

**Gebze Technical University  
Computer Engineering**

**CSE 222 - 2018 Spring**

**HOMEWORK 5 REPORT**

**AHMET MUZAFFER DÜLGER  
131044082**

Course Assistant: Fatma Nur ESİRCİ

# 1 Double Hashing Map

## 1.1 Pseudocode and Explanation

Write pseudocode and explanation about code design. Indicate what you are using that interfaces, classes, structures, etc.

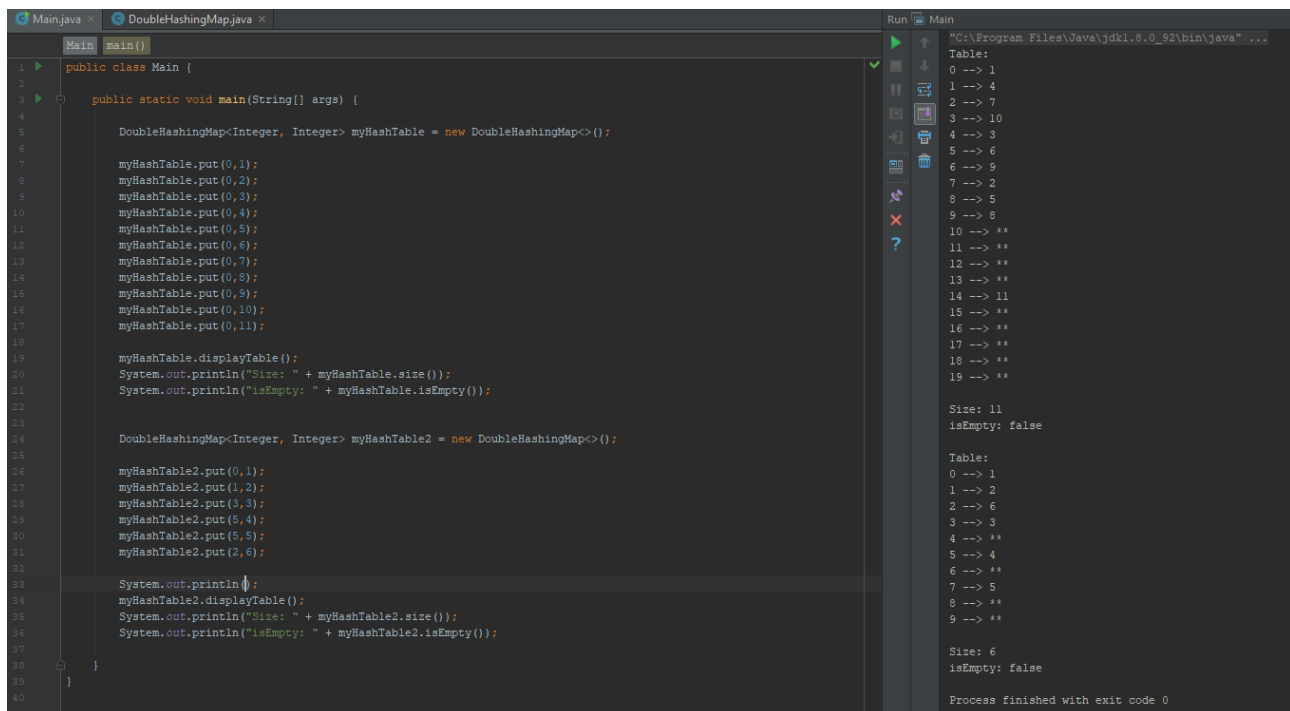
```
begin
    hashValue hesaplanır. (hashCode % table size)
    step hesaplanır. (table[hashValue] dolu ise kaç adım gidilecek)

    eğer eleman sayısı table size'dan büyükse resize yapılır.

    eğer table[hashTable] boş ise
        yeni entry yapılıp buraya atılır
    değilse
        while(table[hashValue] null değilken)
            begin
                hashValue step ile toplanır
                eğer hashValue table size'ı aşarsa
                    hashValue table size'a göre modu alınır
            end
        yeni entry yapılıp table[hashValue] atılır.
End
```

