Gebze Technical University
Computer Engineering

CSE 222 - 2018 Spring

HOMEWORK 5 REPORT

BURHAN ELGÜN
151044060

Course Assistant: Fatma Nur ESİRCİ

1 Double Hashing Map

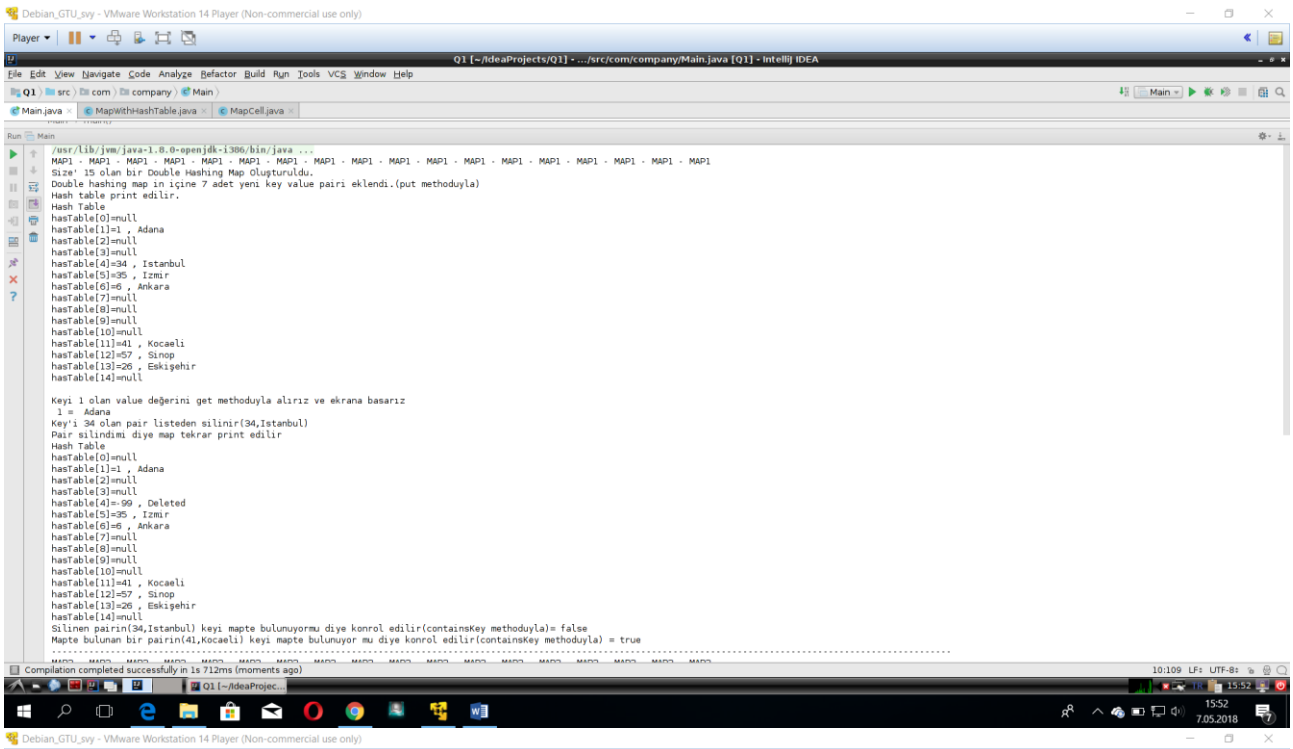
1.1 Pseudocode and Explanation

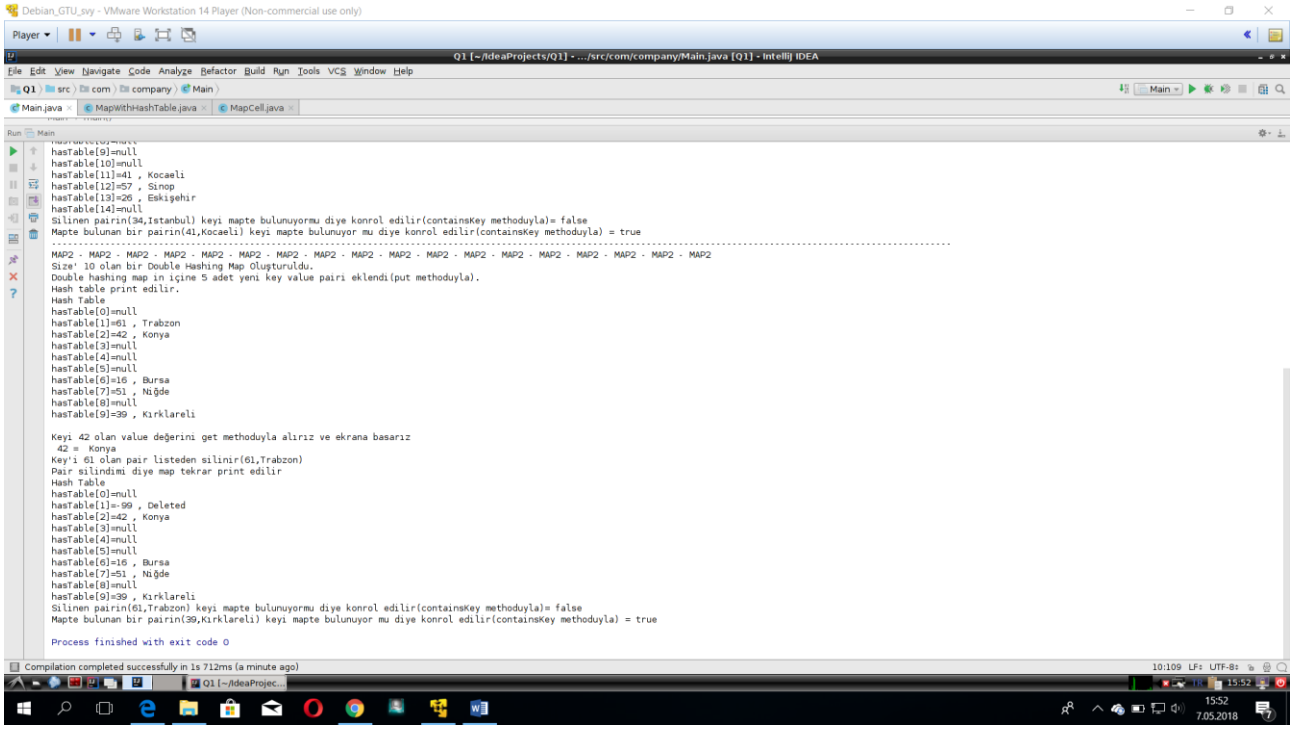
Write pseudocode and explanation about code design. Indicate what you are using that interfaces, classes, structures, etc.

