

**Gebze Technical University  
Computer Engineering**

**CSE 222 - 2018 Spring**

**HOMEWORK 5 REPORT**

**EMİRE KORKMAZ  
141044043**

Fatma Nur Esirci

# 1 Double Hashing Map

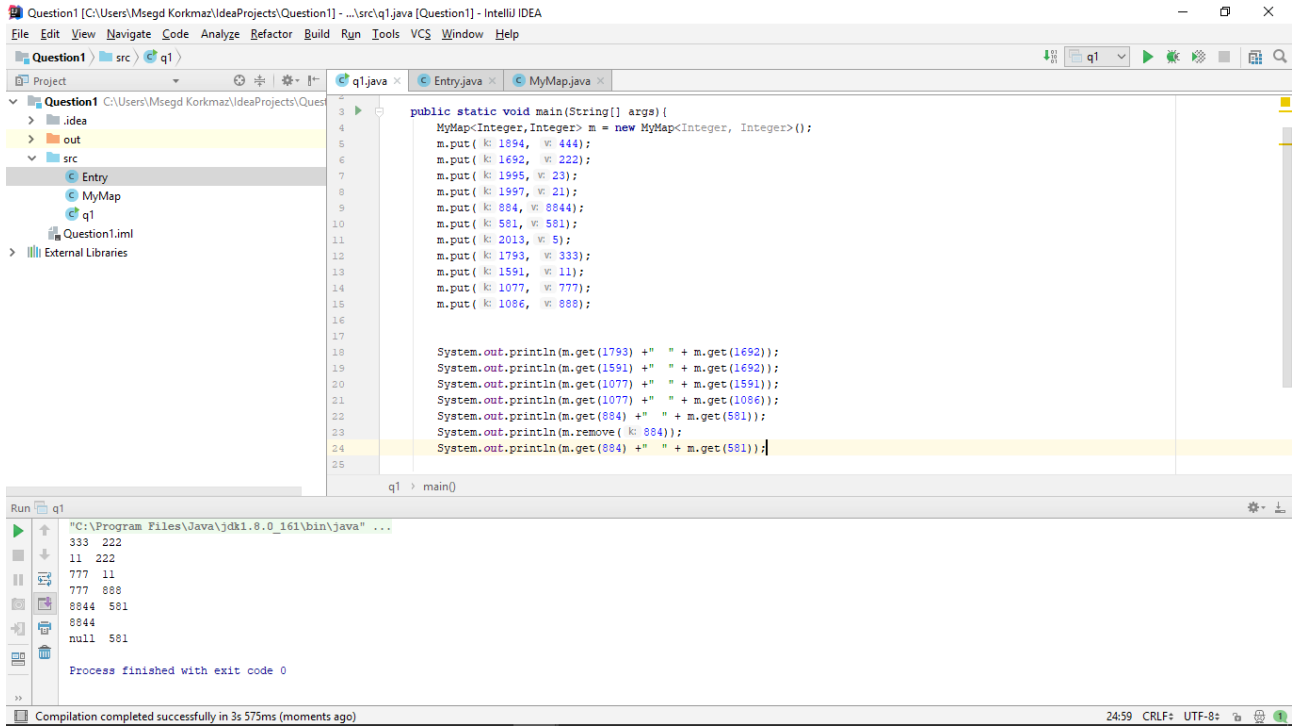
Bu soruda bizden Open Hashing yöntemini kullanarak Map Interface'ini implement etmemiz isteniyor. Collision olması durumunda bunu Double Hashing kullanarak çözmemiz isteniyor.

## 1.1 Pseudocode and Explanation

Öncelikle verileri tutmak için Entry adında bir class yazdım. Bu class'ta key, value, size ve silinip silinmediğini kontrol etmek için bir boolean değişken tuttum. MyMap adında oluşturduğum class'ta Entry tipinde bir array oluşturdum. Bu hash table oldu. Yeni bir eleman alındığında Entry tipindeki bir objeye yazılıp hash table'da gerekli yere yazılıyor. Collision durumunda kullanmak için ikinci bir hash function tanımladım. Ve bu fonksiyonlar kullanarak verilen elemanlar hash table'da gerekli yerlere yerleştiriliyor.

