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

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.

For instructor use only:

Question number	Course ILO	Program ILO	Question weight	Student mark
Q1			4	3.75
Q2	K2		4	4
Q3	I1		7	6
Q4			5	4.5
Q5			6	5.75
Q6			4	3.5
Total /30				27.5

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

3.7.5

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

(first iteration in counter)

- let $n = 0$:

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

$$2^0 = 2^1 - 1$$

$$1 = 1 \checkmark$$

- let $n = k+1$:

$$\sum_{i=0}^{k+1} 2^i = 2^{(k+1)+1} - 1$$

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

$$\underbrace{2^0 + 2^1 + 2^2 + \cdots + 2^k}_{\text{original LHS}} + 2^{k+1} = 2^{2k+2} - 1$$

$$2^{k+1} - 1 + 2^{k+1} = 2(2^{2(k+1)}) - 1$$

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

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

```
Algorithm XYZ (A [0..n − 1])
//Input: An array A[0..n − 1] of n real numbers
val ← 5          C(1)
res1 ← 0         C(1)
res2 ← 1         C(1)
for i ← 0 to n − 1 do    C(n)
    if A[i] <= val
    res1 ← A[i]+res1     sum of all elements ≤5 .  (2)
    if A[i] > val
    res2 ← A[i]*res2     product of all elements >5   (1 + T(n-1))
return res1 – res2
```

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

a) Difference between Sum of elements ~~greater~~ less then or ~~equal~~ to 5 and product of all elements greater then or equal to 5.

b) $O = n + 2(n-1) + (1 + T(n-1))$

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

$= 4n + T(n-1) - 3$

$O(n)$.

4

where ??

(recursive & $n/2$)

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)**

6/

```
findMin (A[0... n-1])
min1 ← A[0]
for i ← 0 to ⌊N/2⌋
    if A[i] < min1
        min1 ← A[i]
min2 ← A[⌈N/2⌉]
for i ← ⌈N/2⌉ to N
    if A[i] < min2
        min2 ← A[i]
if min1 < min2
    return min1
else return min2
```

$$T_n = \begin{cases} 0 & n = 1 \\ 1 + T(n+1) & n > 1 \end{cases}$$

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

4.5

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

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.

a) $T_n = \begin{cases} 1 & n=1 \\ 1+Q(n-1) & n>1 \end{cases}$

b) $Q(n-1)+1$

$Q(n-2)+1$

$Q(n-3)+1$

$Q(n-4)+1$

$T_n = Q(n-4)+1+1+1+1$

$T_n = Q(n-4)+4$

$T_n = Q(n-i)+i$

$Q(1) = 1$

$n - i = 1$

$i = n-1$

$T_n = Q(1) + n - 1$

$T_n = 1 + n - 1$

$O(n)$

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

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

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

5.75

missing first step!

a) A = | -3 | 8 | 4 | 22 | 13 | 6 | 20 | 9 |

take ~~(first~~ , middle and ~~last~~ elements ~~& compare~~:

13 ~~22~~ 9

Set up loops i & j

A = | -3 | 8 | 4 | 22 | 13 | 6 | 20 | 9 |

i j

A = | 8 | 4 | 22 | 13 | 6 | 20 | 9 |

pivot i j

3.75

-3

A = | 8 | 4 | 6 | 13 | 22 | 20 | 9 |

j i

-3

A = | 6 | 4 | 8 | 13 | 22 | 20 | 9 |

| 6 | 4 |

pivot

| 13 | 22 | 20 | 9 |

pivot i j

| 4 | 6 |

| 13 | 9 | 20 | 22 |

| 9 | 13 | 20 | 22 |

| 4 |

| 4 | 6 | 8 |

| -3 | 4 | 6 | 8 |

9

| 20 | 22 |

pivot

22

| 9 | 13 | 20 | 22 |

| -3 | 4 | 1 | 8 | 9 | 13 | 20 | 22 |

b)

- Insertion sort best case: already sorted array O(n)
- Insertion sort worst case: reverse sorted array $O(n^2)$
- ~~Insertion sort average case: shuffled array $O(n^2)$~~
- Quick sort best case: balanced partition using median of three O(nlogn)
- Quick sort worst case: sorted array or reverse sorted array $O(n^2)$

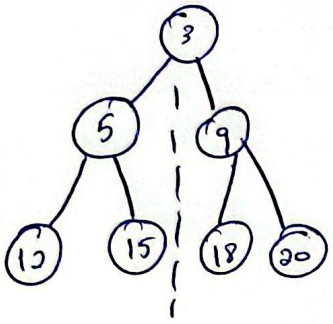
Best case: Quick sort is faster

Worst case: Same time complexity.

2

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)**

3.5



- called at last parent node (9) checks to see if children are within n. Compares to children (18 & 20); 18 > 9, compares 18 to 20; 18 < 20, swaps 9 & 20
- Called at second last parent node (5) compares to 12 and 15, swaps 5 & 15
- called at first parent node (3) compares to 15 & 20, swaps 3 and 20
- called again to swap 3 with 18.
- calls max-heapify recursively.

?!

(splits the heap into 2 smaller heaps for a time complexity of $\log_2 n$.).