TÜRKİYE CUMHURİYETİ
YILDIZ TEKNİK ÜNİVERSİTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

ALGORİTMA ANALİZİ 1.ÖDEV :BÖL VE YÖNET ALGORİTMALAR(KEY-LOCK PROBLEMİ)

17011604 — Havvanur Dervişoğlu
—

ALGORİTMA ANALİZİ ÖDEVİ RAPORU

Danışman
Dr.Öğr.ÜyesiM. Amaç Güvensan

Kasım, 2018

İÇİNDEKİLER

1
KULLANILAN YÖNTEM

1.1 Akış Diyagramı

1.1.1 Main Fonksiyonu

Ana fonksiyonda;

- key ve lock dizilerinin boyut değişkeni olan N sayısı kullanıcıdan girilmesi isteniyor.
- "array-al()" fonksiyonu çağrılarak key ve lock arraylerinin kullanıcıdan girilmesi bekleniyor
- key ve lock eşlemesi için "key-lock-match()" fonksiyonu çağrılıyor
- sonra eşleşen key ve lock çiftleri yazdırılıyor.

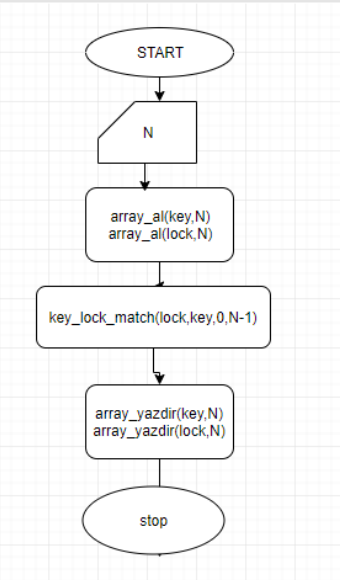

Figure 1.1 main fonksiyonu

1.1.2 Swap Fonksiyonu

Swap fonksiyonuna gelen iki değişkenin değerini birbiriyle değiştiriyor.

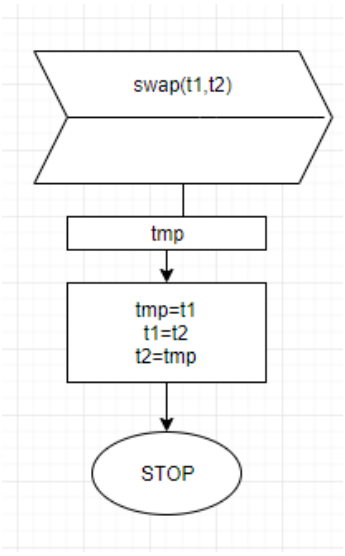

Figure 1.2 swap fonksiyonu

1.1.3 Array-al ve Array-yazdir Fonksiyonları

Array-al fonksiyonu kullanıcıdan bir dizi girmesini bekler. Array-yazdir fonksiyonu diziyi kullanıcıya göstermek için kullanılır.

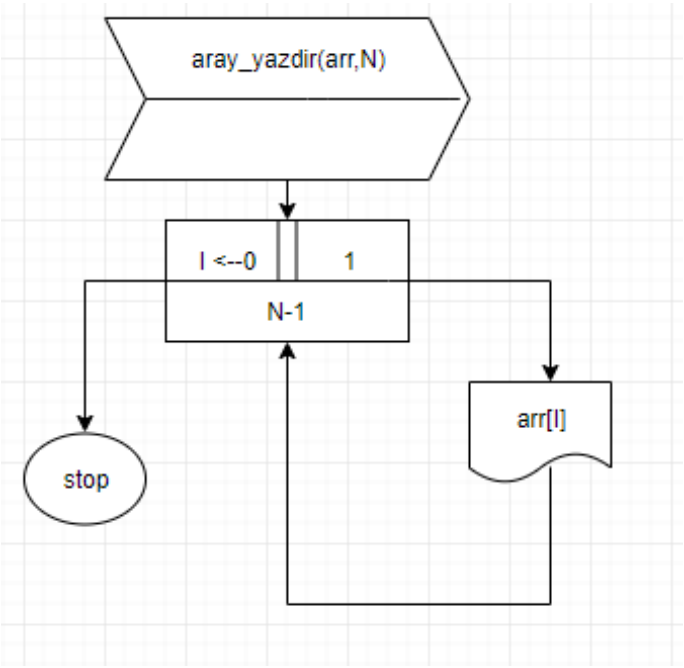

Figure 1.3 Array-yazdir fonksiyonu

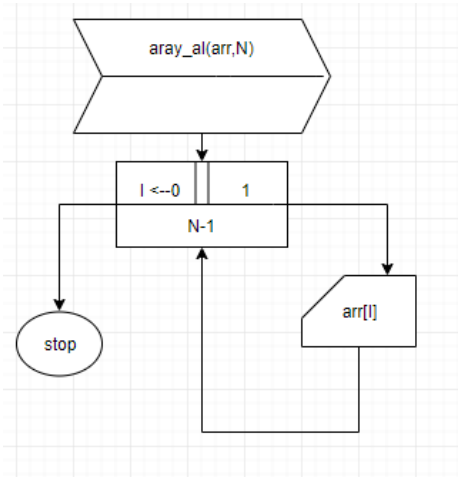

Figure 1.4 Array-al fonksiyonu

1.1.4 Key-Lock-Match Fonksiyonu

Bu fonksiyona key ,lock değerleri ile sıralama dizinin başını gösteren p değeri ve sıralama yapılacak olan dizinin sonunu gösteren r değerleri ile kullanılır.

- önce key değeri random olarak seçilir
- bu key değeri için "partition" fonksiyonu çağrılarak lock fonksiyonunda düzenleme yapılır ve lock dizisindeki değeri belirlenir
- key diziside çıkan lock[q] değerine göre düzenlenir
- özyinelemeli olarak keyden küçük key değerleri ve keydenbüyük key değerleri içinde aynı düzenlemeler yapılarak keylere uygun lock değerleri bulunur.