MapCell adında bir class oluşturuldu.
MapCell classı her bir Map entrysini temsil etmektedir ve içerisinde key value pairlerini tutmaktadır.
MapCell constructoru oluşturuldu ve gelen key ve value değerlerini kendi data fieldlarına atar.

Map interfacesini implement eden MapWithHashTable adında bir class oluşturuldu.
MapWithHashTable classında MapCell titpinde hashTable adında bir array datafield olarak oluşturuldu.Bu array hash table dır.(MapWithHashTable classında MapCell classı kullanıldı)
MapWithHashTable classında datafield olarak capacity tutuldu.
MapWithHashTable classında datafield olarak elementSize tutuldu.
Capacity alan bir constructor yapıldı ve bu constructor capacitye gör hashTable oluşturur
Size methodu eleman sayısını return edecek şekilde implement edildi.
isEmpty methodu eleman sayısını kontrol ederek boş olup olmadığını return edecek şekilde implement edildi.
containsKey methodu get methodunu kullanır ve eğer gelen obje ile birbirine eşitse true return eder.
getPrime methodu capacity den başlayarak en yakın ilk prime sayıyı return eder.
hashCode1 methodu Object tipinde bir key alır ve o keyin hash valuesini, hashcodun capacitye göre modunu alarak return eder.
hashCode2 methodu Object tipinde bir key alır ve keyin hashcodunun capacitye göre modunu bulduktan sonra bir de capacitye en yakın prime sayıya göre modunu alır ve ardından gelen değeri capacitye en yakın prime sayıdan çıkararak return eder.
Get methodu hascode1 ve hashcode2 methodlarını çalıştırarak iki farklı hash değeri üretir ardından has1 kullanılarak get işlemi gerçeklekleşmiyorsa hash2 kullanılır ve parametre olarak gelen key değerinin valuesi return edilir.
Put methodu hem key hem value alır ardından hash1 ve hash2 kodları ürettirilir key için. Daha sonra has1 için put işlemi gerçekleşmiyorsa hash2 için put işlemi gerçekleştirilir.Ve elementSize 1 arttırılır.
Remove methodu object titpinde key alır ve hash1 ve hash2 kullanılarak o key silinmeye çalışılır.Öncelikle hash1 kullanılır ve remove işlemi gerçekleşmezse hash2 kullanılır.
Clear methodu baştan sona tüm Map i null ile doldurur.

1.2 Test Cases





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 Algortihms

Not: ScreenShotlardaki kısaltmaların anlamları:
AQS = Avarege Case Quick Sort
AHS = Average Case Heap Sort
AIS = Average Case Insertion Sort
AMS = Average Case Merge Sort
AmSDLL = Average Case Merge Sort With DLL

