



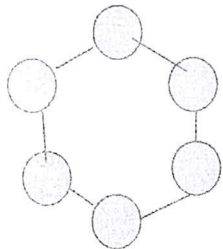
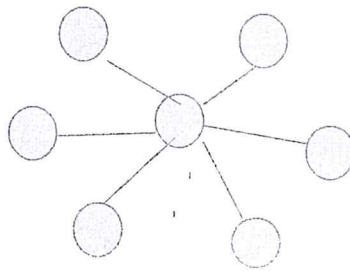
General Instructions:

- Answer the questions precisely.
- Make your logic simple and straight forward

Some formula:

- Stirling formula :  $n! \sim (2n\pi)^{0.5} (n/e)^n$

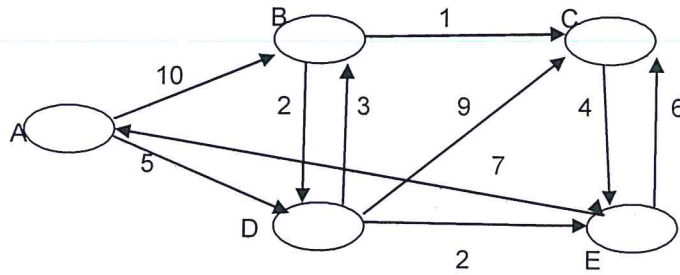
1	a	Compare the growth rate of the following functions as n tends to infinity. I) $n!$ And $(n-1)!$ II) $2^n$ and $2^{n-1}$ III) $\log(n)$ and $\text{sqrt}(n)$	6
	b	<pre> algorithm what(a[0 .. n-1, 0 .. n-1]) res &lt;- 0 for i &lt;- 0 to n - 1 do     found &lt;- false; j &lt;- 0     while not found and (j &lt; n) do         found &lt;- a[i, j] = 0         j &lt;- j + 1     if found then         res &lt;- res + 1 return res </pre> <p>i) what does the algorithm do? ii) What is the basic size? iii) Count the number of comparisons – <math>a[i,j] = 0</math> in the best case and worst case.</p>	2+1+4

	c	<pre> algo what(a[l ..r] , l, r) if l = r then     return a[l] else if a[l] &gt; a[r] then     return what(a, l + 1, r) else     return what(a, l, r - 1)  i) what does the given function do? ii) What is the basic operation? iii) What is the basic size? iv) express and solve the recurrence relation for number of operations? </pre>	$2 + 1$ $+ 1 +$ $3$
2	a	<p>A network topology specifies how computers, printers, and other devices are connected over a network. The figure below illustrates two common topologies of networks: the ring and the star.</p> <div style="display: flex; justify-content: space-around; align-items: center;">   </div> <p>Given a graph G implemented using an adjacency matrix, develop an algorithm to determine whether it represents ring or star or some other topology using brute-force technique.</p> <p>Algorithm find_topology(a[0 .. n-1, 0 .. n-1])  // TODO  // return 1 for ring; 2 for star and 3 for others</p>	6
	b	<p>ALGORITHM Merge(B[0..p - 1], C[0..q - 1], A[0..p + q - 1])  //Merges two sorted arrays into one sorted array  //Input: Arrays B[0..p - 1] and C[0..q - 1] both sorted  //Output: Sorted array A[0..p + q - 1] of the elements of B and C</p> <pre> i ← 0; j ← 0; k ← 0 while i &lt; p and j &lt; q do     if B[i] ≤ C[j]         A[k] ← B[i]; i ← i + 1     else         A[k] ← C[j]; j ← j + 1     k ← k + 1 if i = p     copy C[j..q - 1] to A[k..p + q - 1] else     copy B[i..p - 1] to A[k..p + q - 1] </pre>	6

