Gebze Technical University
Computer Engineering

CSE 222 - 2018 Spring

HOMEWORK 5 REPORT

Burak Özdemir
141044027

FatmaNur Esirci

1 Double Hashing Map

1.1 Pseudocode and Explanation

Public class DoubleHashMap<K,V> implements Map<K,V>

+public static class HashEntry<K,V>
-private int tableSize
-private int size
-private HashEntry<K,V> table;
-private int primeSize;

+public V put(K key,V val):
    If table size is equal to tableSize
        Print “table full”
    İnitialize hash1 and assign hashFirst(key)
    İnitialize hash2 and assign hashSecond(key)
    While hash1 of table is notequal to null
        Add hash2 into hash1
        Add hash1 mod tableSize into hash1
        End of while
    Set hash1 of tablet to new HashEntry
    Add one to size
    Return value of hash1 of table
➔hash1 ve hash2 degeleri hesaplanır . tabloda null görülene kadar ılerlenır ve set yapılır .

+public V remove (Object key):
    İnitialize hash1 and assign hashFirst(key)
    İnitialize hash2 and assign hashSecond(key)
    İnitiliazae val
    While (hash1 of table is not equal to null) and (key is not equal to key of has1 of table)
        Add hash2 to hash1
        Add hash1 mod tablesize to hash1
    Set value of hash1 of table to val
    Set null to hash1 of table
    Extract one to size
    Return val
➔Her defasında hash2 hash1 e eklenerek silinecek olan key degerı tabloda aranır

+public void clear()
    Set zero to size
    For set zero to i;i smaller than tablesize;add one to i
        Set null to i of table
        End of for

+public V get(Object key)
    İnitialize hash1 and assign hashFirst(key)
    İnitialize hash2 and assign hashSecond(key)
    While (hash1 of table is not equal to null) and (key is not equal to key of has1 of table)
        Add hash2 to hash1
        Add hash1 mod tablesize to hash1

Return value of hash1 of table
➔Dongu ıcınde hash2 hash1 e eklenir ve mod alınarak sureklı tekralanır ıslem . eslesme oldgunda return edır

-private int hashFirst(object x)
İnitialize hashVal and set hashCode of x
Set hash1 mod tablesize to hash1
İf hashval is smaller than zero
Add tablesize to hash1
Return hashVal
➔objenın ılk hashdegerı return edılır

-private int hashSecond(object x)
İnitialize hashVal and set hashCode of x
Set hash1 mod tablesize to hash1
İf hashval is smaller than zero
Add tablesize to hash1
Return primeSize – hashVal mod primeSize
➔objenın ıkıncı hash degerı return edılır

1.2 Test Cases

İlk obje için size 5 tir. 5 tane eleman put edılmıstır . Ve dıger get ve remove metodları denenmıstır.
2. obje ıcın sıze 3 secılmıstır yıne farklı 3 eleman put edılıp dıger metodlar denenmıstır. (Basarılı)

2 Recursive Hashing Set

2.1 Pseudocode and Explanation

Public class myRecursiveHashSet<E> implements Set<E>
/* Multidimensional buckets yapısı ile recursive hash saglanmıstır***/**
/* Diger table size 10 ile sabıtlenmıstır.***/**

-private static class Entry<E>
-private Entry[][] buckets
-private int size

-private hashFunction(int hashCode)
İnitialize index and set hashCode
İf index smaller than zero
Decrease one to index and set index
Return index mod buckets.length
➔objenın hashcode unu alarak tablesize a gore hashdegerını return eder.