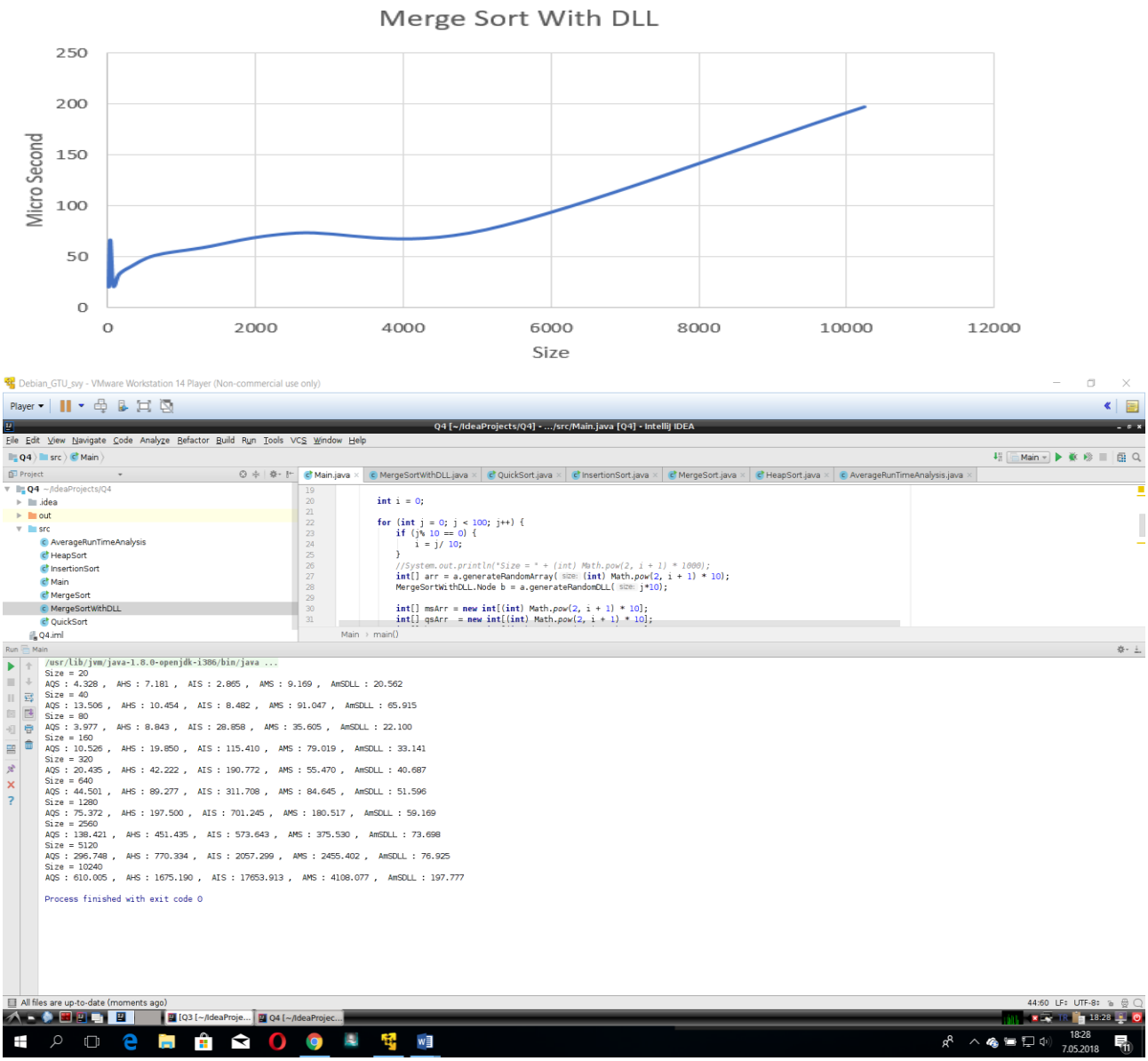
3.1 MergeSort with DoubleLinkedList

3.1.1 Pseudocode and Explanation

MergeSortWithDLL adında bir class oluşturuldu.
Bu classın içerisinde Node inner classı oluşturuldu.
Node classının içerisinde int data, Node next, Node prev tutuldu.
MergeSortWithDLL classında Node head, Node tail, int size tutuldu.
Double Linked Listeye add işlemi yapabilmek için addLast methodu impelement edildi.Bu method tail inden bir sonraki yer için ekeleme yapmaktadır.(3 parametreli Node constructoruyla)
Split methodu yazıldı , burda amaç merge yapabilmek için önce listeyi ortdan ikiye bölmektir.Bu amaçla split methodu listeyi iki parçaya böler.
MergeSort methodu gelen Node ve nexti null değilken(base case) recursive olarak splitten gelen iki farklı listeyide mergeSort methoduna yollar(recursive)
Ardından bunları merge eder.
Merge methodu kendisine gelen iki farklı sıralı listeyi birleştirir.Hangisinin ilk elemanı daha küçükse o ilk liste olur diğeri onun sonuna eklenir.

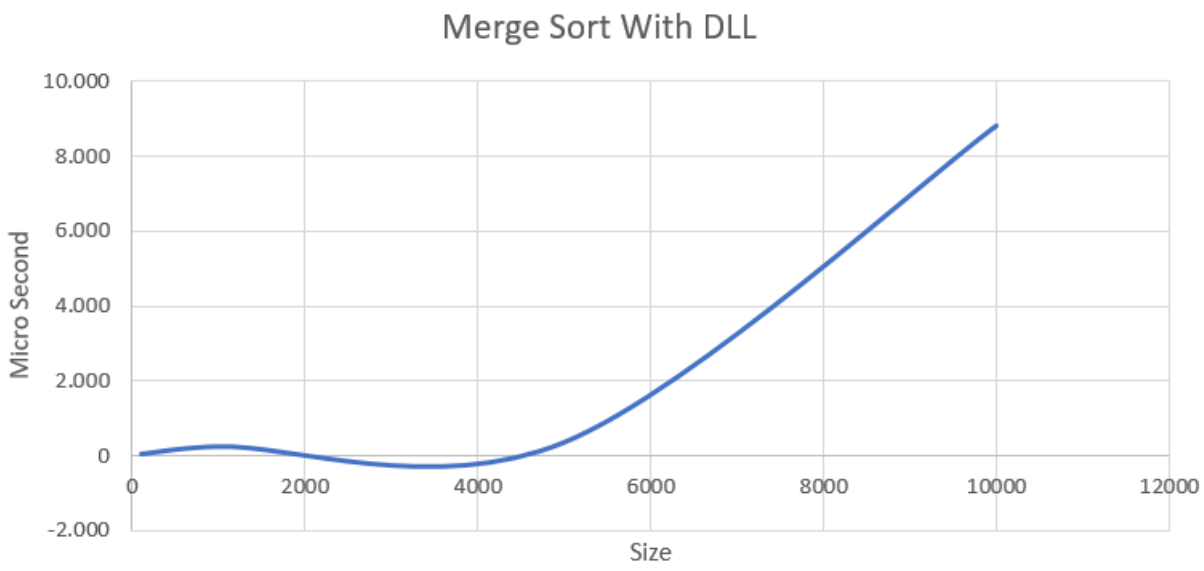
3.1.2 Average Run Time Analysis

Q(nlogn) =Teta(nlogn) zaman alır.
Size = 20 = > 20.562 micro saniye (10 tane testin ortalaması)
Size = 40 = > 65.915 micro saniye (10 tane testin ortalaması)
Size = 80 = > 22.100 micro saniye (10 tane testin ortalaması)
Size = 160 = > 33.141 micro saniye (10 tane testin ortalaması)
Size = 320 = > 40.687 micro saniye (10 tane testin ortalaması)
Size = 640 = > 51.596 micro saniye (10 tane testin ortalaması)
Size = 1280 = > 59.169 micro saniye (10 tane testin ortalaması)
Size = 2560 = > 73.698 micro saniye (10 tane testin ortalaması)
Size = 5120 = > 76.925 micro saniye (10 tane testin ortalaması)
Size = 10240 = > 197.777 micro saniye (10 tane testin ortalaması)



