merge sort algoritmasına ait bir alt fonksiyondur. Fonksiyonun görevi verilen diziyi alt parçalara bölmektir.

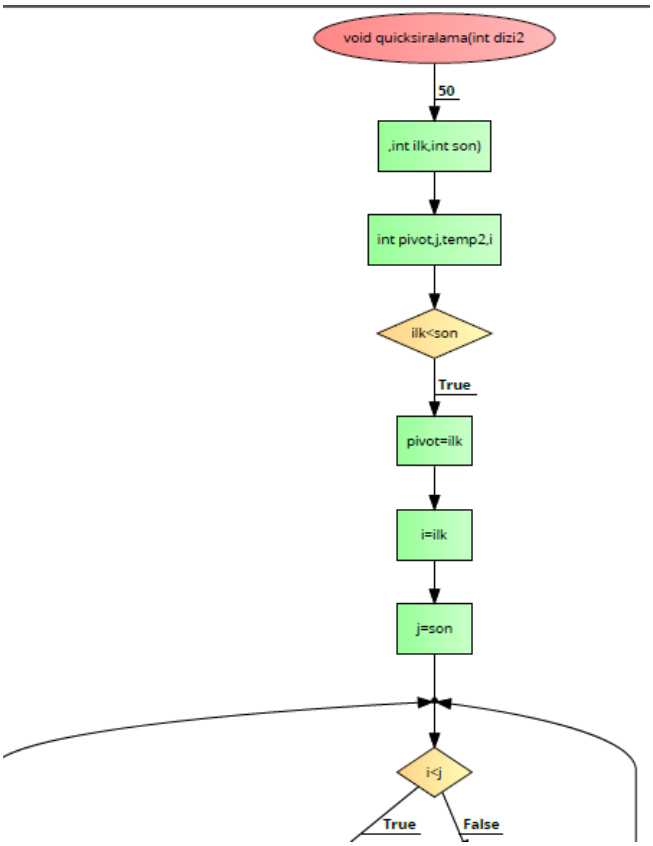


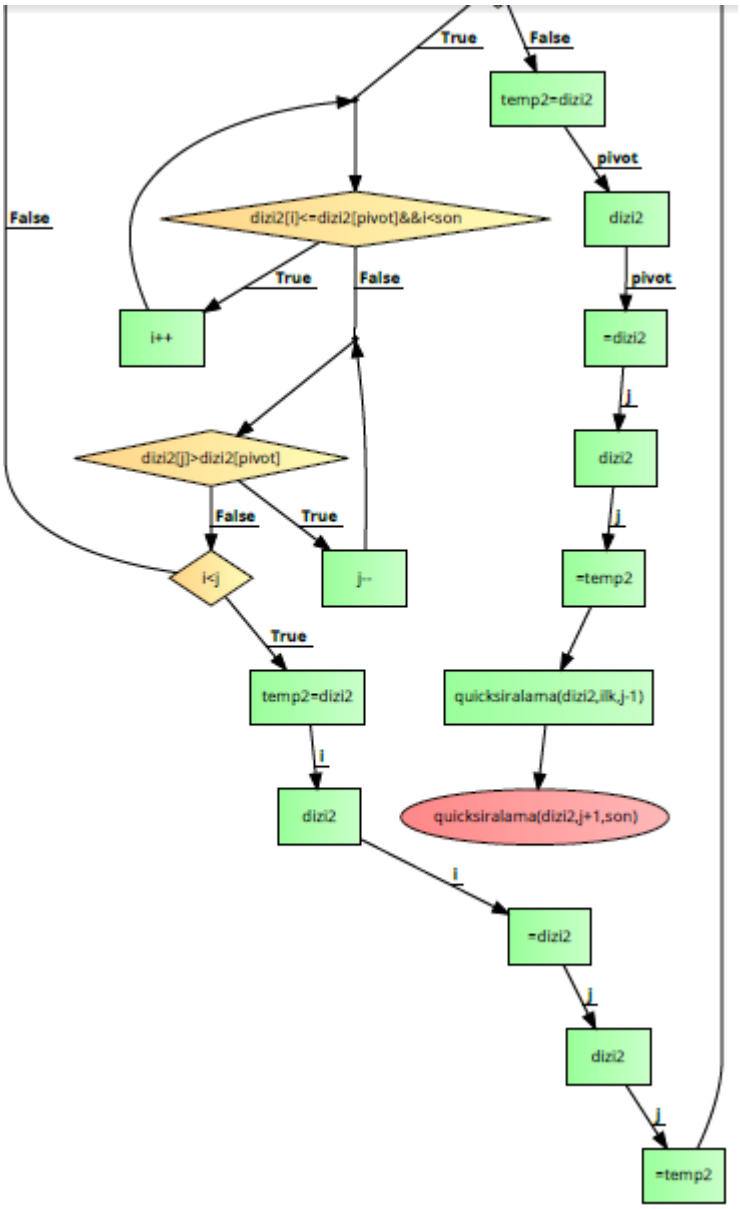
İşlemlerin yapıldığı mergesiralama alt fonksiyonunda ise bölme fonksiyonunda bölünen diziler kendi içinde sıralanarak birleştirilir.