Bu partta bizden Map interface'ini implement ederek open addressing hash table yapmamız isteniyordu. Çakışma olma durumunda ise double hashing yöntemini kullanmamız isteniyordu. Map interface'ini implement ettiğimde metotlarını override etmem gerekiyordu. Ancak ben sadece kullanacağım metotları override ettim. Diğer metotlar ya boş duruyor ya da null return ediyor. numKeys adında array'imde kaç tane eleman olduğunu tutan integer bir değer var. Size metodun da bu numKeys'i return ediyorum. isEmpty'de ise numKeys 0 ise true, değilse false return ediyorum. put metodun da ise önce hashCode'un array boyutuma göre modunu alıyorum. Array'de bu yer boş ise parametre olarak verilen key ve value ile yeni bir entry oluşturup bu alana ekliyorum. Eğer dolu ise devreye step giriyor. Step'i hesaplamak için  $7 - (\text{key} \% 7)$  hesabını kullandım. Bunu hesapladıktan sonra array[hashValue] null olana kadar hashValue'ya step i ekliyorum ve eklediğim value'yu return ediyorum. Ayrıca array'in dolma ihtimaline karşılık resize metodu yazdım. Bu metot array'in boyutunu iki katına çıkarıyor. Array'i console'a yazdırmak için de displayTable metodunu yazdım.

## 1.2 Test Cases



The screenshot shows an IDE with two tabs: 'Main.java' and 'DoubleHashingMap.java'. The 'Main.java' tab is active, displaying the following code:

```
1 public class Main {
2
3     public static void main(String[] args) {
4
5         DoubleHashingMap<Integer, Integer> myHashTable = new DoubleHashingMap<>();
6
7         myHashTable.put(0,1);
8         myHashTable.put(0,2);
9         myHashTable.put(0,3);
10        myHashTable.put(0,4);
11        myHashTable.put(0,5);
12        myHashTable.put(0,6);
13        myHashTable.put(0,7);
14        myHashTable.put(0,8);
15        myHashTable.put(0,9);
16        myHashTable.put(0,10);
17        myHashTable.put(0,11);
18
19        myHashTable.displayTable();
20        System.out.println("Size: " + myHashTable.size());
21        System.out.println("isEmpty: " + myHashTable.isEmpty());
22
23
24        DoubleHashingMap<Integer, Integer> myHashTable2 = new DoubleHashingMap<>();
25
26        myHashTable2.put(0,1);
27        myHashTable2.put(1,2);
28        myHashTable2.put(3,3);
29        myHashTable2.put(5,4);
30        myHashTable2.put(5,5);
31        myHashTable2.put(2,6);
32
33        System.out.println();
34        myHashTable2.displayTable();
35        System.out.println("Size: " + myHashTable2.size());
36        System.out.println("isEmpty: " + myHashTable2.isEmpty());
37
38    }
39 }
```

The 'Run' button is clicked, and the output is displayed in the 'Run' console. The output shows the state of the two hash tables:

```
Table:
0 --> 1
1 --> 4
2 --> 7
3 --> 10
4 --> 3
5 --> 6
6 --> 9
7 --> 2
8 --> 5
9 --> 8
10 --> **
11 --> **
12 --> **
13 --> **
14 --> 11
15 --> **
16 --> **
17 --> **
18 --> **
19 --> **

Size: 11
isEmpty: false

Table:
0 --> 1
1 --> 2
2 --> 6
3 --> 3
4 --> **
5 --> 4
6 --> **
7 --> 5
8 --> **
9 --> **

Size: 6
isEmpty: false

Process finished with exit code 0
```

## 2 Recursive Hashing Set

This part about Question2 in HW5

### 2.1 Pseudocode and Explanation

Write pseudocode and explanation about code design. Indicate what you are using that interfaces, classes, structures, etc.

### 2.2 Test Cases

Try this code least 2 different hash table size and 2 different sequence of keys. Report all of situations.

## 3 Sorting Algorithms

### 3.1 MergeSort with DoubleLinkedList

#### 3.1.1 Pseudocode and Explanation

begin

    eğer array boş ise ya da nexti boş ise  
        array return edilir.

