UCSC

University of Colombo, Sri Lanka

University of Colombo School of Computing

BACHELOR OF SCIENCE IN INFORMATION SYSTEMS

Second Year Examination - Semester II - UCSC AY19 [held in March/ April/May 2023]

IS2110 — Data Structures and Algorithms II

00 019

(Two (2) Hours)

Answer ALL questions

Number of Pages 12 **Number of MCQs = 35; Essay Questions = 1**

Important Instructions to candidates:

I. Students should answer in the medium of English language only using the space provided in this question paper.

II. Note that questions appear on both sides of the paper. If a page or a part of this question paper is not printed, please inform the supervisor immediately.

III. MCQ should be marked on the **MCQ answer sheet** and the essay question on the **paper** provided.

IV. Write your **index number** on the answer sheets.

V. This paper has two (02) parts, **Part I** and **Part II**. Part I consists of **thirty-five (35) MCQ** questions for **seventy (70) marks.** Select **the best suitable answer among the given five (5) choices. Part II** consists of an **essay question for thirty marks (30)**. Answer ALL questions.

VI. Each MCQ question has five (05) answers with **one (01) correct answer**. Each MCQ question is worth two (02) marks. If you **select more than one answer**, you will **receive zero (0) marks** for that particular question. However, there are no negative marks for selecting only one incorrect answer.

VII. Calculators, mobile phones or any electronic device capable of storing and retrieving data are **not allowed**.

PART I

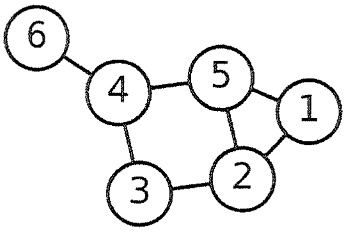
1) What is a **k-partite graph**?

(a) *A* graph whose graph vertices can be partitioned into k/2 disjoint sets so that no two vertices within the same set are adjacent.
(b) *A* graph whose graph vertices can be partitioned into two disjoint sets so that no k/2 vertices within the same set are adjacent.
(c) *A* graph whose graph vertices can be partitioned into k disjoint trees.
(d) *A* graph whose graph vertices can be partitioned into k disjoint sets so that no two vertices within the same set are adjacent.
(e) *A* graph whose graph vertices can be partitioned into k/2 disjoint trees.

2) What is a **hyperedge** in a graph?

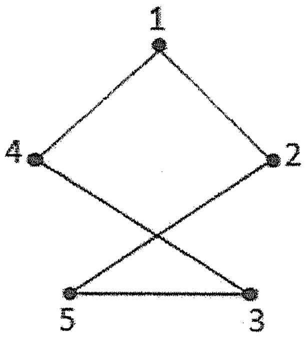
(a) A particular type of line segment joining two vertices in a polygon, polyhedron, or higher-dimensional polytope
(b) A particular type of line segment joining two vertices
(c) A particular type of line segment joining two vertices in a polygon, or higher-dimensional polytope
(d) A connection between two or more vertices of a hypergraph. A hyperedge connecting just two vertices is simply a usual graph edge
(e) A connection between two vertices of a hypergraph. A hyperedge connecting just two vertices is simply a usual graph edge

3) What kind of data structure shows by the following figure?



(a) Undirected graph
(b) Directed graph
(c) Multigraph
(d) Complete Graph
(e) Singleton graph

4) Which **adjacency matrix** shows the correct representation of the following graph.



(a)

	1	2	3	4	5
1		1		1	
2	1				1
3				1	1
4	1		1		
5		1	1		

(b)

	1	2	3	4	5
1		1		1	
2	1				1
3		1			1
4	1		1		
5		1	1		

(c)

	1	2	3	4	5
1		1		1	
2				1	1
3		1			1
4	1		1		
5		1	1		

(d)

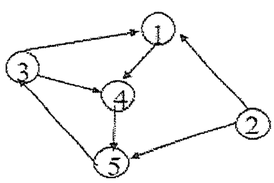
	1	2	3	4	5
1		1		1	
2	1				1
3		1			1
4	1		1		
5		1		1	

(e)

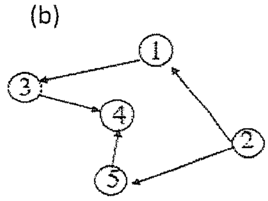
	1	2	3	4	5
1		1		1	
2	1				1
3		1			1
4	1				1
5		1	1		

5) Which of the following graph represents a **universal sink**?