+public boolean contains(Object elem)
İnitialize indY and set hashFunction(hashcode of elem)
Return helperContains(0,indy,elem)
-private boolean helperContains(int x,int y,object elem)
İf key of buckets[y][x] is equal to elem

Return true
İf bucket[y][x] is equal to null
Return false
İf indexY of bucket[y][x] is equal to y
Return helperContains(x+1,y,elem)
Else
Return helperContains(x,y+1,elem)

➔Recursive sekılde hashFunction metodunu kullanarak objenın ıcerde olup olmadgnı kontrol eder

+public boolean remove(Object o)
İnitialize index and set hashFunction(hashCode() of o)
Return helperRemove(0,index,o)
-private boolean helperRemove(int x,int y,object elem)
İf bucket[y][x] is equal to null
Return false
İf key of bucket[y][x] is eqaul to elem
İnitiliaze newBuckets and set new Entry[bucket.length][10]
For set zero to i;i smaller than buckets.length;add one to i
For set zero to j;j smaller than length of buckets[i];add one to j
İf buckets[i][j] is not equal to null
İf key of buckets[i][j] is equal to elem

Else
Set buckets[i][j] to newBuckets[i][j]
End for
Endfor
Set newbuckets to buckets
Decrease one to size
Return true

➔Recursive sekılde hashFunction kullanarak egerkı obje ıcerde varsa remove eder yoksa false return eder.

+public boolean add(Object elem)
İnitialize indexy and set hashFunction(hashCode() of elem)
Return helperAdd(0,indexy,elem)
-private boolean helperAdd(int x,int y,object elem)
İf buckets[y][x] is not equal to null
İf key of buckets[y][x] is equal to elem
Return false
İf buckets[y][s] is equal to null
İnitialize entry and set new Entry(elem,x,y)
Set enry to buckets[y][x]
Add one to size
Return true
İf indexy of buckets[y][x] is equal to y
Return helperAdd(x+1,y,elem)
Else
Return helperAdd(x,y+1,elem)

➔Recursive sekılde hashFunction metodunu kullanarak objeyı uygun yere ekleme yapar . Basarılı durumda true aksı durumda false return eder.

2.2 Test Cases

Obje 1 size 3 ile belirlenmıstır .Sırası ıle elemanlar eklenmıstır . Size ı geçme durumunda hashFunctionun degerıne gore sankı başka bır tablo oluşturmuş gbı arrayın 2. Boyutuna eklenerek devam eder . Aynı eleman ekleme durumunda false return edılır .
Obje 2 size yine 3 secılıp farklı elemanlar ıle denenmıstır.Obje 1 ın tum testlerı bu objeyede yapılmıştır.(Basarılı)

3 Sorting Algortihms

3.1 MergeSort with DoubleLinkedList

3.1.1 Pseudocode and Explanation

Public class MergeSortDoubleLinkedlist
-private static class Node
-private static Node head
➔Sınıf DoubleLinkedList ozelligi için Node sınıfını kullanır ve yapının bas kısmını head ile elinde tutar.

+public Node mergeSort(Node node)
    İf (node is equal)or(next of node is equal null)
        Return node
    Initialize second and set split(node)
    Set mergeSort(node) to node
    Set mergeSort(second) to second
    Return merge(node,second)
➔listeyi en kucuk bırımlerına kadar split metodu ıle ayırır daha sonra merge metoduna vererek sıralamalı bır sekılde bırlestırır.

-private Node split(Node head)
    Initialize fast and set head
    Inıtialize slow and set head
    While (next of fast is not equal to null) and (next of next of fast is not equal to null)
        Set next of next of fast to fast
        Set next of slow to slow
    Inıtialize temp and set next of slow
    Set null to next of slow
    Return temp
➔Verilen node dan ıtıbaren listeyi 2 parcaya ayırır

-private Node merge (Node first,Node second)
    İf first is equal to null
        Return second
    İf second is equal to null
        Return first
    İf data of first smaller than data of second
        Set merge(next of first,second) to next of first
        Set first to prev of next of first
        Set null to prev of first
        Return first

Else

Set merge(first,next of second) to next of second
Set first to prev of next of first
Set null to prev of second
Return second

➔Verilen iki linkedlist yapısının elemanlarını karşılaştırarak sıralı tek bır lıste return eder.