    değilse

        array parçalanır

        ilk parça mergeSort a gönderilir

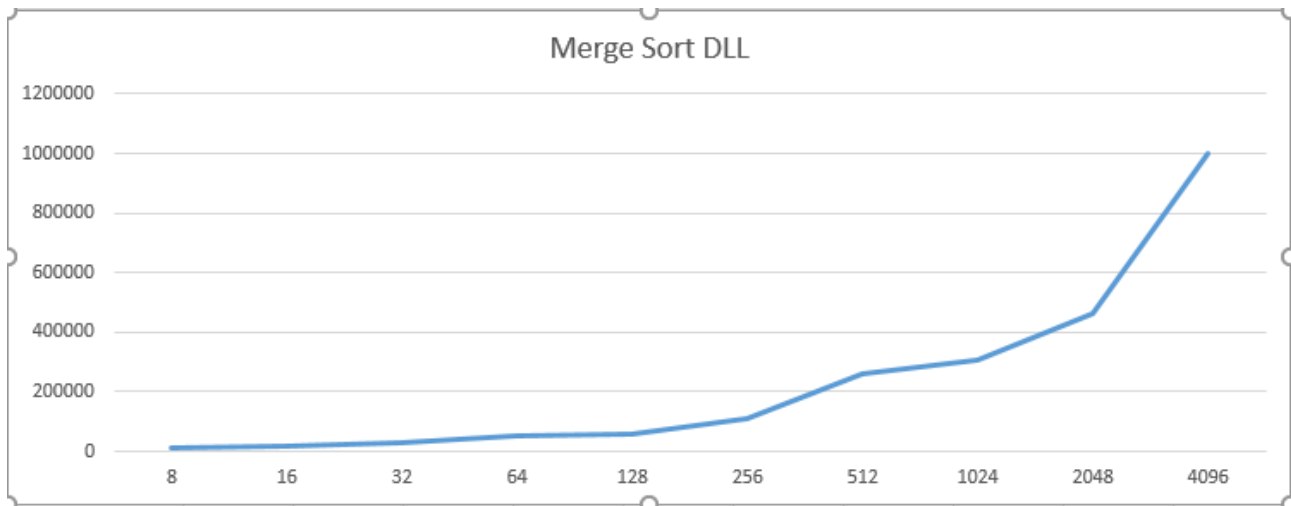
        ikinci parça mergeSort a gönderilir

        karşılaştırma metodu return edilir.

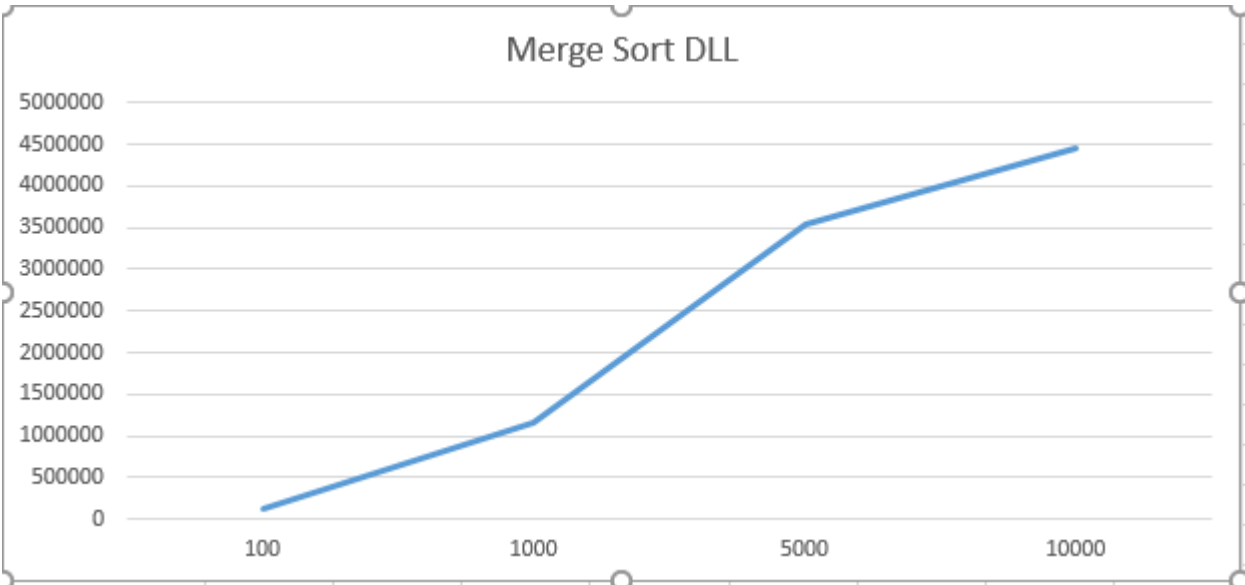
end

Bu partta Double Linked List kullanarak Merge Sort implement etmemiz isteniyordu. İlk önce DoubleLinkedList için Node isminde inner class yaptım. Bu class'ın datası, nexti ve previ var. Daha sonra bu inner class tipinde bir değişken tanımladım. Add(), divide(), mergeSort() ve mergeSortCompare() metotlarını yazdım. Add() metodunu linkliste eleman eklemek için yazdım. Divide() metodunu parametre olarak gelen linklisti ikiye bölmek için yazdım. mergeSort() metodunda ilk önce parçalama işlemi yapılıyor ve bu parçalar sırası ile tekrar mergeSort'a recursion olacak şekilde gönderiliyor. Bu işlemlerden sonra ise karşılaştırma metodu çağırılıyor ve return ediliyor. mergeSortCompare() metodunda ise karşılaştırma yapılıyor ve küçük olan ile büyük olan yer değiştiriliyor.

