Gebze Technical University
Computer Engineering

CSE 222 - 2018 Spring

HOMEWORK 5 REPORT

Furkan AKTAŞ
141044029

Course Assistant:

Her Package'ın kendi Main'i oldugundan unit test, test class vs. yazılmadı.

1 Double Hashing Map

This part about Question1 in HW5

1.1 Pseudocode and Explanation

firstHashCode ve secondHashCode olmak üzere 2 tane hash fonksiyonu var. İlk hash fonksiyonu Java'dan gelen HashCode() kullanılarak gelen key'e göre üretildi. 2. hash fonksiyonunana key değeri olarak 1. hash fonkisyonundan üretilen değer veriliyor.

Genel olarak kitaptakine benzer bir şekilde hash işlemleri gerçekleştirildi. Nested olarak Entry yapısında key value değeri tutuldu. Tablo Entry array şeklinde implement edildi. Tablonun doluluk oranı %75 olduğunda reHash yapılır. Rehash yapılırken tablo kapasitesi asal olacak şekilde büyütüldü.

1.2 Test Cases

100 ve 708 'erlik map'ler oluşturularak get put size remove gibi fonksiyonlar denendi.

2 Recursive Hashing Set

yapılmadı

2.1 Pseudocode and Explanation

yapılmadı

2.2 Test Cases

yapılmadı

3 Sorting Algortihms

3.1 MergeSort with DoubleLinkedList

This part about Question3 in HW5

3.1.1 Pseudocode and Explanation

Merge sort algoritmasındaki array'ler yerine java nın double LinkedList yapısı kullanıldı. Performansı artırmak için kopyalama vs. işlemleri iterator kullanılarak yapıldı.

3.1.2 Average Run Time Analysis

Merge sort ta olduğu gibi çalışma süresi **Teta(n.logn)** dir. Fakat, muhtemelen LinkedList yapısının add methodundan kaynaklı olarak çalışma süresi merge sort a göre oldukça fazla sonuç üretti.

3.1.3 Wort-case Performance Analysis

Worst case olarak merge sort ta olduğu gibi maximum swap durumunda worst case elde edilir. Bu yüzden worst case durumndaki array üretildi ve bu array sıralandı.

3.2 MergeSort

This part about code in course book.

3.2.1 Average Run Time Analysis

Array sürekli 2 ye bölünerek küçülür bundan dolayı bu kısımdan gelen çalışma süresi **Teta(logn)** dir. Aynı zaman her bölünme için sıralama kendi içinde yapıldığından burdan **Teta(n)** gelir. Tüm çalışmas süresi olarak **Teta(n.logn)** ‘e ulaşırız.

3.2.2 Wort-case Performance Analysis

Merge sort her zaman **Teta(n.logn)** süreyle çalışır. Olabileceği en kötü durum maximum sayıda karşılaştırma swap işlemi yapılması ile gerçekleşir. Bu durumun sağlanması için worst case durumunu sağlayan Array üretildi ve bu array sıralandı. Bu array’i üreten kod internetten alındı.

3.3 Insertion Sort

3.3.1 Average Run Time Analysis

Array sıralı veya neredeyse sıralı değilse, 2 döngü olduğundan dolayı çalışma süresi **Teta(n^2)** dir. Eğer array sıralı ise yani solundaki eleman her zaman küçükse içteki döngüye girmez bu yüzden çalışma süresi **Teta(n)** olur. Genel olarak bakıldığında çalışma süresi **O(n^2)** dir.

3.3.2 Wort-case Performance Analysis

Insertion sort ortalama çalışma süresi O(n^2) dir. Worst case durumu listenin ters sıralı olduğu durumda gerçekleşir. Bu durumda elemanlar her seferinde en başa taşınacağından en uzun çalışma süresi elde edilir.

3.4 Quick Sort

3.4.1 Average Run Time Analysis

Quick sort ta 1 pivot belirlenir ve bu pivot’a göre liste 2 ye bölünür. Bölünen bu parçalar sort edilir. Bu bölünmeden dolayı logn çalışma süresi ve sıralama kısmında n çalışır. Sonuç olarak çalışma süresi O(n.logn) dir.

3.4.2 Wort-case Performance Analysis

Pivot en büyük veya en küçük eleman olduğunda çalışma süresi yapılan **O(n^2)** olur. Kitaptaki kod karşılaştırma kısmında pivot olarak ilk elemanı seçiyor bu yüzden , sıralı liste vererek en küçük eleman pivot olarak seçilmiş oldu ve worst case sağlandı.