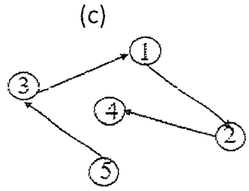
(a)



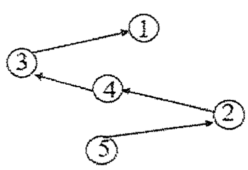
(b)



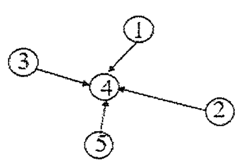
(c)



(d)



(e)



6) What can **Breadth First Search** in graphs be effectively used for?

(a) To find the shortest path between two nodes
(b) For topological sorting
(c) To find the outdegree of a vortex
(d) For solving sudoku puzzle
(e) For scheduling problems

7) What can **Depth First Search** in graphs be effectively used for?

(a) To find Minimum Spanning Tree
(b) In Peer to Peer Networks
(c) To find the indegree of a vortex
(d) To test if a graph is Bipartite
(e) To find Strongly Connected Components

8) What is a **Strongly connected component**?

(a) An edge with both ends as the same vertex
(b) A closed walk which does not repeat edges or vertices except for the starting and ending vertex
(c) A leaf vertex with degree one.
(d) The portion of a directed graph in which there is a path from each vertex to another vertex
(e) A closed walk which repeat edges or vertices except for the starting and ending vertex

9) What is a disadvantage of **Open addressing Hash Table Organization**?

(a) The use of outer hashing function
(b) Consider a static set of keys
(c) Overhead of multiple linked lists
(d) Two parameters which govern performance need to be estimated
(e) Maximum number of elements must be known

10) What is the **folding technique** in hashing function does?

(a) Divide the key into several parts and perform an arithmetic operation (such as summation) on the parts.
(b) Only a part of the key is used to compute the address.
(c) The key is transformed into another number base.

(d) The key is squared and the middle part/mid part of the result is used as the address.
(e) Multiply the key by a constant.

11) What is the ***running time*** of **Depth First Search** in a graph?

(a) O(V+E)	(d) O(log n)
(b) O(n)	(e) O(n log n)
(c) O(log n^2)	

12) *"Explore if possible, Backtrack otherwise"* is a feature of,

(a) Breadth First Search
(b) Depth first search
(c) Insertion sort
(d) Interpolation search
(e) linear search

13) What could be the possible ***missing step*** of the following source code related to **Breadth First Search**?

```
while Q ≠ Ø
      do u ← DEQUEUE(Q)
      for each v ∈ Adj[u]
            do if color[v] = WHITE
                  then color[v] ← GRAY
                     d[v] ←d[u] + 1
                      p[v] ← u

                     ___________
      color[u] ← BLACK
```

(a) `do color[u] ← WHITE`
(b) `ENQUEUE(Q,v)`
(c) `color[s] ← GRAY`
(d) `p[u] ←null`
(e) `Q ← {s}`

14) What could be the possible ***missing step*** of the following source code related to **Depth First Search**?

```
DFS-Visit(u){
  color[u] ← GRAY //vertex u has been discovered
  d[u] ← time ← time + 1
  for each v ∊ Adj[u] //explore edge (u,v)
      do if color[v] = WHITE
            then p[v] ← u
                 ________________
  color[u] → BLACK //blacken u; it is finished
  f[u] ← time ← time + 1
}
```

(a) `time ← time + 1`
(b) `u ← pop(S)`
(c) `DFS-Visit(v)`
(d) `push(v, S)`
(e) `color[v] = GRAY`

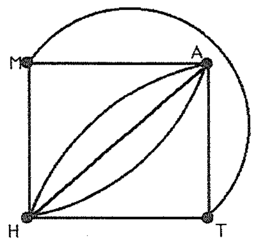
15) What is the main advantage of **Adjacency Matrix Representation** of Graphs?

