CS483 Analysis of Algorithms
Lecture 04 – Graph *

Jyh-Ming Lien

February 13, 2008

*this lecture note is based on *Algorithms* by S. Dasgupta, C.H. Papadimitriou, and U.V. Vazirani and *Introduction to the Design and Analysis of Algorithms* by Anany Levitin.

▷ Introduction
What can be represented as graph?
What are the problems that can be solved using graphs?
Graph Representation
Explore graphs
Topological sort
Strong connected components
Conclusion

Introduction

What can be represented as graph?

Introduction
▷ What can be represented as graph?
What are the problems that can be solved using graphs?
Graph Representation
Explore graphs
Topological sort
Strong connected components
Conclusion

- Internet
-
-
-
-
-

What are the problems that can be solved using graphs?

Introduction
What can be represented as graph?
▷ What are the problems that can be solved using graphs?
Graph Representation
Explore graphs
Topological sort
Strong connected components
Conclusion

- Network routing
-
-
-
-
-

Graph Representation

Introduction
What can be represented as graph?
What are the problems that can be solved using graphs?
▷ Graph Representation
Explore graphs
Topological sort
Strong connected components
Conclusion

- Adjacency matrix
  - Space

- Adjacency list
  - Space

Introduction

▷ Explore graphs
Graph Search
Graph Search
Graph Search
Depth-first search
Depth-first search
DFS Application
DFS Application
DFS Application
DFS Application

Topological sort

Strong connected components

Conclusion

Explore graphs

Graph Search

Introduction

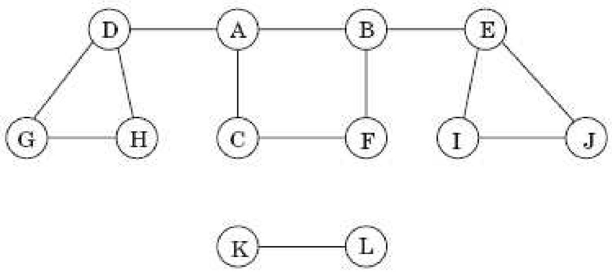
Explore graphs
- Graph Search
- Graph Search
- Graph Search
- Depth-first search
- Depth-first search
- DFS Application
- DFS Application
- DFS Application
- DFS Application

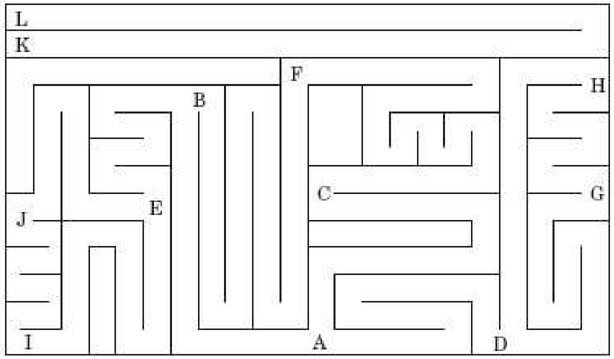
Topological sort

Strong connected components

Conclusion

- What parts of the graph are reachable from a given vertex? (i.e., connected components)
- Many problems require processing all graph vertices (and edges) in systematic fashion
- Basic tools to safely explore an unknown environment
  - –
  - –





Introduction
Explore graphs
Graph Search
▷ Graph Search
Graph Search
Depth-first search
Depth-first search
DFS Application
DFS Application
DFS Application
DFS Application
Topological sort
Strong connected components
Conclusion

