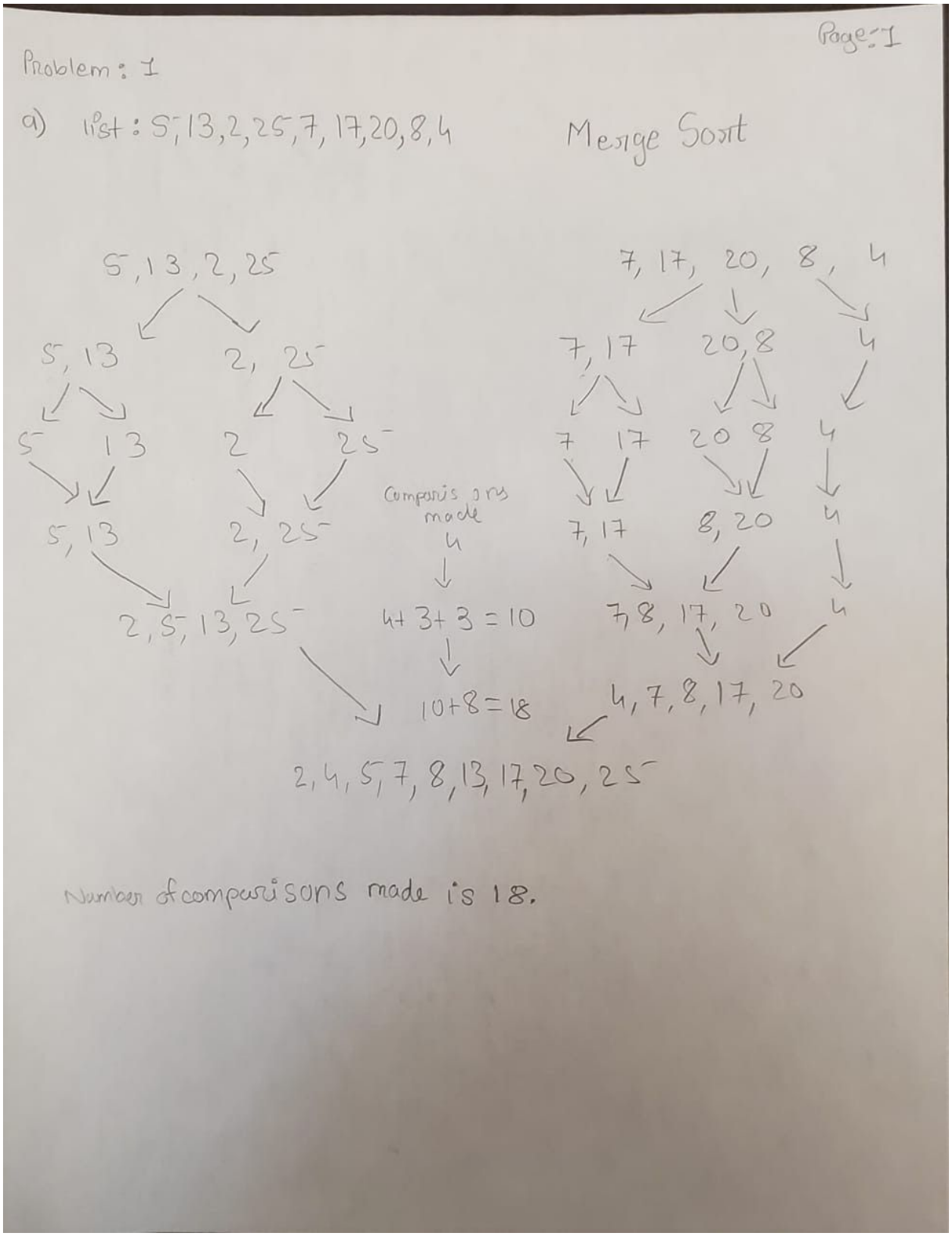
Saad Saiyed | 200415258

Assignment 5

Problem 1 (4+3 marks).