		<p>a) What is the number of comparisons <math>B[i] \leq C[j]</math> in the best case? Which is the best case?</p> <p>b) What is the number of comparisons <math>B[i] \leq C[j]</math> in the worst case? Which is the worst case?</p> <p>c) What will happen if the comparison is changed to <math>B[i] &lt; C[j]</math>?</p>	
	c	Express the recurrence relationship for number of comparisons in quicksort in the best case. Solve the recurrence.	4
	d	With respect quicksort, a. are arrays made up of all equal elements the worst-case input, the best-case input, or neither? b. are strictly decreasing arrays the worst-case input, the best-case input, or neither. State the reason in a sentence or two.	4
3	a	<p>ALGORITHM what(<math>A[0..n - 1]</math>)  for <math>i \leftarrow 1</math> to <math>n - 1</math> do      <math>j \leftarrow i - 1</math>      while <math>j \geq 0</math> and <math>A[j] &gt; A[j + 1]</math> do          swap(<math>A[j]</math>, <math>A[j + 1]</math>)          <math>j \leftarrow j - 1</math></p> <p>a) What does this algorithm do?  b) Show the working of this algorithm on the array  <math>A = \{ 20, 10, 20, 30, 15 \}</math></p>	2 + 4
	b	Topological sort of a connected graph is unique only if out degree of each node is 1 but for the last node whose out degree is 0. State whether the above statement is true or false. Give your reason in a sentence or two.	4
	c	<p>algorithm what(<math>a[0 .. n-1]</math>, <math>s</math>)  for <math>i \leftarrow 0</math> to <math>n - 2</math> do      for <math>j \leftarrow 1</math> to <math>n - 1</math> do          if <math>a[i] + a[j] = s</math> then              return true  return false</p> <p>i) What does the algorithm do?  ii) What is the time complexity of this algorithm?  lii) Rewrite the algorithm to have complexity <math>O(n \log n)</math>.</p>	1 + 1 + 4
	d	State how to find the range (biggest key – smallest key) in a 2-3 tree. What will be the complexity of this algorithm?	2 + 2



4	a	Find the value of $C(6, 4)$ using dynamic programming with memory function – Do fill the table cells only if required.	4
	b	<p>Complete this algorithm to check whether a given graph represented by its adjacency matrix A has a cycle</p> <pre> Algorithm isDag(A[0 .. n-1, 0 .. n-1]) // result of call to Warshall is stored in A itself Warshall(A[0 .. n-1, 0 .. n-1]) // TODO  // Hint: check what would be the significant change on closure in the matrix while applying Warshall </pre> <p>c</p> <p>i) construct the shift table for the pattern TCCTATTCTT  ii) show matching of this pattern on text  TTATAGATCTCGTATTCTTTTATAGATCTCCTATTCTT  using Horspool algorithm.  lii) count the number of comparisons</p> <p>d</p> <p>i) Give an example of a graph or a digraph with negative weights for which Floyd's algorithm does not yield the correct result.  ii) ALGORITHM Floyd(W [1..n, 1..n])  //Implements Floyd's algorithm for the all-pairs shortest-paths problem  //Input: The weight matrix W of a graph with no negative-length cycle  //Output: The distance matrix of the shortest paths' lengths  <math>D \leftarrow W</math> //is not necessary if W can be overwritten  for <math>k \leftarrow 1</math> to <math>n</math> do      for <math>i \leftarrow 1</math> to <math>n</math> do          for <math>j \leftarrow 1</math> to <math>n</math> do              <math>D[i, j] \leftarrow \min\{D[i, j], D[i, k] + D[k, j]\}</math>  return D</p> <p>The longest path not having a cycle has a length <math>n - 1</math>.  Why are all these loops executed <math>n</math> times?</p> <p>What would be the max length of the path whose weight is stored in <math>D[i, j]</math> after <math>k</math>th iteration?</p>	4
5	a	<p>Apply Dijkstra's algorithm to find <b>single destination shortest paths</b> for the given graph with respect to the destination node E .</p> <p>Hint : you may want to reverse the directions of edges to change destination to source.</p>	5



- b Develop the Prim's algorithm with the constraint that the specified edge has to be included in the solution.

Algorithm my\_prim( $G(V,E)$ ,  $e(v_1, v_2)$ )

// TODO

// edge  $e$  connecting  $v_1$  and  $v_2$  to be included in the solution.

- c Comment.

i) Huffman code for 'e' is 0 and 'a' is 01

ii) Verifying Hamiltonian circuit is a polynomial time algorithms

iii) In branch and bound technique for solving the knapsack problem, an estimate of the maximum value in the beginning is made based on total weight of all items

iv) Lower bound of multiplication of two matrices can be  $\Omega(n * \log n)$

v) In backtracking, we construct solutions one component at a time and evaluate the feasibility