3.1.3 Wort-case Performance Analysis

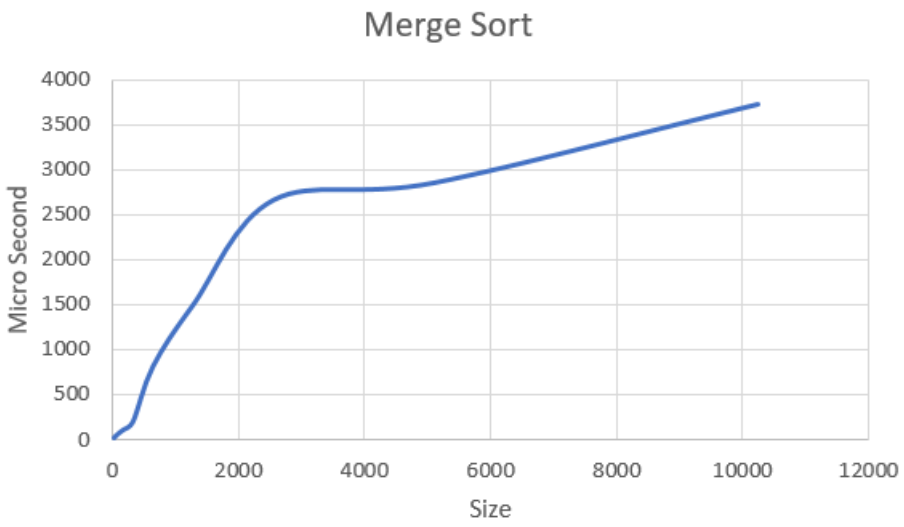
Size = 100 = > 51,239 micro saniye
Size = 1000 = > 252,704 micro saniye
Size = 5000 = > 365,777 micro saniye
Size = 10000 = > 8818,719 micro saniye