(a) Illustrate how the list 5, 13, 2, 25, 7, 17, 20, 8, 4 is sorted with Mergesort. Count the number of comparisons Made (b) Illustrate how the list 5, 13, 2, 25, 7, 17, 20, 8, 4 is sorted

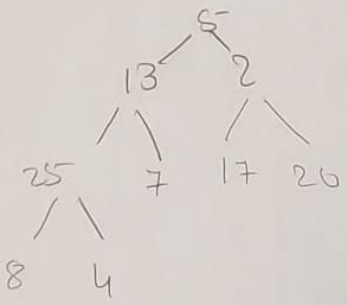
with Heapsort.



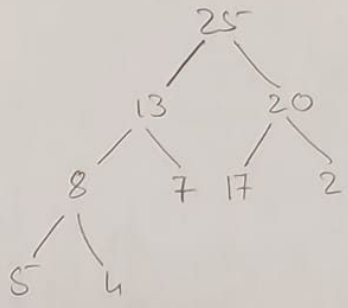
b.

List: 5, 13, 2, 25, 7, 17, 20, 8, 4 Heapsort

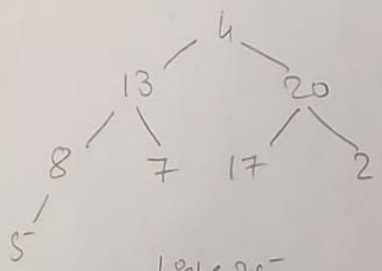
Creating tree



heapify

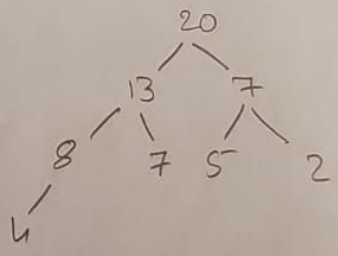


Deleting root

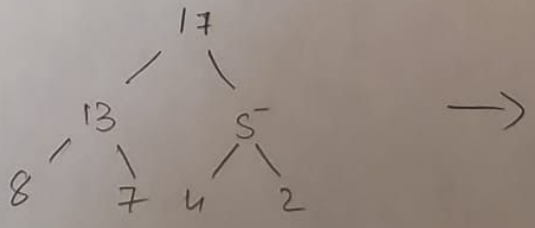


List: 25

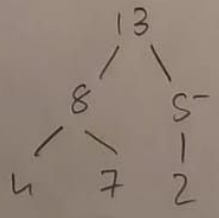
heapify



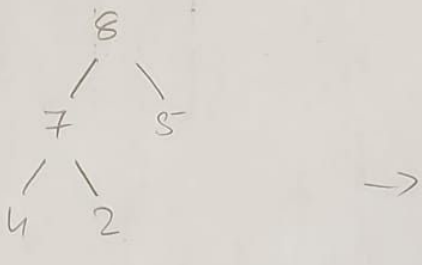
Deleting root and hepify at the same step



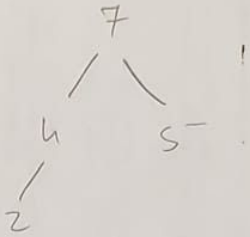
List: 20, 25



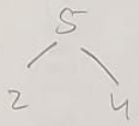
List: 17, 20, 25



List : 13, 17, 20, 25



List : 8, 13, 17, 20, 25



List : 7, 8, 13, 17, 20, 25

List : 5, 7, 8, 13, 17, 20, 25

Finally : 2, 4, 5, 7, 8, 13, 17, 20, 25.

Problem 2 (2+3 marks). Assume your sorting algorithms have to deal with lists that can potentially contain duplicates. Assume the same sorting algorithms as discussed in class / in the textbook.

(a) What is the running time of Insertion Sort when all elements in a given list of length N are equal? Explain your answer.

Ans: If all the elements in a given list with size N are the same, the running time complexity would be O(N) for insertion sort as it is the best case. One loop through the array will give you the result. The basic fundamental of insertion sort is to make a separate array to store sorted lists. This is done by having a pointer which moves along the array checking the last inserted value is bigger than current value. Which is neglected for any disordered insertion as all the elements are the same.