3.1.2 Average Run Time Analysis

Q(nlogn) zaman alır . Listeyi bölerken logn zamanda bölme işlemi yapılır. En küçük birime kadar bölünceği için her turlu logn olur daha sonra bırlestırken her eleman ıcın kontrol edılme ıslemı yapılır buda n zaman alır . bölme ıslemı sırasında karsılastrma oldgu ıcın logn*n zaman alır avarage case durumunda.

3.1.3 Wort-case Performance Analysis

Algoritma worst case , avarage case ve best case de de en küçük elemana kadar bölme ve her defasnda tüm elemanları her turlu karsılastracagı ıcın yine Q(nlogn) zaman alır .

3.2 MergeSort

3.2.1 Average Run Time Analysis

Q(nlogn) zaman alır . Listeyi bölerken logn zamanda bölme işlemi yapılır. En küçük birime kadar bölünceği için her turlu logn olur daha sonra bırlestırken her eleman ıcın kontrol edılme ıslemı yapılır buda n zaman alır . Bölme ıslemı sırasında karsılastrma oldgu ıcın logn*n zaman alır avarage case durumunda.

3.2.2 Wort-case Performance Analysis

Algoritma worst case , avarage case ve best case de de en küçük elemana kadar bölme ve her defasnda tüm elemanları her turlu karsılastracagı ıcın yine Q(nlogn) zaman alır .

3.3 Insertion Sort

3.3.1 Average Run Time Analysis

Sıra ile seçilen her eleman yeni listenın her elemanı ıle her durumda karsılastrma yapılacağı ıcın n*(n+1)/2 karsılastrma yapacağı ıcın Q(n^2) zaman karmaşıklığı olacaktır.

3.3.2 Wort-case Performance Analysis

Worst-case durumu insertion sort da elemanlar ters sıralı oldgu durumda gerceklesır . Yine avarage case de oldgu gıbı her eleman yenı lıstenın her elemanı ıle karsılasacagından O(n^2) complexity değeri çıkar .

3.4 Quick Sort

3.4.1 Average Run Time Analysis

Her satırdaki her elemana bakar eger n tane sayı varsa n tane sayıya bakılır .Adım sayısı logn olur .Ortalama durumda elemanların pivottan kucuk mu buuyk mu olacagnı kestrmek mumkun

degıldır o yüzden zaman karmasklıgı O(nlogn) olur .

3.4.2 Wort-case Performance Analysis

Her satırdaki her elemana bakar eger n tane sayı varsa n tane sayıya bakılır.En kotu durumda butun elemanlar eger pivottan kucuk ıse elemanlar ıkıye bölünmez . Her zaman pıvot bır soldakı secılcegı ıcın n-1 tane sayı olacaktır .Ve bu ıslem tekralanır . böylece O(n^2) karmaşıklığı elde edılır.

3.5 Heap Sort

3.5.1 Average Run Time Analysis

Bu algoritma her eleman için bir kere silme metodunu çağıracağı için(her eleman için oldgundan n-1 kere) ve her silme operasyonu logn zaman alacağından algorıtmanın zaman karmaşıklığı O(nlogn) olacaktır .

