Decomposition of Graphs: Strongly connected components

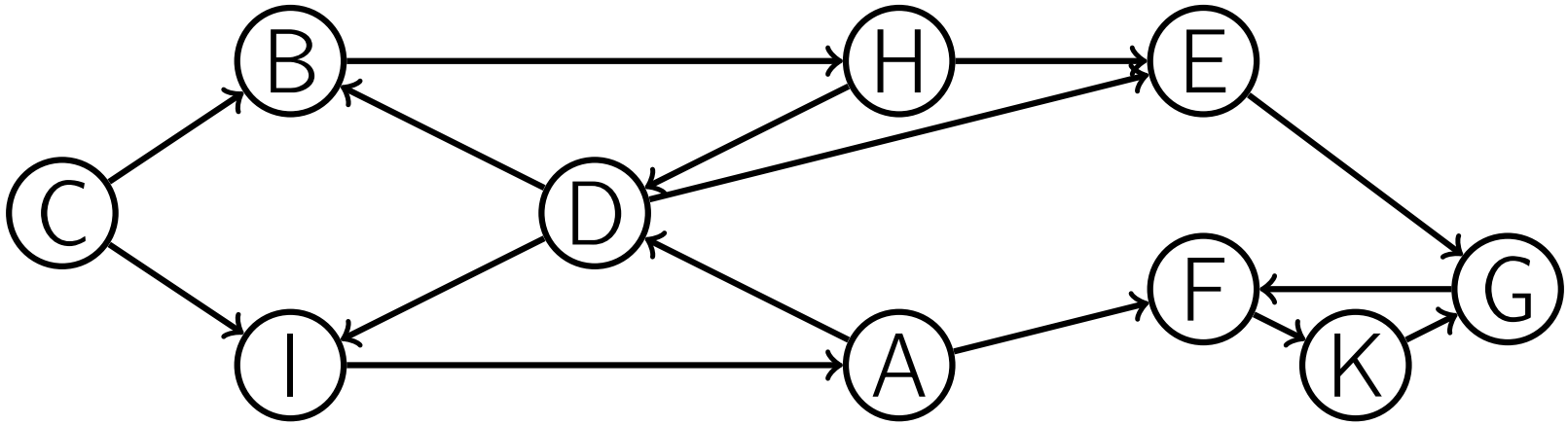
Daniel Kane

Department of Computer Science and Engineering
University of California, San Diego

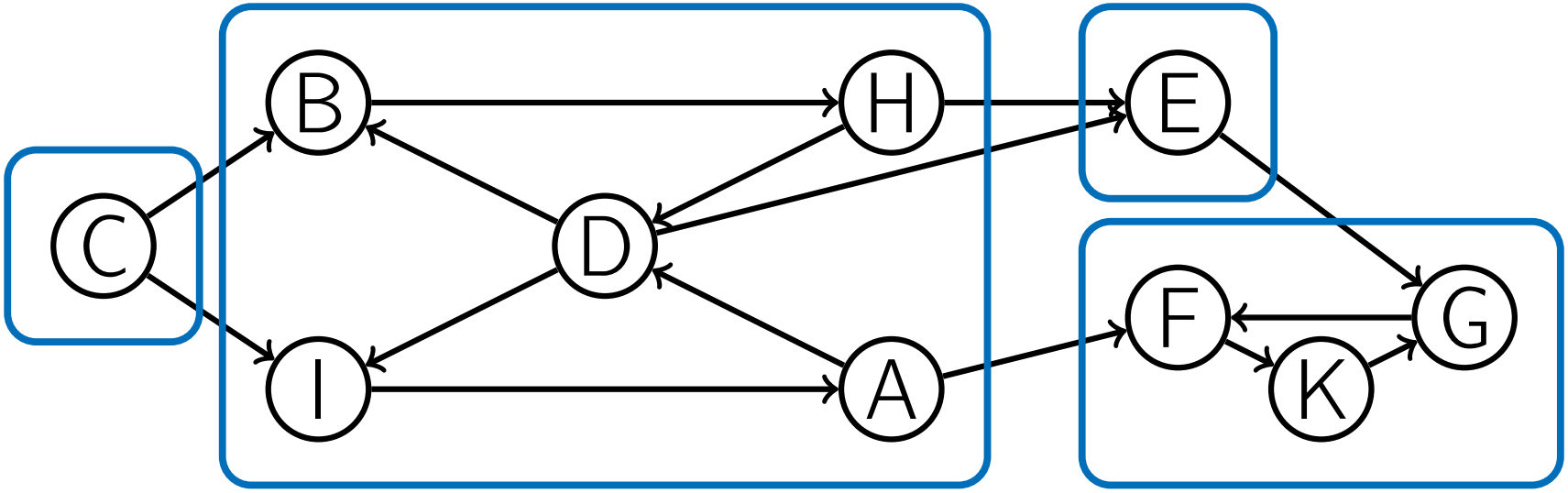
Graph Algorithms
Data Structures and Algorithms

