CMPS 101
Summer 2010
Homework Assignment 6 Solutions

1. (1 Point) p.548: 22.3-9
Modify the pseudocode for depth-first search so that it prints out every edge in the directed graph together with its type. (Hint: use the result stated in the last paragraph of page 546.)
Solution:

DFS(G)
1. for each $u \in V[G]$
2. color[u] $\leftarrow$ white
3. p[u] $\leftarrow$ nil
4. time $\leftarrow 0$
5. for each $u \in V[G]$
6. if color[u] = white
7. ModifiedVisit(u)

ModifiedVisit(u)
1. color[u] $\leftarrow$ gray
2. d[u] $\leftarrow$ time $\leftarrow$ (time+1)
3. for all $v \in Adj[u]$ (Explore edge (u,v))
4. if color[v] = white
5. print (u, v) ' is a Tree edge'
6. p[v] $\leftarrow$ u
7. ModifiedVisit(v)
8. else if color[v] = gray
9. print (u,v) ' is a Back edge'
10. else if d[u] > d[v]
11. print (u,v) ' is a Cross edge'
12. else
13. print (u,v) ' is a Forward edge'
14. color[u] $\leftarrow$ black
15. f[u] $\leftarrow$ time $\leftarrow$ (time+1)

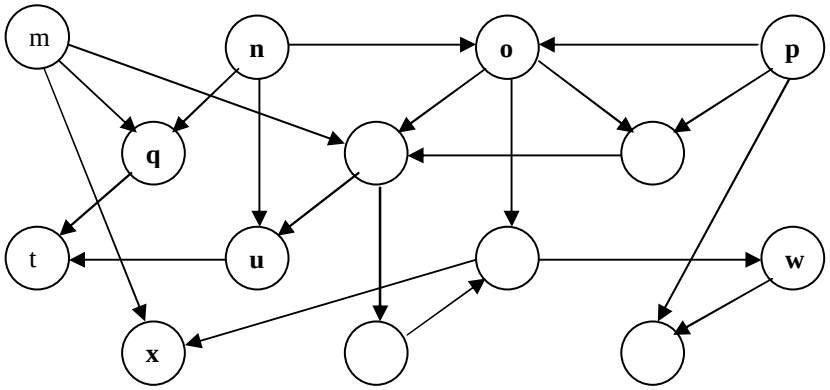
2. (1 Point) p.549: 22.3-11
Show that a depth-first search of an undirected graph G can be used to identify the connected components of G, and that the depth-first forest contains as many trees as G has connected components. More precisely, show how to modify depth-first search so that each vertex v is assigned an integer label $cc[v]$ between 1 and k, where k is the number of connected components of G, such that $cc[u] = cc[v]$ if and only if u and v are in the same connected component.
Solution:
Let $cc[u]$ be another attribute associated with each vertex u in the graph, and let k be a new variable which is local to DFS, but static over all recursive calls to Visit (similar to time.) We alter DFS() and Visit() as follows.

DFS(G)
1) for each $u \in V(G)$
2) color[u] $\leftarrow$ white
3) $\pi[u] \leftarrow$ nil
4) time $\leftarrow 0$
5) $k \leftarrow 0$
6) for each $u \in V(G)$
7) if color[u] = white
8) $k \leftarrow k+1$
9) Visit(u)

Visit(u)
1) color[u] $\leftarrow$ gray
2) $d[u] \leftarrow$ time $\leftarrow$ time $+1$
3) $cc[u] \leftarrow k$
4) for each $v \in$ adj[u]
5) if color[v] = white
6) $\pi[v] \leftarrow u$
7) Visit(v)
8) color[u] $\leftarrow$ black
9) $f[u] \leftarrow$ time $\leftarrow$ time $+1$

Observe that k is incremented only when a new root in a DFS tree is established, and that that root and all its descendents will have the same cc value. Thus when DFS() is complete, k will equal the number of trees in the DFS forest, and all vertices in each tree will have the same cc value. Theorem 22.10 on page 547 says that after running DFS() on an undirected graph G, there are no edges in G joining distinct DFS trees (i.e. no cross edges). Hence if x and y are vertices lying in distinct DFS trees, then G contains no x-y path. On the other hand, if x and y lie in the same DFS tree, then G does contain an x-y path, namely one consisting of tree edges. Thus each DFS tree spans a connected component of G, and in particular, the number of trees in a DFS forest is the same as the number of connected components in G. We see that after this modified version of DFS() is run, k will equal the number of connected components in G, and for any two vertices u, v we will have $cc[u] = cc[v]$ if and only if u and v lie in the same connected component. ///

