İçindekiler

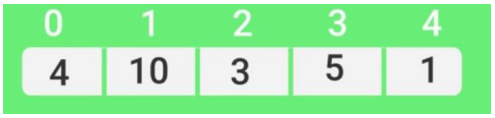
ALGORİTMA

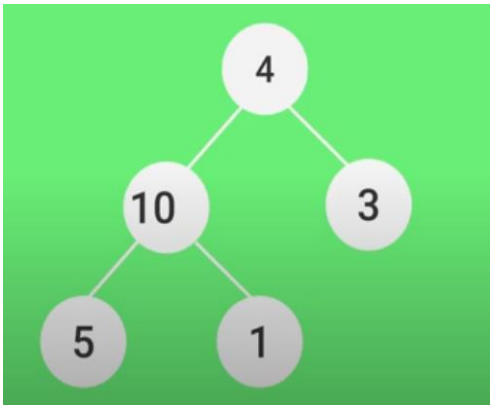
Verinin hafızada sıralı tutulması için geliştirilen sıralama algoritmalarından (sorting algorithms) bir tanesidir. Yıpınlama sıralaması, arka planda bir yığın ağacı (heap) oluşturur ve bu ağacın en üstündeki sayıyı alarak sıralama işlemi yapar.

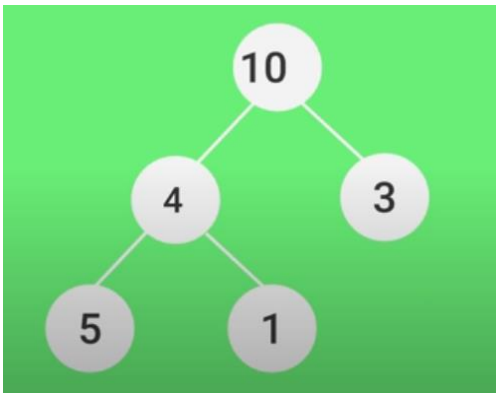
```
void heapolustur(int *a,int size){
    int i,m;
    m=(size-2)/2;

    for(i=m ; i>=0;i--)
        maxheapify(a,i,size);
}

void maxheapify(int *a,int i,int size){
    int largest=i,left,right;
    left=(2*i)+1;
    right=(2*i)+2;
    if(left<size && a[left]>a[largest])
    largest=left;
    if(right<size && a[right]>a[largest])
    largest=right;
    if(largest!=i)
    {
    swap(&a[largest],&a[i]);
    maxheapify(a,largest,size);
    }
}
```

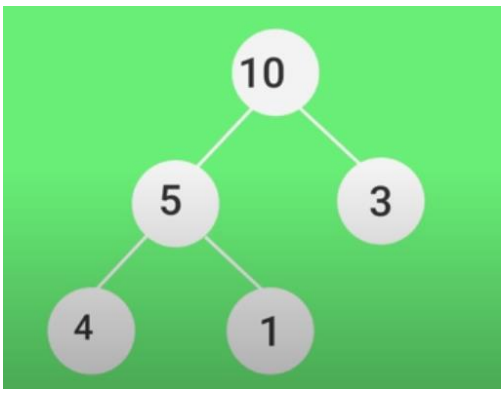
YUKARIDAKİ KODUN YAPTIĞI İŞLEMLER ŞU ŞEKİLDEDİR :

Örnek dizi:



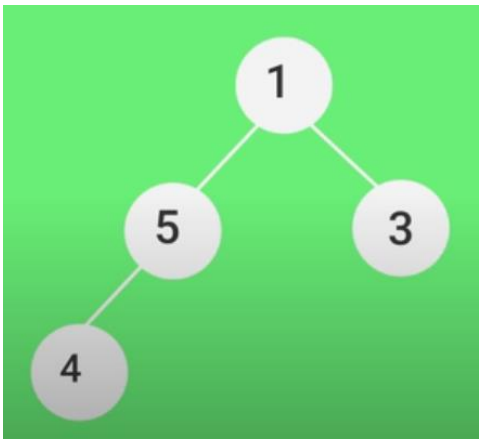




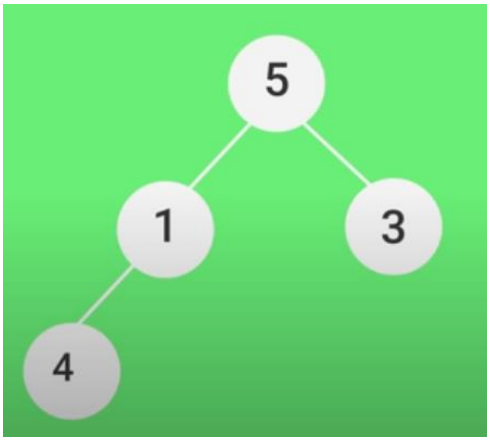



```
void heapsort(int *a,int size){
    heapolustur(a,size);
    while(size>1)
    {
    swap(&a[0],&a[size-1]);
    --size;
    maxheapify(a,0,size);
    }
}
```

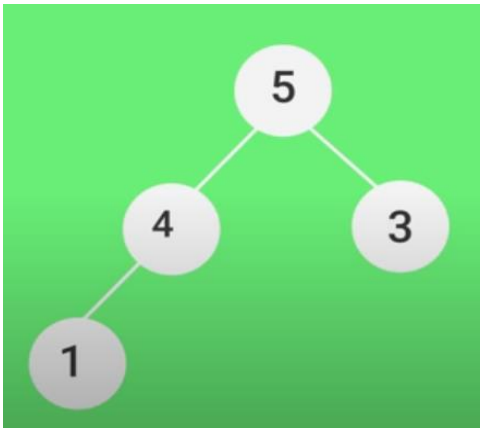
YUKARIDAKİ KODUN YAPTIĞI İŞLEMLER ŞU ŞEKİLDEDİR :