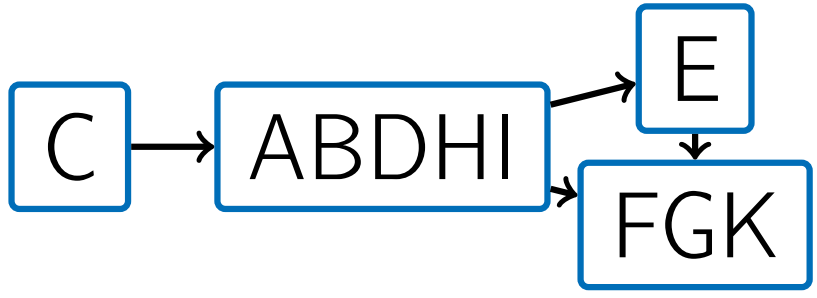
Strongly connected components

A strongly connected component of a directed graph is an inclusion-wise maximal subset of vertices such that there is a (directed) path between any two of them in both directions.



Metagraph





Lemma

If C and C' are strongly connected components, and there is an edge from a node in C to a node in C', then the highest `post` number in C is bigger than the highest post number in C'.

Proof

Case 1. If DFS visits C before C' then all of C and C' will be examined before we end processing a vertex from C, so C will receive a higher `post`

Lemma

If C and C' are strongly connected components, and there is an edge from a node in C to a node in C', then the highest `post` number in C is bigger than the highest post number in C'.

Proof

Case 1. If DFS visits C before C' then all of C and C' will be examined before we end processing a vertex from C, so C will receive a higher `post`

