Gebze Technical University
Computer Engineering

CSE 222 - 2018 Spring

HOMEWORK 5 REPORT

EBRU KARDAŞ
141044049

Course Assistant: Fatma Nur Esirci

1 Double Hashing Map

I think as key is hash code and value is the element and implementation is according to this logic. Some methods are implemented (isEmpty, size, get, containsKey, containsValue, put)

1.1 Pseudocode and Explanation

1.2 Test Cases

2 Recursive Hashing Set

2.1 Pseudocode and Explanation

2.2 Test Cases

3 Sorting Algortihms

3.1 MergeSort with DoubleLinkedList

3.1.1 Pseudocode and Explanation

In algorithm, using linked list , instead of array, is more efficient. One of the reasons is removing from the head of the array when lower one is added to merged list. Also we can use iterator which gives us more efficient access opportunity. It may cause a disadvantage about space but we generally consider about time and when size becomes huge numbers, time is an important criterion.

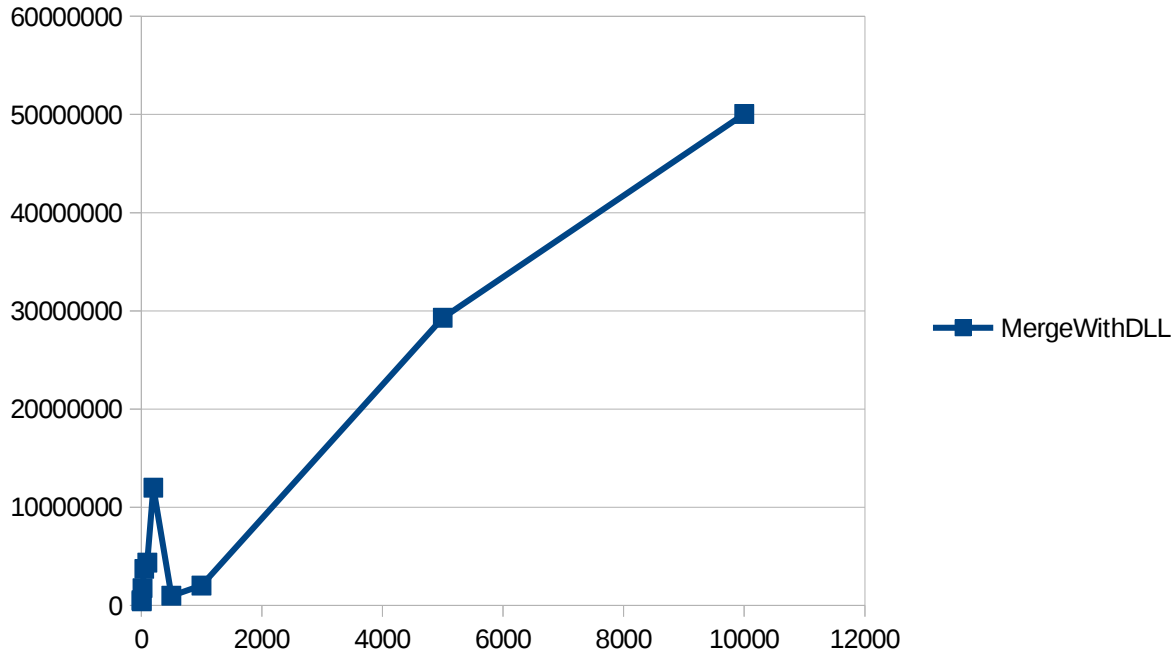
Merge sort is a divide&conquer sort algorithm with **worse-case time** is O(nlogn) which gives great opportunity to sort in less time as against most sort algorithm.

In implementation, list is divided by half recursively until size is 1. Then another private method is called which takes divided lists. Left sublist and right sublist elements is compared starting from zero index. Lower element is added to local linked list to return, the lower one is removed from the list that belongs to and this operation continues until one of the lists is empty. If any element left in any list, these elements are added to local linked list to return. Then no element left, operation is done, method returns the sorted linked list.

