Homework No. 1 - Introduction & Growth of Functions

Neha Goud Baddam 11519516

Q1. Suppose we are comparing implementations of insertion sort and merge sort on the same machine. For inputs of size n, insertion sort runs in 8n 2 steps, while merge sort runs in 64nlg n steps. For which values of n does insertion sort beat merge sort? (Exercise 1.2-2)

To compare implementations of insertion sort and merge sort on the same machine., we need to compare the time complexities of insertion and merge sort

Solution 1: Insertion sort beats merge sort (for input sizen), when 8n² (insertion sort steps) is less than canlogn (merge sort steps)

Then 8n² × 64n logn

8n² × 64n logn

n × 8 logn

n × 8 logn

n × 100 logn

hets substitute in value to balance the equation

Firstly less assume that n=8, then 24× × n

When we keep substituting n values, we notice that for n=43, insertion sort starts to beat merge sort

For n=43, 218 = 42.4 × n

Therefore for in × 43, insertion sort heats merge sort 2

for n=44, 211/2 = 44.8 > n

Homework No. 1 - Introduction & Growth of Functions

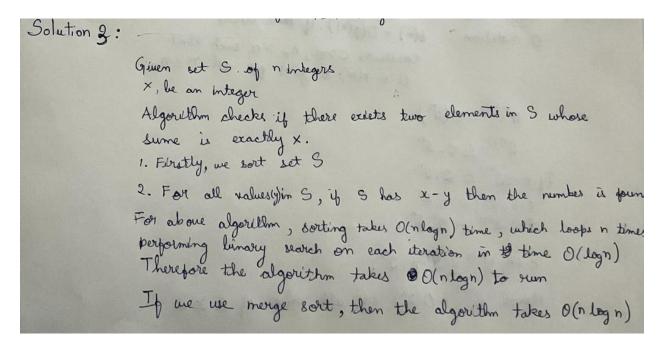
Neha Goud Baddam 11519516

Q2. We can express insertion sort as a recursive procedure as follows. In order to sort A [1 .. n], we recursively sort A [1 .. n-1] and then insert A [n] into the sorted array A [1 .. n-1]. Write a recurrence T(n) for the running time of this recursive version of insertion sort. (Exercise 2.3-4)

Q3. Describe a Θ (n lg n) - time algorithm that, given a set S of n integers and another integer x, determines whether there exist two elements in S whose sum is exactly x. (Exercise 2.3-7) we can sort the array with merge sort Θ (n log n) and then for each element (say y) in the array, we can do a binary search for (x - y) on the sorted array. So, the algorithm will run in Θ (nlog n).

Homework No. 1 - Introduction & Growth of Functions

Neha Goud Baddam 11519516



Q4. Sort all the functions below in increasing order of asymptotic (big-O) growth. If some have the same asymptotic growth, then be sure to indicate that. As usual, lg means base 2.

- 1. 5n
- $2. n^4$
- 3. 4 lg n
- 4. n^{n/4}
- 5. $n^{1/2}log_n^4$

Answer: $4 lgn < 5n < n^4 < n^{1/2} log_n^4 < n^{n/4}$

Solution 4: Time complexities for Big-O are in the following order $O(\log n) < O(n) < O$

Homework No. 1 - Introduction & Growth of Functions

Neha Goud Baddam 11519516

Q5. Prove that 2 n+1 = O(2n). (Exercise 3.1-4) (Hint: Try to satisfy the definition of O-notation with some constants c, n0 > 0)

Solution 5: Prove that
$$2^{n+1} \ge O(2^n)$$
 0 -notation: $f(n) \ge O(g(n))$, if there exists

constants $c > 0$, $n_0 > 0$ such that

 $constants c > 0$, $n_0 > 0$ such that

 $0 \le f(n) \le cg(n)$ for all $n \ge n_0$
 $2^{n+1} \ge O(2^n)$
 $2^{n+1} \le c \cdot 2^n$
 $c \ge 3$, $2^{n+1} \le 0 \cdot 2^n$
 $c \ge 3$, $2^{n+1} \le 0 \cdot 2^n$
 $c \ge 7$, $2^{n+1} \le 0 \cdot 2^n$
 $c \ge 7$, $2^{n+1} \le 0 \cdot 2^n$

Hence $2^{n+1} \ge O(2^n)$ for all values of $c \ge 2$