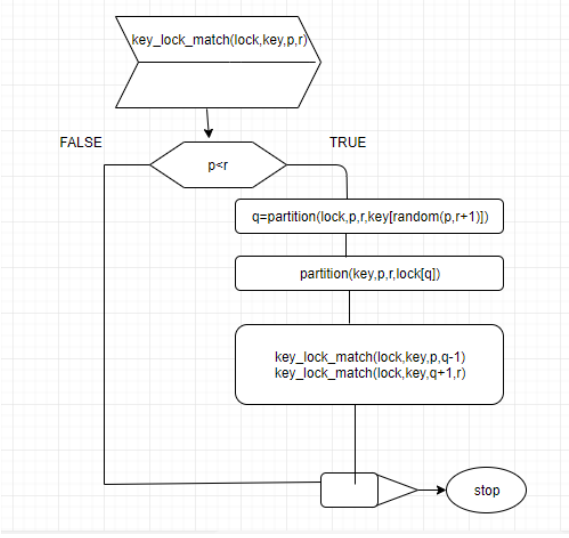

Figure 1.5 Key-Lock-Match fonksiyonu

1.1.5 Partition Fonksiyonu

Partition fonksiyonuna gelen dizi ,dizinin başlangıç ve bitiş indisi değeri, ve dizinin ona göre düzenlenecek olan key değeri ile kullanılır.Bu foksiyondaki "i" değişkeni key değerinde küçük olan değerlerin sınırını belirtmek için ve j ise keyden büyük olanları bulmak için kullanacağımız değişkenlerdir.

Bu fonksiyon çalıştığında dizi 3 farklı bölgeye ayrılır;