#### 3.1.2 Average Run Time Analysis

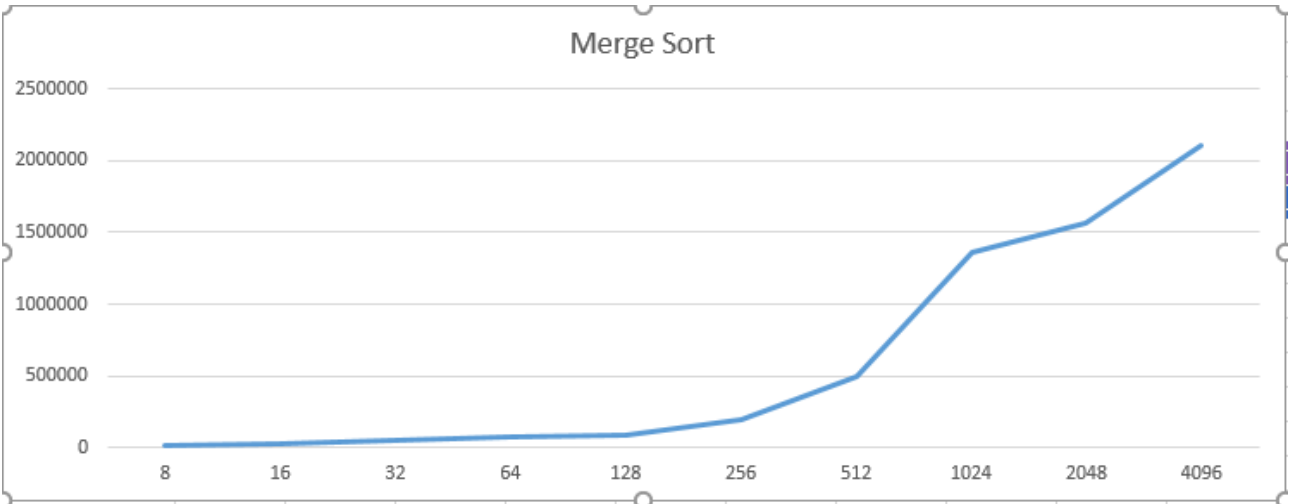


3.1.3 Wort-case Performance Analysis

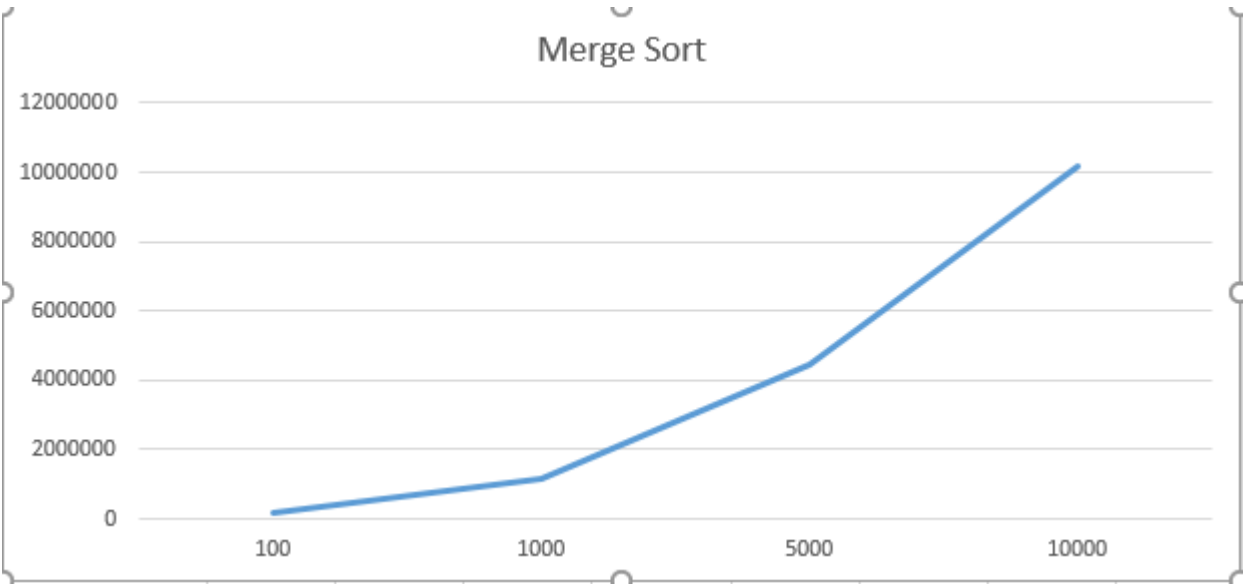


3.2 MergeSort