## 1.2 Test Cases

Hash table size : 11



```
public static void main(String[] args){
    MyMap<Integer,Integer> m = new MyMap<Integer, Integer>();
    m.put(1694, 444);
    m.put(1692, 222);
    m.put(1995, 23);
    m.put(1997, 21);
    m.put(884, 8844);
    m.put(581, 581);
    m.put(2013, 5);
    m.put(1793, 333);
    m.put(1591, 11);
    m.put(1077, 777);
    m.put(1086, 888);

    System.out.println(m.get(1793) + " " + m.get(1692));
    System.out.println(m.get(1591) + " " + m.get(1692));
    System.out.println(m.get(1077) + " " + m.get(1591));
    System.out.println(m.get(1077) + " " + m.get(1086));
    System.out.println(m.get(884) + " " + m.get(581));
    System.out.println(m.remove(884));
    System.out.println(m.get(884) + " " + m.get(581));
}
```

Run q1

```
"C:\Program Files\Java\jdk1.8.0_161\bin\java" ...
333 222
11 222
777 11
777 888
8844 581
8844
null 581
Process finished with exit code 0
Compilation completed successfully in 3s 575ms (moments ago)
```

## 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

## 3 Sorting Algorithms

### 3.1 MergeSort with DoubleLinkedList

This part about Question3 in HW5

#### 3.1.1 Pseudocode and Explanation

Merge Sort algoritmasını array'lere uygularken yaptığımız gibi LinkedList'i de ikiye bölerek yapmaya çalıştım. Ama recursive olarak sağ listeyi sort methoduna parametre olarak verdiğümde Stackoverflow hatası aldım ve bu hatayı çözemedim. Sağ liste yerine sol listeyi aynı şekilde parametre olarak verince hiçbir sorun çıkmıyor ama sağda hata veriyor ve bunun sebebini bulamadım.

#### 3.1.2 Average Run Time Analysis

This part about Question4 in HW5

#### 3.1.3 Worst-case Performance Analysis

This part about Question5 in HW5

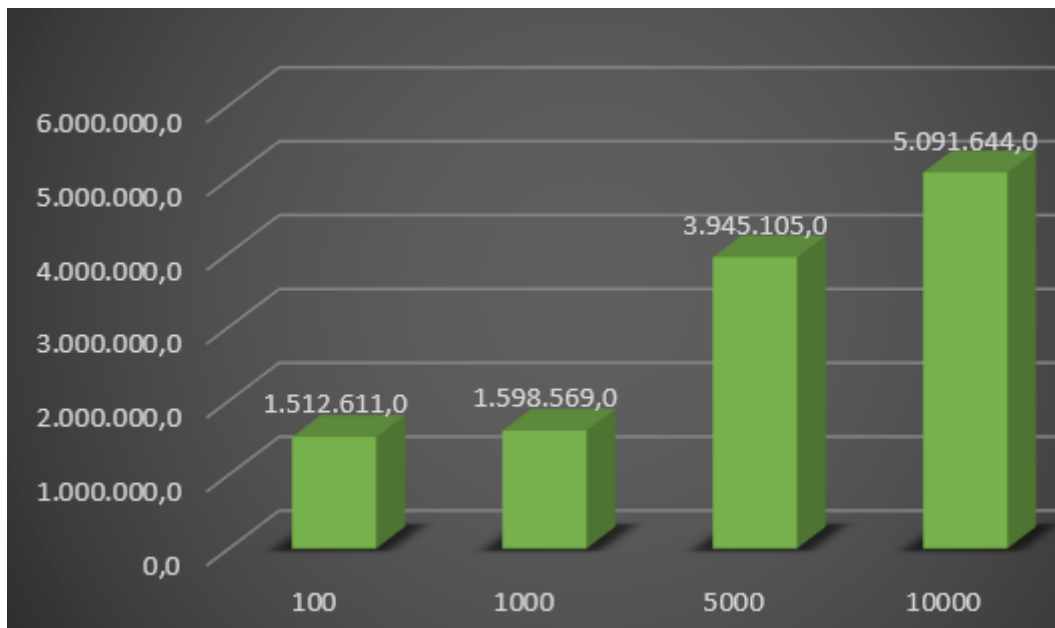
### 3.2 MergeSort

This part about code in course book.