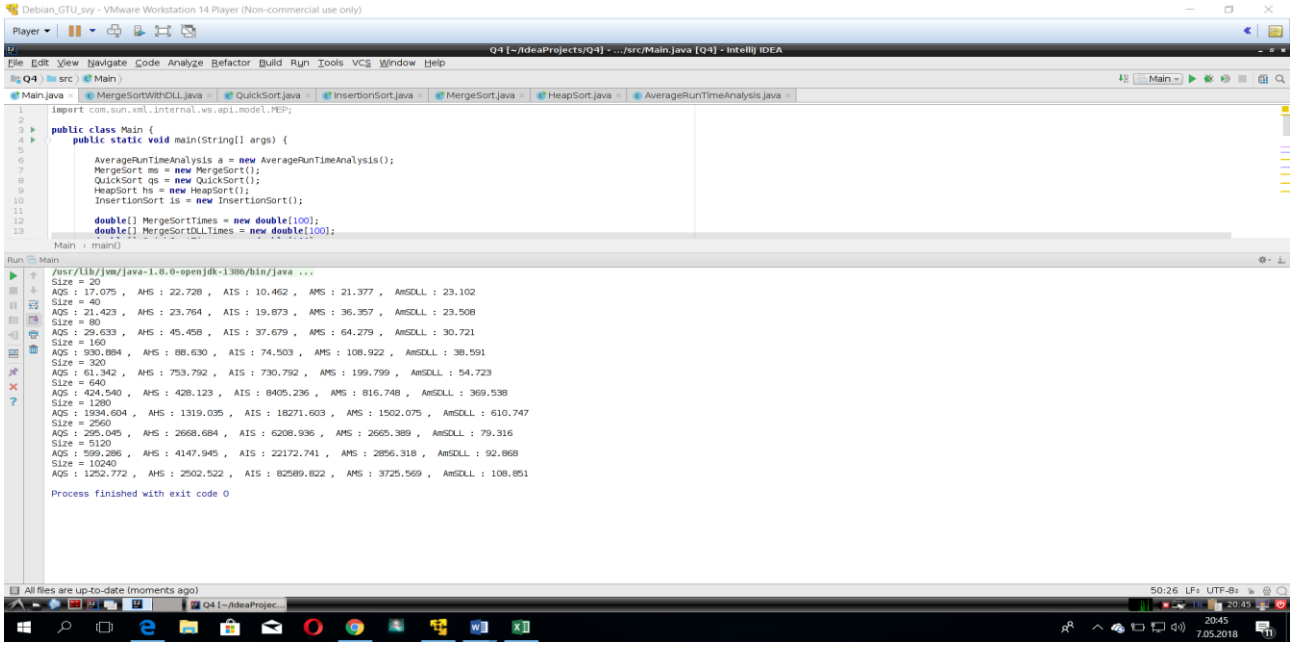


3.2 MergeSort

3.2.1 Average Run Time Analysis

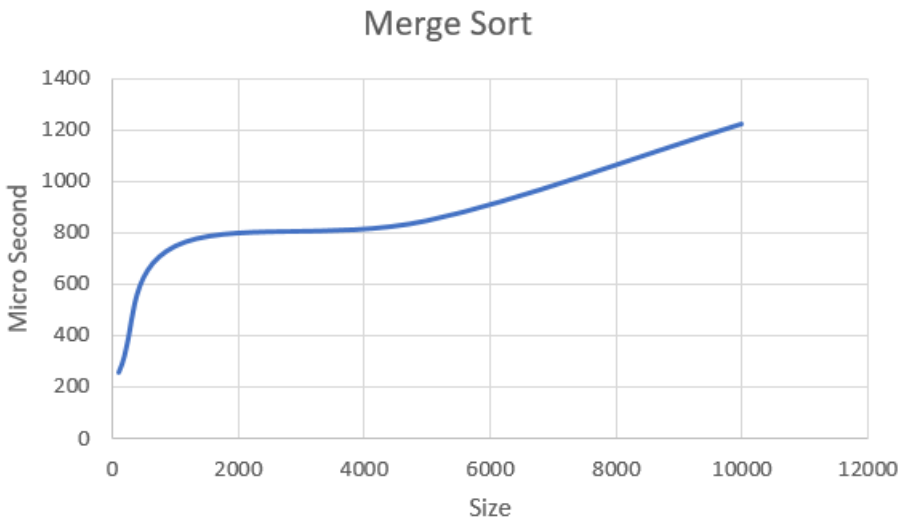
Size = 20 = > 21.377 micro saniye (10 tane testin ortalaması)
Size = 40 = > 36.357 micro saniye(10 tane testin ortalaması)
Size = 80 = > 64.279 micro saniye (10 tane testin ortalaması)
Size = 160 = > 108.922 micro saniye (10 tane testin ortalaması)
Size = 320 = > 199.799 micro saniye (10 tane testin ortalaması)
Size = 640 = > 816.748 micro saniye (10 tane testin ortalaması)
Size = 1280 = > 1502.075 micro saniye (10 tane testin ortalaması)
Size = 2560 = > 2665.389 micro saniye (10 tane testin ortalaması)
Size = 5120 = > 2856.318 micro saniye (10 tane testin ortalaması)
Size = 10240 = > 3725.569 micro saniye (10 tane testin ortalaması)





3.2.2 Wort-case Performance Analysis

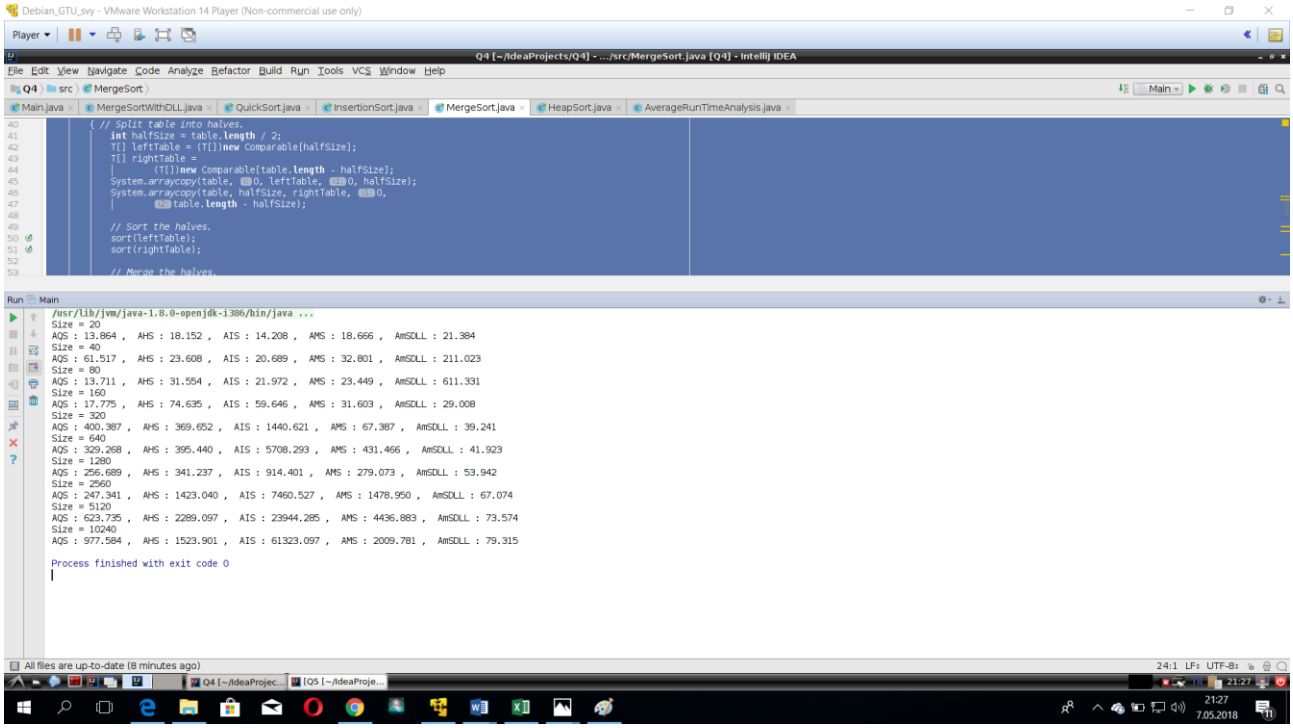
SIZE : 100
Worst Case Merge Sort : 255.916
SIZE : 1000
Worst Case Merge Sort : 748.375
SIZE : 5000
Worst Case Merge Sort : 847.684
SIZE : 10000
Worst Case Merge Sort : 1224.95