(b) Give a Θ-bound on the running time of Mergesort for the case that all elements in a given list of length N are equal (assuming N is a power of 2). Explain your answer.

Ans: Considering a list of all same elements, O(log N) for merge sort. This is achieved as merge sort consists of three steps; First separation of array, second arrangement and last combining the results. Separation takes time complexity of O(1). Arrangement takes O(log n) where n != N. Finally combining has time complexity of O(n). This means that if the N is equal to some number which is a power of 2 let's say k, then by increasing the number of N the value of k will be slightly increased. Therefore time complexity of merge sort for large input is negligible.

Problem 3 (4 marks). A sorting algorithm is stable if the relative order of any two equal entries in the given array stays the same: when two records a[i] and a[j] are equal in

content, and i < j, then the algorithm sorts the array in a way that the record originally stored in a[i], still appears to the left of the record originally stored in a[j], when the array is sorted. Which of the algorithms Insertion Sort, Shellsort, Heapsort, and Mergesort (as presented in class) are stable, which are not?

Ans: Stable sorting algorithms are Insertion sort, Merge sort, Radix sort, etc as they are reliable if the two data have the same content but their values are different. Unstable sorting algorithms are Shell sort, Heap sort, Quick sort.

Problem 4 (2+2+2+2 marks). Analyze the following recurrence relations using the Master Theorem, and give a - bound for each.

(a) T(N) = 2T(N=4) + 1.

(b) T(N) = 2T(N=4) + pN.

(c) T(N) = 2T(N=4) + N2.

(d) T(N) = 9T(N=3) + N.

Ans:

Q.4 Page: 1

Master Theorem can be applied if the recurrence relation satisfy $T(n) = a \cdot T(n/b) + f(n)$ format.

There are 3 cases in which we can solve a relation.

i) $f(n) = \Theta(n^c)$, $c < \log_b a$ then $T(n) = \Theta(n^{\log_b a})$

ii) $f(n) = \Theta(n^c)$, $c = \log_b a \Rightarrow T(n) = \Theta(n^c \log n)$

iii) $f(n) = \Theta(n^c)$, $c > \log_b a \Rightarrow T(n) = \Theta(f(n))$

a. $T(N) = 2T(N/4) + 1$

$a=2, b=4, \log_4 2 = 0.5, c=0$

$\therefore c < \log_b a \Rightarrow 0 < 0.5$

$\therefore$ case i is applicable $\Rightarrow T(N) = \Theta(N^{0.5})$

b. $T(N) = 2T(N/4) + \sqrt{N}$

$a=2, b=4, \log_4 2 = 0.5, c = 0.5$

$c = \log_b a \Rightarrow 0.5 = 0.5$

$\therefore$ case ii is applicable $\Rightarrow T(N) = \Theta(\sqrt{N} \log N)$

C. $T(N) = 2T(N/4) + N^2$

$a=2,\ b=4,\ \log_4 2 = 0.5,\ c = 2$

Now $c > \log_b a \Rightarrow 2 > 0.5$

$\therefore$ case iii) is applicable, $T(N) = \Theta(N^2)$

D. $T(N) = 9T(N/3) + N$

$a=9,\ b=3,\ \log_3 9 = 2,\ c=1$

$c < \log_b a \Rightarrow 1 < 2$

$\therefore$ case i) is applicable, $T(N) = \Theta(N^2)$

Problem 5 (2 marks). Explain why the Master Theorem cannot be applied to the recurrence relation T(N) = 2T(N=2)+ N log(N).

Ans: The master theorem only applies to the recurrence relation which satisfies a particular format of T(n) = a.T(n/b) + f(n) where a >= 1 and b > 1 and f(n) = polynomial. In the case of *T(N) = 2T(N=2)+ N log(N)*, f(n) is a logarithmic function which doesn't satisfy Master Theorem's conditions. This is why the master theorem is not applicable to given recurrence relation.