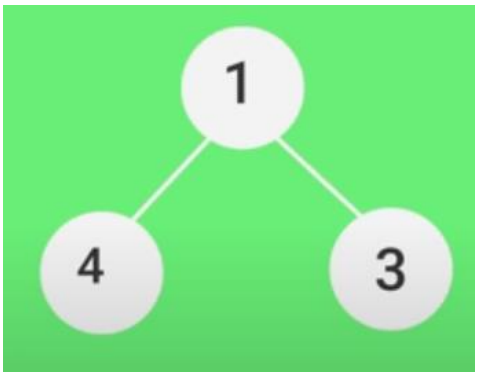
0	1	2	3	4
1	5	3	4	10



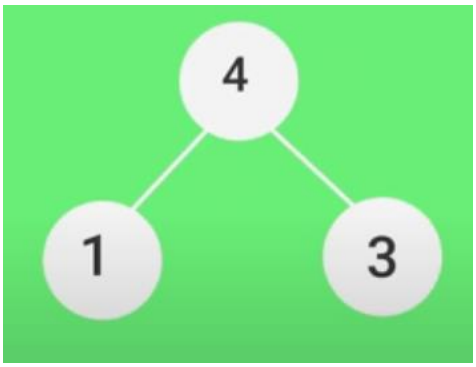
0	1	2	3	4
5	1	3	4	10



0	1	2	3	4
5	4	3	1	10

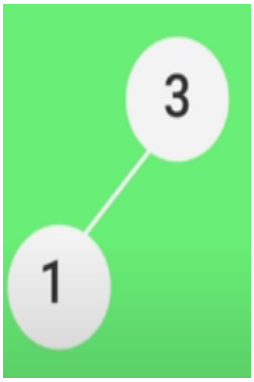


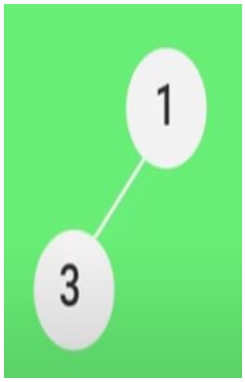
0	1	2	3	4
1	4	3	5	10



0	1	2	3	4
4	1	3	5	10

0	1	2	3	4
3	1	4	5	10





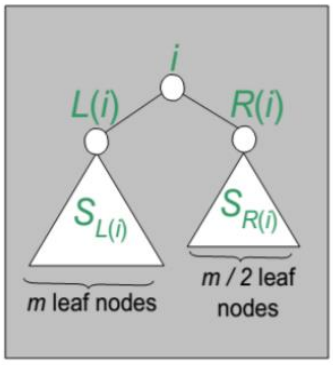
0	1	2	3	4
1	3	4	5	10

ANALİZ

ZAMAN KARMAŞIKLIĞI

Heapify() İçin Zaman Karmaşıklığı Analizi:

- Let m be the number of leaf nodes in $S_{L(i)}$
- $|S_{L(i)}| = \underbrace{m}_{\text{ext}} + \underbrace{(m-1)}_{\text{int}} = 2m-1$;
- $|S_{R(i)}| = \underbrace{m/2}_{\text{ext}} + \underbrace{(m/2-1)}_{\text{int}} = m-1$



- $|S_{L(i)}| + |S_{R(i)}| + 1 = n$

  $(2m-1) + (m-1) + 1 = n \Rightarrow m = (n+1)/3$

  $|S_{L(i)}| = 2m - 1 = 2(n+1)/3 - 1 = (2n/3+2/3) - 1 = 2n/3 - 1/3 \leq 2n/3$

- $T(n) \leq T(2n/3) + \Theta(1) \Rightarrow \boxed{T(n) = O(\lg n)}$

By case 2 of Master Thm

BuildHeap() İçin Zaman Karmaşıklığı Analizi:

$$- T(n) = \sum_{h=0}^{\lg n} O(n/2^h)O(h) = O(n \times \left[\frac{1}{2} + \frac{2}{4} + \cdots \frac{\lg n}{n}\right])$$

$$\sum_{h=0}^{\lfloor \lg n \rfloor} \left\lceil \frac{n}{2^{h+1}} \right\rceil O(h) = O\left(n \sum_{h=0}^{\lfloor \lg n \rfloor} \frac{h}{2^h}\right)$$

$$O\left(n \sum_{h=0}^{\lfloor \lg n \rfloor} \frac{h}{2^h}\right) = O\left(n \sum_{h=0}^{\infty} \frac{h}{2^h}\right) \quad \sum_{h=0}^{\infty} \frac{h}{2^h} = \frac{1/2}{(1-1/2)^2} = 2 .$$

$$T(n) = O(n)$$

Heapsort İçin Toplam Zaman Karmaşıklığı:

BuildHeap(): O(n)

Heapify() : O(log n) (n-1) kez çağrılır

HeapSort() :

T(n) = O(n) + (n-1)*O(log n)

T(n) = O(n) + O(n logn) => o(n log n)

Time Complexity	
Best	O(nlog n)
Worst	O(nlog n)
Average	O(nlog n)
Space Complexity	O(1)
Stability	No

Heap Sort tüm durumlar için (best case, average case, and worst case) O(n logn) zaman karmaşıklığına sahiptir.

