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

Advanced Algorithms
601326

[illegible]

Your N

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, etc.) not allowed
- Using extra papers or sheets not allowed

For instructor use only:

Question number	Course ILO	Program ILO	Question weight	Student mark
Q1			5	3
Q2			5	5
Q3			2	0.75
Q4	I2		4	2
Q5			5	5
Q6			5	4
Q7	I2		4	3
Q8			3	1.5
Q9			7	3.5
Total			40	27.75

This exam has 9 Questions. The total mark is 40

3

Question 1) Choose the correct answer for each of the following: **(5 marks)**

1. What is time complexity of TSP using Branch and Bound in the worst case? (where n is number of vertices and E is number of edges)
   a) O(1)
   b) O(n*E)
   c) O(n^2)
   [✓ d) O(n!)] ✓
2. What is the space complexity of the Insertion Sort algorithm?
   a) O(1)
   b) O(n^2)
   [✓ c) O(n)] ✗
   d) O(n log n)
3. Given Build_Max_Heap Algorithm:

```
Build_Max_Heap(A){
    n= length(A)
    for i= n/2 down to 1
        max_heapify(A, i, n)
}
```

Based on your understanding of the algorithm, why does i start from n/2 ?

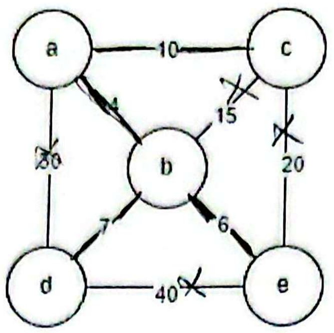
   [✓ a) to get value of non-leaf vertices] ✗
   b) to get index of non-leaf vertices
   c) to divide A into two sub-arrays and sort each one separately
   d) to call max_heapify only twice
   e) none of the above

4. Which of the following algorithms design techniques support backtracking:
   a) Divide and conquer
   b) Greedy programming
   [✓ c) Dynamic programming] ✓
   d) None of the above

5. What is the time complexity of the Dynamic algorithm of 0/1 Knapsack Problem? (where n is number of items, and W is knapsack capacity)
   a) O(n+W)
   [✓ b) O(n*W)] ✓
   c) O(n log W)
   d) O(n^2)

Question 2) Apply **Prim's** algorithm to the following graph to find **MST** showing detailed steps.

(5 marks)



5

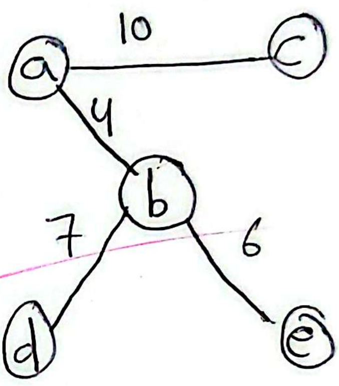
$h = \{4, 10\}$

$h = \{\cancel{6}, 7, 10, 15\}$

$h = \{\cancel{7}, 10, 15, 20, 40\}$

$h = \{\cancel{10}, 15, 20, 30, 40\}$

$h = \{\cancel{15}, \cancel{20}, \cancel{30}, \cancel{40}\}$



cost = {4, ~~6~~, 7, 10} → summation

cost = 4 + 6 + 7 + 10 = 10 + 17 $\boxed{= 27}$

Question 3) Based on your understanding of Floyd's algorithm (for weighted graphs), write a recursive relation for finding R^k_{ij} .

(2 marks)

0.75

$$Q(n) = \begin{cases} w[w_{i-1}, \cancel{v_i} + a_i] & , n \geq [illegible] \\ \cancel{0} & , n = 0 \end{cases}$$

v_i like (a, b)
a_i like (c, a)
by example

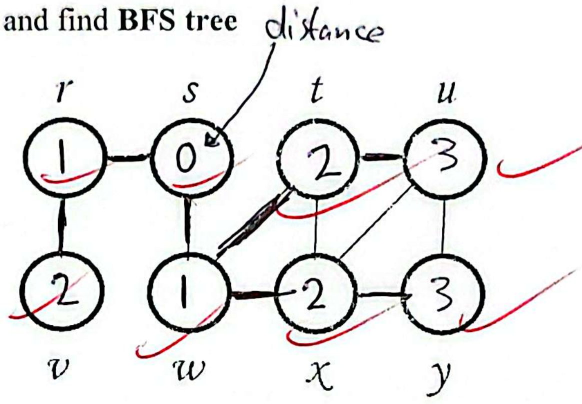
If min(previous value , the num of node number +

Question 4) Fill in the bellow table best and worst cases of the following algorithms: **(4 marks)**