- Eğer p<=k<=i Array[k]<key
- Eğer i+1<=k<=j-1 Array[k]>key
- k=r Array[k]=key

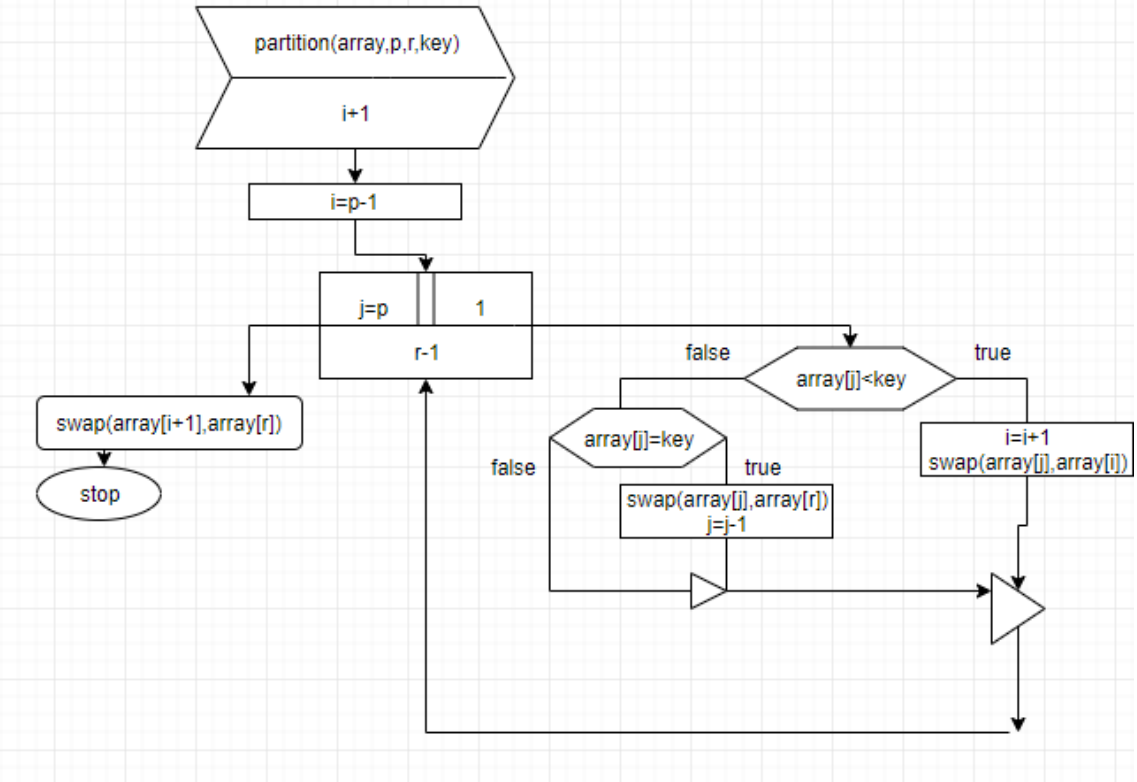

Figure 1.6 Partition fonksiyonu

2
UYGULAMA

2.1 Rastgele Sayılardan Oluşan Dizi Örneği

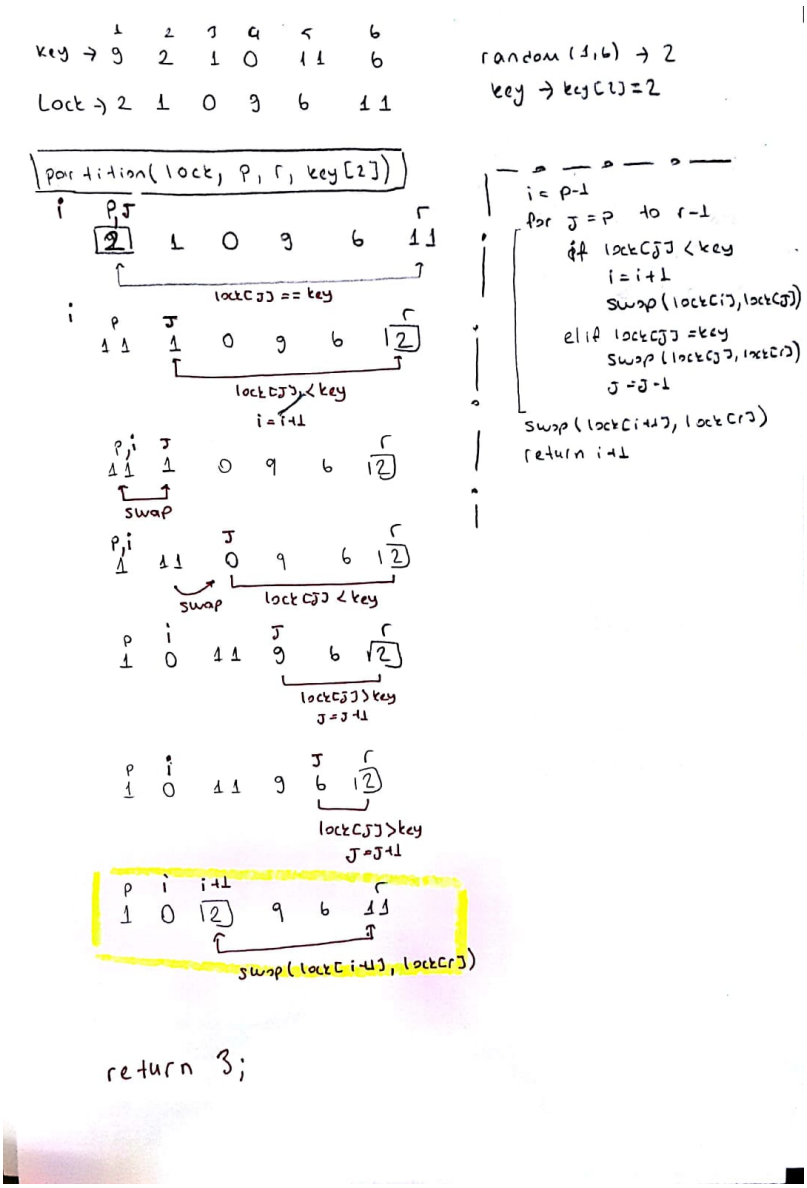

Figure 2.1 Partition fonksiyonu-lock için

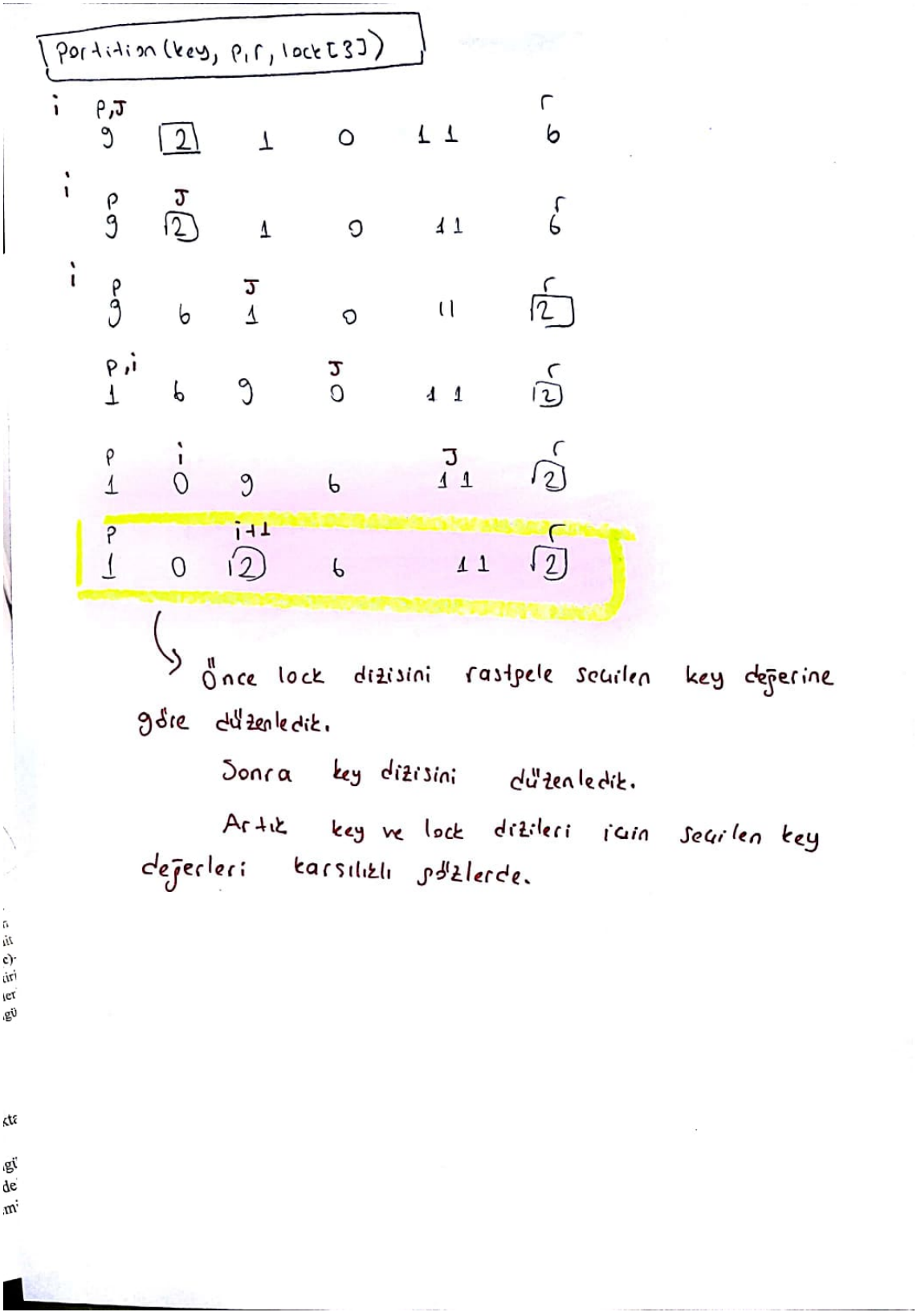

Figure 2.2 Partition fonksiyonu-key için

2.2 Sıralı Dizi Örneği

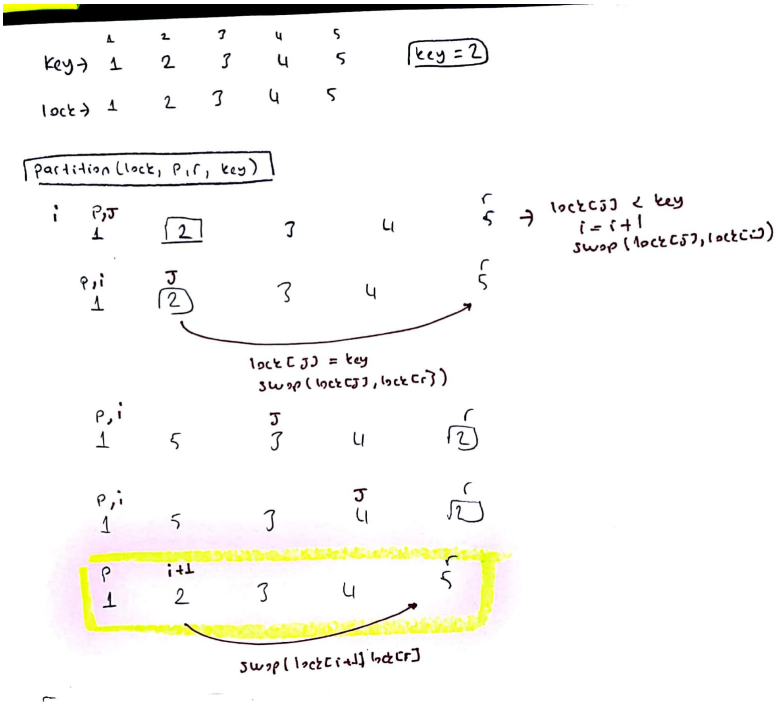

Figure 2.3 Partition fonksiyonu-lock için

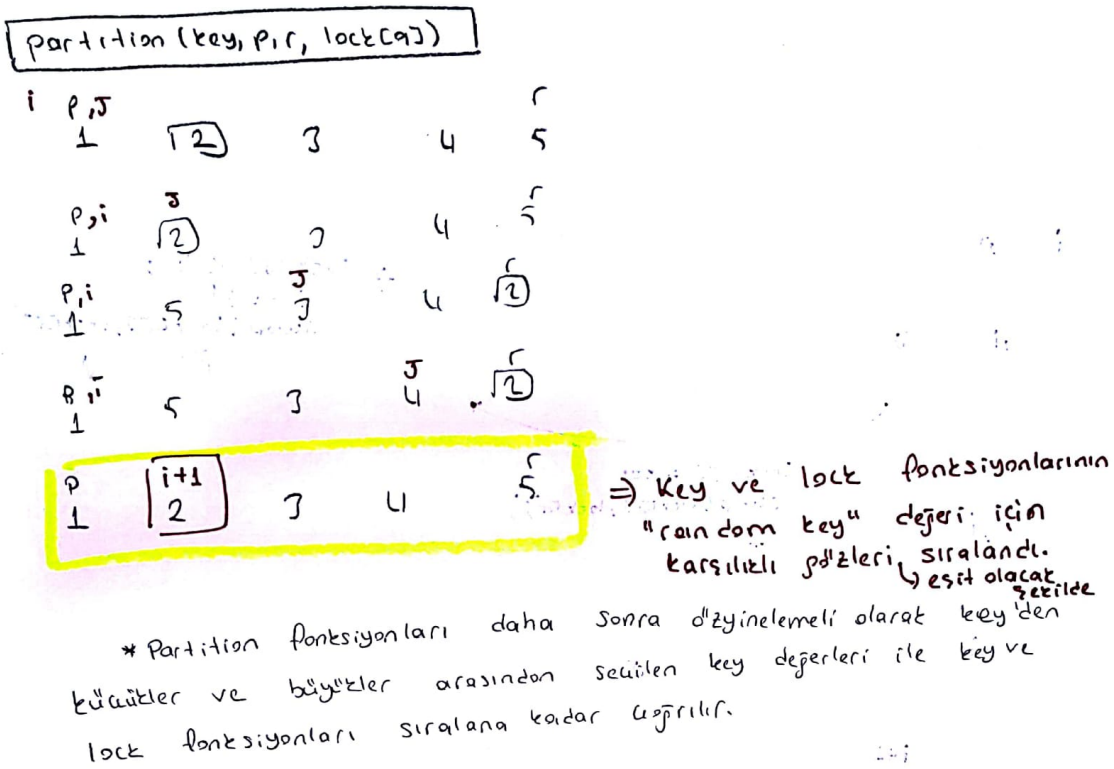

Figure 2.4 Partition fonksiyonu-key için

3
SONUÇ

3.1 Algoritmanın Analizi

Algoritma QuickSort benzeri bir algoritmadır, farkı pivot değerini her seferinde ilk değeri almak yerine burada random olarak belirliyoruz.Pivot rastgele belirlendiğinden girdi dizisi ayrımları ortalamada makul bir şekilde dengelenmiştir diyebiliriz.

Bu algoritmanın analizini key-lockmatch ve partition fonksiyonlarına bakarak çıkarabiliriz.Partition 'a giden key değeri diğer özyinelemelerden partition'a bu key değeri gitmez yani Partition'a en fazla n çağrı yapıyoruz.Partition'a her çağrı o(1) zaman alır ve for döngüsündeki iterasyonların sayısı ile orantılı olan zaman kadar süre alır.For döngüsünün içerisinde key değeri ile lock dizisinin diğer elemanları karşılaştırılır , biz bu karşılaştırmanın çalışma sayısını bulabilirsek ,yaptığımız matching işlemi boyunca for döngüsü için harcanan zamanı bulabiliriz.

- Temel işlem= for döngüsü içinde yapılan karşılaştırma

