Cesar Camarena
T00223324

CHAPTER I HOMEWORK

Exercise 1.1:

5.) Design an algorithm to find all the common elements in two sorted lists of numbers. For example, for the lists 2, 5, 5, 5, and 2, 2, 3, 5, 5, 7, the output should be 2, 5, 5. What is the maximum number of comparisons your algorithm makes if the lengths of the two given lists are *m* and *n*, respectively?

If (m < n || m == n)

then answer array is length n

else

answer array is length m.

for 0 to size m

x ← m[i]

for 0 to size n

y ← n[i]

if x == y

then answer array = y

return answer array

O(n^2)

9.) a. What is the minimum number of divisions made by Euclid's algorithm among all inputs $1 \le m, n \le 10$?

1, that is the smallest number of divisions for any 2 numbers.

b. What is the maximum number of divisions made by Euclid's algorithm among all inputs $1 \le m, n \le 10$?

5, because it takes 5 divisions for the GCD of (5,8) which is the maximum for inputs between 1 – 10.

Cesar Camarena
T00223324

Exercise 1.2:

5.) Describe the standard algorithm for finding the binary representation of a positive decimal integer

a. in English. Divide the integer by 2, then the remainder will either be 0 or 1. Then divide the quotient by 2 and repeat. Record the 0 and 1 by left to right.

b. in pseudocode.

w ←0

while n is not equal to 0

then store the number mod 2

divide the number by 2

repeat

Exercise 1.3:

1.) Consider the algorithm for the sorting problem that sorts an array by counting, for each of its elements, the number of smaller elements and then uses this information to put the element in its appropriate position in the sorted array:

ALGORITHM *ComparisonCountingSort*$(A[0..n-1])$

```
//Sorts an array by comparison counting
//Input: Array A[0..n − 1] of orderable values
//Output: Array S[0..n − 1] of A's elements sorted
//    in nondecreasing order
for i ← 0 to n − 1 do
    Count[i] ← 0
for i ← 0 to n − 2 do
    for j ← i + 1 to n − 1 do
        if A[i] < A[j]
            Count[j] ← Count[j] + 1
        else Count[i] ← Count[i] + 1
for i ← 0 to n − 1 do
    S[Count[i]] ← A[i]
return S
```

a. Apply this algorithm to sorting the list 60, 35, 81, 98, 14, 47.
Um?... Idk what to do here. 14, 35, 47, 60, 81, 98.

b. Is this algorithm stable?
No because there is swapping.

c. Is it in-place?
I don't think so since it uses more arrays to compute the list.

Cesar Camarena
T00223324

Exercise 1.4:

1.) Describe how one can implement each of the following operations on an array so that the time it takes does not depend on the array's size *n*.

a. Delete the *i*th element of an array ($1 \le i \le n$).

Replace the element with a different number and decrease the array size by 1.

b. Delete the *i*th element of a sorted array (the remaining array has to stay sorted, of course).

Either replace with the ith element with null or place a number there that signifies a deleted element

2.) If you have to solve the searching problem for a list of *n* numbers, how can you take advantage of the fact that the list is known to be sorted? Give separate answers for

a. lists represented as arrays.

I think a binary search tree would reduce the amount of time to find your element by a lot.

b. lists represented as linked lists.

Just take the element that is less than or equals to the key, or vice versa depending on what you're looking for, and then stop.