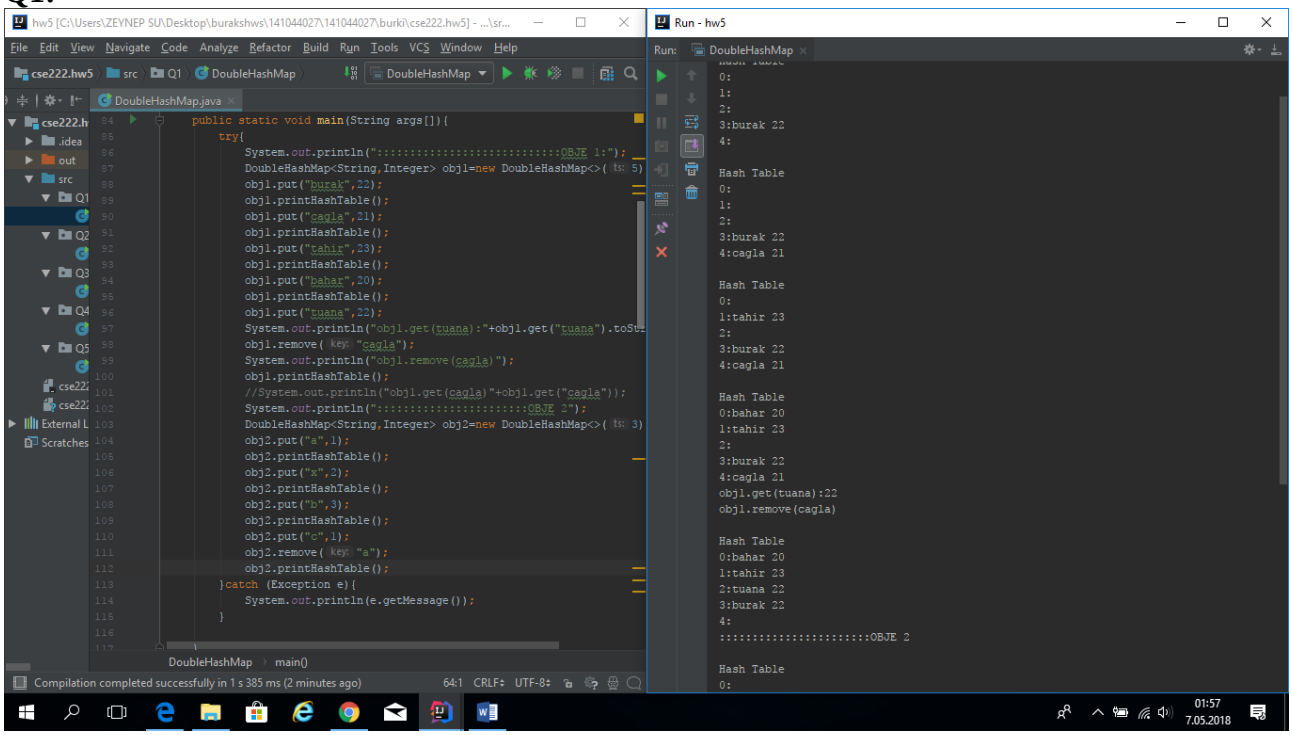
3.5.2 Wort-case Performance Analysis

O(nlogn) karmaşıklığına sahıp olacaktır . Cunku her turlu ağaç yukseklıgı logn karmasıklıgna sahıp olacaktır ve worstcase durumunda tek bır dal seklınde düşünürsek n-1 tane karsılastrma olacaktır ve yıne n*logn karmsıklıgını elde etmıs olacağız.

