Math 416

Worksheet 14. Applications of DFS

The DFS algorithm is reproduced here for your convenience.

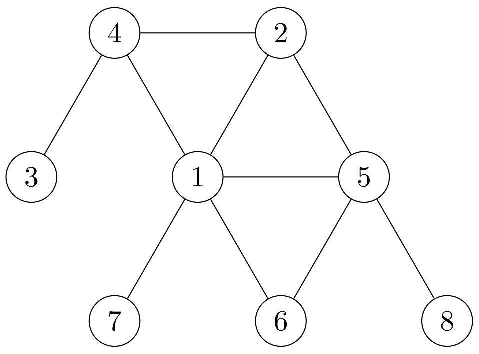
Algorithm 1: the explore subroutine

```
1 explore($v$):
2     visited($v$) = true;
      // (previsit work)
3     foreach neighbor $u$ of $v$ do
4         if not visited($u$) then
5             add $(u, v)$ to $T$ ;
6             explore ($u$);
      // (postvisit work)
```

Algorithm 2: Depth-first search

```
1 DFS($G, s$):
      Input: a graph $G = (V, E)$, a start vertex $s$
      Output: a tree spanning the connected component of $s$
      // initialize
2     foreach $v \in V$ do
3         set visited($v$) = false.
4     explore ($s$);
5     return $T$
```

Problem 1. Run DFS and BFS on this graph, starting at $s = 1$ and using the numerical ordering of the vertices. Indicate in which order the vertices are discovered, and draw the BFS and DFS trees.



(1)

The previsit and postvisit orderings.

Definition. Suppose that G is a graph with a given ordering of its n vertices. Each vertex u of G is assigned two numbers, pre(u) and post(u), as follows. We start with pre(s) = 1, where s is the first vertex in the given ordering. Whenever the subroutine `explore`(u) starts, pre(u) is the least unassigned integer. When the subroutine `explore`(u) finishes, then post(u) is the least unassigned integer. (Note: we use the *same counter* for the pre- and postvisit numbers.)

Problem 2.

(a) Add some lines to the DFS code to fill in the arrays pre and post.
(b) Find the pre- and post-visit times of all vertices in the graph in (1).

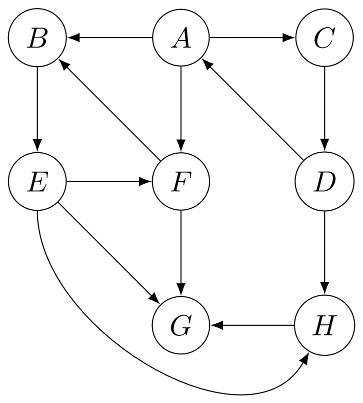
Problem 3. Explain why the previsit and postvisit numberings satisfy the following property.

> For any two vertices u and v, the intervals $[\text{pre}(u), \text{post}(u)]$ and $[\text{pre}(v), \text{post}(v)]$ are either disjoint or comparable. That is, if $\text{pre}(u) \in (\text{pre}(v), \text{post}(v))$ then $\text{post}(u) < \text{post}(v)$.

DFS for directed acyclic graphs. DFS as we described it can be used on directed graphs, taking care to traverse edges only in their prescribed order.

The previsit and postvisit numberings can also be used in this context.

Problem 4. Run DFS on the directed graph below, giving the previsit and postvisit numbers. Draw the DFS tree and (in a different color) add in the edges from the graph that aren't part of the tree.

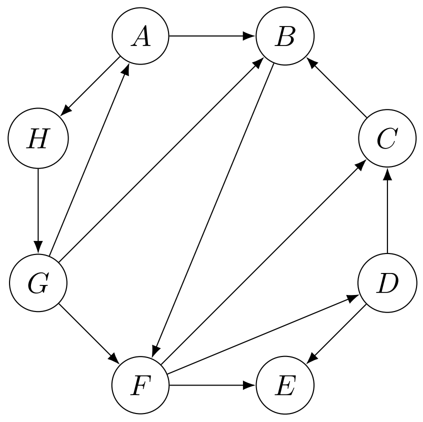


The edges in the graph but not the DFS tree T can be classified as

Forward edges: from a vertex to a non-child descendant. (e.g. $A \to F$ or $E \to G$ in the example.)
Back edges: from a vertex to one of its ancestors. (e.g. $F \to B$ or $D \to A$ in the example.)
Cross edges: lead to neither a descendant nor an ancestor, i.e., to an already-visited node. (e.g. $H \to G$ or $D \to H$ in the example.)

Problem 5. Prove that if $u \to v$ is an edge and u is visited first by DFS, then $u \to v$ is either a tree edge or a forward edge. (This justifies the phrase after 'i.e.' in the definition of 'cross edge.')