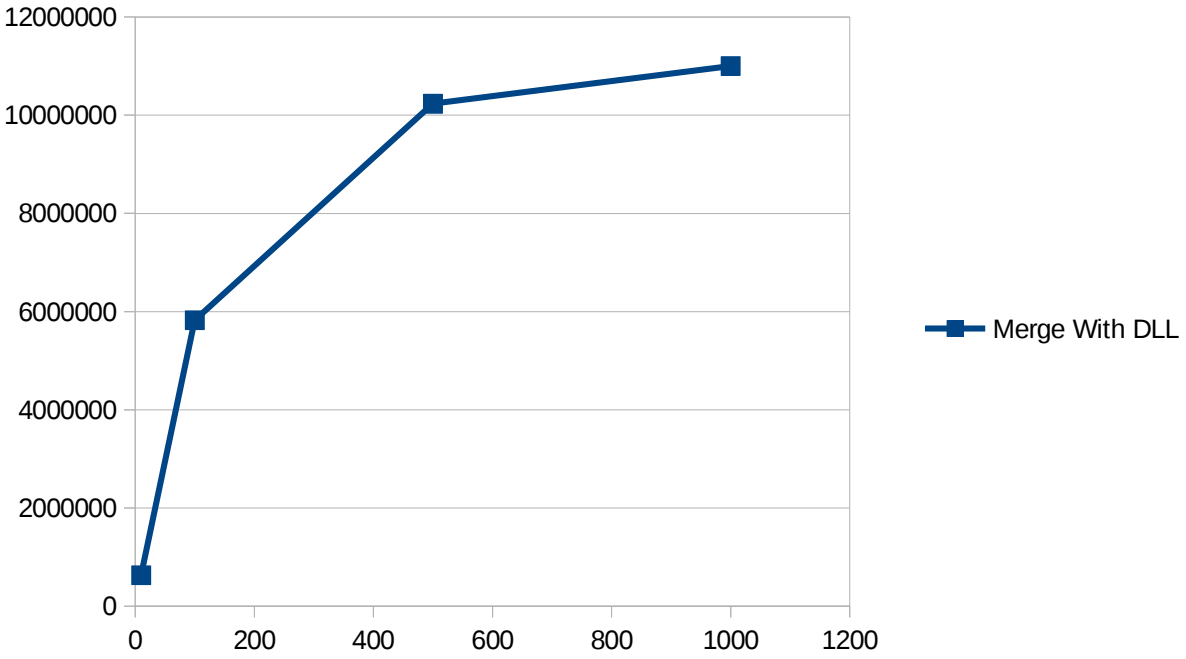
```
merge(list)
        if size is 1, return list
        create 2 LinkedList, divide argument list 2 and assign
        recursive call for left sublist
        recursive call for right sublist
        return merge(leftsublist, rightsublist)

merge(leftsublist, rightsublist)
   create a  LinkedList to return sorted merged list
   while leftsublist or rightsublist is not empty
```

if head of leftsublist is bigger than head of rightsublist,
add head of rightsublist to linkedlist and remove the element from the list that belongs to
else,
add head of leftsublist to linkedlist and remove the element from the list that belongs to
while there is any element left in leftsublist
add to linkedlist and remove from the list that belongs to
while there is any element left in righttsublist
add to linkedlist and remove from the list that belongs to
return linkedlist

