University of Petra	جامعة البترا - ثلاثون عاماً University of Petra
Faculty of Information Technology	كلية تكنولوجيا المعلومات
Department of Computer Science	قسم علم الحاسوب

جامعة البترا

Advanced Algorithms
601326
Midterm Exam – 2024 2

Instructions for the Exam:

- Write your name and ID number on the exam and answer sheets.
- Write the number of the section that you enrolled in.
- Write the name of your instructor.
- Questions in the exam not allowed.
- Using any type of technology (mobiles, smart watches) not allowed
- Using extra papers or sheets not allowed
- The exam consists of Six questions.

did i improve [illegible]?

For instructor use only:

Question number	Course ILO	Program ILO	Question weight	Student mark
Q1			4	1.5
Q2	K2		4	4
Q3	I1		7	1.5
Q4			5	2.5
Q5			6	2.5
Q6			4	3.25
Total /30				15.25

Q1) Prove by induction that: **(4 marks)**

$$\sum_{i=0}^{n} 2^{i} = 2^{n+1} - 1$$

1.5

Bare = 0

when i = 0

$2^0 = 2^{0+1} - 1$

$1 = 2 - 1$

$1 = 1$ ✓

$n = K$

$2^i = 2^{K+1} - 1$

$n = K+1$ ✓

$\sum 2^i = 2^{K+2} - 1$ ✓

$\sum 2^i = 2^K \cdot 2^2 - 1$?

$2^i = 4(2^K) - 1$

I don't Know

Q2) Consider the following algorithm. **(4 marks)**

4

```
Algorithm XYZ (A [0..n − 1])
//Input: An array A[0..n − 1] of n real numbers     1
val ← 5                 o(1)
res1 ← 0                0 1
res2 ← 1                1
for i ← 0 to n − 1 do   n-1
   if A[i] <= val       1
   res1 ← A[i]+res1     current res1   1
   if A[i] > val        1
   res2 ← A[i]*res2     1
return res1 – res2      1
```

a. What does this algorithm compute?
b. What is the time complexity for the algorithm?

b: o(n)

a) sets a value (val)

and then search in the array

iF ther is avalue less than=it

it will add it to res1

iF > then multiely by res2

then after it

it returns res1 - res2

ب

Q3) Design a **divide-and-conquer** algorithm to find **Minimum** Number in an unsorted array of N elements (**without using any of the sorting algorithms covered in the class**). **Setup** a recurrence relation for your algorithm. **(7 marks)**

1.5

```
• i = 0
• min (arr, i) {
• minVal = arr[1]
~~if [illegible]~~ • if (arr[i] > arr[i+1])
    •• min Val = arr[i+1]

    ~~return (arr, i+1)~~

    ~~return min(arr, i+1)~~
    • if i <= arr.length
        •• return min(arr, i+1)
• return minval
```

$T(n) = ?$

}

Q4) Consider the following recursive algorithm: **(5 marks)**

```
Algorithm Q(n)
//Input: A positive integer n
if n =1
    return 1
else
    return Q(n − 1) + 2 ∗ n − 1
```

2.5

a) **Setup** a recurrence relation for the number of multiplications made by this algorithm.

b) **Solve** the recurrence relation using backward substitution and find Big O.

I know but ~~[illegible]~~ i forgot how to get it so i solve it

$T(n) = ?!$

~~b) (n-1)+2~~

1

b) $t(n) = (n-1)+2$

$t(n-1) = T(n-2)+2$

$t(n-2) = T(n-3)+2$

$t(n-3) = t(n-4)+2$

$t(n) = t(n-4)+4(2)$

$i = 4$

$t^{o}(n-i)+i(2)$

when

$n=1$ $t(n) = \cancel{1}\,8$ 0

$t(1)+2(n-1)$

$n = 1+i$

$1+2n-2$

$n-i = 1$

$-i = 1-n$

$i = n-1$

$O(n)$

i think

-3 4 8 6 9 13 20 22

Q5) Given the following array, A= -3 ,8, 4, 22, 13, 6, 20, 9 **(6 marks)** 2.5

a) Apply Quick Sort algorithm to sort the array elements in ascending order **(show detailed steps)**

9 6 13 22

b) Compare Quick sort to Insertion Sort in terms of best and worst cases.

n log n n^2

n n^2

B: quick: best: n log n
worst: n^2

insersion: best: n
worst: n^2

2

Pi~~[illegible]~~ -3, 8, 4, 22, 13, 6, 20, 9

artition (s) ← 1. we choose a pivot = ~~[illegible]~~ 13 0.5

2. we Find its place

2.1 compare every element untill every element to its left is ≤ pivt and right side > pivot

assign l/r to compare and do it

~~[illegible]~~

array becomes

-3, 8, 4, 9, 6, 13, 20, 22
L s r

then we solve Left and right the same way

method[L...s-1]
method[s+1...r]] → repeating the steps to the left side and right

s → what the partition returns

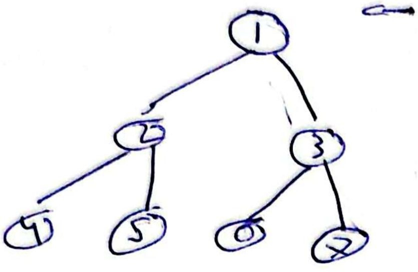
But You should apply these steps to the given array!!

(I didn't Know you want most left.)

you said you mentioned it in one of the lectures I didn't attend and I don't have time to rewrite :(

Q6) Show that the worst case running time of MAX-HEAPIFY on a heap of size *n* is ***log n*** (Hint: For a heap with n nodes, give node values that cause MAX-HEAPIFY to be called recursively at every node on a simple path from the root down to a leaf.) **(4 marks)**

3.25



→ its gonna check the
children of 1.
and both kids are > root
its gonna compare
and make ~~it~~ 3 root
~~but then theres~~
and then the ①
Is still not in place
so its gonna check its children aswell
here it checks every right side
then the ② same checking
then the ⑤ has to go up
and thats how you check all

?!