Problem 6. Perform DFS on this digraph, using the alphabetical ordering. Classify each edge as a tree edge, forward edge, back edge, or cross edge, and give the pre- and post-number of each vertex.



(2)

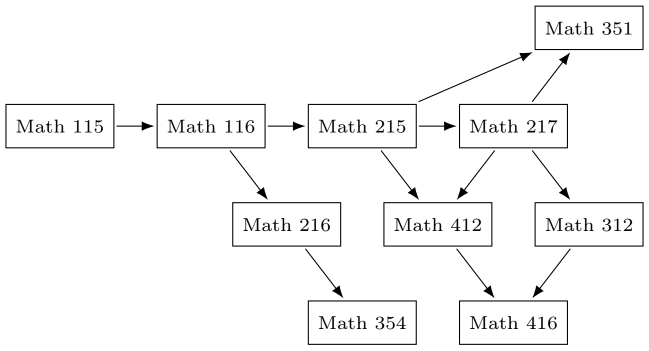
In a directed graph, a **cycle** is a path

$$v_0 \to v_1 \to v_2 \to \cdots \to v_{k-1} \to v_0$$

that does not repeat vertices except for v_0 at the beginning and end.

In the example (2) above, $B \to F \to D \to C \to B$ is a cycle. A directed graph without cycles is called **acyclic**. A DAG is a **directed acyclic graph**.

Among other things, a DAG can model a sequence of tasks: $u \to v$ means that u needs to be completed before v. For example, course prerequisites:



Proposition. A digraph has a cycle iff in DFS there is a back edge.

Problem 7. Prove the Proposition.

(*Hint:* For the direction $\Rightarrow$, suppose that $v_0 \to v_1 \to \cdots \to v_{k-1} \to v_0$ is a cycle and suppose that v_i is the first vertex visited by DFS. Argue that (v_{i-1}, v_i) is a back edge.)

Definition. A **linearization** or **topological sorting** of a directed graph is an ordering of the vertices in which every edge goes from an earlier vertex to a later one:

$$v_0, v_1, v_2, \ldots, v_{n-1}, \quad (v_i, v_j) \in E \text{ implies } i < j.$$

Problem 8. Find any linearization of the graph below. Find another one.

$$\begin{tikzcd}
A \arrow[dr] & & B \\
& D \arrow[ur] & \\
C \arrow[uu] \arrow[ur] \arrow[rr] & & E \arrow[uu]
\end{tikzcd} \qquad (3)$$

Proposition. Every DAG has a linearization.

Problem 9. Finish the following idea for a proof of the Proposition.

(a) Suppose that $u \to v$ is a back edge in DFS. What can you say about post(u) and post(v)?
(b) If, in a DAG, you order the vertices in _________ order of post($-$), then you will get a linearization. Explain.

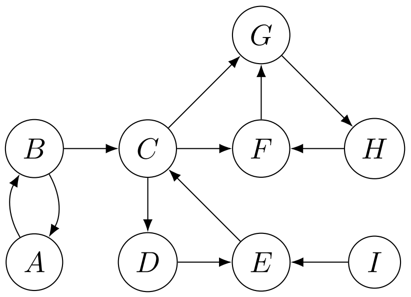
Problem 10. Run DFS on the graph in (3) (starting over as necessary until all vertices have been visited). Which linearization does the postvisit ordering give?

Corollary. Every DAG has a **source** and a **sink**.

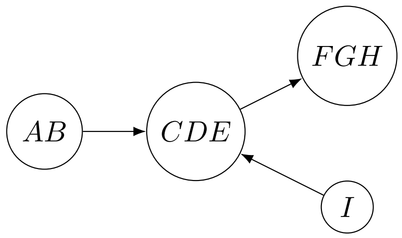
(A **source** is a vertex v for which there are no edges $u \to v$, and a **sink** is a vertex w for which there are no edges $w \to v$.)

Connectivity in directed graphs We say that v is **reachable** from u if there is a path $u \to v_1 \to \cdots \to v_k \to v$ from u to v. In a directed graph, we say that u is **(strongly) connected to** v iff u is reachable from v and v is reachable from u. This gives an equivalence relation, and the equivalence classes are called **strongly connected components** (SCCs).

We would like to use DFS to decompose a digraph into its SCCs. The set of SCCs is naturally thought of as a digraph too, called the **metagraph**: the vertices of the metagraph are the strongly connected components of G, and $C \to D$ is an edge if and only if there are vertices $u \in C$ and $v \in D$ and an edge $u \to v$ in G.



digraph



metagraph

The SCCs are $\{A, B\}$, $\{C, D, E\}$, $\{F, G, H\}$, and $\{I\}$. The metagraph is on the right.

Problem 11. Explain why the following observation is true.