3.1.2 Average Run Time Analysis



3.1.3 Wort-case Performance Analysis

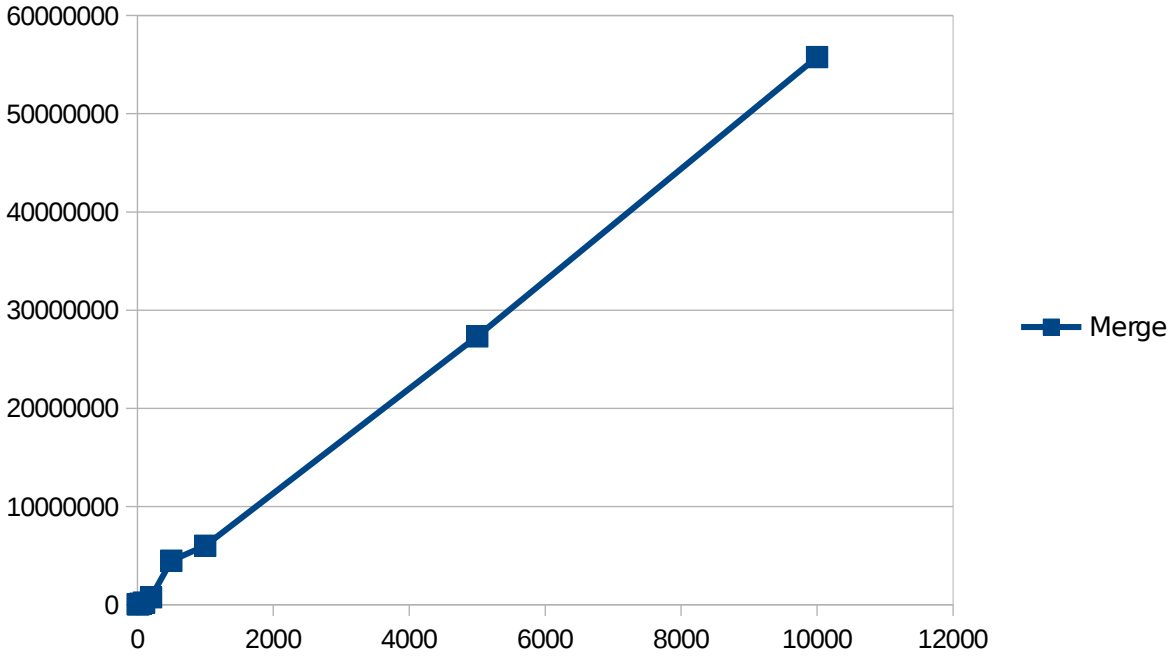


3.2 MergeSort

Merge sort is a sort algorithm with divide and conquer approach. Algorithm divides array until 1 or 2 element(s) left, sort them, and merge sorted arrays until done.

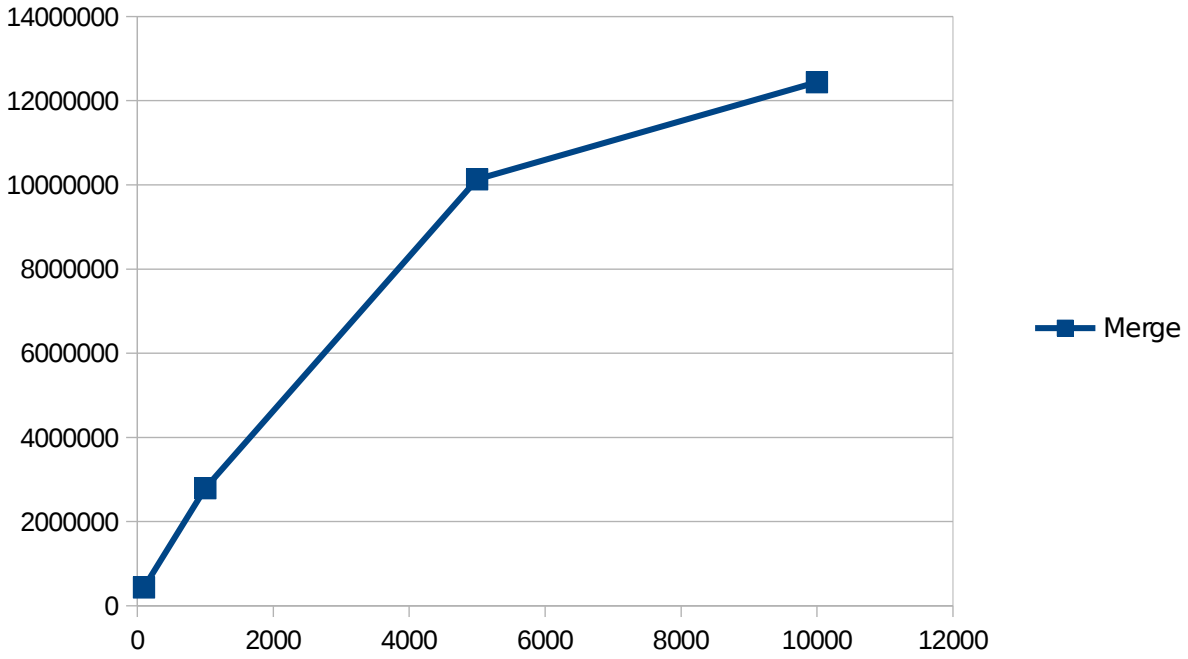
3.2.1 Average Run Time Analysis

In merge algorithm, if array is not sorted , then run time will be T(n)=O(nlogn).



