Chennai Mathematical Institute
Advanced Programming
Final Examination, Spring Semester, 2022–2023

Date : 28 Apr, 2023 (2:00pm-5:00pm)
Duration : Three hours

Marks : 50
Weightage : 30%

1. (a) Give an example of a directed graph on 4 vertices and 5 edges that does not have a topological sort. *(1 marks)*

   (b) Give an example of a directed graph on 4 vertices and 5 edges that has more than one topological sorts. *(1 marks)*

   (c) Give an example of a directed graph on 4 vertices and 5 edges that has a unique topological sort. *(1 marks)*

   (d) Prove that if a directed graph has a unique topological sort then for every pair of nodes u, v in the graph either there is a directed path from u to v or from v to u but not both. $(u \neq v)$ *(2 marks)*

   (e) Prove the converse of the previous part. *(2 marks)*

2. Assume we have a Python class `List` in which each node has the basic structure indicated by the definition to the right and every list is terminated by a dummy node with `value` and `next` set to `None`.

```
class List:
  def __init__(self):
    self.value = None
    self.next = None
    return
```

   Write a Python function to reverse such a list. The function should take as its argument the head of the original list and return a pointer to the head of the reversed list. *(7 marks)*

3. Consider a commutative semigroup $(S, +)$ (i.e. the operation $+$ is associative and commutative). You have access to a procedure `hasSubsetSum(X,t)` that, takes as arguments a set $X \subseteq S$ and a target element $t \in S$. It answers `True` if there exists a $Y \subseteq X$ such that the sum of all elements in Y (under the operation $+$) is precisely t.

   Describe, preferably as pseudocode, an algorithm `findWitnessSubset(Z,t)` that takes two arguments: a set $Z \subseteq S$ and an element $t \in S$ and returns a witness set for t if it exists and `None` otherwise. In other words it returns a subset $Y \subseteq Z$ such that the elements of Y sum to t. Your algorithm should run in time at most $O(n)$ apart from making $O(n)$ calls to `hasSubsetSum` where $n = |Z|$. *(7 marks)*

4. Let L_1, L_2 be two sorted list of distinct integers, containing respectively ℓ_1, ℓ_2 elements. We say that $L_1 \leq L_2$ in lexicographic order if there exists an integer $i \leq \ell_1$ such that $L_1[j] = L_2[j]$ for each $j \leq i$ and the following occurs: Either, ($i < \ell_1$ and $L_1[i+1] \lneq L_2[i+1]$), or $i = \ell_1$. (indexing starts at 1, i.e., L contains $L(1), \cdots, L(\ell)$).

   Let `BST` be a binary search tree class with fields `value`, `left` and `right`. Write python code for the following:

   - Given a binary search tree T produce the list of elements L contained in it in $O(\ell)$ time where n is the number of elements contained in L *(2 marks)*
   - Given two binary search trees T_1, T_2 output if for the corresponding sorted lists of elements L_1, L_2 whether $L_1 \leq L_2$. The running time of your algorithm should be ~~$O(h_1 + h_2)$~~ where ~~h_1, h_2 are the heights of the two trees~~. *(6 marks)*

     $O(\ell_1 + \ell_2)$ $\ell_1 = |L_1|, \ell_2 = |L_2|$

     (Assume $l_1 < l_2$
     - elts of L have to be produced in sorted order where $l = |L|$
     - $O(\ell_1 + \ell_2)$; $\ell_1 = |L_1|, \ell_2 = |L_2|$)

5. This question asks you to implement Algorithm-P using the Fibonacci Heaps taught in class. (Assume graph is connected)

   Algorithm-P takes as input an undirected weighted graph and outputs a tree thus: it initialises the tree to a single vertex with no edges. Then in $n - 1$ steps it add one edge per step to the tree. At every step, the edge added is the lightest amongst all edges with one endpoint on the tree and the other not on the tree.

   You need write to pseudocode to implement Algorithm-P in $O(m + n \log n)$ time where m, n are respectively the number of edges and vertices in the graph. You may assume the complexity values for various operations supported by Fibonacci heaps but you must name and state the correct value for each operation you use. There is no need to give pseudo-code for or argue the complexity of these operations. *(7 marks)*

6. Let D be a directed acyclic graph with integer weights (not necessarily positive) on edges. The weight of a path in D is the sum of weights of the edges on the path. Write pseudocode to find the weight of a longest path between two specific vertices s, t in D as well as report such a longest path itself. Argue the correctness of your algorithm. State and prove the complexity of your algorithm in terms of m, n. How will you achieve the same complexity if s, t are not part of the input i.e. you are allowed to pick any starting and ending point for the path. (longest path = path with max. weight ; m = # edges, n = # vertices in D) *(7 marks)*

7. Assume that we have defined a class `Node` to build a linked list, and a function `insert(l,n)` that inserts a new value `n` at the head of the list `l`, as given below. The function `new(Node())` allocates a fresh object of type `Node` and returns a reference (pointer) to the location of this object.

```
class Node {
  int value;
  Node next;
  ...
}
```

```
boolean insert(Node head, int newvalue){
  Node newnode;
  newnode = new(Node());
  newnode.next = head.next;
  newnode.value = head.value;
  head.next = newnode;
  head.value = newvalue;
  return(True);
}
```

   Let S and H be the memory stack and heap, respectively, before a call to `insert(..)`. Using diagrams, explain how the stack and heap are transformed with respect to the original S and H when `insert(..)` is running, and after `insert(..)` returns. *(7 marks)*