NATIONAL INSTITUTE OF TECHNOLOGY ROURKELA

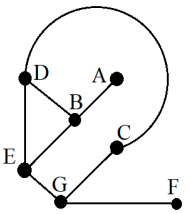
■ CS2005 Data Structures and Algorithms ■ End Sem Autumn 2019 ■ UG ■ Pg 2 ■ FM 50 ■ 3 Hrs ■

■ Answer ALL questions. ■ All parts of a question MUST be answered together. ■ Mere answers without proper explanation will not fetch marks. ■ Variation in quality of answer will vary the secured marks.

1. (a) Given a 2-tree with n total nodes, what is the maximum and minimum number of leaf nodes? Give your answers as a function of n. [2]
(b) What is the maximum number of edge changes (insertions and deletions) for a given graph $G(V, E)$ might result in producing a minimum spanning tree for G? Give answer as a function of V and E. [2]
(c) How long does it take to determine if an undirected graph contains a vertex that is connected to no other vertices? Consider both cases that the graph is stored as adjacency matrix and adjacency list. Justify. [2]
(d) Consider a binary max-heap with n nodes. How long does it take to find the third smallest key in the heap? You need not delete the key but just report its value. Explain briefly. [2]
(e) Show every possible binary min-heap that can be generated with elements 3, 9, 11, and 15 arriving in no particular order. [2]

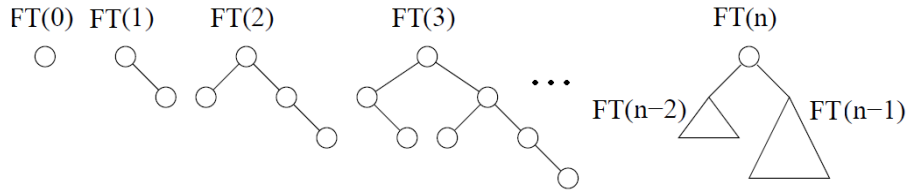
2. (a) The `dequeue()` function of a queue `Q` is tampered such that it returns and deletes the second element. However, when only one element is present in `Q`, that element is returned and deleted. Refer to the pseudocode given beside. Perform the breadth first search (BFS) on the graph shown beside starting from vertex `A` using the modified queue. Show the content of queue in every interim step clearly. [6]

```
char dequeue(Q, FRONT, REAR)
{If (FRONT == NULL) then
   Print Underflow; Exit;
Else if (FRONT == REAR) then
   PTR=FRONT; t= PTR -> DATA;
   FRONT = NULL; REAR = NULL;
   FREE PTR; Return t;
Else if (FRONT ≠ NULL & FRONT → LINK ≠ NULL) then
   PTR=FRONT → LINK; t= PTR -> DATA;
   FRONT → LINK = FRONT → LINK → LINK;
   If PTR = REAR then REAR = FRONT;
   FREE PTR; Return t;}
```



(b) Consider a modified stack (with linked implementation) with capacity of 3 elements where push operation is always performed at `top` location, whereas pop operation can be performed at any location. On arrival of a new element, if it is already existing in stack, then it is popped and pushed onto the `top` location. If the new element is not previously existing in stack, it is simply pushed at the `top` location. If there is no place for pushing the newly arrived element, rather than overflow, the stack pops out the bottom element present in the stack, and pushes the newly arrived element in `top` location. Show the content of the stack beginning with an empty state and pushing the following elements: 4, 3, 2, 4, 5, 5, 3, 2, 3. What is the asymptotic worst case complexities of the push and pop operations for this modified stack with generalized capacity n. [3+1]

3. (a) A lower triangular matrix of order $n \times n$ is to be stored using a one-dimensional array. Only the lower triangular elements needs to be stored in 1-D array for efficient memory usage. An indexing function is used to map the value at $[i, j]$ to index p in this array. Derive functions that map the index pair $[i, j]$ to index p in the array if the lower triangular elements of the matrix are stored in (i) row-major order and (ii) column major order. Your functions should run in O(1) time. [4]
(b) A modified Fibonacci binary tree (FT) is defined recursively as follows: FT of order zero (i.e. FT(0)) is just a single node. FT of order one (i.e. FT(1)) consists of a root node and a single right child node. In general, for FT of order n (≥ 2) denoted as FT(n), it consists of a root node, a left subtree FT($n-2$), and a right subtree FT($n-1$) as shown in figure below.



$N(\text{FT}(n))$ denotes the number of nodes in a FT of order n. $N(\text{FT}(n)) = N(\text{FT}(n-2)) + N(\text{FT}(n-1)) + 1$. Prove by method of induction that for all $n \geq 4$, $N(\text{FT}(n)) \leq (1.9)^n$. Use N(FT(4)) and N(FT(5)) as basis case. It may be helpful to use the fact that $1/(1.9)^k \leq 0.077$ for all $n \geq 4$. [4]

(c) Considering a connected undirected graph G with positive edge weights where no two edges have the same weight, answer the following: [2]

`Statement #1`: The edge of minimum weight must be in MST of G. True pr false? Justify.

`Statement #2`: The edge of maximum weight must not be in MST of G. True or false? Justify.

4. (a) Construct an AVL tree with the following elements arriving in given order: `b, e, f, h, i, g`. [4]

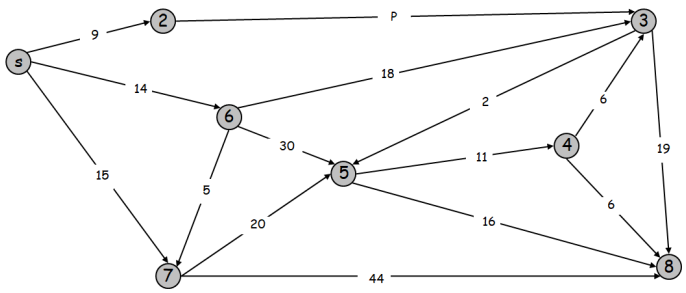
(b) Use stack to demonstrate through an algorithm whether the expression $[m - (f + g) * d] \uparrow \{c/d\} * \{[(r + t) * (b + e)] + (s \uparrow j)\}$ has balanced parentheses or not. Demonstrate clearly the content of the stack in each step and mention the decisive condition and terminating condition. [3]

(c) Fill the table beside with expected time for each operation using O notation. The operations are `Insert` (to place a new item in the data structure), `isMember` (to test if a given item is present in the data structure), `delMin` (to return the value of minimum item and to delete it from the data structure. [3]

Data Structure ↓	`Insert`	`isMember`	`delMin`
sorted array			
unsorted array			
sorted linked list			
unsorted linked list			

5. (a) Given an array `A={h, e, b, f, l}`, sort its elements in increasing order using Bubble sort. [2]

(b) Given the directed graph beside, apply Dijkstra's algorithm and decide the maximum value of P such that the edge $< 2, 3 >$ becomes a member of the shortest path from `s` to `4`. Note that the shortest path from any node `X` to any node `Y` including the edge $< \texttt{I}, \texttt{J} >$ is composed of shortest path from `X` to `I`, shortest path from `I` to `J`, and shortest path from `J` to `Y`. In order to consider the path from `X` to `Y` through $< \texttt{I}, \texttt{J} >$ to be the shortest, it should bear lower cost than the best path excluding $< \texttt{I}, \texttt{J} >$. [8]



–END–