Observation 1. If the `explore` subroutine is begun at u, then it terminates precisely when all nodes reachable from u have been visited.

So start at a *sink* SCC in the metagraph! Will get exactly the SCC and nothing more. Difficult to find a sink. Easier to find a source:

Observation 2. The node that receives the highest post number in DFS must lie in a source SCC.

This follows from:

Observation 3. If C and D are different SCCs, and there is an edge from C to D in the metagraph, then the highest post number in C is greater than the highest post number in D.

Problem 12. Why is Observation 3 true?
(*Hint:* Consider two cases separately. Either DFS visits a vertex in D first, or it visits a vertex in C before any vertex in D.)

So we have a generalization of the earlier linearization algorithm:

SCCs can be linearized by max-post in decreasing order.

So we can get a source SCC by linearizing the metagraph. To get a sink, instead consider G^{rev}, the **reverse graph** of G: $u \to v$ in G^{rev} iff $u \leftarrow v$ in G.

Problem 13.

(a) Explain why G and G^{rev} have the same SCCs.
(b) A vertex is a sink in G iff it is a ______ in G^{rev}.

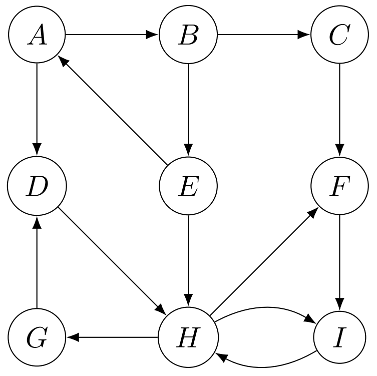
Procedure to find SCCs:

(1) Run DFS on G^{rev}, restarting at new vertices if necessary until all vertices have been visited. Record all pre- and postvisit numbers and identify the vertex v of highest postvisit number.

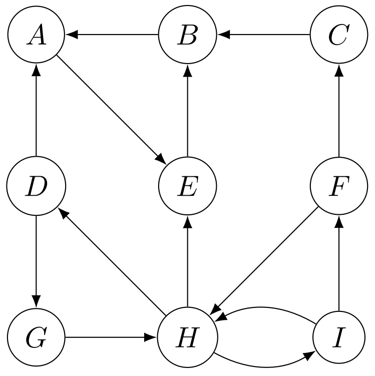
(2) Starting at v, run DFS on G, processing vertices in decreasing order of postvisit number from (1).
(3) The SCCs are delineated by the beginnings of the outer loop of DFS in (2).

Problem 14. (a) Run our algorithm (using the alphabetical ordering of vertices) to find the strongly connected components of the graph G below. (A picture of the reverse graph G^{rev} is provided for your convenience.)
(b) Which SCCs are sinks and which are sources? Draw the metagraph.
(c) What is the minimum number of edges that must be added to this graph to make it strongly connected? Explain your answer.

G:

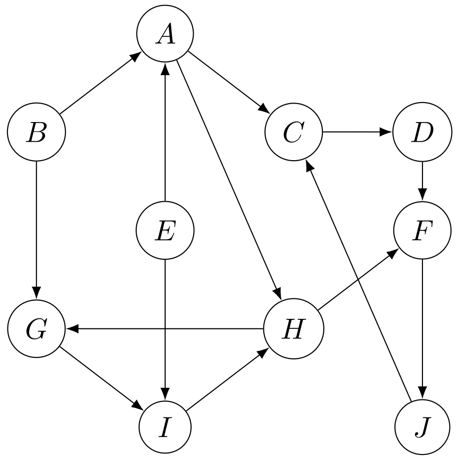


G^{rev}:

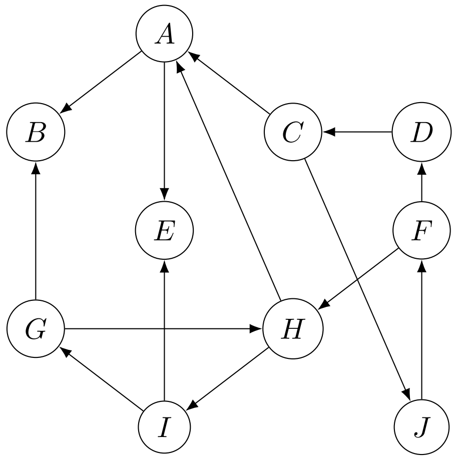


Problem 15. (a) Run our algorithm (using the alphabetical ordering of vertices) to find the strongly connected components of the graph H below. (A picture of the reverse graph H^{rev} is provided for your convenience.)
(b) Which SCCs are sinks and which are sources? Draw the metagraph.
(c) What is the minimum number of edges that must be added to this graph to make it strongly connected? Explain your answer.

H:



H^{rev}:


A
B
C
D
E
F
G
H
I
J