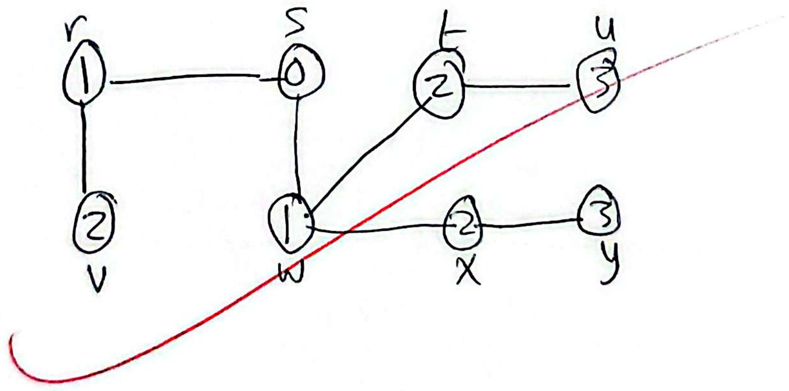
2

	Breadth-First Search	**Quick Sort**	**Selection Sort**	**Recursive Towers of Hanoi**
Best Case	$\Omega(1)$	$\Omega(n \log n)$	$\Omega(n^2)$	$\Omega(1)$
Worst Case	$O(n \log n)$	$O(n \log n)$	$O(n^2)$	$O(2^n)$

Question 5) Given the below graph, apply the BFS algorithm to find **distance** from source vertex ***s*** to all other vertices and find **BFS tree** **(5 marks)**



5



$$\begin{tikzcd}
 & s \arrow[dl, dash] \arrow[dr, dash] & \\
r \arrow[d, dash] & & w \arrow[dl, dash] \arrow[d, dash] \\
v & t \arrow[d, dash] & x \arrow[d, dash] \\
 & u & y
\end{tikzcd}$$

Question 6) Given the following set of characters and their frequencies: **(5 marks)**

4

Character	A	B	C	D	E	F
Frequency	5	9	12	13	16	45

(a) Construct the Huffman coding tree for the given characters by clearly showing details

(b) From the constructed tree, determine the binary Huffman code for each character.

a) A(5) + b(9) = 14

C(12) + d(13) = 25

Ab	E	Cd	F
14	16	25	45

Ab(14) + E(16) = 30

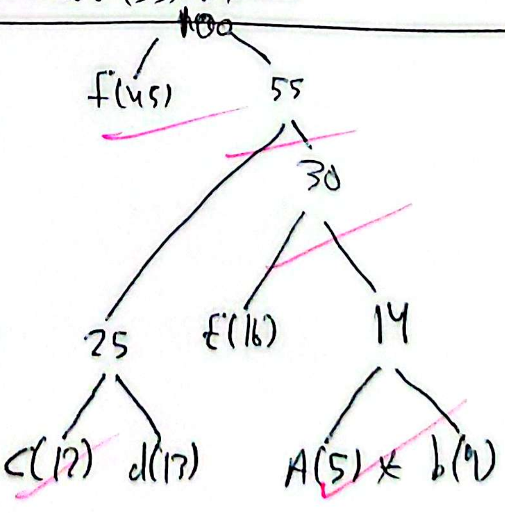
cd	AbE	F
25	30	45

cd(25) + AbE(30) = 55

F	Ab cd E
45	55

AbcdEF = 100

F(45) + AbcdE(55) = 100



b)

binary

F = 0

E = 110 ✗

D = 101

C = 100

B = 111 ✗

A = 1110 ✗

3

Question 7) Discuss **briefly (two sentences maximum)** the **difference** between the following:
(4 marks)

- **P vs NP problems:**

1

P(Polynomial) is a problem that it can easy to find the solution to it in an appropriate time.

NP(Nondeterministic-Polynomial) is a problem that is hard to find the solution to it, but if you give me the solution, I can decompute it

- **Decision vs Optimization problems:**

0.75

Decision problem the answer to it yes or no, example, does exited a link between 2 vertices

Optimization problem is to find all solutions in a problem like TSP, —— the best, worst, least - -

- **Branch & Bound vs Brute Force techniques:**

× Branch and bound has a techniques, but not all the examples in technique can be solved.

0.25 Brute Force technique has a guarantee to find the solution.

- **Traveling Salesman Problem vs Hamiltonian Cycle:**

1

Hamilton cycle is a cycle that starts at a source node, then transfers to each node one time ~~and~~ one by one, then returns back to the source node.

TSP is a problem of NP tries to find the least distance.

Question 8) **(3 marks)**

Write the recursive formulation for the 0/1 Knapsack Problem for a Dynamic Programming solution.

1.5

let n = number of item
W = Knapsack capacity
w_i = weight of item i
v_i = Value (benefit) of item(i).

$$Q(n) = \begin{cases} w[w_i] = w[w_i - 1] & , w_i > W \\ V_i + (w \cancel{-} w_i) & , w_i \le W \end{cases}$$

?

Question 9) **(7 marks)**

Consider the problem of scheduling n jobs of known durations t1, ..., tn for execution by a single processor. The jobs can be executed in any order, one job at a time. You want to find a schedule that minimizes the total time spent by all the jobs in the system. (The time spent by one job in the system is the sum of the time spent by this job in waiting plus the time spent on its execution.)

3.5

a) Design a greedy algorithm for this problem.

b) Does your greedy algorithm always yield an optimal solution?

```
a) Minimize(A,n)
     time = tn → number That user enters.
      sum = 0
         n = number of all jobs → plus = 0
      for  i = 0  to  ~~time~~ n
          sum = time + plus  →  sum = time + plus

          return  sum

          ~~return  time[i-1]~~ time (i-1)

    for (int w = 0 to n)
          make_set(n)
           for i=0  to w(u,v)
                    return  w(u,v) - w(u-1, v-1)
                        find_set(n)
```

?

b) Yes

Good Luck