3.5 Heap Sort

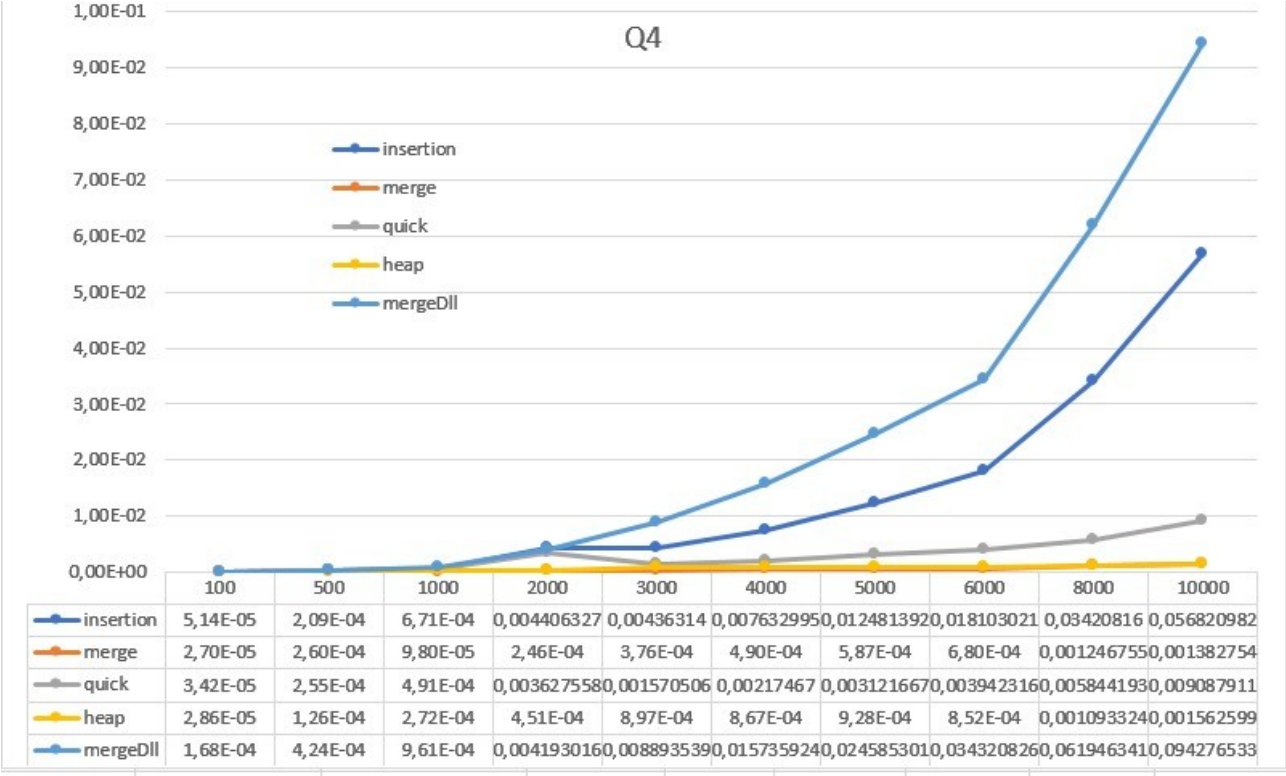
3.5.1 Average Run Time Analysis

Heap sort verilen arrayi önce heap yapısına çevirir sonra bu heap yapısını sıralar. Max heap kullanılarak yapılır. En başta her seferinde en büyük eleman olur bu eleman en sona koyulur(bir nevi silinir). Bu sayede ekstra bir yer almaya da gerek yoktur. Bu işlemleri **Teta(n.logn)** sürede yapar.