3.2.1 Average Run Time Analysis

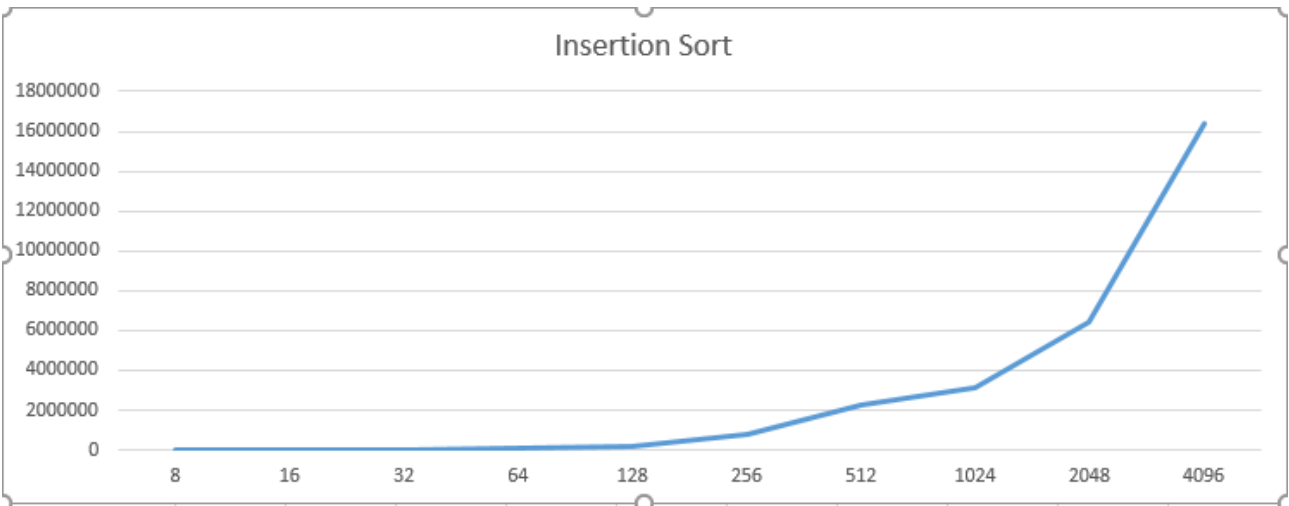


3.2.2 Wort-case Performance Analysis

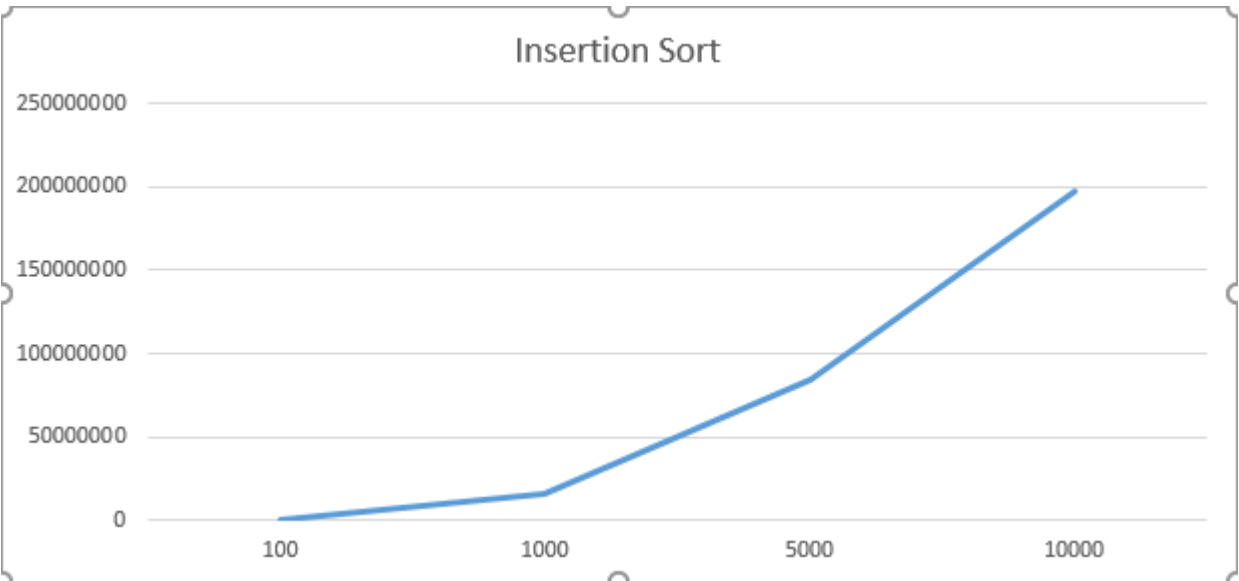


3.3 Insertion Sort

3.3.1 Average Run Time Analysis