3.2.2 Wort-case Performance Analysis

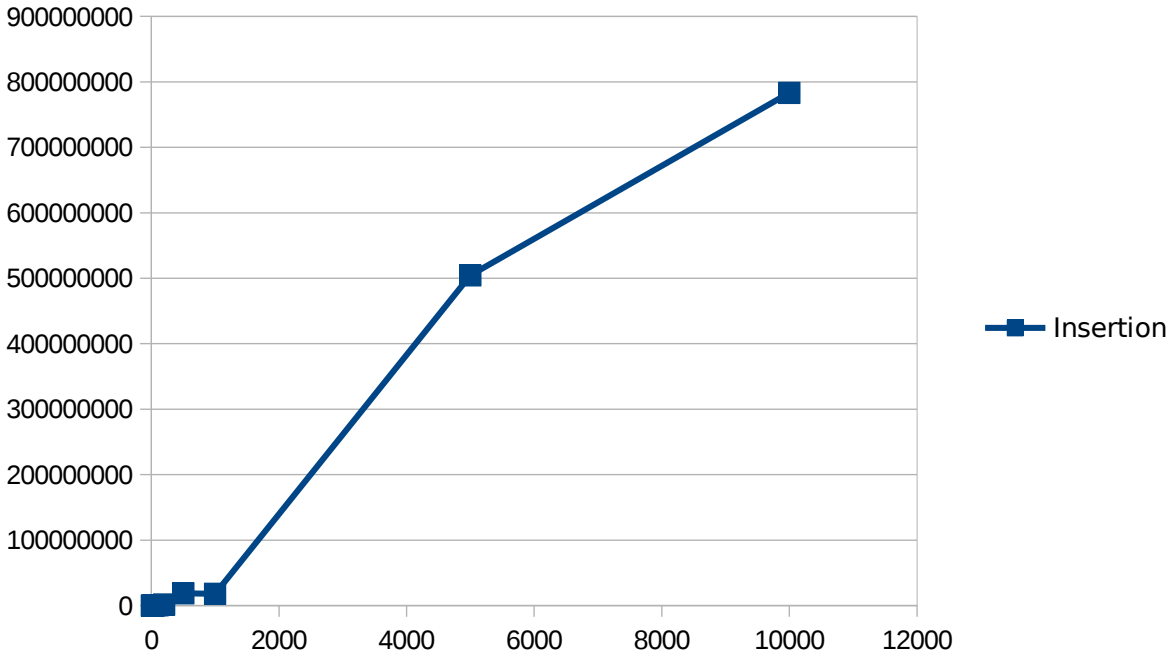
In merge algorithm, if array is already sorted, then run time will be T(n)=O(nlogn), similar to average run time.