Amacımız bu karşılaştırma sayısını bulmaktır, bu karşılaştırma sayısına X diyelim, o zaman key-lock-match fonksiyonunun çalışması O(n+X) olur.

Bu X'i bulmak için algoritmanın key değeri ve lock dizisi elemanlarını karşılaştırdığını ve ne zaman karşılaştırmadığını anlamalıyız.Lock dizisi elemanları key değeri ile karşılaştırılır ve partition kısmı bittiğinde bu çağrıda kullanılan key değeri bir daha diğer elemanlarla karşılaştırılmaz.Yani "her eleman çifti bir kere karşılaştırılır " diyebiliriz.

Karşılaştırmanın , algoritmanın çalışması sırasında herhangi bir zamanda yer alıp almadığı;

Xij=Lock dizisindeki key ile lock[j] elemanın karşılaştırılması

$X = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} Xij$

Key dizisinde eleman key olarak seçilmeden önce hepsi aynı parçadadır ilk durumda.Bu yüzden key dizisinin her bir elemanının key değeri olarak seçilme olasılıkları aynıdır.Key dizimiz p-r+1 elemana sahiptir ve keyler random olarak seçildiğinden herhangi bir elemanın key olma olasılığı:

$$\frac{1}{p-r+1}$$

key ve lock[j] ile karşılaştırılma olasılığı=key'in gelme olasılığı+lock[j]'nin key değeri olarak gelme olasılığı:

$$\frac{2}{p-r+1}$$

O zaman X için aşağıdakini yazabiliriz:

$X=\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\frac{2}{p-r+1}=\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\frac{2}{k+1}<\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\frac{2}{k}=\sum_{i=1}^{n-1}O(lgn)=O(nlgn)$

şeklinde buluruz.