Nedeni;

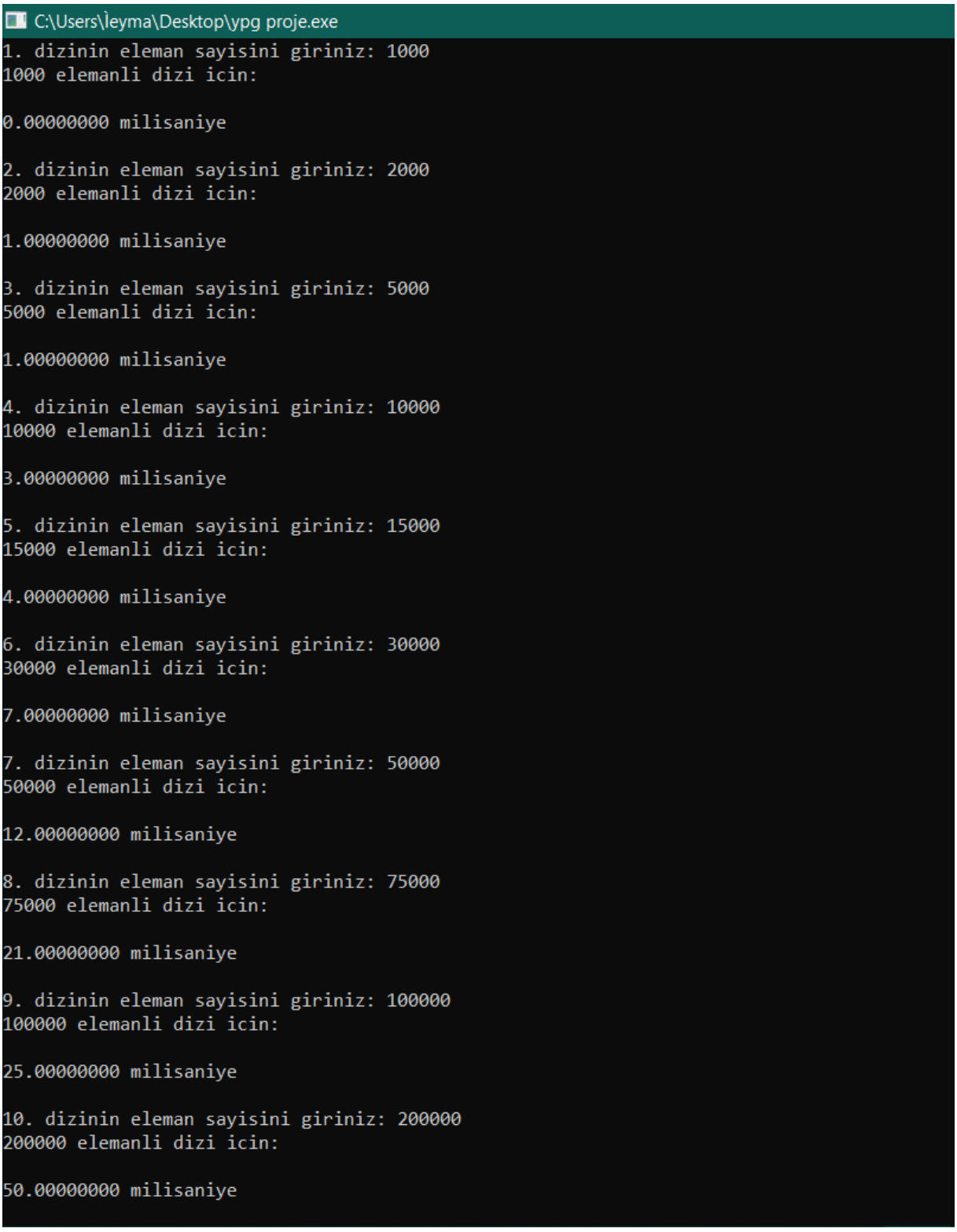
N eleman içeren bir binary ağacın yüksekliği logn dir.Alt ağaçları max_heaps şeklinde olan bir öğeyi sıralamak için elemanı sol ve sağ çocukları ile karşılaştırmaya devam etmeli ve her iki çocuğunun da kendisinden daha küçük olduğu bir noktaya gelinceye kadar aşağı doğru itmeliyiz.En kötü senaryoda çoklu log(n) karşılaştırmaları ve takasları yaparak bir ögeyi kökten yaprak düğüme taşımamız gerekecek.build_max_heap aşaması sırasında,bunu n/2 öge için yapıyoruz bu nedenle build_heap adımının en kötü durum karmaşıklığı n/2*logn ~ nlogn dir.