#### 3.2.1 Average Run Time Analysis

10 tane array oluşturdum; 10, 15, 20, 25, 30, 35, 40, 45, 50, 55 boyutlarında. Her bir array'i 10 kere çalıştırdım. Ortalama olarak 11593.7, 6102.61, 2553.09, 4563.05, 2805.39, 212231.67, 249202.34, 302205.79, 21156.05, 25411.15 buldum

### 3.2.2 Wort-case Performance Analysis

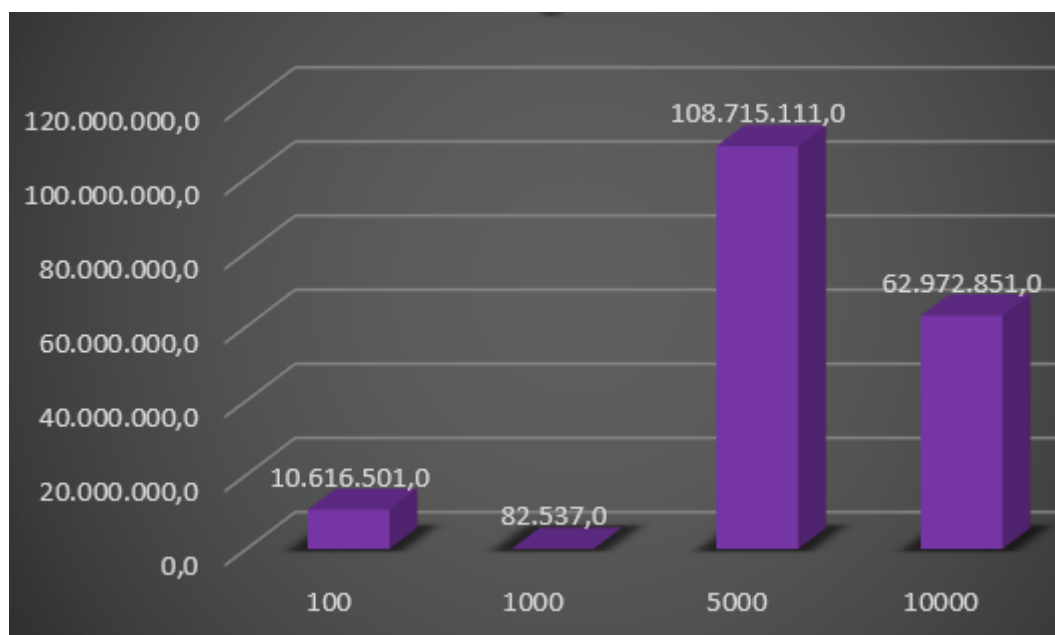


## 3.3 Insertion Sort

### 3.3.1 Average Run Time Analysis

10 tane array oluşturdum; 10, 15, 20, 25, 30, 35, 40, 45, 50, 55 boyutlarında. Her bir array'i 10 kere çalıştırdım. Ortalama olarak 500.37, 4994.98, 372.02, 641.47, 825.39, 3549.55, 821.11, 1205.95, 1235.91, 1595.15 buldum.