A
AKIŞ DİYAGRAMI

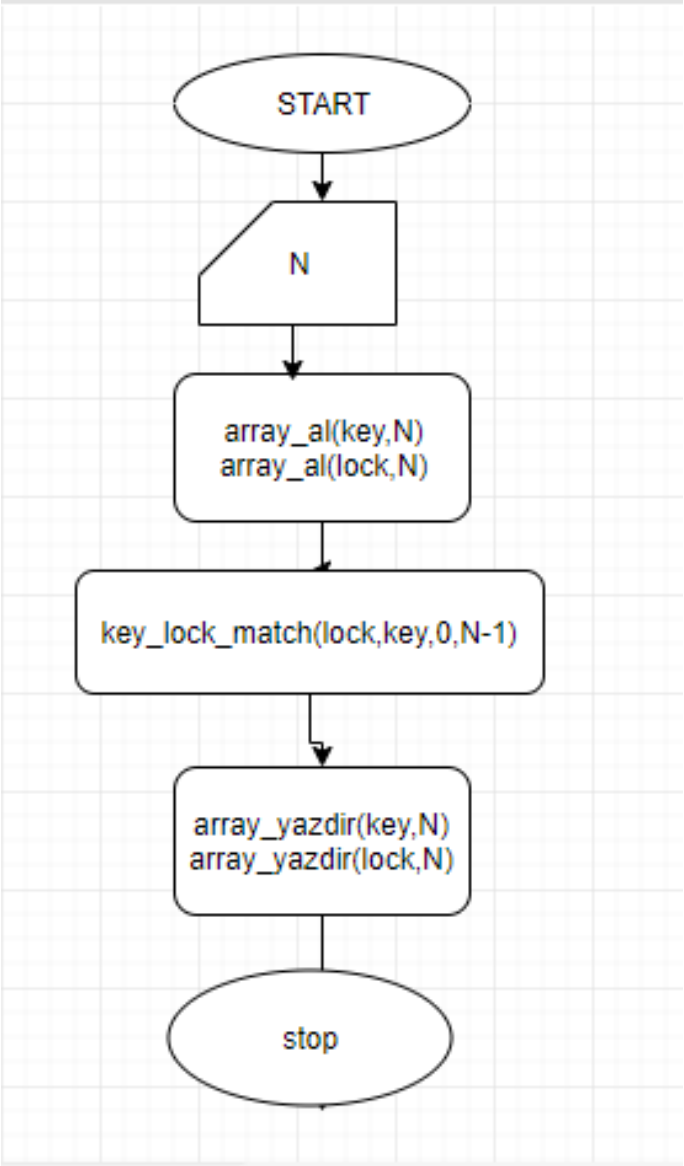

Figure A.1 main fonksiyonu

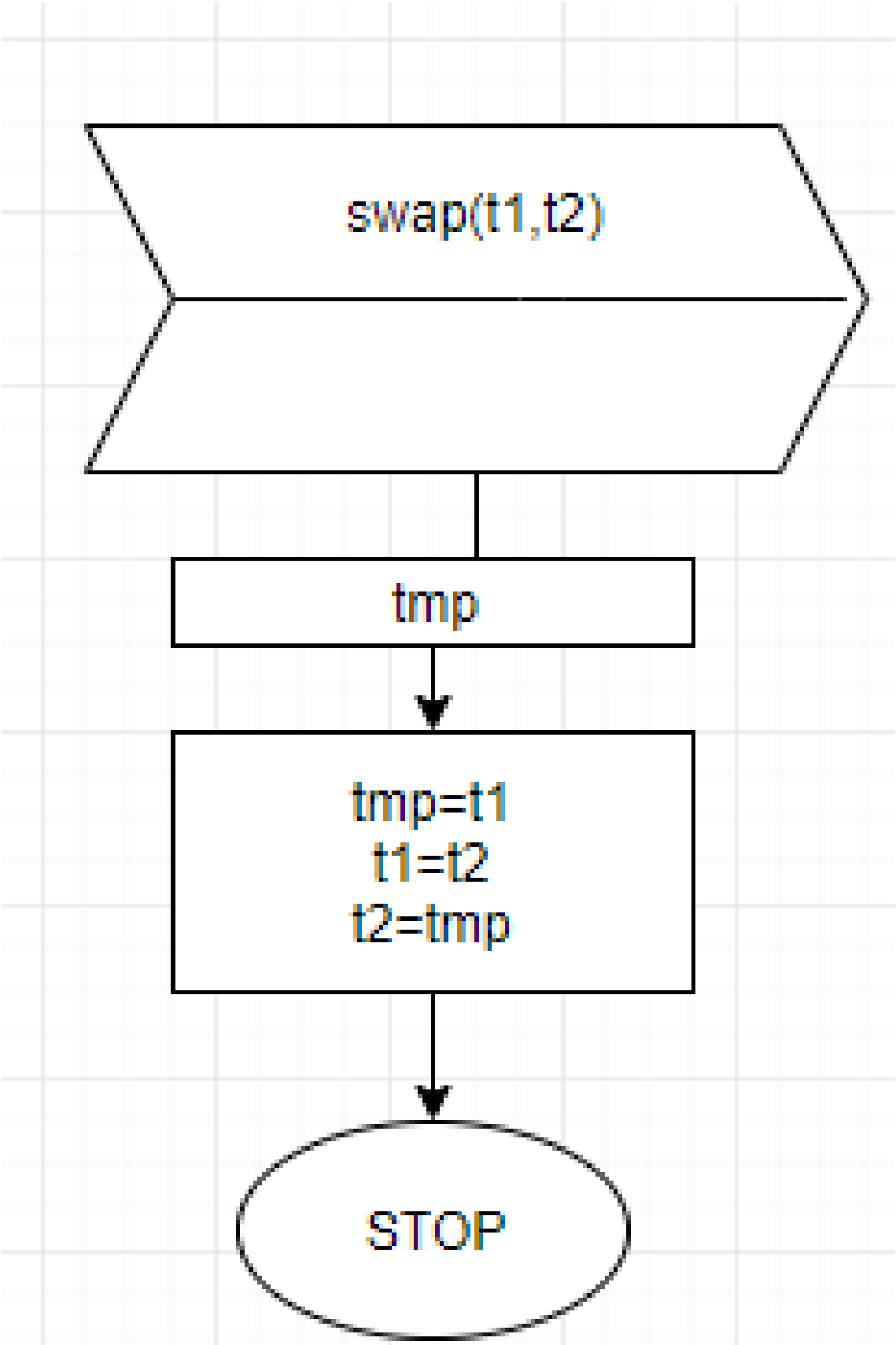

Figure A.2 swap fonksiyonu

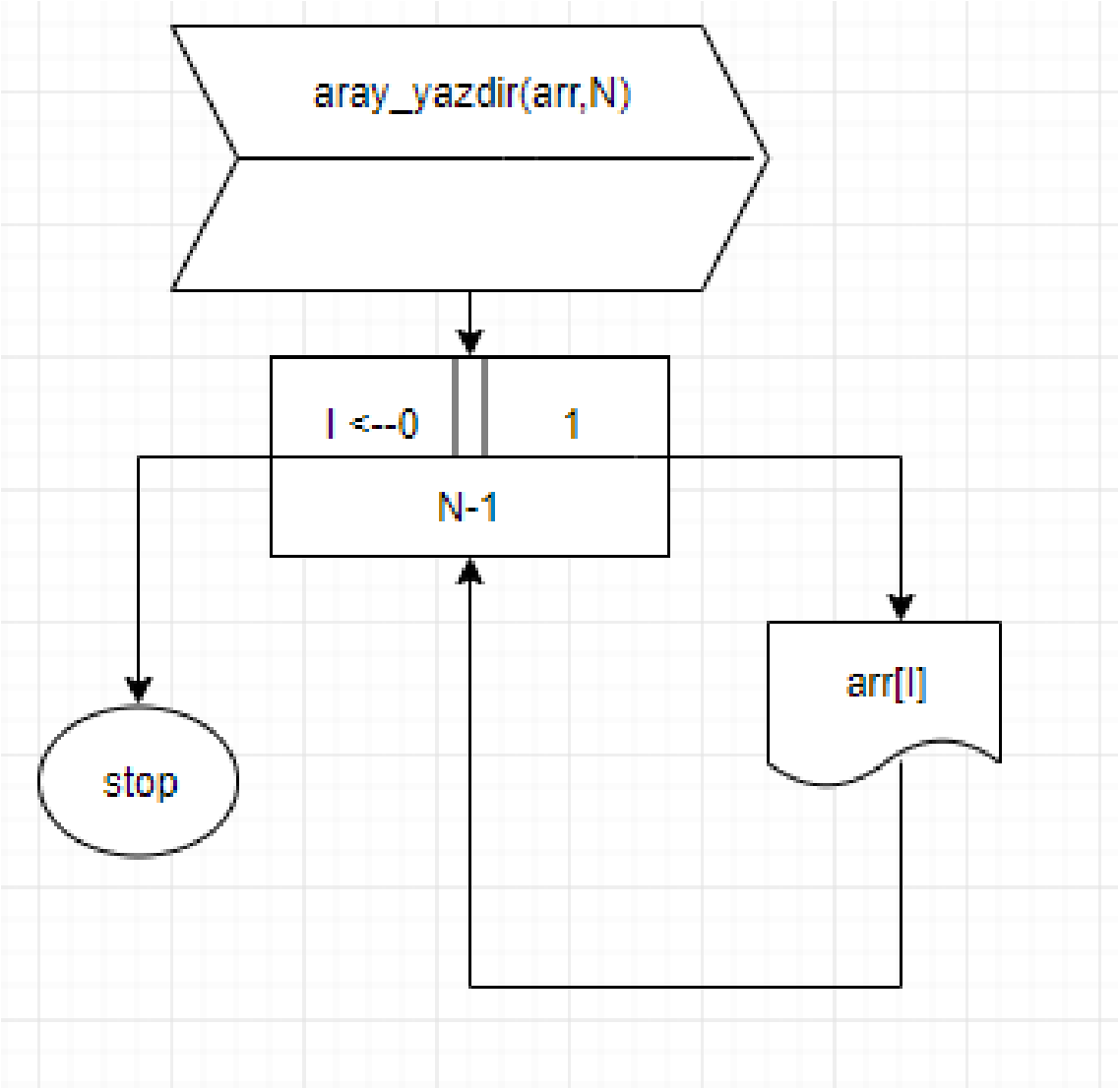

Figure A.3 Array-yazdir fonksiyonu

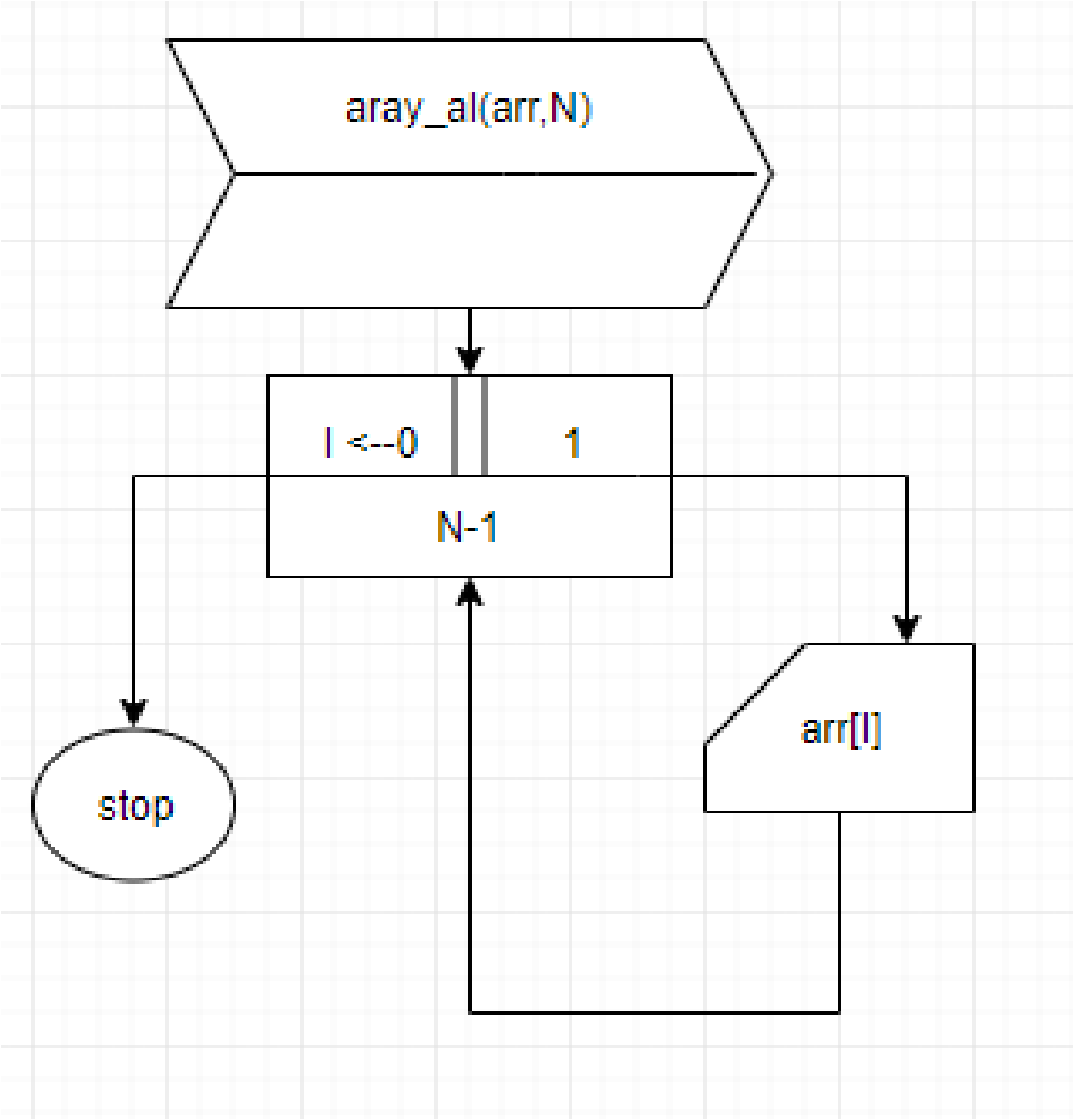

Figure A.4 Array-al fonksiyonu

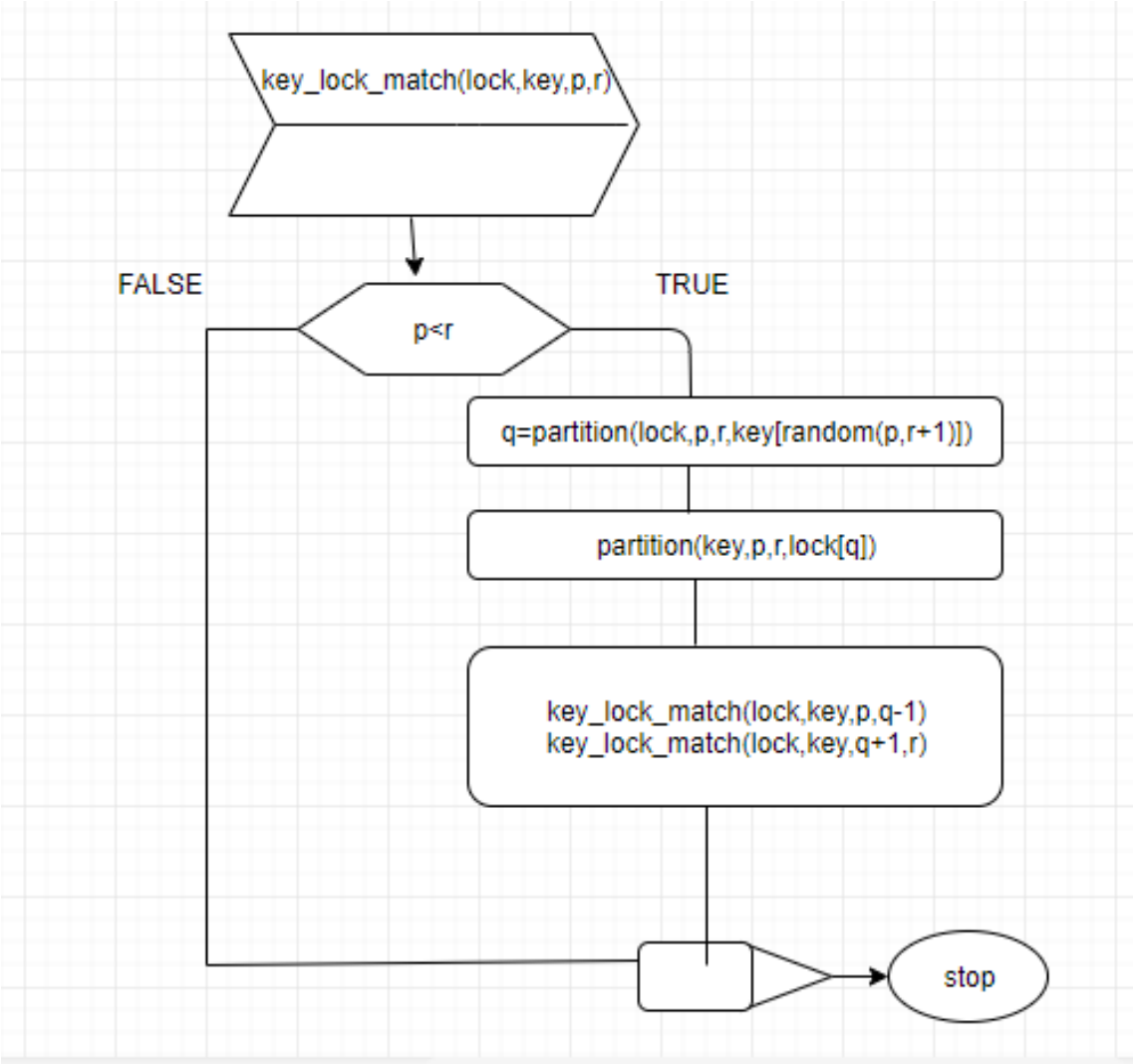

Figure A.5 Key-Lock-Match fonksiyonu

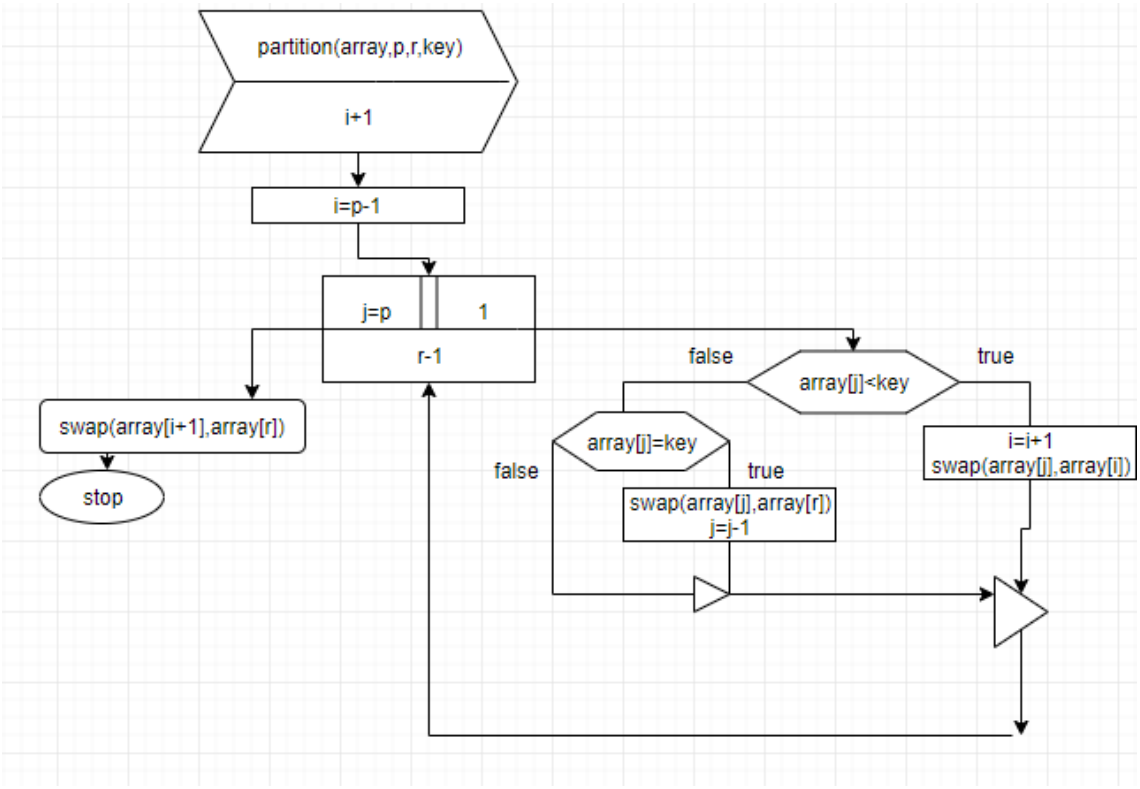

Figure A.6 Partition fonksiyonu

B
C KODU

```c
//17011604-Havvanur Dervisoglu-Algoritma Analizi 1.Ödev
//ödev:key dizisinden rastgele secilen anahtari lock dizisini keye göre düzenliyerek ,ve key dizisini de lock[x] e göre duzenleyerek sonra
//özyinelemeli olarak keyden büyük key degerleri ve kücük key degerleri icinde ayni islemleri yaparak "kilit-anahtar uyumunu" bulma
#include<stdio.h>
#include<stdlib.h>
#include<time.h>

//swap islemi icin kullanacagimiz fonksiyon
//fonksiyona verilen iki degeri ,birbiriyle degistirir
void swap(int* t1, int* t2)
{
    int tmp = *t1;
    *t1= *t2;
    *t2= tmp;
}
```

```c
//parçalama fonksiyonu her seferinde girilen key degerinin yerini dönüyor