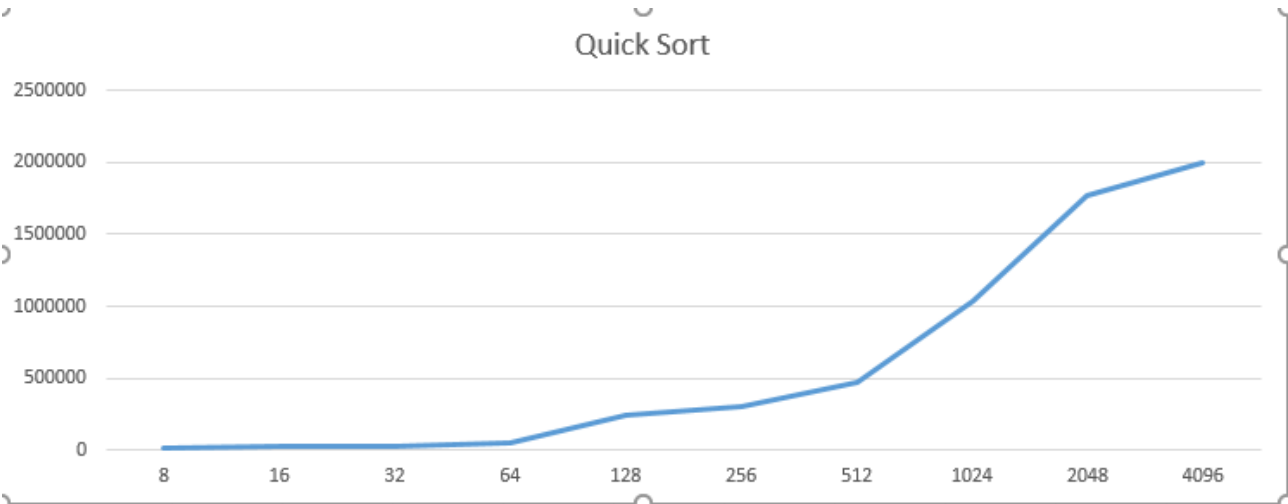


3.3.2 Wort-case Performance Analysis

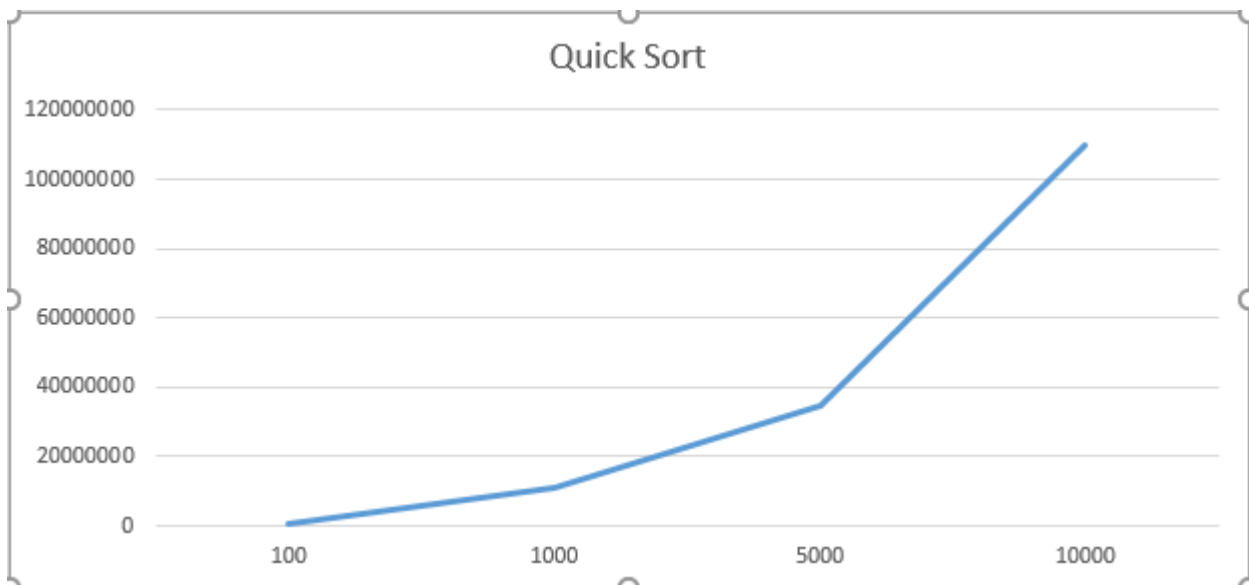


3.4 Quick Sort

3.4.1 Average Run Time Analysis