3.3 Insertion Sort

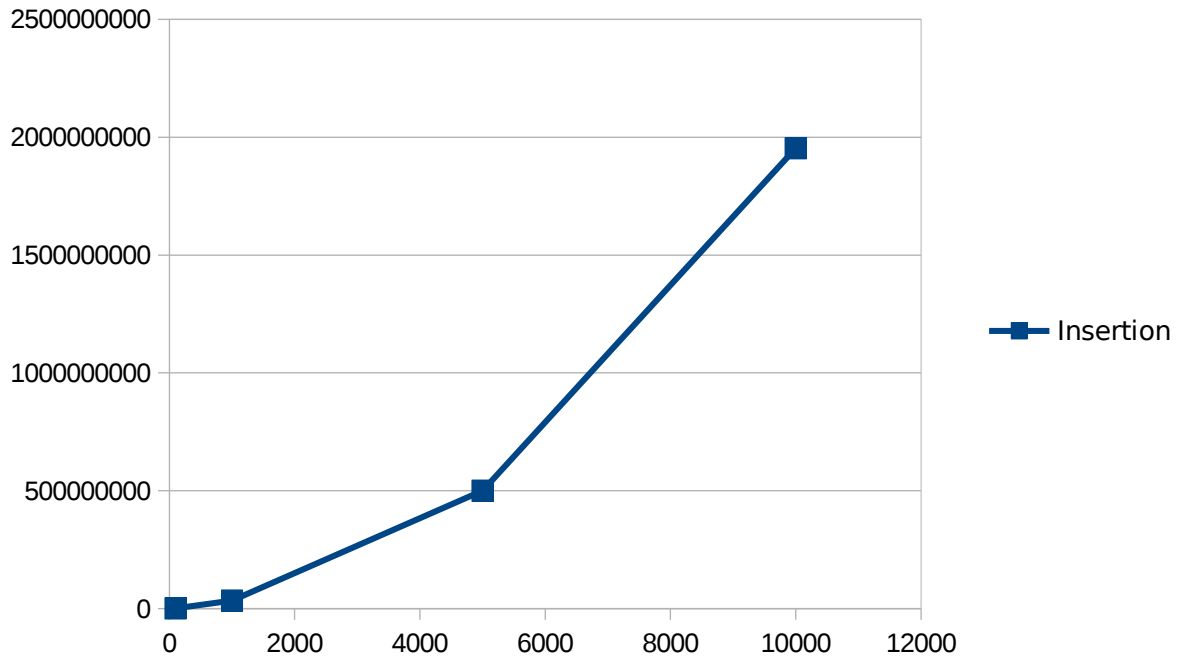
3.3.1 Average Run Time Analysis

Insertion sort is a sort algorithm which select elements from array respectively and swap until array is sorted. In best case, no swap and n comparison occurs.



3.3.2 Wort-case Performance Analysis

In this algorithm, worst-case scenario occurs when array is in reverse order. Time complexity is O(n^2) because of swap operations in each iteration. Results of the worst-case is shown below in graphic:

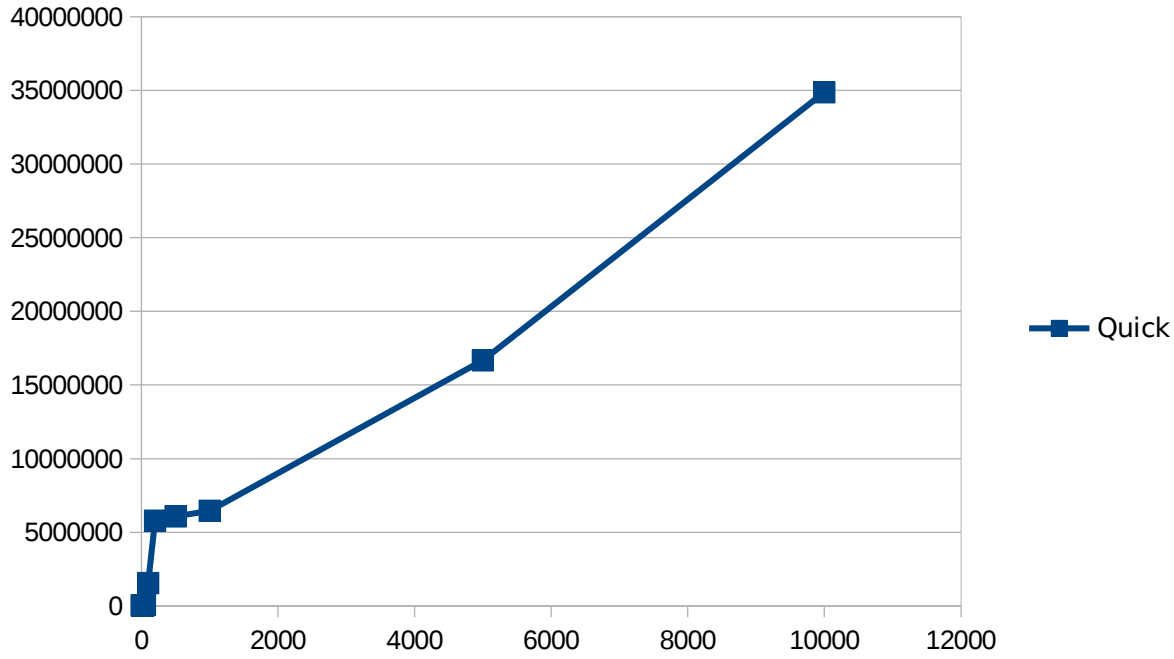


3.4 Quick Sort

In quick sort algorithm, like merge algorithm, divide and conquer approach is used. In algorithm, a pivot is chosen and sorts until done.