void mergesiralama(int dizi1
,int kucuk,int ortaeleman,int buyuk)
int i,m,s,k,temp
500
k=kucuk
i=kucuk
m=ortaeleman+1
(k<=ortaeleman)&&(m<=buyuk)
False
True
k>ortaeleman
dizi1[k]<=dizi1[m]
False
True
False
True
for(s=k
for(s=m
temp
temp
i
i
s<=ortaeleman
s<=buyuk
=dizi1
=dizi1
m
k
s++)
s++)
m++
k++
temp
temp
i++
i
i



Böylece Merge sort algoritması belirtilmiştir. Son olarak Quick sort algoritması ele alınacaktır. Aşağıda Quick Sort algoritmasına ait akış diyagramı verilmiştir.





Böylece algoritmik akış ele alınmıştır. Bundan sonraki bölümde yazılan C kodu değerlendirilecektir.

4. Yazılan C kodunun belirlenmesi

Dev C++ derleyecesinde derlenen C kodu aşağıda belirlenmiştir. 3. Bölümde incelenen algoritmaya uygun olarak tasarlanan C koduna ait sonuçlar ise 5. Bölümde değerlendirilmiştir.

```
#include<stdio.h>                    //İLGİLİ KÜTÜPHANELER
 #include<conio.h>
 #include <time.h>
 #include<stdlib.h>

void mergesiralama(int dizi1[],int kucuk,int ortaeleman,int buyuk);
```

```
void bolme(int dizi1[],int kucuk,int buyuk);          //FONKSİYON PROTOTİPLERİ
void quicksiralama(int [100],int,int);

int main(){             //SIRALAMA METODUNU VE DIZI URETIM YONTEMININ DE SEÇİLDİĞİ ANA METOT

     srand((unsigned)time(0));

   int dizi[1000],i,n,secim,secim2;
   printf("Siralama metodunu seciniz\n");
   printf("1-Merge Sort algoritmasiyla siralama yapmak icin seciniz\n");
   printf("2-Quick Sort algoritmasiyla siralama yapmak icin seciniz\n");

     scanf("%d",&secim);
     //printf("Seciminiz %d dir\n",secim);

   printf("Toplam eleman sayisini giriniz: ");
   scanf("%d",&n);

   printf("Dizi olusturma yontemi ile ilgili metodu seciniz\n");
   printf("1-Diziyi el ile kendiniz girmek icin seciniz\n");
   printf("2-Diziyi random uretmek icin seciniz\n");
   scanf("%d",&secim2);
   //printf("Seciminiz %d dir\n",secim2);

     if(secim2==1){

     printf("Sirasiz dizinizi giriniz:\n");
   for(i=0;i<n;i++){
       scanf("%d",&dizi[i]);
   }
}
else if(secim2==2){


     for(i=0;i<n;i++)
     dizi[i]=rand()%100 ;

}

if(secim==1)
   bolme(dizi,0,n-1);
else if(secim==2)
   quicksiralama(dizi,0,n-1);

   printf("\nDiziniz siralandiktan sonra asagidaki gibi olur:");
   for(i=0;i<n;i++){
       printf("%d ",dizi[i]);
   }
```

```
  getch();
  return 0;
}

void bolme(int dizi1[],int kucuk,int buyuk){      //MERGE SORT İÇİN REKURSİF BÖLÜMLEME METODU

   int ortaeleman;

   if(kucuk<buyuk){
      ortaeleman=(kucuk+buyuk)/2;
      bolme(dizi1,kucuk,ortaeleman);
      bolme(dizi1,ortaeleman+1,buyuk);
      mergesiralama(dizi1,kucuk,ortaeleman,buyuk);
   }
}

void mergesiralama(int dizi1[],int kucuk,int ortaeleman,int buyuk){  //MERGE SORT İÇİN BÖLÜMLENMİŞ DİZİLERİN KENDİ İÇİNDE SIRALANDIĞI METOT

   int i,m,s,k,temp[500];

   k=kucuk;
   i=kucuk;
   m=ortaeleman+1;

   while((k<=ortaeleman)&&(m<=buyuk)){

      if(dizi1[k]<=dizi1[m]){
         temp[i]=dizi1[k];
         k++;
      }
      else{
         temp[i]=dizi1[m];
         m++;
      }
      i++;
   }

   if(k>ortaeleman){
      for(s=m;s<=buyuk;s++){
         temp[i]=dizi1[s];
         i++;
      }
   }
   else{
      for(s=k;s<=ortaeleman;s++){
         temp[i]=dizi1[s];
         i++;
      }
```

```
    }

    for(s=kucuk;s<=buyuk;s++){
        dizi1[s]=temp[s];
    }
}


 void quicksiralama(int dizi2[50],int ilk,int son){ // QUICK SORT İÇİN SIRALAMA METODU
    int pivot,j,temp2,i;

     if(ilk<son){
        pivot=ilk;
        i=ilk;
        j=son;

        while(i<j){
            while(dizi2[i]<=dizi2[pivot]&&i<son)
                i++;
            while(dizi2[j]>dizi2[pivot])
                j--;
            if(i<j){
                temp2=dizi2[i];
                 dizi2[i]=dizi2[j];
                 dizi2[j]=temp2;
            }
        }

        temp2=dizi2[pivot];
        dizi2[pivot]=dizi2[j];
        dizi2[j]=temp2;
        quicksiralama(dizi2,ilk,j-1);
        quicksiralama(dizi2,j+1,son);

    }
}
```