(a) Has a quick way to determine the vertices adjacent to another vertex
(b) Can quickly determine if there is an edge between two vertices
(c) Memory required is `O(V + E)`
(d) Preferred when the graph is sparse
(e) Perform well for Directed graphs

16) What is the main advantage of **Adjacency List Representation** of Graphs?

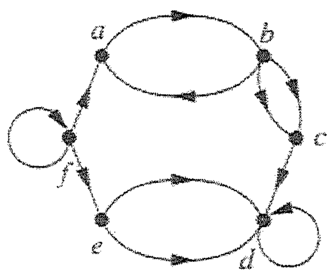
(a) Perform well for Directed graphs
(b) Preferred when the graph is dense
(c) Memory required is O(V + 2E)
(d) Can quickly determine if there is an edge between two vertices
(e) Has a quick way to determine the vertices adjacent to another vertex

17) What is the **degree** of the vertex **H**?



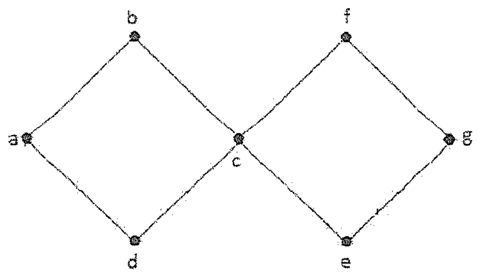
(a) 3	(d) 2+3/2
(b) 5	(e) 10
(c) 5/3	

18) What is the best explanation for the graph given below?



(a) Directed multigraph	(d) Disconnected Graph
(b) Simple graph	(e) Regular Graph
(c) Complete graph	

19) What is the best explanation for the graph given below?



(a) Regular Graph	(d) Cyclic Graph
(b) Wheel graph	(e) Circuit
(c) Complete graph	

20) Which of the following statement is ***false*** regarding tree structures?

(a) Trees contain no cycles
(b) Has V-1 number of edges
(c) Addition of any new edge to a tree creates exactly one cycle.
(d) A tree need not be a graph but a graph must be a tree.
(e) Trees represent relations of a hierarchical type.

21) Which of the following statement is ***false*** regarding **Jump Search**?

(a) Works only for sorted arrays
(b) The optimal size of a block to be jumped is ($\sqrt{n}$).
(c) The time complexity of Jump Search is between Linear Search and Binary Search.
(d) Jump search may traverse back-and-forth few times before finding the search element.

(f) The time complexity of Jump Search O($\sqrt{n}$).

22) A **compression map hash function** is defined as:

`h(k) = k mod m`

if `m = 101`, what are the expected hash values for

`{300, 305, 310, 400, 405, 410, 500, 505, 510}`

(a) {0, 5, 10, 0, 5, 10, 0, 5, 10}
(b) {99, 3, 8, 98, 2, 7, 97, 1, 6}
(c) {98, 2, 7, 97, 1, 6, 96, 0, 5}
(d) {97, 1, 6, 96, 0, 5, 95, 99, 4}
(e) {1, 6, 11, 1, 6, 11, 1, 6, 11}

23) In a **Radix Transformation** hashing function base 11 is used and m=99. What is the output hash value of $(453)_{10}$?

(a) 382
(b) 85
(c) 84
(d) 271
(e) 57

24) Based on the **Pigeonhole Principle**, find the ***minimum*** number of students in a class such that three of them are born in the same month?

(a) 36	(d) 27
(b) 37	(e) 25
(c) 12	

25) **Perfect hashing** is a hashing technique which,

(a) Provides O(1) memory access to perform a search in a worst case
(b) Create a three-level hashing scheme with universal hashing at each level
(c) Is an open-addressing technique
(d) Successively examine the hash table until an empty slot is found to insert the key.
(e) A sequence of probe depends on the key being inserted.