3. (1 Point) p.551: 22.4-1
Show the ordering of vertices produced by TOPOLOGICAL-SORT when it is run on the dag of Figure 22.8, under the assumption of Exercise 22.3-2.
Solution:



Vertex	Adjacency List	Discover Time	Finish Time
m	q r x	1	20
n	o q u	21	26
o	r s v	22	25
p	o s z	27	28
q	t	2	5
r	u y	6	19
s	r	23	24
t		3	4
u	t	7	8
v	w x	10	17
w	z	11	14
x		15	16
y	v	9	18
z		12	13

A topological sort of the vertices is obtained by ordering them by decreasing finish time:
p n o s m r y v x w z u q t ///

4. (1 Point) p.557: 22.5-2

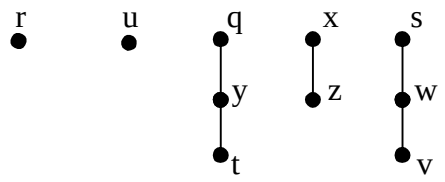
Show how the procedure STRONGLY-CONNECTED-COMPONENTS works on the graph of Figure 22.6. Specifically, show the finishing times computed in line 1 and the forest produced in line 3. Assume that the loop of lines 5-7 of DFS considers vertices in alphabetical order and that the adjacency lists are in alphabetical order.

Solution:

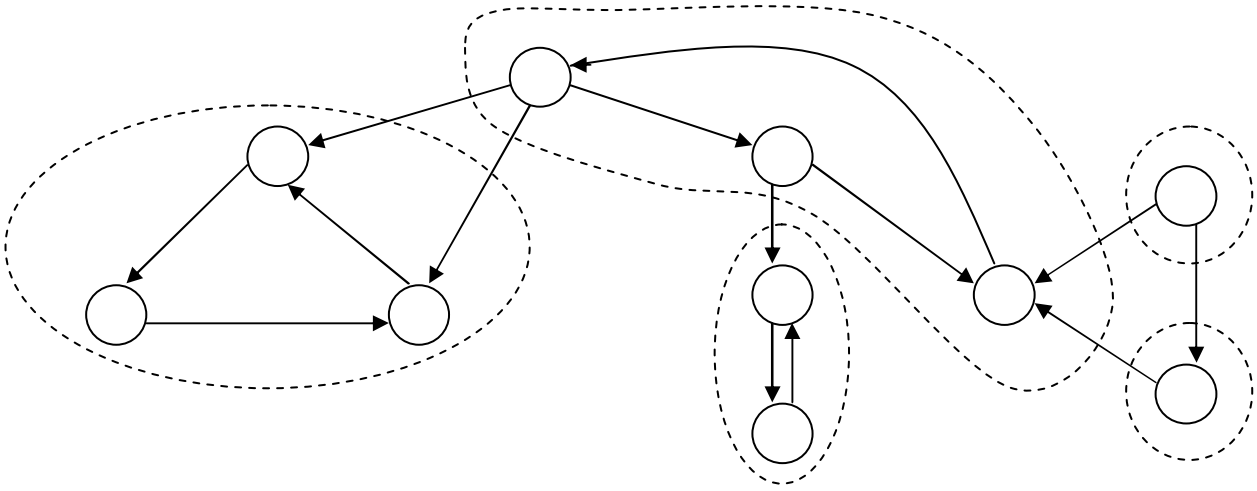
The first call to DFS was traced in problem 22.3-2, which is problem 2 of hw6, and the finish times are given in the solutions to hw6. We order the vertices by decreasing finish times from that call, then run DFS a second time on the transposed graph below.

Vertex	Adj	Discover	Finish
r		1	2
u	r	3	4
q	y	5	10
t	q	7	8
y	r t u	6	9
x	t z	11	14
z	x	12	13
s	q w	15	20
v	s	17	18
w	q v	16	19

DFS Forest



The strong components of the graph in Figure 22.6 are therefore: {r}, {u}, {q, y, t}, {x, z}, {s, w, v}.