HAFIZA KARMAŞIKLIĞI

Heapsort için hafıza karmaşıklığı O(n) dir.

UYGULAMA

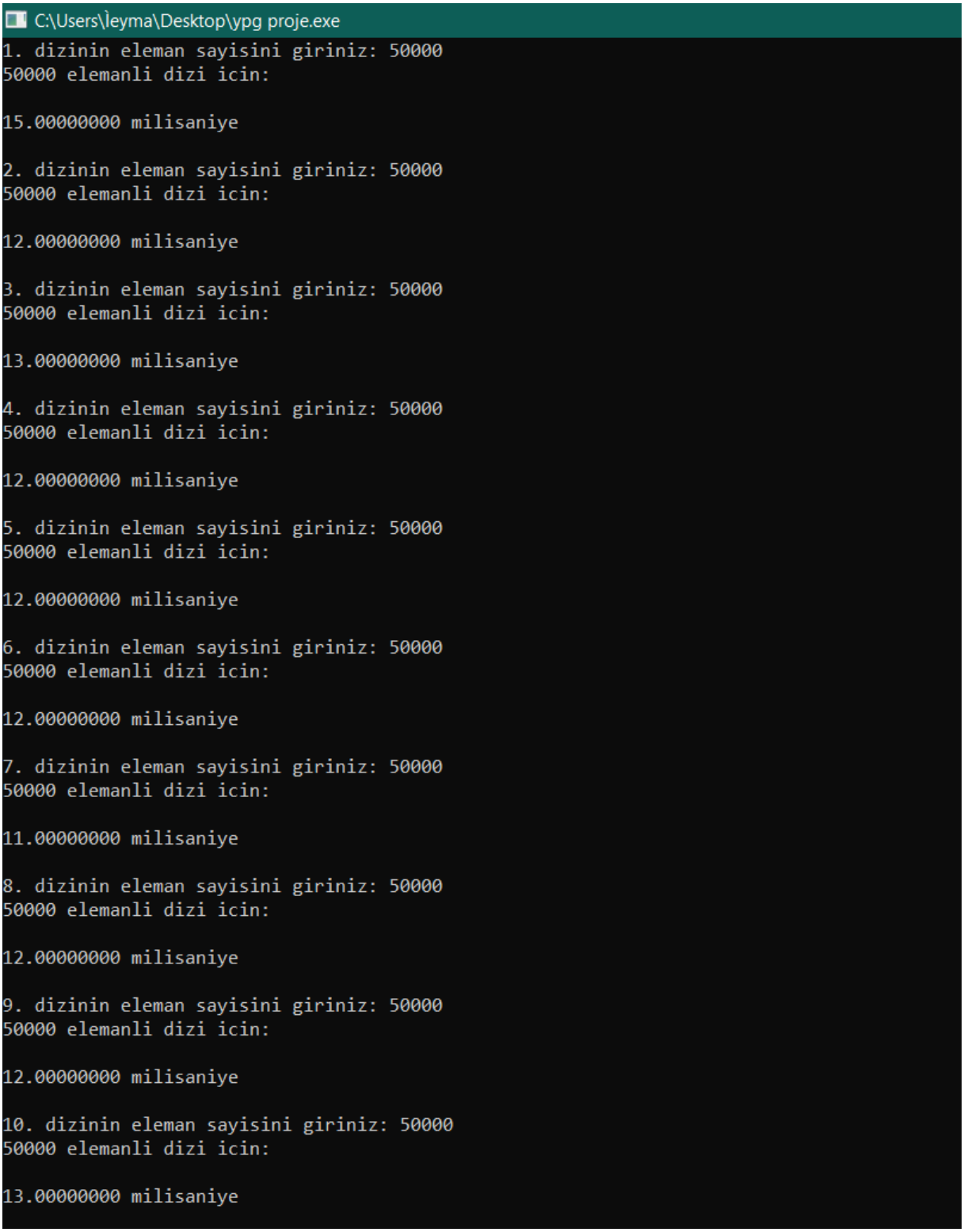
Uygulama ekran çıktısı şu şekildedir:

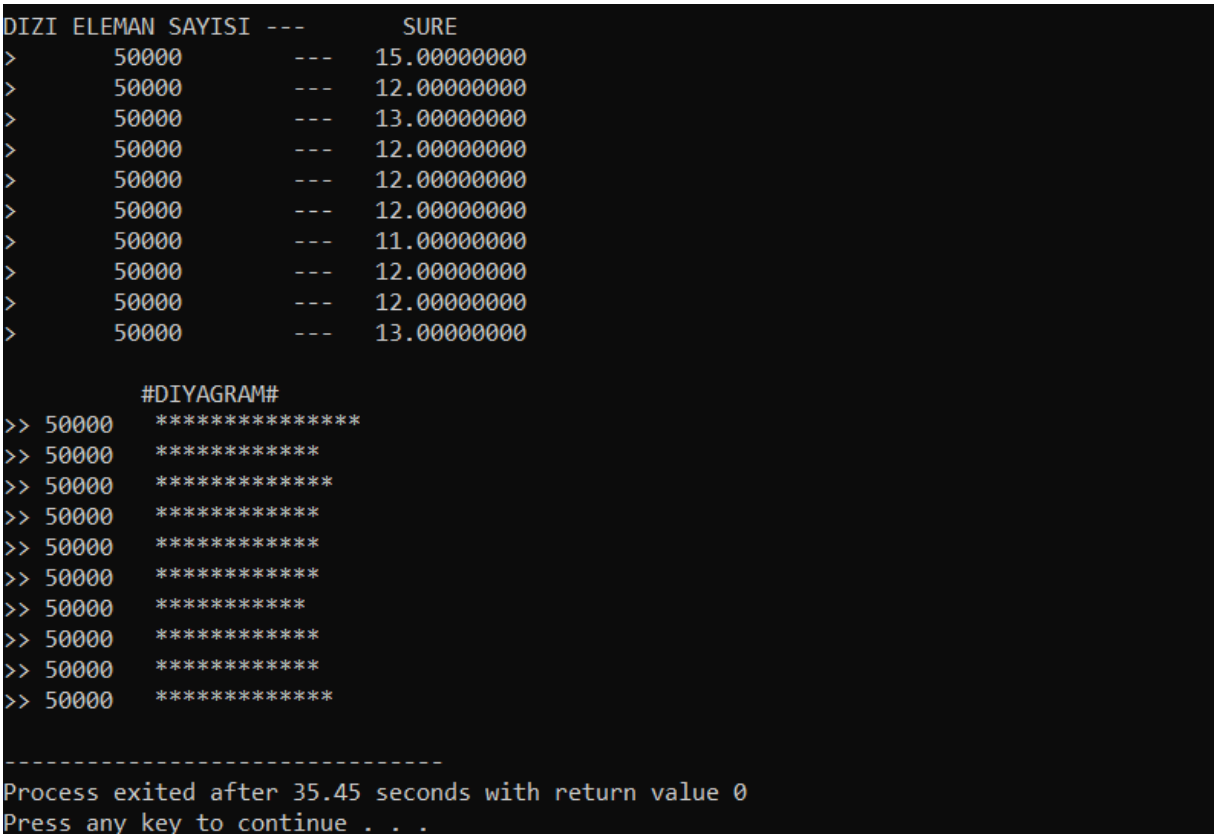


```
DIZI ELEMAN SAYISI ---       SURE
>        1000        ---   0.00000000
>        5000        ---   2.00000000
>       10000        ---   4.00000000
>       20000        ---   7.00000000
>       30000        ---   10.00000000
>       50000        ---   18.00000000
>       75000        ---   29.00000000
>      100000        ---   41.00000000
>      150000        ---   58.00000000
>      200000        ---   79.00000000

          #DIYAGRAM#
>>  1000
>>  5000   **
>> 10000   ****
>> 20000   *******
>> 30000   **********
>> 50000   ******************
>> 75000   ****************************
>> 100000   ************************************
>> 150000   ******************************************************
>> 200000   ************************************************************************

--------------------------------
Process exited after 56.5 seconds with return value 0
Press any key to continue . . .
```