3.3 Insertion Sort

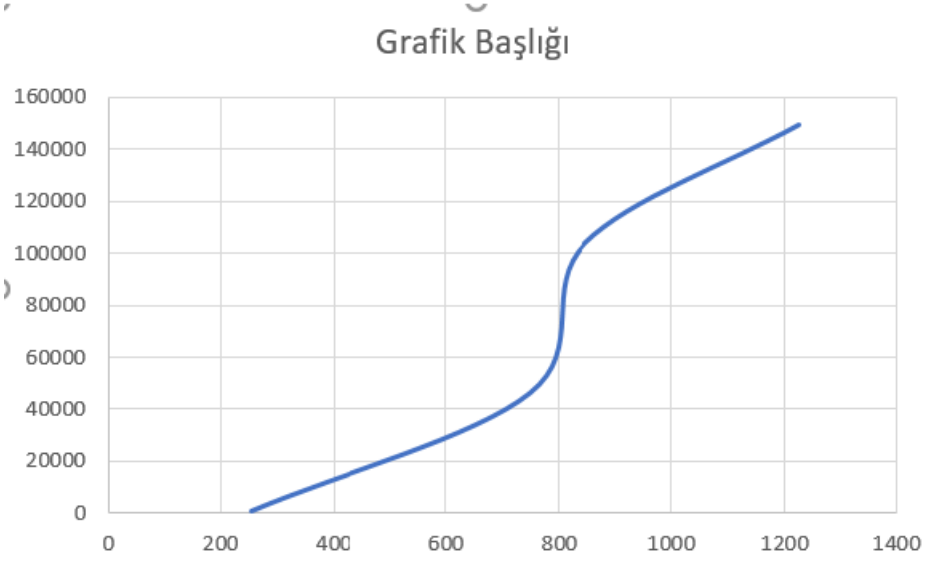
3.3.1 Average Run Time Analysis

Size = 20 = > 14.208 micro saniye (10 tane testin ortalaması)
Size = 40 = > 20.689 micro saniye(10 tane testin ortalaması)
Size = 80 = > 21.972 micro saniye (10 tane testin ortalaması)
Size = 160 = > 59.645 micro saniye (10 tane testin ortalaması)
Size = 320 = > 1440.621 micro saniye (10 tane testin ortalaması)
Size = 640 = > 5708.293 micro saniye (10 tane testin ortalaması)
Size = 1280 = > 914.401 micro saniye (10 tane testin ortalaması)
Size = 2560 = > 7460.527 micro saniye (10 tane testin ortalaması)
Size = 5120 = > 23944.285 micro saniye (10 tane testin ortalaması)
Size = 10240 = > 61323.097 micro saniye (10 tane testin ortalaması)



3.3.2 Wort-case Performance Analysis

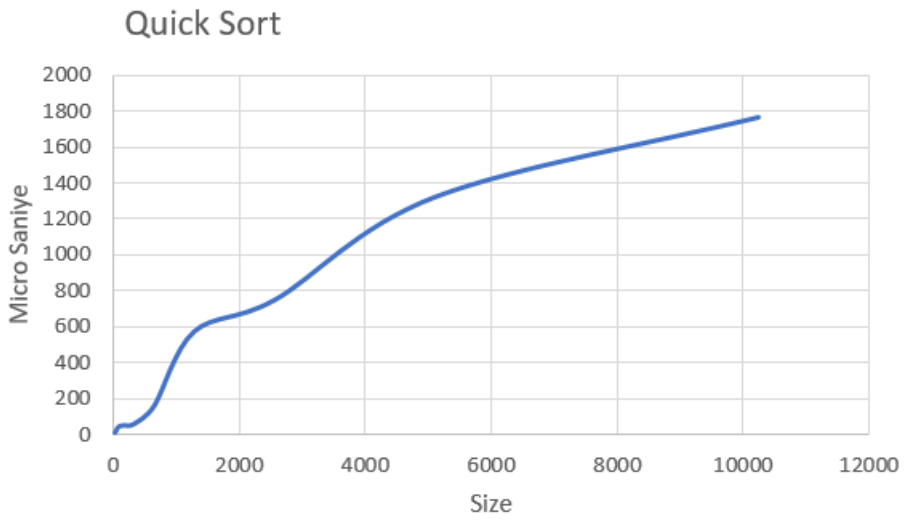
SIZE : 100
Worst Case Insertion Sort : 915.651
SIZE : 1000
Worst Case Insertion Sort : 46272.134
SIZE : 5000
Worst Case Insertion Sort : 103901.391
SIZE : 10000
Worst Case Insertion Sort : 149353.707