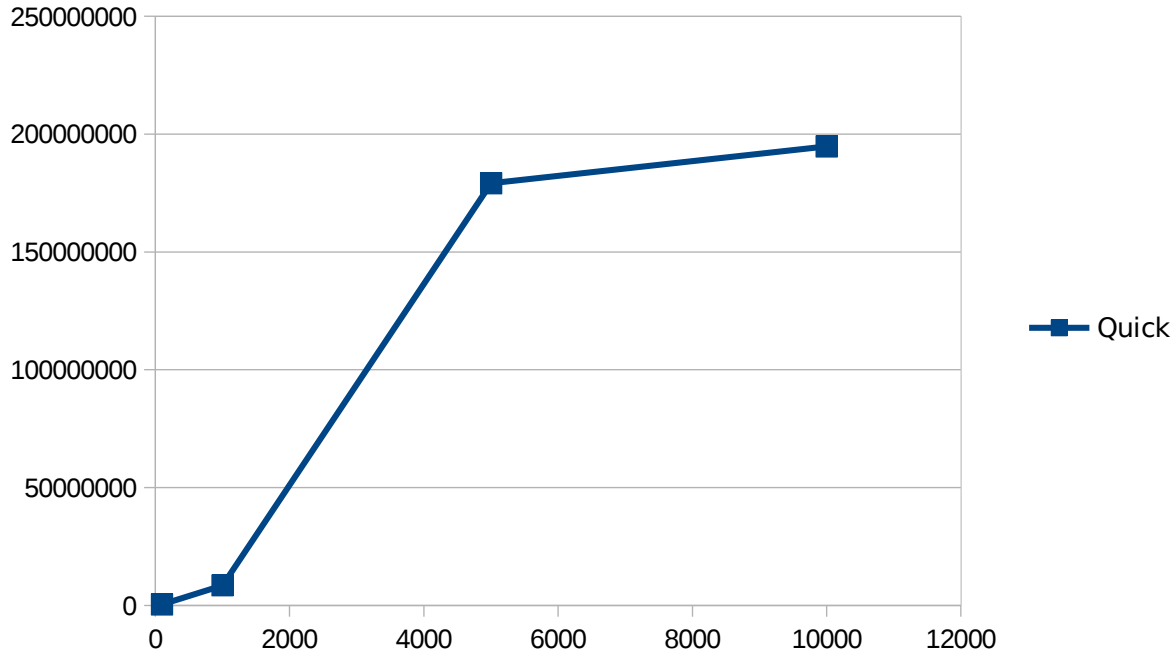
3.4.1 Average Run Time Analysis

It is not guaranteed that chosen pivot is the best choice but in average-case, run time is T(n)=O(nlogn) like shown in graphic in different size of arrays:



3.4.2 Wort-case Performance Analysis

If chosen pivot is a bad choice, then its run-time goes n^2 ($T(n)=O(n^2)$) shown below *with time that takes:*