int partition(int array[],int p,int r,int key){
    //p dizinin dusuk indisi ilk durumda 0, r ise buyuk indisidir
    //key degerine göre array dizisi duzenlenecektir
    //i degiskeni keyden kucuk olan elemanlarin sinir inidisi,yani i.elemandan öncekiler keyden kucuk
    //j degeri ise keyden buyuk olan elemanlarin sinir degeridir ,yani i+1 ile j-1 arasindakiler keyden buyuk
    int i=p-1,j;

    for(j=p;j<=r-1;j++){
        if(array[j]<key){
            //eger key degeri büyükse girilen dizinin j. elemanindan
            //keyden küçük olan indisleri tutan i degerini arttir
            //sonra küçüklerin yanina koy yani swap yapiyoruz
            i++;
            swap(&array[i], &array[j]);

        }
        else if(array[j]==key){
            //eger esitse key degerine bunu son yere koy en son yani dizinin sonuna
            // diger elemanlari da kontrol devam etmek için j'yi bir azaltki degistirdigi sayiyi da kontrol edebilsin
            swap(&array[j], &array[r]);
            j--;
        }
    }//döngünün bitme kosulu j=r dir, döngünün sonunda dizideki bütün girdiler keyden buyuk olnlar ,keyden buyuk olanlar ve key olarak 3'e böldü