5. Sonuçlar ve analizi

Öncelikle Merge sort algoritmasını test edelim. Bunun için öncelikle testi random sayı üretilen dizilerle gerçekleştirelim. Bunun için ilgili seçimler yapılırsa üretilen sonuç dizisi aşağıdaki gibi olacaktır.

```
Siralama metodunu seciniz
1-Merge Sort algoritmasiyla siralama yapmak icin seciniz
2-Quick Sort algoritmasiyla siralama yapmak icin seciniz
1
Toplam eleman sayisini giriniz: 70
Dizi olusturma yontemi ile ilgili metodu seciniz
1-Diziyi el ile kendiniz girmek icin seciniz
2-Diziyi random uretmek icin seciniz
2

Diziniz siralandiktan sonra asagidaki gibi olur:0 0 1 2 5 6 8 9 10 14 14 17 18 22 23 27 31 38 42 44
45 45 45 45 47 49 50 51 52 54 55 55 56 57 59 60 60 60 61 61 62 63 66 67 67 68 69 69 73 73 74 76 76 7
8 78 79 82 82 83 83 83 89 90 94 95 96 96 97 98 99 _
```

Merge Sort için el ile üretilen dizide bir sıralama yapılması için oluşturulan örnek aşağıdaki gibidir.

```
Siralama metodunu seciniz
1-Merge Sort algoritmasiyla siralama yapmak icin seciniz
2-Quick Sort algoritmasiyla siralama yapmak icin seciniz
1
Toplam eleman sayisini giriniz: 5
Dizi olusturma yontemi ile ilgili metodu seciniz
1-Diziyi el ile kendiniz girmek icin seciniz
2-Diziyi random uretmek icin seciniz
1
Sirasiz dizinizi giriniz:
-5
6
1
3
8

Diziniz siralandiktan sonra asagidaki gibi olur:-5 1 3 6 8
```

Quick sort için bir uygulama örneği sunulacak olursa, sunulan örneğe ait ekran çıktısı aşağıdaki gibi olacaktır.

```
Siralama metodunu seciniz
1-Merge Sort algoritmasiyla siralama yapmak icin seciniz
2-Quick Sort algoritmasiyla siralama yapmak icin seciniz
2
Toplam eleman sayisini giriniz: 70
Dizi olusturma yontemi ile ilgili metodu seciniz
1-Diziyi el ile kendiniz girmek icin seciniz
2-Diziyi random uretmek icin seciniz
2

Diziniz siralandiktan sonra asagidaki gibi olur:0 2 3 4 4 5 5 8 8 8 8 9 11 12 13 13 14 15 16 16 17 1
9 20 23 23 25 26 31 31 32 34 34 35 36 38 42 42 48 50 51 51 55 56 57 62 63 65 67 70 71 74 75 76 76 77
 77 77 81 83 83 84 85 86 88 90 93 94 96 97 98 _
```

Yukarıdaki örnek dizi random elemanlar ile üretilmiştir. Aşağıda görülen ekran çıktısında ise Quick sort algoritması için oluşturulacak dizi el ile üretilmiştir.

```
Siralama metodunu seciniz
1-Merge Sort algoritmasiyla siralama yapmak icin seciniz
2-Quick Sort algoritmasiyla siralama yapmak icin seciniz
2
Toplam eleman sayisini giriniz: 6
Dizi olusturma yontemi ile ilgili metodu seciniz
1-Diziyi el ile kendiniz girmek icin seciniz
2-Diziyi random uretmek icin seciniz
1
Sirasiz dizinizi giriniz:
9
1
5
11
-3
8

Diziniz siralandiktan sonra asagidaki gibi olur:-3 1 5 8 9 11 _
```

Böylece tasarlanan algoritma ve akabinde geliştirilen C koduna ait olası senaryolar değerlendirilerek ilgili testler gerçekleştirilmiş ve algoritmanın doğru çalıştığı gözlemlenmiştir.

6. Kaynakça

[1] http://tr.wikipedia.org/wiki/Birle%C5%9Ftirmeli_s%C4%B1ralama

[2] http://bidb.itu.edu.tr/seyirdefteri/blog/2013/09/08/merge-sort-%28bile%C5%9Ftirme-s%C4%B1ralamas%C4%B1%29-algoritmas%C4%B1

[3] http://tr.wikipedia.org/wiki/H%C4%B1zl%C4%B1_s%C4%B1ralama

[4] http://bilgisayarkavramlari.sadievrenseker.com/2008/08/09/hizli-siralama-algoritmasi-quick-sort-algorithm/