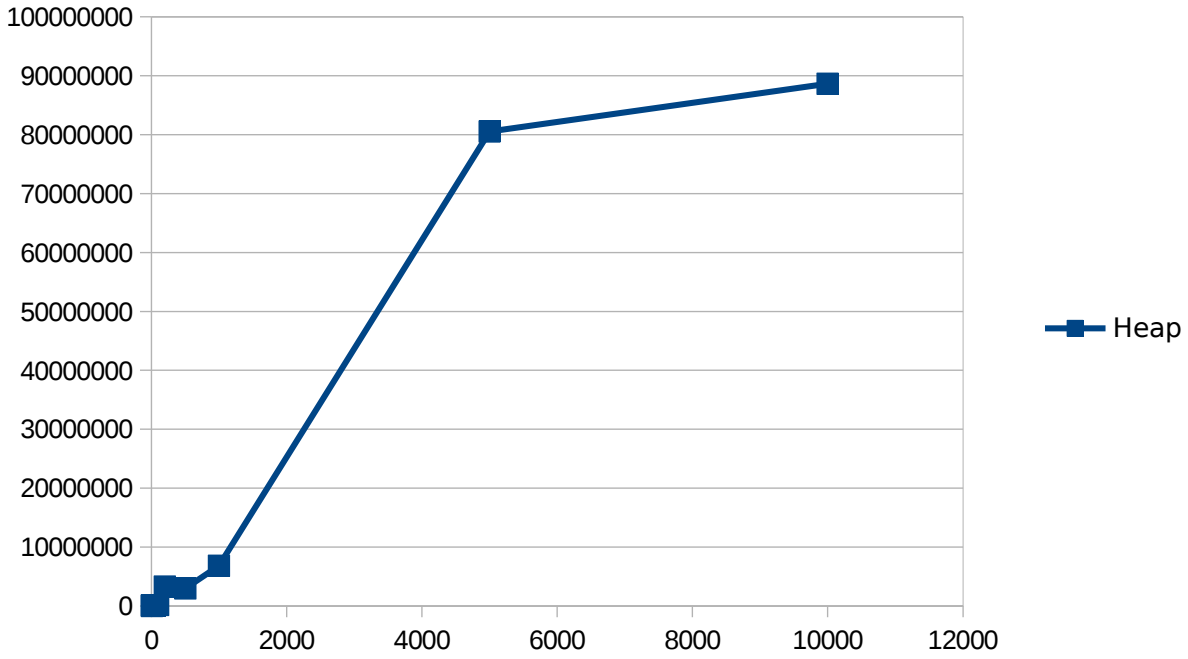


3.5 Heap Sort

In heap algorithm, it build a (max) heap tree and then pull elements from tree to array.

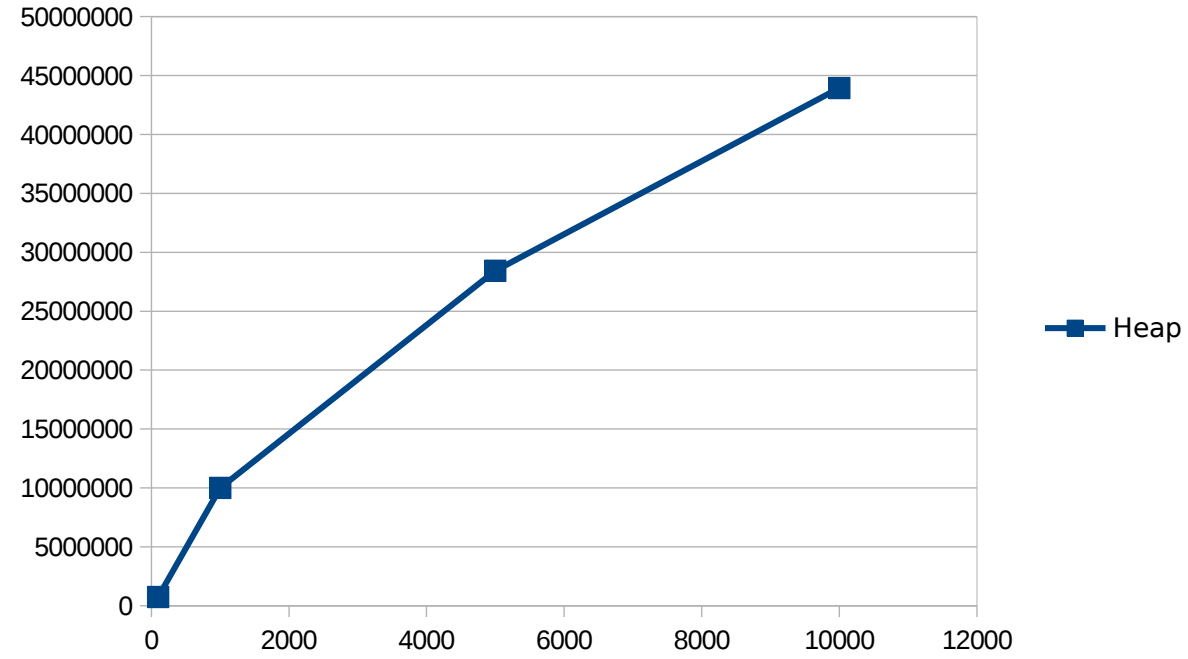
3.5.1 Average Run Time Analysis

Reading elements heap and traverse, takes T(n)=O(nlogn).