    //artik key degerimizin yeri kucuklerden bir sonraki olarak yerini koyuyoruz
    //key degeri ondan kucukler ve ondan buyuklerin arasina yerlestiridk
    swap(&array[i+1], &array[r]);
    //dönus degeri keyin yeri bölmeye burdan basliycaz
    return (i+1);
}
```

```c
void key_lock_match(int lock[],int key[],int p,int r){
    //key ve lock degerleri giris degeri olarak aliniyor
    //p dizinin kucuk,baslangic indisi,r ise buyuk,bitis indisi

    if(p<r){//özyinelemeden çikma için bu kosulu kullandim

        //random olarak bir key degeri seciliyor sonrasinda lock dizisinde
        //bu key dizisinin yeri berilirleniyor
        int q=partition(lock,p,r,key[(p+ (rand() % (r - p + 1)))]);
        //sonra bu deger ,lock[q] key degeri key dizisndede duzenleniyor
        //bu drumda key dizisinde ve lock dizisinde lock[q] ve key[rand] degerleri karsilikli
        partition(key,p,r,lock[q]);

        //daha sonrasinda keyden öncekiler yani kucuk olanlar arasindan anahtar secilerek özyineleme olarak cagirilarak sýralanir
        key_lock_match(lock,key,p,q-1);
        //keyden büyük key degeri icinde  özyinelemeli olarak siralama yapilir
