College of Engineering

Wellesley Road Shivajinagar Pune 411005 (MS)

Year: 2013-2014 **MIS Number:** | | | | | | | | | | |

Examination: End Semester **Semester:** V **Class:** TY Comp

Course: Design and Analysis of Algorithms **Code:** CT09002

Date: Nov 13, 2013 **Maximum Marks:** 60 **Time:** 03 Hrs

N.B.

(i) Attempt any **five** the questions. Question paper contains **six** questions.

(ii) Figures to the right indicate full marks.

(iii) Assume suitable data if necessary.

(iv) Unspecified symbols have usual meanings.

Q.1 (a) (i) Give the increasing order of asymptotic complexity of the following functions f_1, f_2, f_3, and f_4?

- $f_1(n) = 2^n$
- $f_2(n) = n^{3/2}$
- $f_3(n) = nLogn$
- $f_4(n) = n^{Logn}$

(03)

(ii) Show that for any real constants a and b where $b > 0$

$$(n + a)^b = \Theta(n^b)$$

(03)

(b) (i) Write INSERTION-SORT algorithm.

(ii) Demonstrate its working with suitable example.

(iii) Which one of the following in place sorting algorithms needs the minimum number of swaps?

QUICK-SORT, INSERTION-SORT, SELECTION-SORT, HEAP-SORT **(06)**

Q.2 (a) (i) Write pseudocode for LEFT-ROTATE in red-black tree.

(ii) Where in a max-heap might the smallest element reside assuming that all elements are distinct?

(iii) Is the array with values $< 23, 17, 14, 6, 13, 10, 1, 5, 7, 12 >$ a max-heap? **(06)**

(b) (i) Write Breadth-First-Search algorithm (BFS).

(ii) What is the running time of BFS if its input graph is represented by an adjacency matrix and the algorithm is modified to handle this form of input? **(06)**

Q.3 (a) (i) Write Floyd-Warshall algorithm and illustrate its working with suitable example.

(ii) How can we use the output of the Floyd-Warshall algorithm to detect the presence of a negative-weight cycle? **(06)**

(b) Consider a complete undirected graph with vertex set $0, 1, 2, 3, 4$. Entry w_{ij} in the matrix W below is the weight of the edge i, j.

$$W = \begin{pmatrix} 0 & 1 & 8 & 1 & 4 \\ 1 & 0 & 12 & 4 & 9 \\ 8 & 12 & 0 & 7 & 3 \\ 1 & 4 & 7 & 0 & 2 \\ 4 & 9 & 3 & 2 & 0 \end{pmatrix}$$

What is the minimum possible weight of a path P from vertex 1 to vertex 2 in this graph such that P contains at most 3 edges? Give path and total weight. Justify your answer. **(06)**

Q. 4 (a) (i) The **incidence matrix** of a directed graph $G = (V, E)$ is $|V| \times |E|$ matrix $B = (b_{ij})$ such that

$$b_{ij} = \begin{cases} -1 & \text{if edge } j \text{ leaves vertex } i \\ 1 & \text{if edge } j \text{ enters vertex } i \\ 0 & \text{otherwise.} \end{cases}$$

Describe what the entries of the matrix product BB^T represent, where B^T is the transpose of B.

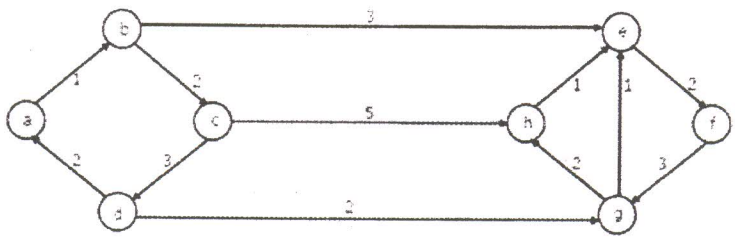
(ii) Given an adjacency-list representation of a directed graph, how long does it take to compute the out-degree of every vertex?

(iii) How long does it take to compute the in-degrees? **(06)**

(b) (i) Compare Dijkstra's Algorithm with other shortest path algorithms.

(ii) Dijkstras single source shortest path algorithm when run from vertex a in the below graph, computes the correct shortest path distance to:

- Option A: only vertex a
- Option B: only vertices a, e, f, g, h
- Option C: only vertices a, b, c, d
- Option D: all the vertices



(06)

Q. 5 (a) List the string matching algorithms. Write KNUTH-MORRIS-PRATT Algorithm and illustrate its working with suitable example. **(06)**

(b) Professor decides to write a procedure that produces at random any permutation besides the identity permutation. S/he proposes the following procedure:

```
PERMUTE-WITHOUT-IDENTITY (A)
1  n = A.length
2  for i = 1 to n − 1
3        swap A[i] with A[RANDOM(i+1, n)]
```

Does this code do what Professor intends? **(06)**

Q. 6 (a) Let X be a problem that belongs to the class NP. Then which one of the following is TRUE? State your answer in each option.

- Option A: There is no polynomial time algorithm for X.
- Option B: If X can be solved deterministically in polynomial time, then P = NP.
- Option C: If X is NP-hard, then it is NP-complete.
- Option D: X may be undecidable.

(06)

(b) Show that an otherwise polynomial-time algorithm that makes at most a constant number of calls to polynomial-time subroutines runs in polynomial time, but that a polynomial number of calls to polynomial-time subroutines may result in an exponential-time algorithm. **(06)**