3.5.2 Wort-case Performance Analysis

Same as average, T(n)=O(nlogn)



4 Comparison the Analysis Results

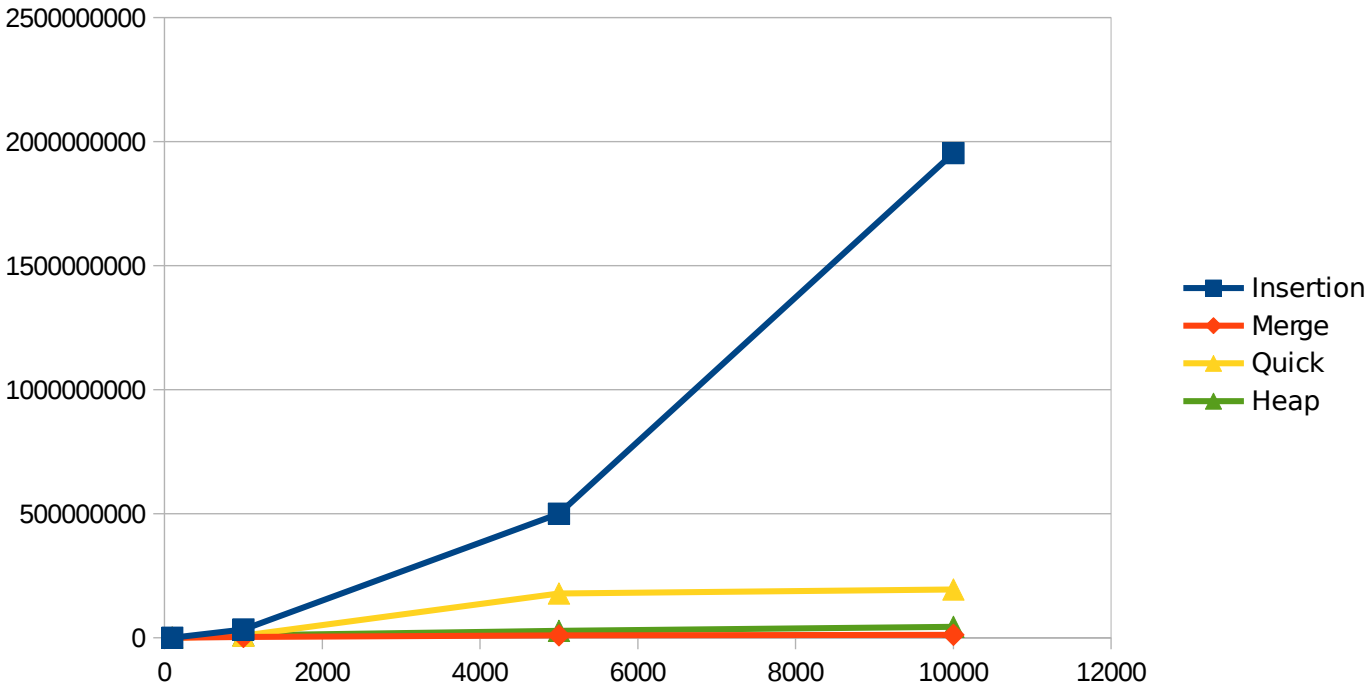


Figure 4.1. Comparison of sorting algorithms (this figure just a example)

Insertion sort takes more more time than other (n^2) like mentioned. Heap and merge sorts are similar and O(nlogn) time takes. Quick sort is not better than heap or merge and not worse than insertion, of course. Random list may not show the exactly worse case but quick sort is close in worst-time with bad pivot (O(n^2)) but posibility of chosing bad pivot is not high so can be considered in average time which is nlogn.