### 3.3.2 Wort-case Performance Analysis



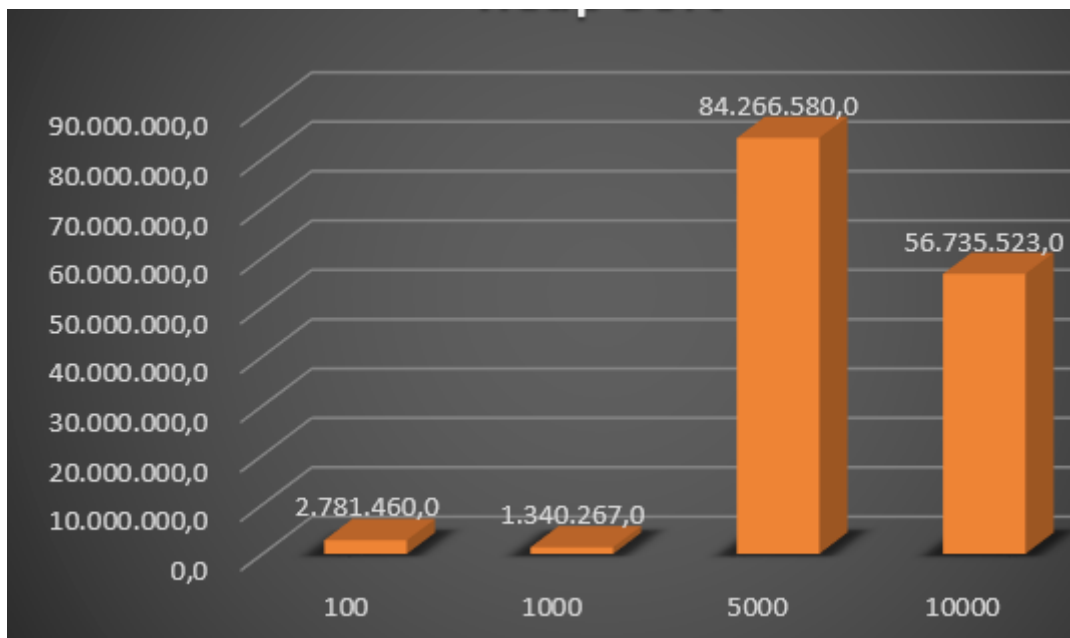
### 3.4 Quick Sort

#### 3.4.1 Average Run Time Analysis

10 tane array oluşturdum; 10, 15, 20, 25, 30, 35, 40, 45, 50, 55 boyutlarında. Her bir array'i 10 kere çalıştırdım. Ortalama olarak 7526.7, 6868.1, 13300.07, 11730.58, 1813.23, 28990.62, 301504.4, 14292.15, 18384.78, 24265.09 buldum.

#### 3.4.2 Wort-case Performance Analysis

### 3.5 Heap Sort

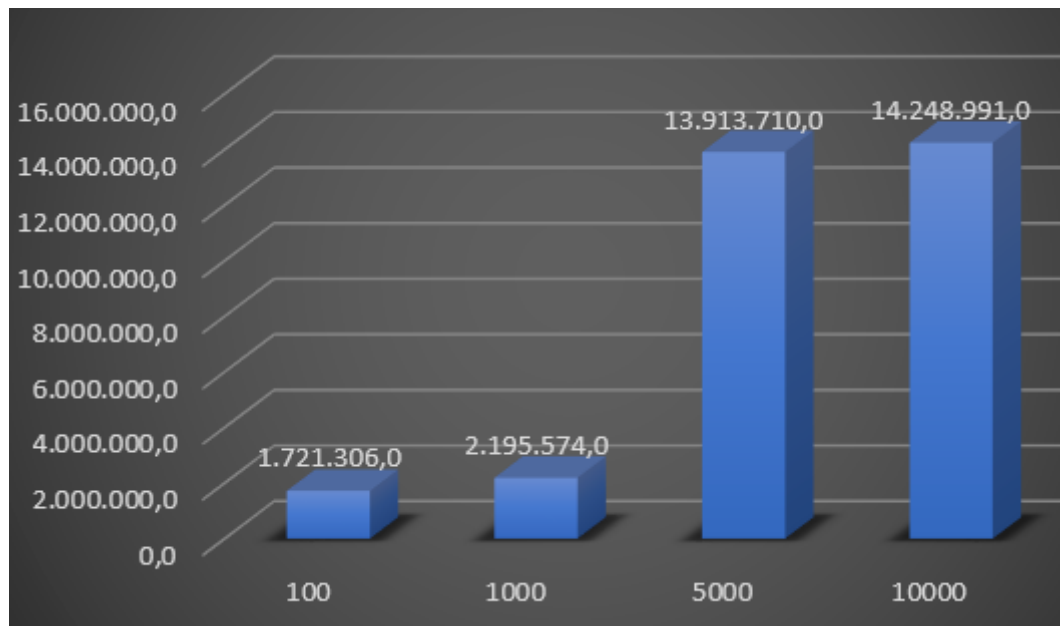


Sort

#### 3.5.1 Average Run Time Analysis

10 tane array oluşturdum; 10, 15, 20, 25, 30, 35, 40, 45, 50, 55 boyutlarında. Her bir array'i 10 kere çalıştırdım. Ortalama olarak 1890.22, 2108.33, 4058.41, 7509.61, 11700.59, 17503.88, 4994.98, 6898.07, 10276.52, 9438.32 buldum.

#### 3.5.2 Wort-case Performance Analysis



#### 4 Comparison the Analysis Results