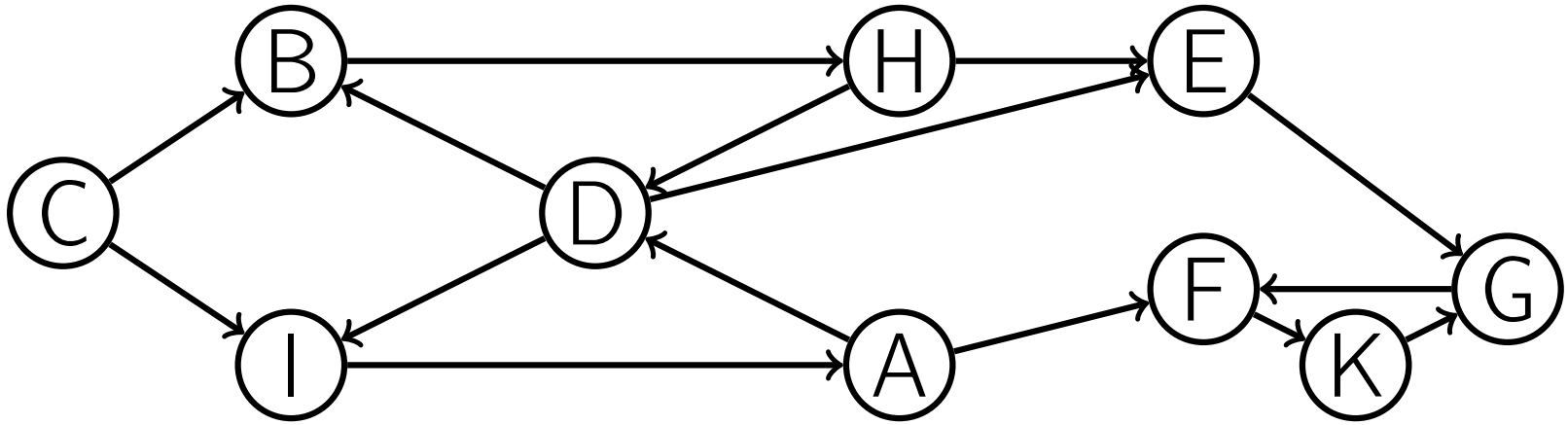
Corollaries

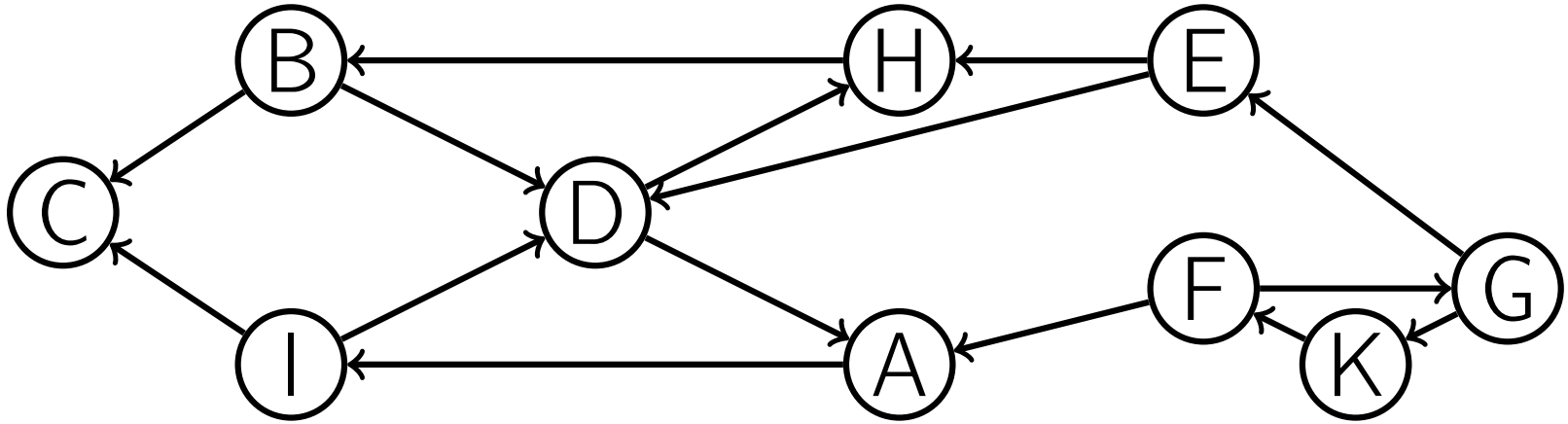
1. The vertex with the highest `post` number lies in a source SCC.

Corollaries

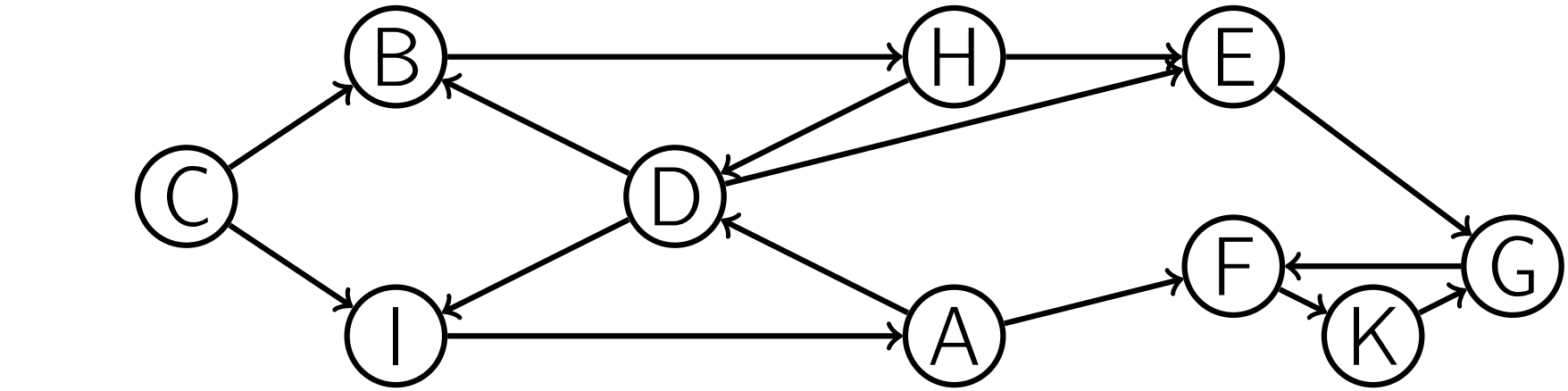
1. The vertex with the highest `post` number lies in a source SCC.
2. SCC's can be linearized by arranging them in decreasing order of their highest `post` numbers.