26) A **greedy strategy** usually progresses in,

(a) A bottom-up fashion
(b) A top-down fashion
(c) Following divide and conquer approach
(d) Following decrease and conquer approach
(e) The virtual memory of a computer system

27) Which is ***not*** an application of **Greedy algorithms**?

(a) Pathfinding
(b) Fractional knapsack problem
(c) Huffman Coding
(d) Strongly connected components
(e) Coin problem

28) As shown in the table below, there are 6 activities with corresponding start and end time, the objective is to compute an execution schedule having the maximum number of non-conflicting activities. What is the possible **activity execution schedule** using **greedy approach**?

Start Time (s)	Finish Time (f)	Activity Name
5	9	a1

1	2	a2
3	4	a3
0	6	a4
5	7	a5
8	9	a6

(a) a2, a3, a5, a6
(b) a2, a3, a4, a5
(c) a1, a2, a5, a6
(d) a1, a3, a2, a6
(e) a4, a5, a1, a6

29) **Huffman's greedy algorithm** can be effectively used for,

(a) Lossless data compression
(b) Hashing large set of numbers
(c) Appointing shorter codes for the data which appear with lower frequency
(d) Finding strongly connected components
(e) Finding cycles in graphs

30) The main difference between the **divide and conquer algorithms** and **Decrease and Conquer** algorithms is,

(a) Decrease and conquer is a greedy approach while divide and conquer is not
(b) There is only one subproblem in decrease and conquer techniques while divide and conquer has many
(c) Length of coding
(d) Decrease and conquer uses threads while divide and conquer uses processes.
(e) Decrease and conquer cannot be used for sorting while divide and conquer is mainly used for sorting.

31) Which algorithm ***cannot*** be considered as an implementation of the **decrease and conquer approach**?

(a) Insertion Sort
(b) Depth First Search
(c) Topological Sorting

(d) Binary Search
(e) Merge sort

32) What is the time complexity of **topological sort** using depth-first search?

(a) O(n)
(b) O(log n)
(c) $O(V^2)$
(d) $O(n^2)$
(e) O(V+E)

33) What is the time complexity of the **insertion sort** algorithm?

(a) O(n)
(b) O(log n)
(c) $O(V^2)$
(d) $O(n^2)$
(e) O(V+E)

34) What can be the missing step in the following **binary search** implementation?

```
while (l<r)

  ____________________

  if K = A[m] then
    return m
  else if K < A[m] then
    r = m-1
  else
    l = m+1
return -1
```

(a) `m = [(l+r)/2]`
(b) `m = [(l+r]]`
(c) `m = m+1`

(d) m = m-1

(e) m = [(l-r⌋]

35) What can be the missing step in the following **insertion sort** implementation?

```
INSERTION-SORT(A)
for j = 2 to A.length
  key = A[j]
  i = j - 1
  while i > 0 and A[i] > key
    ____________________
    i = i - 1
  A[i+1] = key
```

(a)	A[i] = A [i+1]
(b)	A[i+1] = A [i]
(c)	A[j] = A [j+1]
(d)	A[j+1] = A [j]
(e)	j=j+1

Part II

Use separate answer sheets to answer this question.

Q1

a. With a proper example discuss how you would insert keys {15, 38, 29, 25, 30, 43, 48, 53} to a hash table using **Coalesced Chaining** method (m=9). Make necessary assumptions if needed.

(9 marks)

b. Briefly describe how **Hashing** can be effectively used in the following application areas:

i. Dictionary
ii. Board Games
iii. Unix Shell

(9 marks)

c. Compare and contrast **divide and conquer algorithms** and **greedy algorithms** on problem-solving.

(6 marks)

d. Briefly describe the following three variations of **decrease and conquer** algorithms

i. Decrease by a constant
ii. Decrease by a constant factor
iii. Variable size decrease

(6 marks)

***