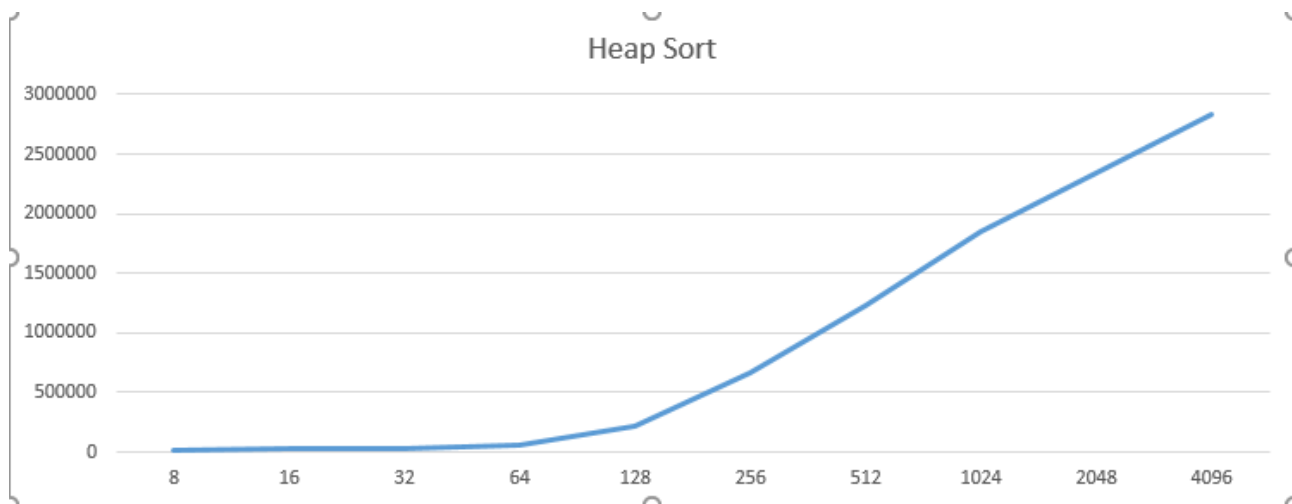


3.4.2 Wort-case Performance Analysis



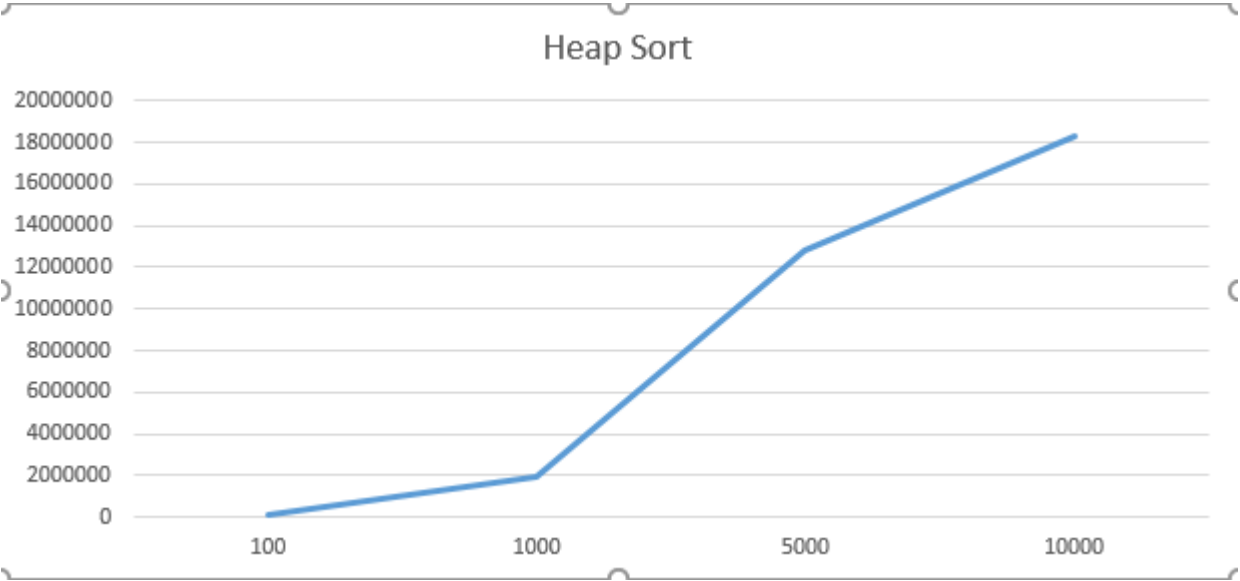
3.5 Heap Sort

3.5.1 Average Run Time Analysis

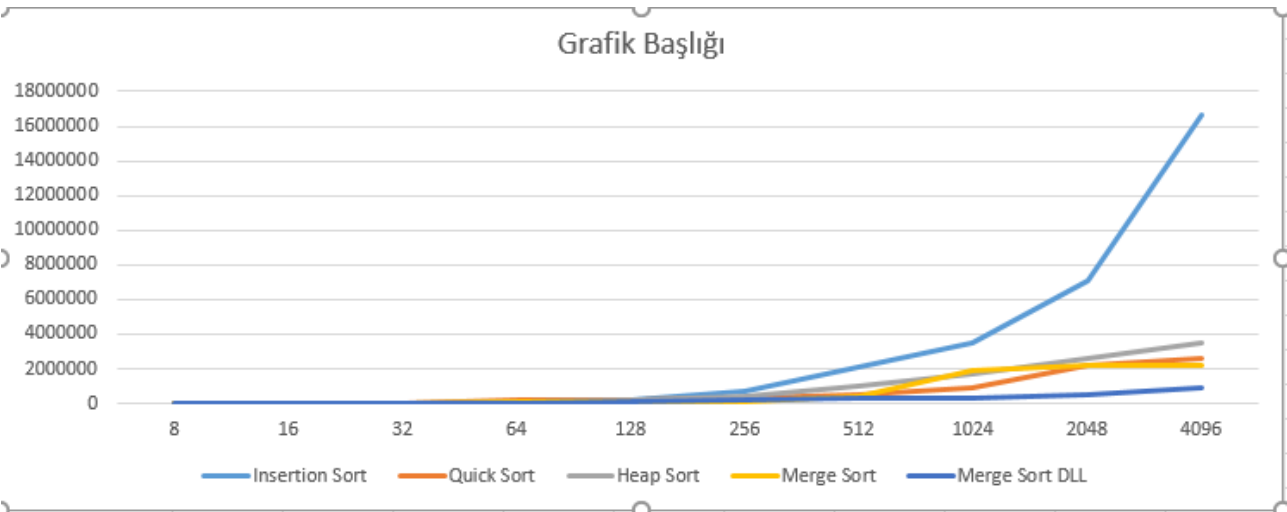