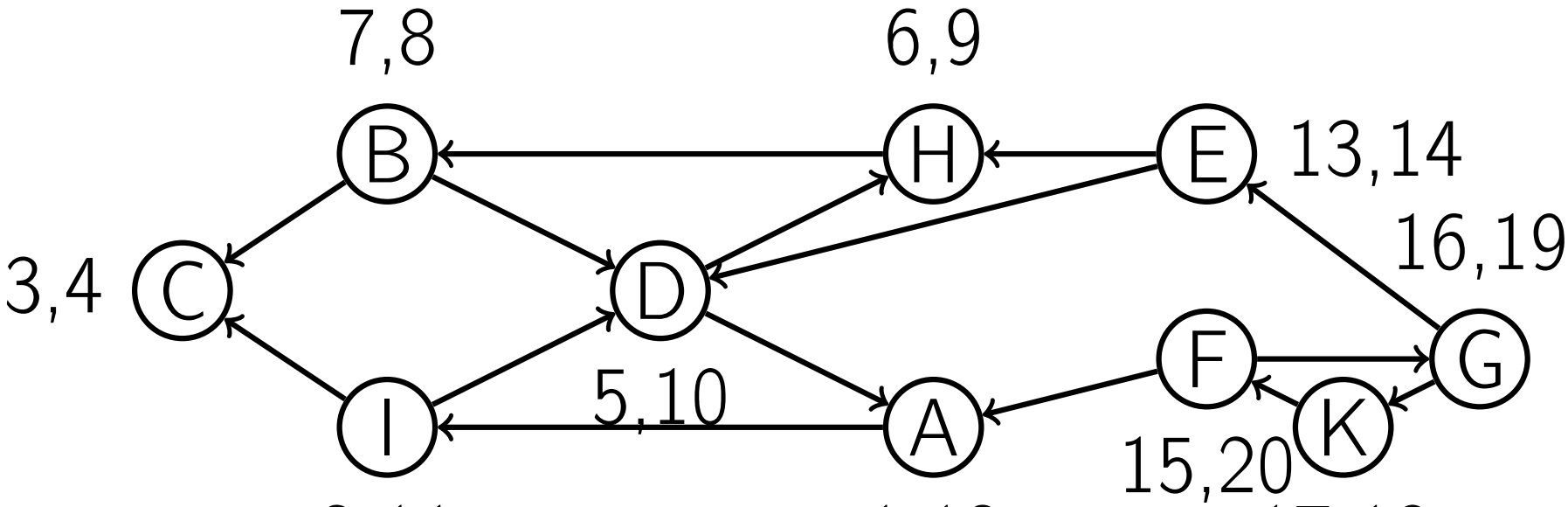
Reverse graph



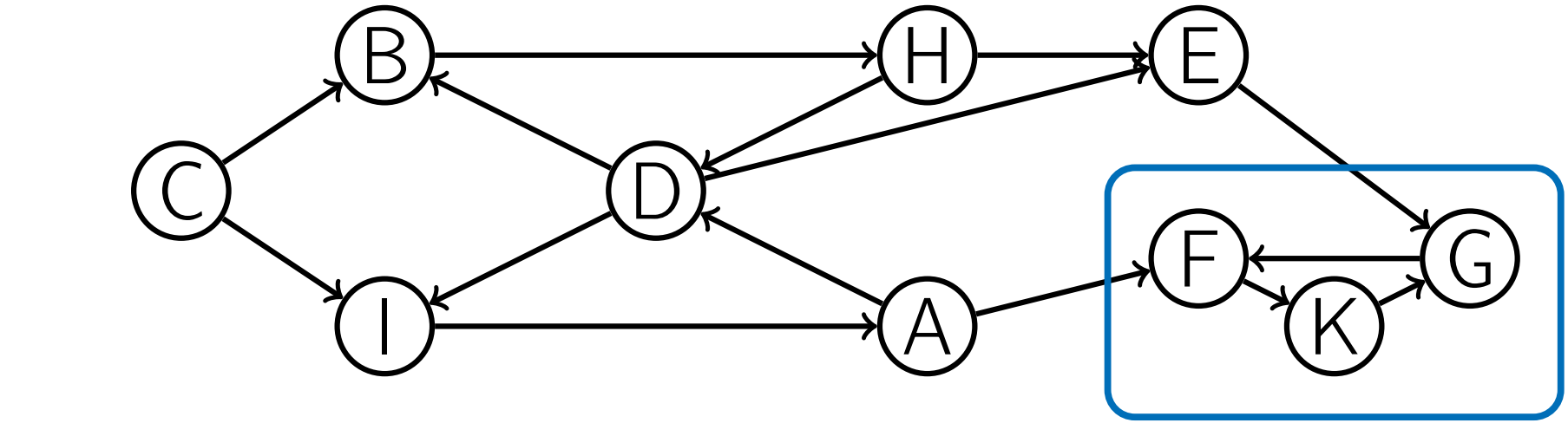


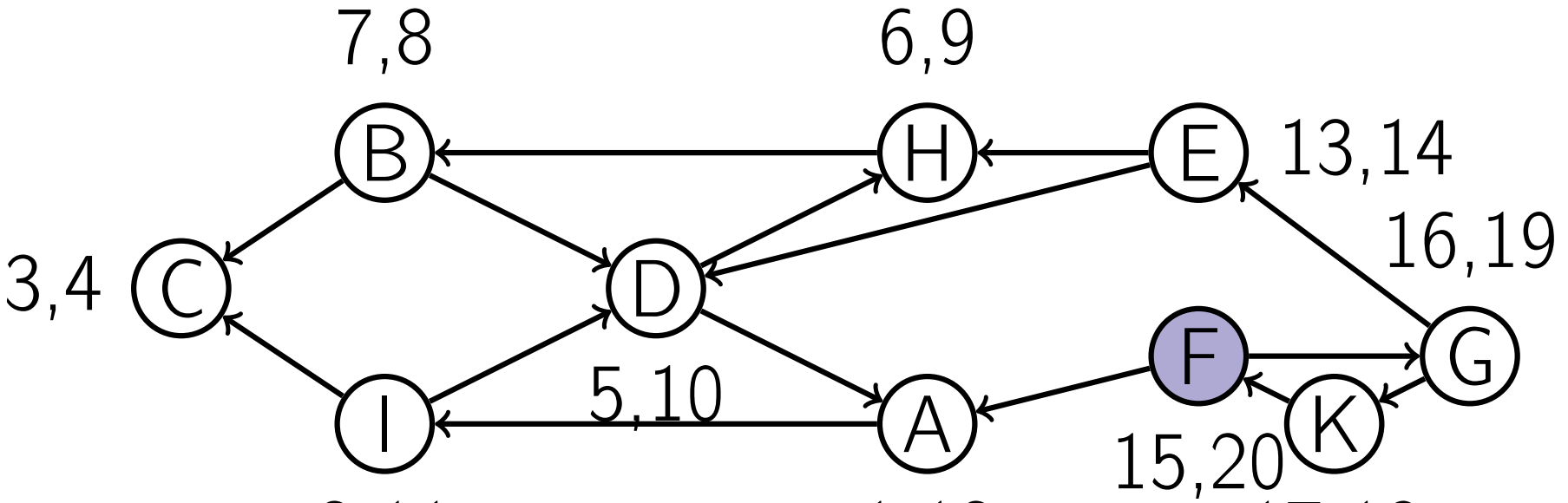
Reverse graph