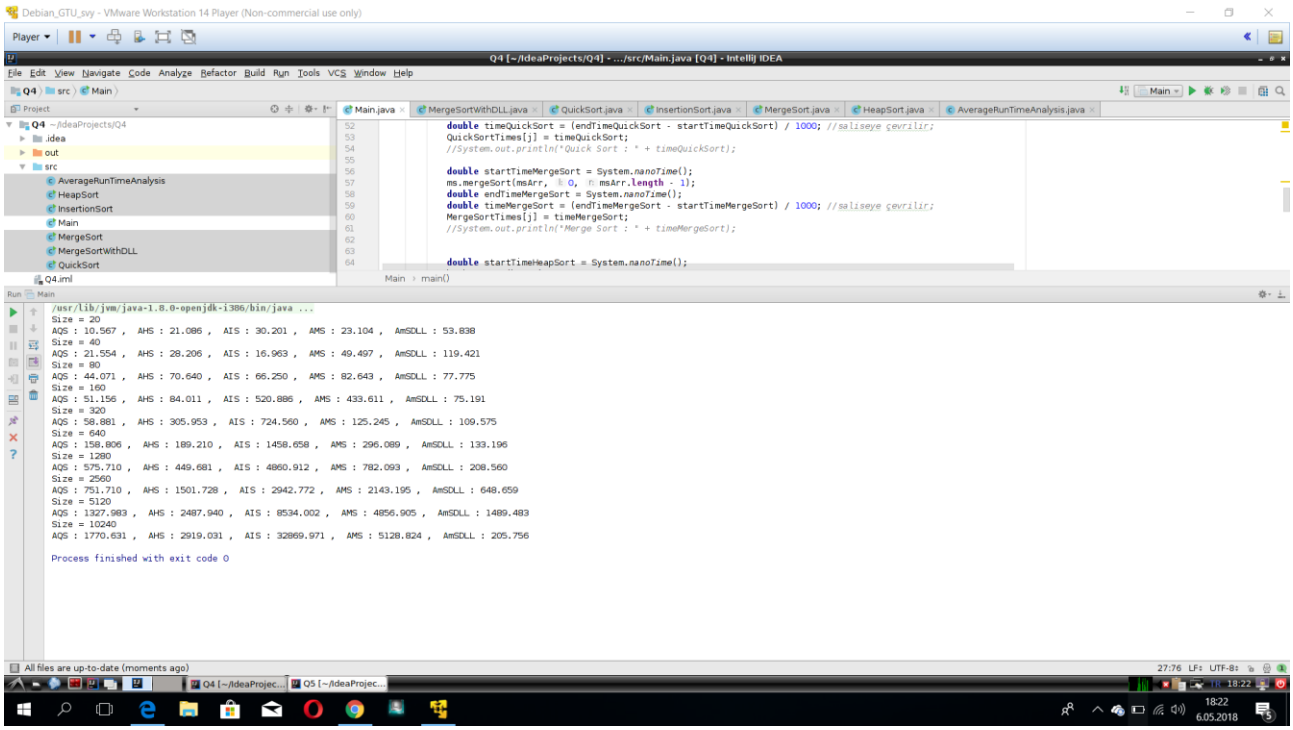


3.4 Quick Sort

3.4.1 Average Run Time Analysis

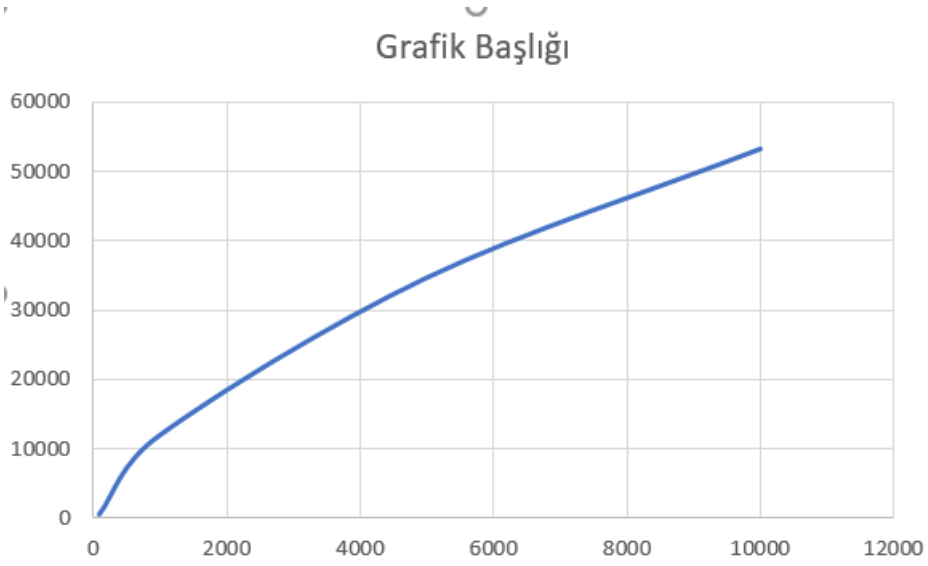
Size = 20 = > 10.567 micro saniye (10 tane testin ortalaması)
Size = 40 = > 21.554 micro saniye(10 tane testin ortalaması)
Size = 80 = > 44.071 micro saniye (10 tane testin ortalaması)
Size = 160 = > 51.156 micro saniye (10 tane testin ortalaması)
Size = 320 = > 58.881 micro saniye (10 tane testin ortalaması)
Size = 640 = > 158.806 micro saniye (10 tane testin ortalaması)
Size = 1280 = > 575.710 micro saniye (10 tane testin ortalaması)
Size = 2560 = > 751.710 micro saniye (10 tane testin ortalaması)
Size = 5120 = > 1327.983 micro saniye (10 tane testin ortalaması)
Size = 10240 = > 1770.631 micro saniye (10 tane testin ortalaması)





3.4.2 Wort-case Performance Analysis

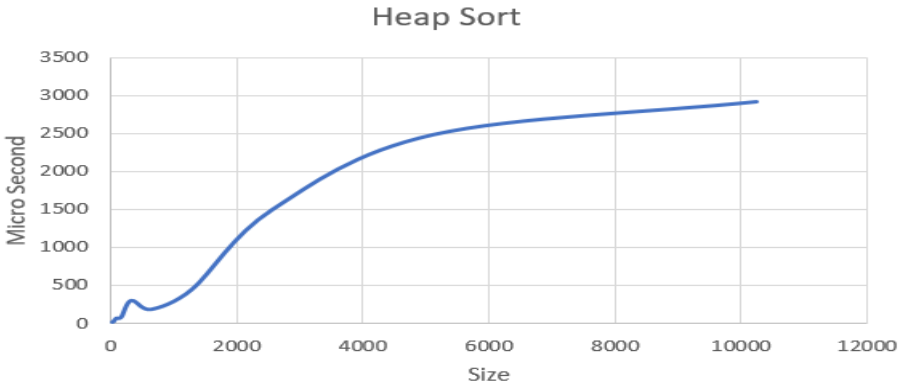
SIZE : 100
Worst Case Quick Sort : 433.821
SIZE : 1000
Worst Case Quick Sort : 11832.456
SIZE : 5000
Worst Case Quick Sort : 34615.998
SIZE : 10000
Worst Case Quick Sort : 53254.557