5. (1 Point) p.1091: B.5-4
Use induction to show that a nonempty binary tree with n nodes and height h satisfies $h \geq \lfloor \lg n \rfloor$. Hint: use the recursive definition of height discussed in class:

$$h(T) = \begin{cases} -\infty & n(T) = 0 \\ 0 & n(T) = 1 \\ 1 + \max(\, h(L), h(R)\,) & n(T) > 1 \end{cases}$$

Here $n(T)$ denotes the number of nodes in a binary tree T, $h(T)$ denotes its height, L denotes its left subtree, and R its right subtree. Note that this proof can be phrased equally well as an induction on $n(T)$ or on $h(T)$. Additional hint: use (and prove) the following fact: $\lfloor \lg(2k+1) \rfloor = \lfloor \lg(2k) \rfloor$ for any positive integer k.

Proof:
We show that $h(T) \geq \lfloor \lg n(T) \rfloor$ whenever $n(T) \geq 1$, by induction on $n(T)$. If $n(T) = 1$, then $h(T) = 0$. In this case the inequality $h(T) \geq \lfloor \lg n(T) \rfloor$ reduces to $0 \geq 0$, which is true and the base case is satisfied. (Note that the result also holds when $n(T) = 0$ if we interpret $\lfloor \lg(0) \rfloor$ to be $-\infty$.) Let T be a binary tree with $n(T) > 1$, and assume for any binary tree T' with $n(T') < n(T)$ that $h(T') \geq \lfloor \lg n(T') \rfloor$. Since both L and R have fewer nodes than T, the induction hypothesis gives $h(L) \geq \lfloor \lg n(L) \rfloor$ and $h(R) \geq \lfloor \lg n(R) \rfloor$. Therefore

$$h(T) \;=\; 1 + \max(\, h(L), h(R)\,) \;\geq\; 1 + \max(\, \lfloor \lg n(L) \rfloor, \lfloor \lg n(R) \rfloor\,).$$

Now suppose that $n(L) \geq n(R)$. (The proof is entirely similar in the case $n(L) < n(R)$ and we omit it.) It follows that $\max(\, \lfloor \lg n(L) \rfloor, \lfloor \lg n(R) \rfloor\,) \;=\; \lfloor \lg n(L) \rfloor$, and hence

$$\begin{aligned} h(T) \;&\geq\; 1 + \lfloor \lg n(L) \rfloor && \text{by the above assumption} \\ &= \lfloor 1 + \lg n(L) \rfloor && \text{since } 1 + \lfloor x \rfloor = \lfloor 1 + x \rfloor \text{ for any } x \in \mathbf{R} \\ &= \lfloor \lg 2 + \lg n(L) \rfloor \\ &= \lfloor \lg(\, 2 \cdot n(L)\,) \rfloor \\ &= \lfloor \lg(\, 2 \cdot n(L) + 1\,) \rfloor && \text{by the Lemma * below} \\ &\geq \lfloor \lg n(T) \rfloor && \text{since } n(T) \;=\; n(L) + n(R) + 1 \leq 2n(L) + 1 \end{aligned}$$

Therefore $h(T) \geq \lfloor \lg n(T) \rfloor$ whenever $n(T) \geq 1$, as claimed. ///

Note: The above proof could also be phrased as induction on the height $h(T)$. Carry this out as an exercise. Hint: The second sentence of the induction step would read "Since both L and R have lower height than T, …".

Lemma *: $\lfloor \lg(2k+1) \rfloor = \lfloor \lg(2k) \rfloor$ for any integer $k \geq 1$.

Proof: $N = \lfloor \lg(2k+1) \rfloor \;\Leftrightarrow\; N \leq \lg(2k+1) < N+1 \;\Leftrightarrow\; 2^N \leq 2k+1 < 2^{N+1}$. Observe that $2k+1$ is odd while 2^N and 2^{N+1} are even, so the last inequality is equivalent to $2^N \leq 2k < 2^{N+1}$, which is equivalent to $N \leq \lg(2k) < N+1$. Therefore $N = \lfloor \lg(2k) \rfloor$, as required. ///