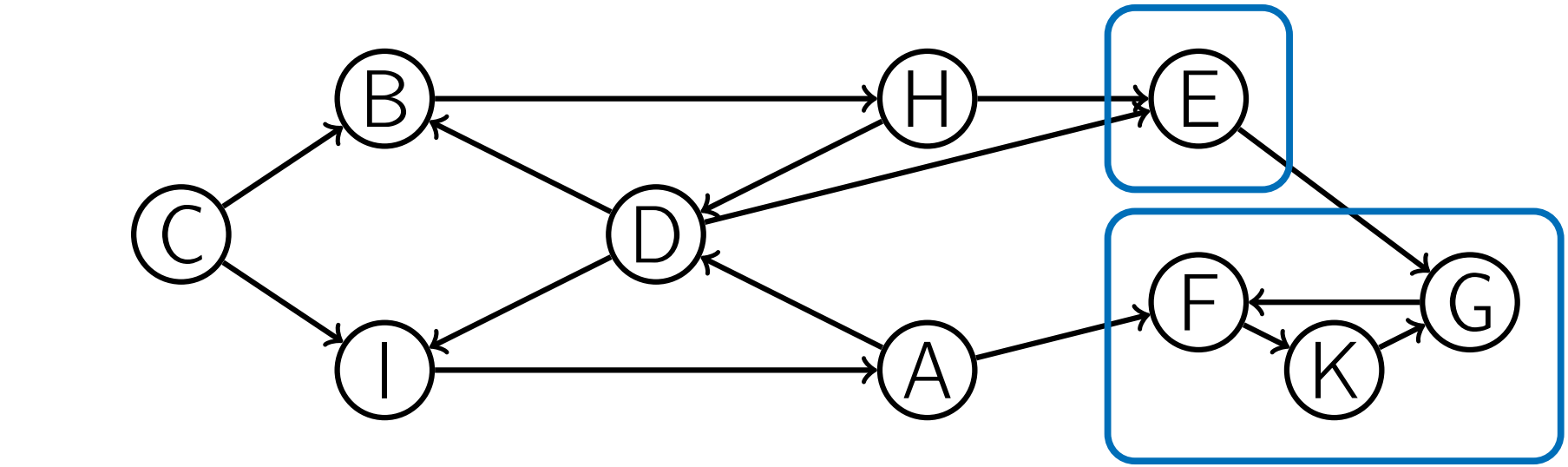


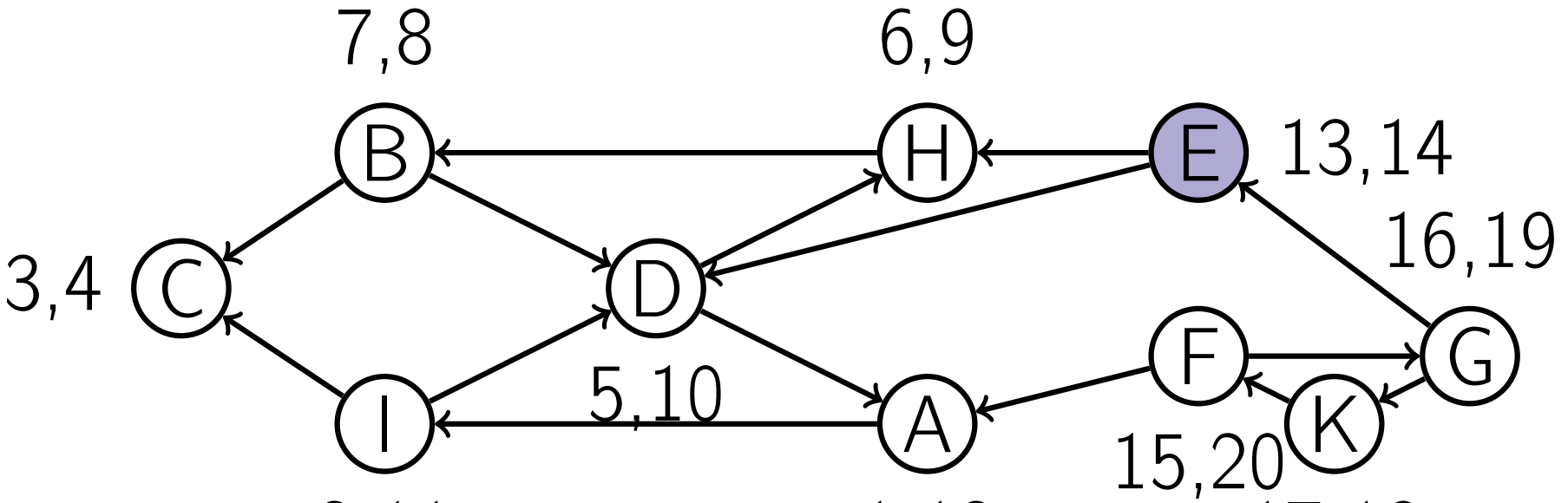
Reverse graph



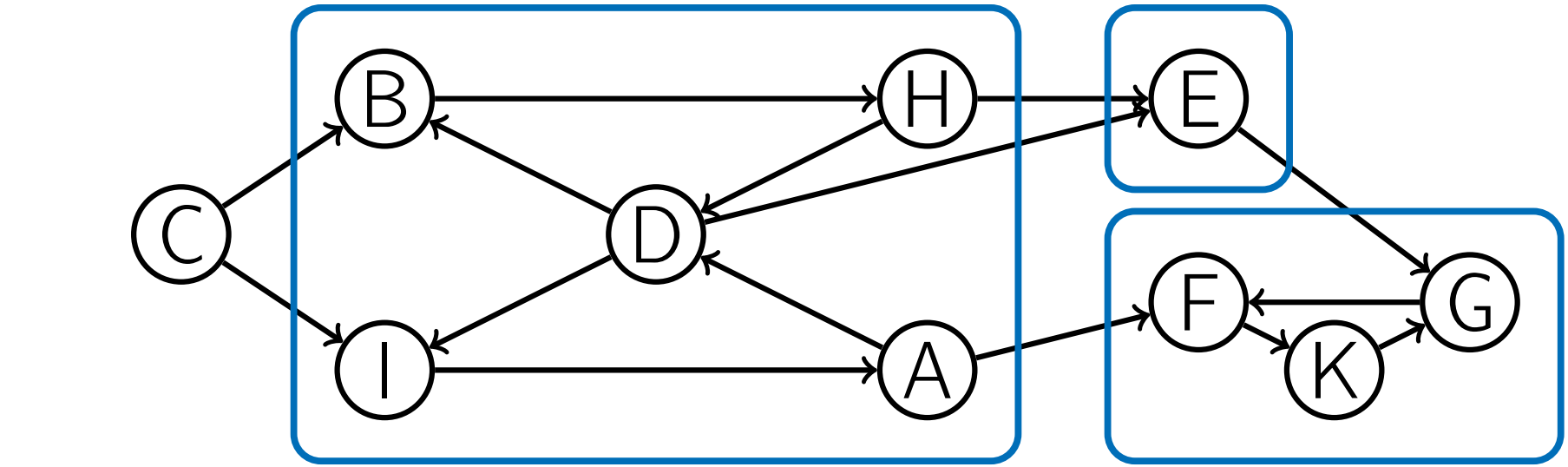


Reverse graph