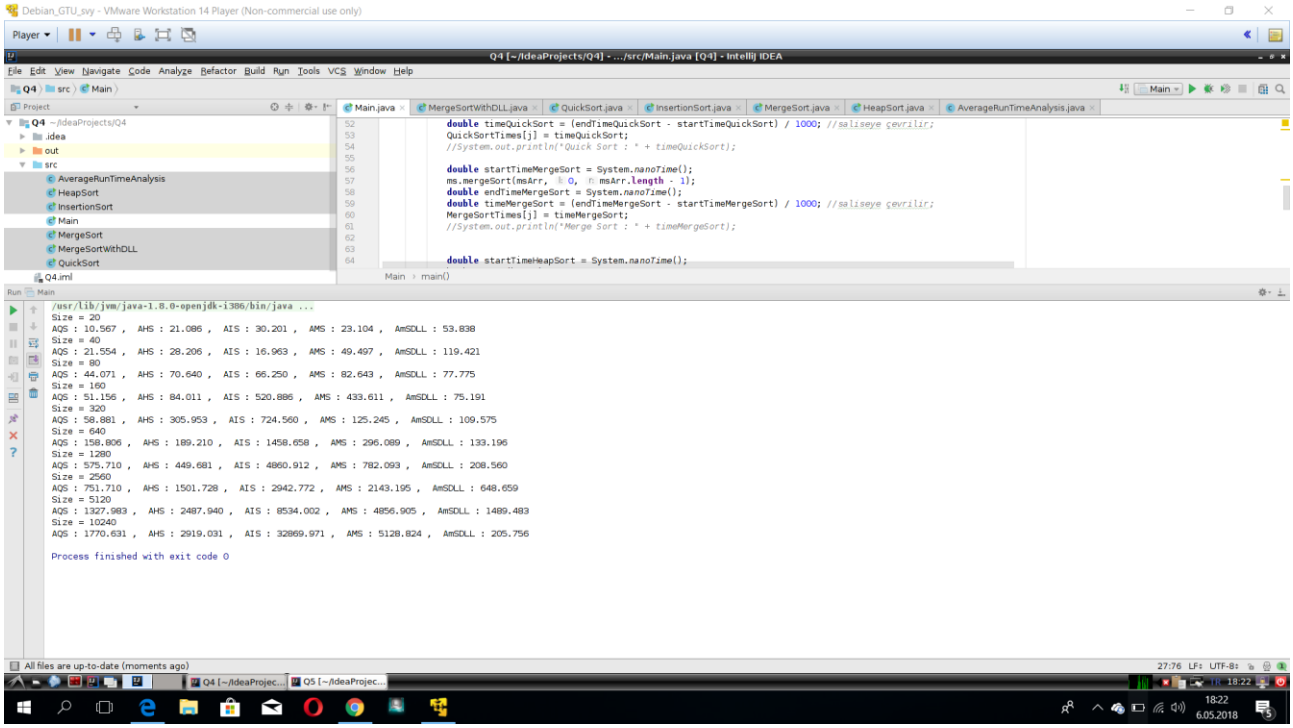


3.5 Heap Sort

3.5.1 Average Run Time Analysis

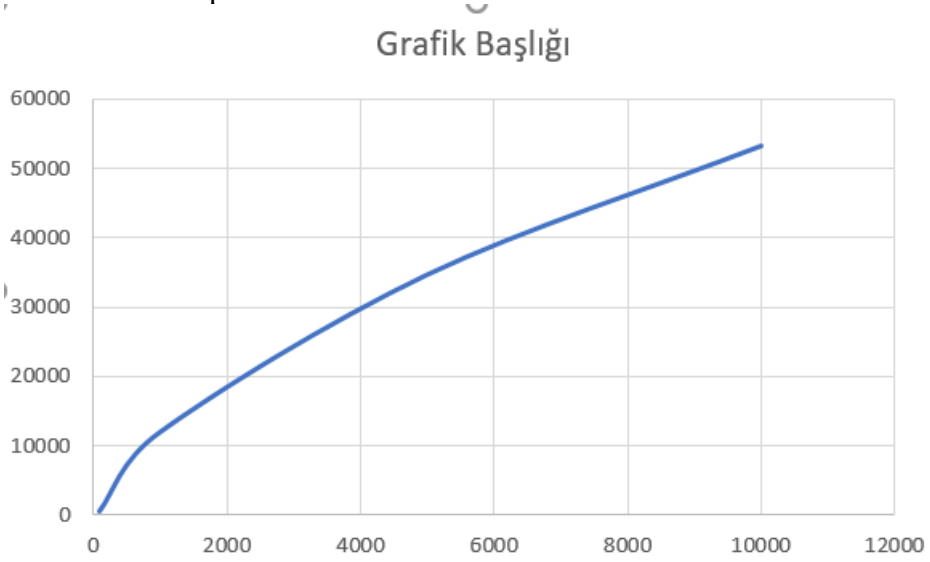
Size = 20 = > 21.086 micro saniye (10 tane testin ortalaması)
Size = 40 = > 28.206 micro saniye(10 tane testin ortalaması)
Size = 80 = > 70.640 micro saniye (10 tane testin ortalaması)
Size = 160 = > 84.011 micro saniye (10 tane testin ortalaması)
Size = 320 = > 305.953 micro saniye (10 tane testin ortalaması)
Size = 640 = > 189.210 micro saniye (10 tane testin ortalaması)
Size = 1280 = > 449.681 micro saniye (10 tane testin ortalaması)
Size = 2560 = > 1501.728 micro saniye (10 tane testin ortalaması)
Size = 5120 = > 2487.940 micro saniye (10 tane testin ortalaması)
Size = 10240 = > 2919.031 micro saniye (10 tane testin ortalaması)





3.5.2 Wort-case Performance Analysis

SIZE : 100
Worst Case Heap Sort :62.93
SIZE : 1000
Worst Case Heap Sort : 140.73
SIZE : 5000
Worst Case Heap Sort : 500.613
SIZE : 10000
Worst Case Heap Sort : 563.837



4 Comparison the Analysis Results