        key_lock_match(lock,key,q+1,r);
    }
}
void array_yazdir(int arr[],int n){
    //dizi yazdirma fonksiyonu
    int i;
    for(i=0;i<n;i++){
        printf("%d\t",arr[i]);
    }
}
void array_al(int arr[],int n){
    //diziyi kullanicidan alma fonksiyonu
    int i;
    for( i=0;i<n;i++){
        scanf("%d",&arr[i]);
    }
}
```

```c
int main(){
    //n burada key lock eleman sayisi
    //i döngüdegiskeni
    //key anahtar degerleri tutan array
    //lock kilit degerleri tutan array
    int n,i,key[100],lock[100];

    //N degeri kullanicidan aliniyor
    printf("N?\n");
    scanf("%d",&n);

    //key ve lock degerleri kullanicidan aliniyor
    printf("key?\n");
    array_al(key,n);
    printf("lock?\n");
    array_al(lock,n);

    //sonra key lock esleme yapiyoruz
    key_lock_match(lock,key,0,n-1);

    //match olan key ve lock degerlerini yazdiriyoruz
    printf("key::\n");
    array_yazdir(key,n);
    printf("\n");
    printf("lock::\n");
    array_yazdir(lock,n);

    printf("\n");
    system("pause");
    return 0;
}
```

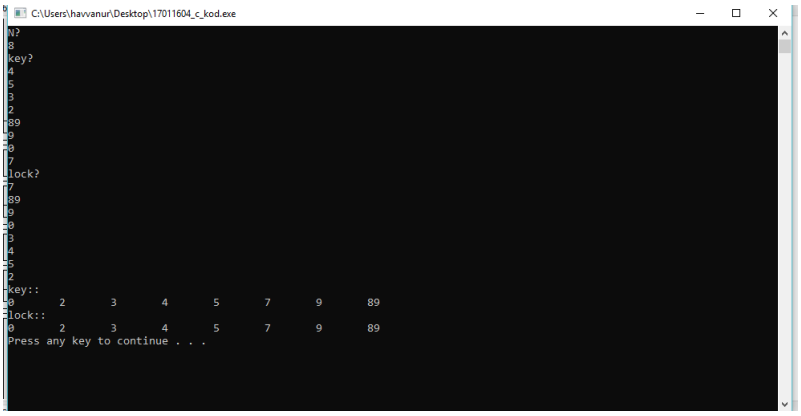

Figure B.1 Örnek Sonuç