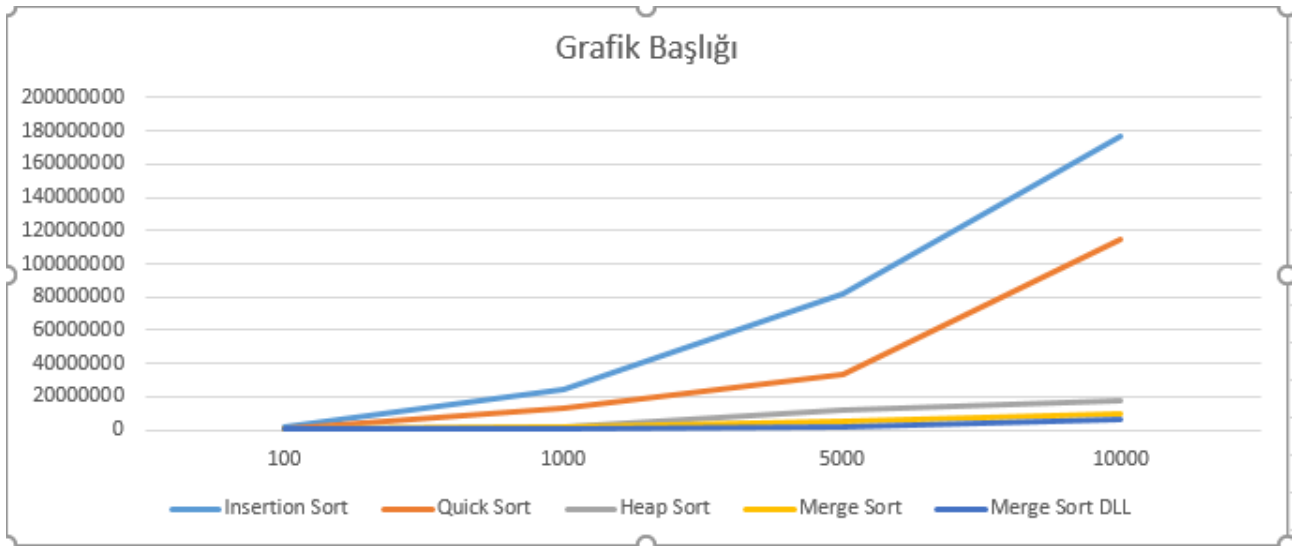
3.5.2 Wort-case Performance Analysis



4 Comparison the Analysis Results



Avarage Runtime Graphics



**Worst Case Graphics**