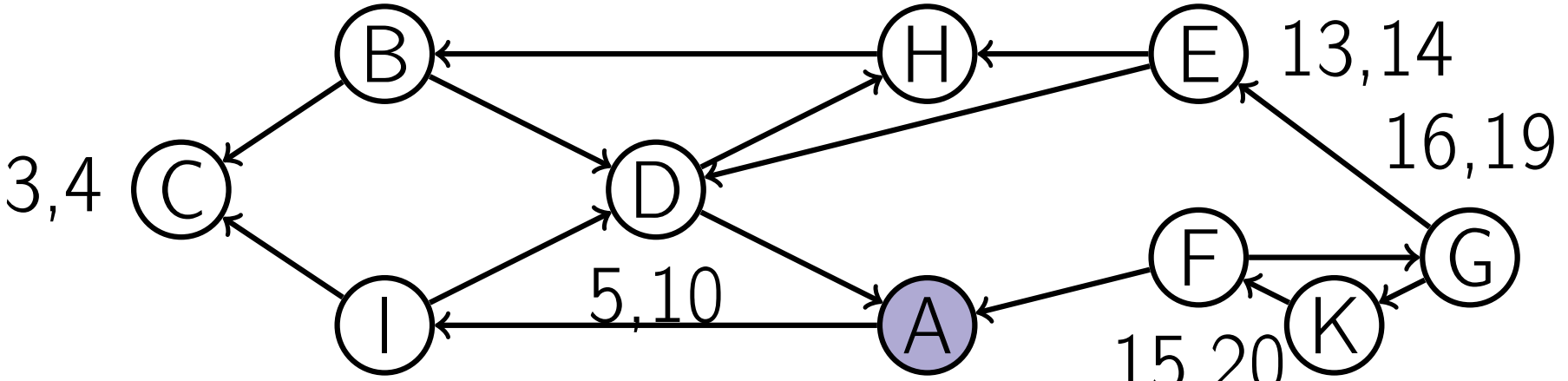




Reverse graph



7,8 6,9



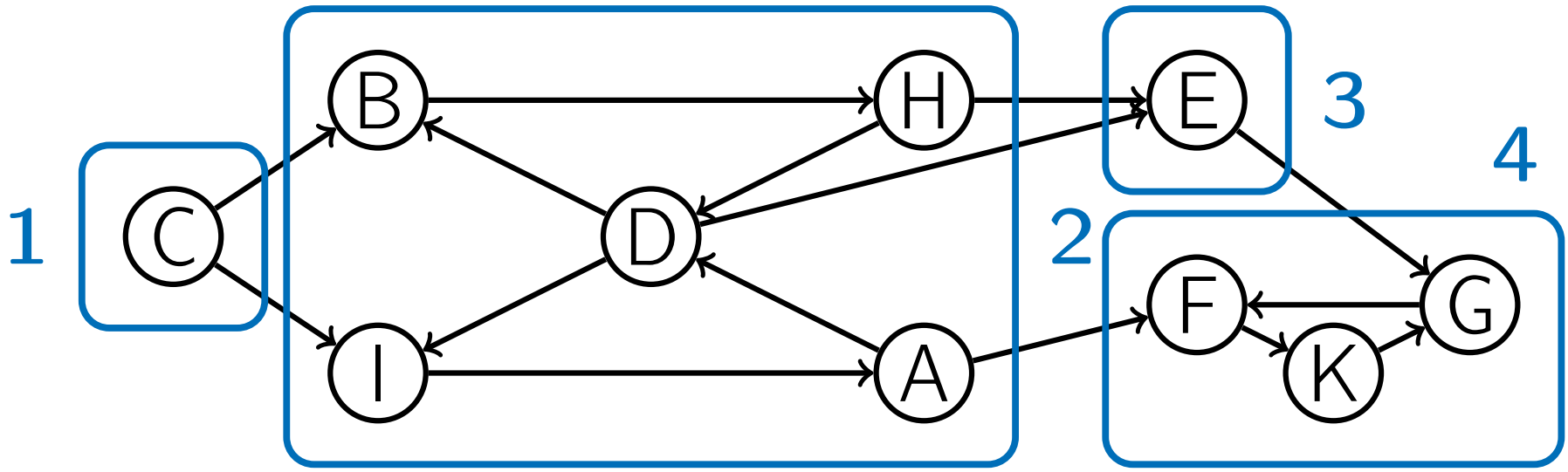
Computing SCC's

SCC(G)

run DFS on G^R

run the undirected connected components algorithm on G processing the vertices in decreasing order of their post numbers from step 1

Constructing metagraph in linear time



Turn the list of edges $\{(A, D), (C, B), \ldots\}$ into $\{(2, 2), (1, 2), \ldots\}$, sort it by calling counting sort twice, remove all duplicates