Website for CS5800

Directed edges are specified using parentheses and undirected edges are specified using braces. 1. (15 points) Run the DFS algorithm on the following undirected graph, showing each step. When there is a choice for the next vertex, use the one that is smaller numerically.

Edge	Edge	Edge	Edge
{1,4}	{1,5}	{1,6}	{2,3}
{2,4}	{2,9}	{3,10}	{5,7}
{5,9}	{6,4}	{6,8}	{6,10}
{7,2}	{7,5}	{8,7}	{8,9}
{9,5}	{9,6}	{10,6}	{10,8}

2. (20 points) Perform depth-first search on the following graph. Whenever there is a choice of vertexes, pick the one that this numerically smaller. Classify each edge as a tree edge, forward edge, back edge, or cross edge, and give the pre and post number of each vertex. Determine whether the graph is acyclic. If the graph is acyclic, topologically sort it.

Edge	Edge	Edge	Edge
(1,3)	(1,4)	(1,10)	(2,8)
(3,4)	(5,2)	(5,10)	(6,10)
(7,3)	(7,4)	(7,10)	(8,3)
(9,2)	(9,3)	(9,4)	(9,5)
(9,8)	(10,3)	(10,4)	(10,8)

3. (20 points) The merge operation in the mergesort algorithm is itself an algorithm. The textbook specifies the merge algorithm recursively. Here is an iterative specification of the merge algorithm based on the primitive operation moveFirst(x, y) that removes the first element of x and appends this element to y (as the last element).

```
input: two arrays, a and b
output: an array containing the union of the elements of the arrays a and b

merge(array a, array b)
  set c to the empty array
  loop forever
    if a is empty then
      if b is empty then
        return c
      else
        moveFirst(b, c)
    else if b is empty then
      moveFirst(a, c)
    else if the first element of a is less than the first element of b then
      moveFirst(a, c)
    else
      moveFirst(b, c)
    end if
```

```
  end loop
```

Prove the following properties of the merge algorithm:

a. (10 points) The merge algorithm terminates.
b. (15 points) The output array contains the union (without duplicate removal) of the elements of the input arrays.
c. (20 points) If the input arrays are in order (i.e., have been sorted) then the output array is also in order.

4. (20 points) Develop an algorithm for finding cycles. This algorithm should have two parameters: a directed graph and a node. It must find a cycle containing the specified node or determine that there is no cycle containing the node. Your algorithm must run in linear time, and you must prove that it runs in linear time. Perform your algorithm on the following graph and the node labeled "4":

Edge	**Edge**	**Edge**	**Edge**
(1,6)	(1,8)	(2,3)	(2,5)
(2,7)	(3,7)	(4,5)	(5,2)
(5,10)	(6,2)	(6,4)	(7,5)
(7,6)	(8,6)	(8,10)	(9,2)
(9,3)	(9,8)	(10,3)	(10,8)

Return to the Course Website