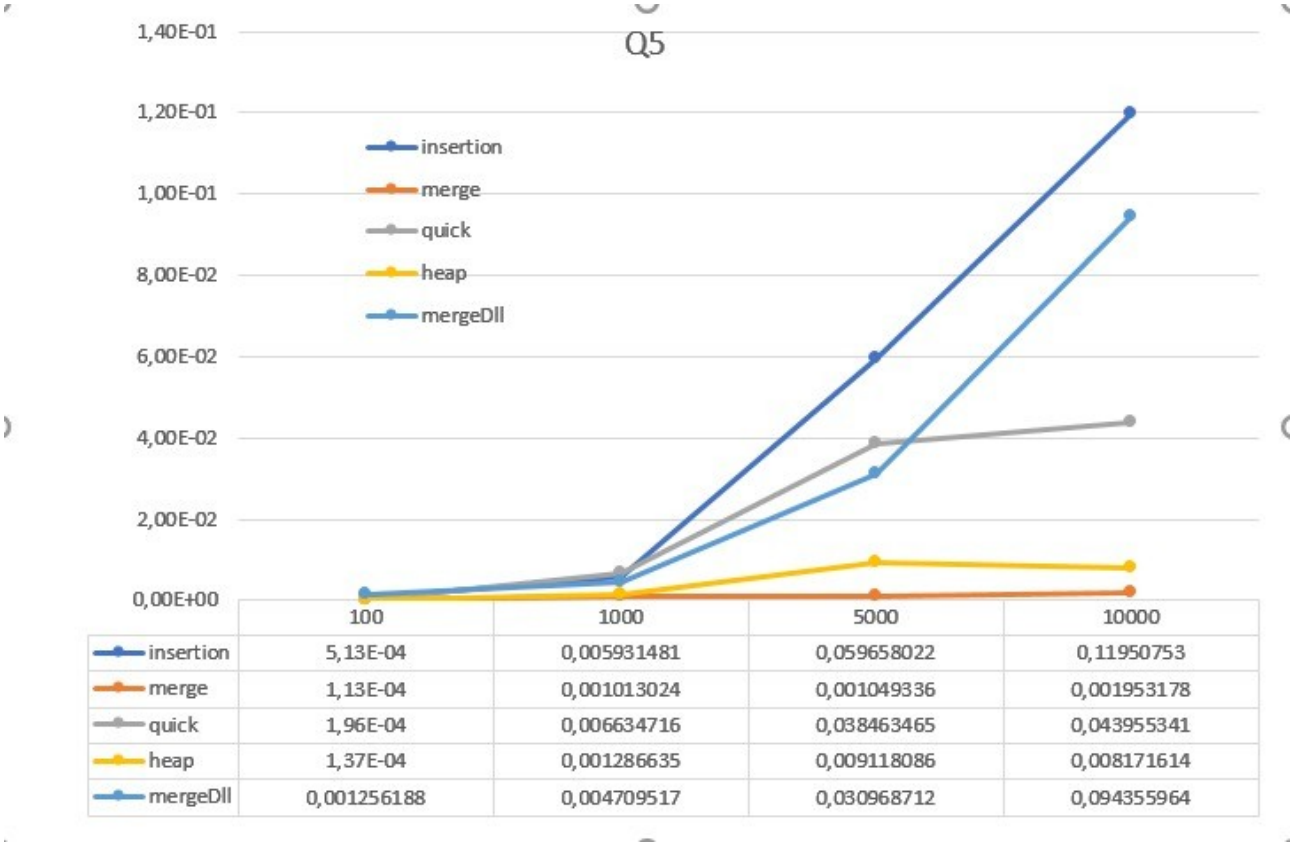
3.5.2 Wort-case Performance Analysis

Heap sort her zaman **Teta(n.logn)** çalışır. Worst durumu düşünüldüğünde swap sayısı göz önünde bulundurulabilir. Bu yüzden heap oluştururken maximum sayıda swap işlemi gerçekleşmesi için sıralı array verip bu array i heap sort ile sıraladım.

4 Comparison the Analysis Results



	100	500	1000	2000	3000	4000	5000	6000	8000	10000
insertion	5,14E-05	2,09E-04	6,71E-04	0,004406327	0,00436314	0,007632995	0,012481392	0,018103021	0,03420816	0,056820982
merge	2,70E-05	2,60E-04	9,80E-05	2,46E-04	3,76E-04	4,90E-04	5,87E-04	6,80E-04	0,001246755	0,001382754
quick	3,42E-05	2,55E-04	4,91E-04	0,003627558	0,001570506	0,00217467	0,003121667	0,003942316	0,005844193	0,009087911
heap	2,86E-05	1,26E-04	2,72E-04	4,51E-04	8,97E-04	8,67E-04	9,28E-04	8,52E-04	0,001093324	0,001562599
mergeDll	1,68E-04	4,24E-04	9,61E-04	0,004193016	0,008893539	0,015735924	0,024585301	0,034320826	0,061946341	0,094276533



	100	1000	5000	10000
insertion	5,13E-04	0,005931481	0,059658022	0,11950753
merge	1,13E-04	0,001013024	0,001049336	0,001953178
quick	1,96E-04	0,006634716	0,038463465	0,043955341
heap	1,37E-04	0,001286635	0,009118086	0,008171614
mergeDll	0,001256188	0,004709517	0,030968712	0,094355964