Aynı elemanlı dizileri farklı zamanlarda sıralayabilir.Bu elemanların dizi içinde ne kadar sıralı dağıldığıyla ilgilidir.

Örneğin;





C PROGRAM KODU

```
#include<stdio.h>

#include <time.h>

#include <conio.h>

#define MAX 500000

void createandbuildheap(int *arr,int size){

        int i;

        for(i=(size-2)/2; i>=0;i--)

        maxheapify(arr,i,size);

}

void maxheapify(int *arr,int i,int size){

         int largest=i,left,right;

        left=(2*i)+1;

        right=(2*i)+2;

         if(left<size && arr[left]>arr[largest])
```

```
        largest=left;
        if(right<size && arr[right]>arr[largest])
        largest=right;
        if(largest!=i){
                swap(&arr[largest],&arr[i]);
                maxheapify(arr,largest,size);
        }
}
void heapsort(int *arr,int size){
        createandbuildheap(arr,size);
        while(size>1){
                swap(&arr[0],&arr[size-1]);
                --size;
                maxheapify(arr,0,size);
        }
}
void swap(int *a,int *b){
        int temp;
        temp=*a;
        *a=*b;
        *b=temp;
}

void dizi_doldur(int a[],int n){
        int i;
        for(i=0;i<n;i++){
                a[i]=1+rand()%100;
        }
}

int main(){
```

```
int i,j,arr[MAX],n,k=0;
srand(time(NULL));
clock_t start;
clock_t end;
int diyagram[10],tekrar=10;
double sonuc[10];
while(k<tekrar){
	printf("%d. dizinin eleman sayisini giriniz: ",k+1);
	scanf("%d",&n);
	diyagram[k]=n;
	printf("%d elemanli dizi icin: \n",n);
	dizi_doldur(arr,n);
	/*printf("Sirasiz random dizi:\n");
	for(i=0;i<n;i++){
		printf(" %d ",arr[i]);
	}*/
	start=clock();
	heapsort(arr,n);
	end=clock();
	double time_passed = ((double)(end-start)/CLOCKS_PER_SEC)*1000;
	sonuc[k]=time_passed;
	/*printf("Random dizinin siralanmis hali:\n");
	for(i=0;i<n;i++){
		printf(" %d ",arr[i]);
	}*/
	printf("\n%.8lf milisaniye",time_passed);
	printf("\n\n");
	k++;
}
printf("DIZI ELEMAN SAYISI ---	SURE\n");
for(i=0;i<tekrar;i++){
```

```
                printf(">     %6d       ",diyagram[i]);
                printf("---");
                printf("   %.8lf\n",sonuc[i]);
        }
        printf("\n");
        printf("         #DIYAGRAM#        \n");
        for(i=0;i<tekrar;i++){
                printf(">> %5d   ",diyagram[i]);
                for(j=0;j<sonuc[i];j++){
                        printf("*");
                }
                printf("\n");
        }
return 0;
}
```

KAYNAKÇA

http://bilgisayarkavramlari.com/2008/08/09/yiginlama-siralamasi-heap-sort/

https://www.hanmurat.com/wp-content/uploads/2019/06/4.Hafta-Algoritma-Analizi_S%C4%B1ralama-Algoritmalar%C4%B1-v1_degisti.pdf

https://www.programiz.com/dsa/heap-sort