4 Comparison the Analysis Results

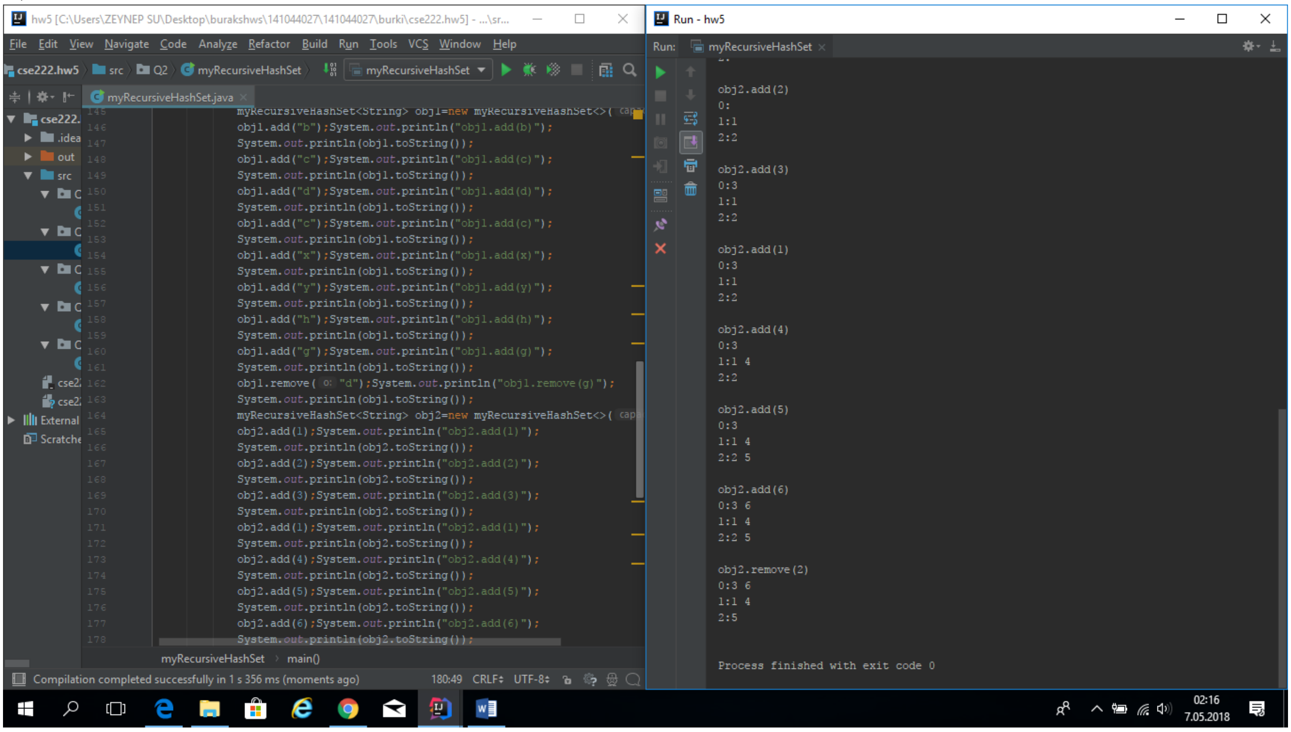
/*Bazı eleman sayılı testlerde bazı algoritmalar stack hatası verdiği için o algoritmalar alınmamıstır*/

Q1:



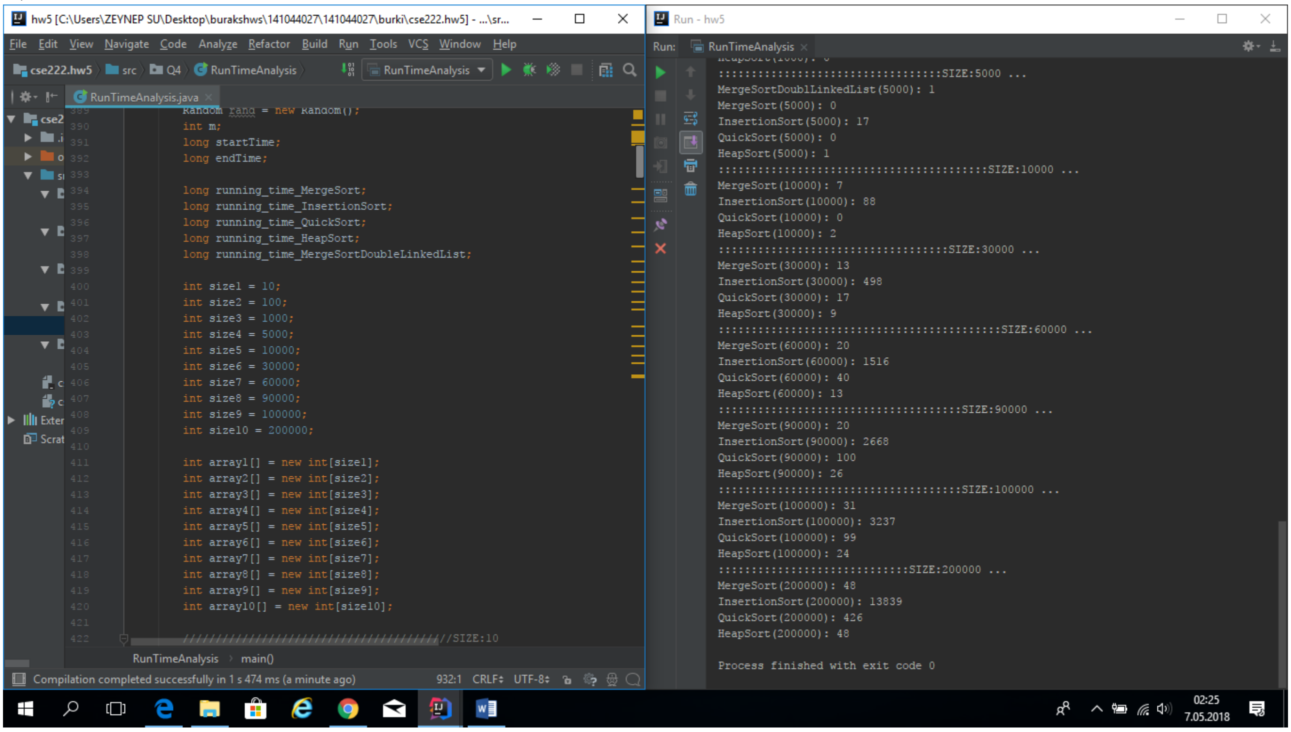
Q1 Main Test

Q2:

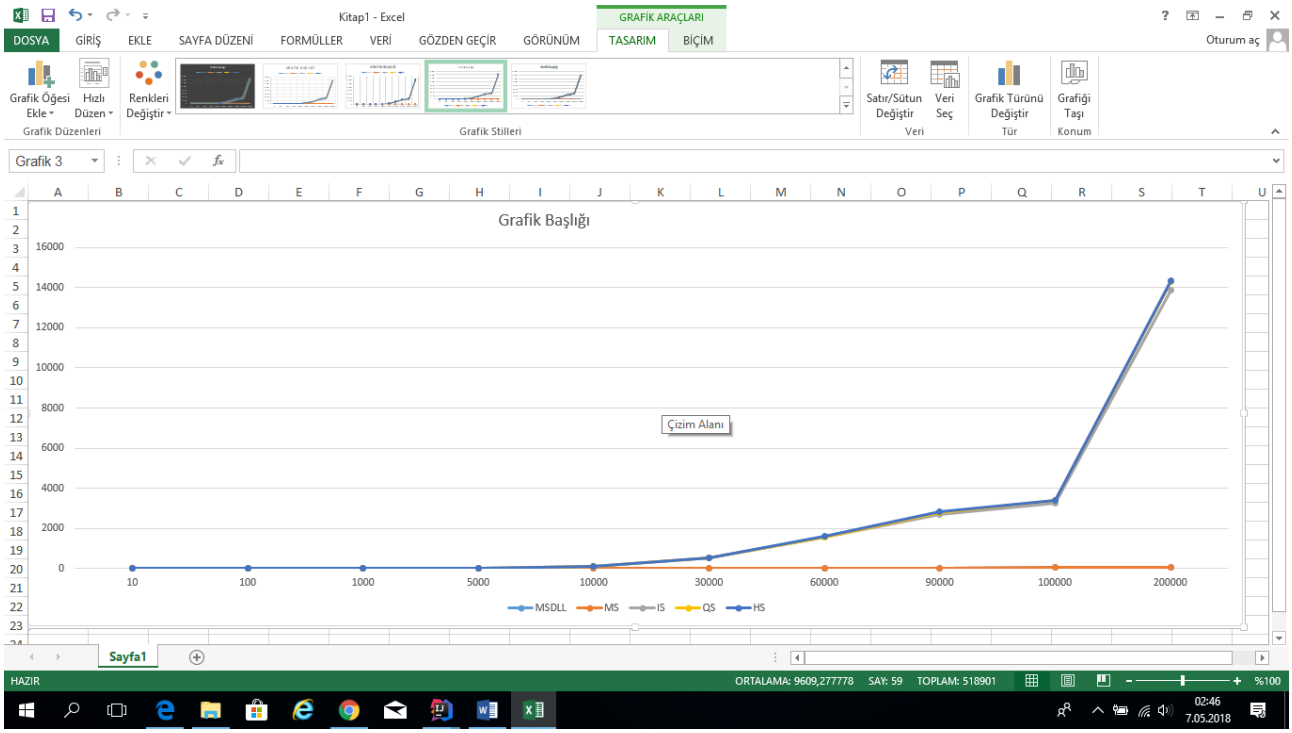


Q2 Main Test

Q4:

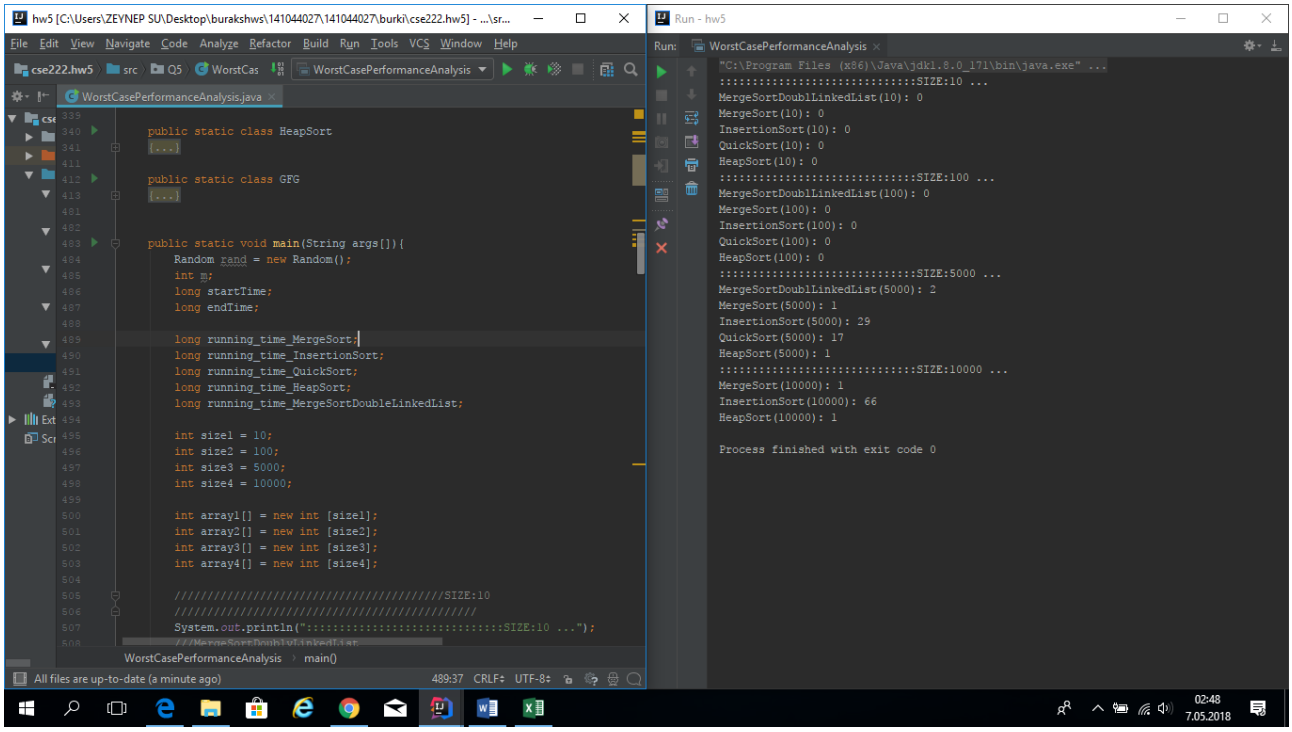


Q4 main Test

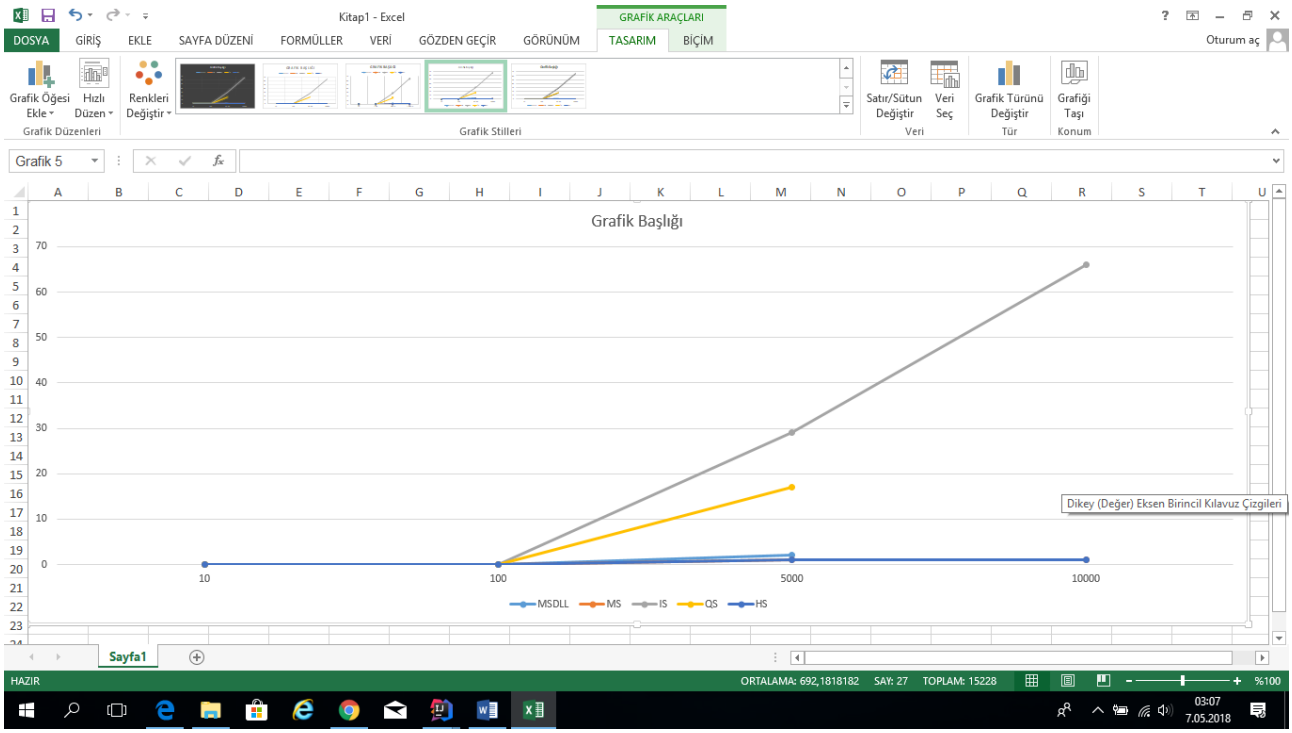


Q4-Grafik

Q5:



Q5-Main Test


Kitap1 - Excel
GRAFİK ARAÇLARI
DOSYA
GİRİŞ
EKLE
SAYFA DÜZENİ
FORMÜLLER
VERİ
GÖZDEN GEÇİR
GÖRÜNÜM
TASARIM
BİÇİM
Oturum aç
Grafik Öğesi Ekle
Hızlı Düzen
Renkleri Değiştir
Grafik Düzenleri
Grafik Stilleri
Satır/Sütun Değiştir
Veri Seç
Veri
Grafik Türünü Değiştir
Tür
Grafiği Taşı
Konum
Grafik 5
Grafik Başlığı
70
60
50
40
30
20
10
0
10
100
5000
10000
MSDLL
MS
IS
QS
HS
Dikey (Değer) Eksen Birincil Kılavuz Çizgileri
Sayfa1
HAZIR
ORTALAMA: 692,1818182
SAYI: 27
TOPLAM: 15228
%100
03:07
7.05.2018

Q5-Grafik