Habib University - Algorithms: Design and Analysis (CS/CE 412/471) – Spring 2025

Quiz 01

Name: ______________________________________ ID: __________ Section: ________

Q1 [15 points]. You need to design an efficient algorithm for the following problem. **No credit** if you use any programming language specifics or methods e.g. list.append(), etc. Where arrays are mentioned, assume a simple and crude array A[1..k] having k elements starting from index 1 to k inclusive. Note that you must design a simple algorithm and try to use the building blocks wherever possible.

Consider an array A[1..n], where A[k] is from the set {0, 1, 2} where k = 1..n. Sort the array.

Answer in the space provided; only part d on back side:

a. [1 point] State the input(s): *An unsorted array A[1..n] having elements 0, 1, and 2.*

b. [1 point] State the output(s): *A sorted array A[1..n] having elements 0, 1, and 2.*

c. [5 points] Basic Idea in Simple English i.e. Pseudocode using the notation stated in CLRS. If you're using a building block, clearly mention how you are using/modifying it in your algorithm.

1. *Set pivot = 1*
2. *Apply Procedure Partition on A*
3. *Set pivot = 2*
4. *Apply Procedure Partition on B (the array returned in 2.)*

```
MODIFIED-PARTITION(A, n, pivot)
let B[1..n] be a new array
left = 1

for i = 1 to n do
    if A[i] < pivot then
        B[left] = A[i]
        left = left + 1

for i = 1 to n do
    if A[i] >= pivot then
        B[left] = A[i]
        left = left + 1

return B
```

d. [3 points] Show one example to show the working of your algorithm. Include illustrations. **(back side)**

e. [2 points] Time complexities for upper and lower bounds. $\Omega(\ n\ )$, $O(\ n\ )$

f. [2 point] Is your algorithm stable? If not, how will you make it stable?

Yes. The relative orderings of 0s, 1s, and 2s do not change.

g. [1 point] Is your algorithm in-place? **Yes / No**

No, as we need another array B.

Q2. [5 points] Show that if f(n) = O(g(n)) and g(n) = O(h(n)), then f(n) = O(h(n)).

A function $f(n)$ is said to be $O(g(n))$ if there exist positive constants c_1 and n_1 such that:

$$f(n) \leq c_1 \cdot g(n), \quad \text{for all } n \geq n_1$$

A function $g(n)$ is said to be $O(h(n))$ if there exist positive constants c_2 and n_2 such that:

$$g(n) \leq c_2 \cdot h(n), \quad \text{for all } n \geq n_2$$

Substitute the 2nd inequality into the 1st:

$$f(n) \leq c_1 \cdot c_2 \cdot h(n), \quad \text{for all } n \geq n_0 \text{ where } n_0 = \max(n_1, n_2)$$

Let $c = c_1 \cdot c_2$,

$$f(n) \leq c \cdot h(n), \text{ for all } n \geq n_0$$

Thus, we have proved that if $f(n) = O(g(n))$ and $g(n) = O(h(n))$, then $f(n) = O(h(n))$.