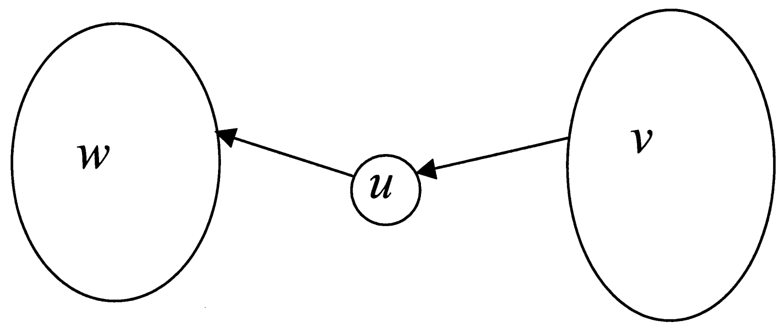
Solution to Ass. 3.

Problem 22.3-10

Explain how a vertex u of a directed graph can end up in a depth-first tree containing only u, even though u has both incoming and outgoing edges in G.

A vertex u can end up in a depth-first tree containing only u if all of its outgoing edges have been visited and then u is chosen to be visited.

Ex:



If all the nodes in w are visited first, they will form their own tree(s). If u is selected next then it will be in it's own tree by itself since it has no edges to visit.

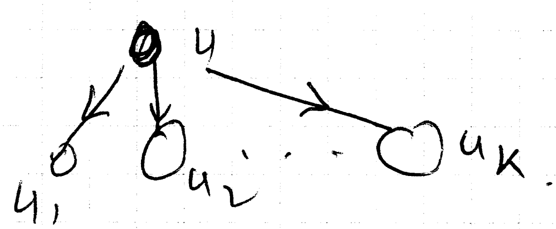
Saying that u could have a self-edge was not an acceptable answer.

Assignment 3

22.4-2

We are only interested in finding the paths from s to t. We delete all the incoming edges to ~~s~~ and all outgoing edges from ~~t~~.

We now apply DFS from s. Consider a vertex u. In DFS, u was initially white, then gray & then black. When u is gray, we are exploring the vertices reachable from u. Consider the children $u_1, u_2, \cdots u_k$ of u which are reachable (directly) from u.



When all $u_1, u_2, \cdots, u_k$ are colored black, the color of u is also turned black.

Suppose we know the # of paths from u_i to t (denoted by $\#(u_i,t)$) for all i, $i=1,2,\cdots,k$. Then the # of paths from u to t is $\#(u,t) = \sum_{i=1}^{k} \#(u_i,t)$.

(This algorithm is very similar to finding the longest path in a diagraph).

22.5-6

Observes that E' is not a subset of E in $G=(V,E)$.

We first compute the SCC (strongly connected components) of $G=(V,E)$. Let $C_1, C_2, \cdots, C_k$ be the components. We now form $G'=(V,E')$ as follows:

a) for each $i=1,2,\cdots,k-1$,

- let x_i be a node of C_i
  x_{i+1} be a node of C_{i+1}
- $E' \leftarrow E' \cup \{(x_i, x_{i+1})\}$.

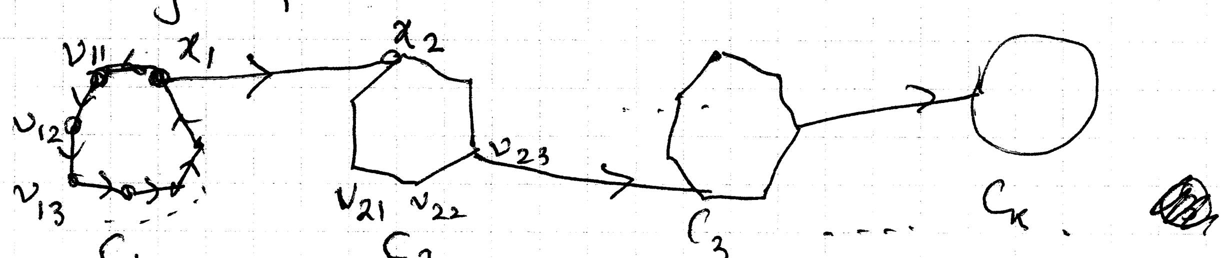
b) for each $i=1,2,\cdots,k$ do the following

/* let $v_{i_1}, v_{i_2}, \cdots v_{i_t}$ be the vertices/nodes of C_i */

for $j=1,2,\cdots,t$ do $E' \leftarrow E' \cup \{(v_{ij}, v_{ij+1})\}$

# of edges $|E'|=|V|+k-1$

The graph G' looks like



2(b) The diameter of a directed or undirected graph is the length of its longest simple path. The problem is very difficult in general. However, for a directed acyclic graph (DAG) this problem can be nicely solved by a depth-first search. Write a function that, give a DAG, computes its diameter.

To compute the diameter, we alter the Depth-first search algorithm found on page 541 of the textbook to keep track of the largest diameter found so far at each node.

```
DFS(G)
    for each vertex u ∈ V[G]
      do colour[u] <- WHITE
      diameter[u] <- -1
    for each vertex u ∈ V[G]
       do if colour[u] = WHITE
         then DFS-VISIT(u)
    maxDiameter <- -1

DFS-VISIT(u)
    colour[u] <- GRAY
    diameter[u] <- 0
    for each v ∈ Adj[u]
       do if colour[v] = WHITE
          DFS-VISIT(v)
       diameter[u] <- max{ diameter[u], diameter[v] + 1 }
    colour[u] <- BLACK
    maxDiameter = max {diameter[u], maxDiameter }
```

In this alteration, when we visit a node, we set its diameter to 0. We then check all of the outgoing edges of this node, if the node is WHITE then we visit it, and all its edges to retrieve its maximum diameter, if it is BLACK then its diameter has all ready been calculated and we check what the highest diameter it has is. We record at the current node the highest diameter found plus 1 for the edge that connects to the highest diameter.

Problem 2(c).

~~Use DFS, the ...~~

Use DFS starting from the root.
When a vertex is turned black,
the operation with two operands
(given by the children) are performed. We
distinguish the child nodes as left & right child nodes.

For the figure the following operations
are performed:

① $3*4$; ② $2+3*4$; ③ $5/(3*4)$

④ $2+3*4+5/(3*4)$