- Basic exploration algorithm

Algorithm 0.1: $\text{EXPLORE}(G = \{V, E\}, v \in V)$

- Can the algorithm always work?
  - *proof*

Graph Search

Introduction

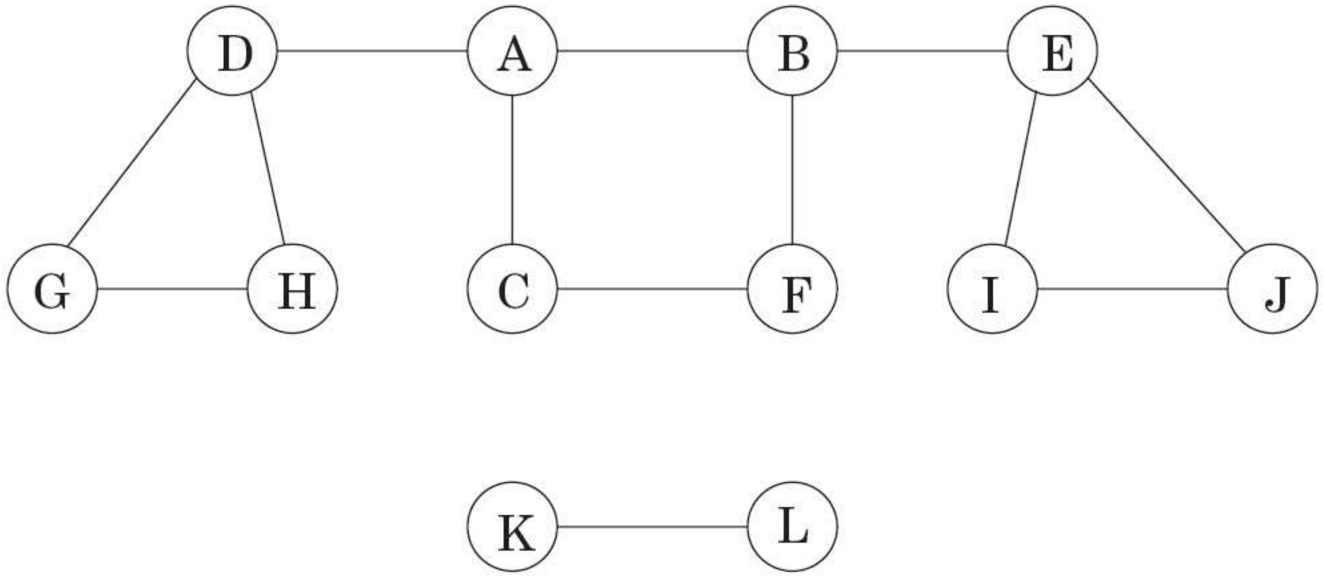
Explore graphs
Graph Search
Graph Search
▷ Graph Search
Depth-first search
Depth-first search
DFS Application
DFS Application
DFS Application
DFS Application

Topological sort

Strong connected components

Conclusion

- Example: EXPLORE(B)



Introduction

Explore graphs
Graph Search
Graph Search
Graph Search
▷ Depth-first search
Depth-first search
DFS Application
DFS Application
DFS Application
DFS Application

Topological sort

Strong connected components

Conclusion

- DFS

Algorithm 0.2: DFS$(G = \{V, E\}, v \in V)$

Depth-first search

Introduction

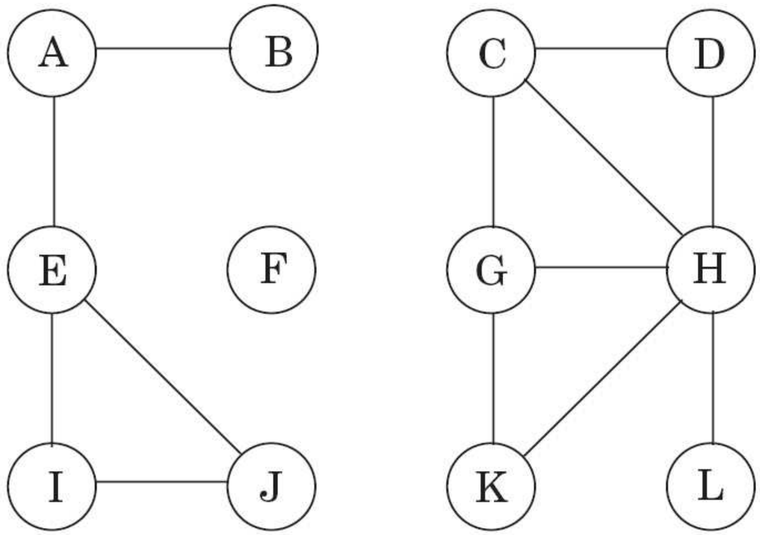
Explore graphs
Graph Search
Graph Search
Graph Search
Depth-first search
▷ Depth-first search
DFS Application
DFS Application
DFS Application
DFS Application

Topological sort

Strong connected components

Conclusion

- Example:



- Time complexity:

DFS Application

Introduction
Explore graphs
Graph Search
Graph Search
Graph Search
Depth-first search
Depth-first search
▷ DFS Application
DFS Application
DFS Application
DFS Application
Topological sort
Strong connected components
Conclusion

- Connect component
  - Given a graph G, report the number of connect components in G.
  - Given a graph G, can you preprocess G so that you can check if two nodes u and v from G are from the same connect component?

DFS Application

Introduction

Explore graphs
Graph Search
Graph Search
Graph Search
Depth-first search
Depth-first search
DFS Application
▷ DFS Application
DFS Application
DFS Application

Topological sort

Strong connected components

Conclusion

- Ancestor/Descendant relationship of tree
  - Given a tree T, can you preprocess T so that you can answer if u is the ancestor of v in *constant time*, where u and v are two nodes from T.

DFS Application

Introduction
Explore graphs
Graph Search
Graph Search
Graph Search
Depth-first search
Depth-first search
DFS Application
DFS Application
▷ DFS Application
DFS Application
Topological sort
Strong connected components
Conclusion

- Given a *directed* graph G, convert G to a tree whose nodes and edge are the vertices and edges of G

Introduction

Explore graphs
Graph Search
Graph Search
Graph Search
Depth-first search
Depth-first search
DFS Application
DFS Application
DFS Application
▷ DFS Application

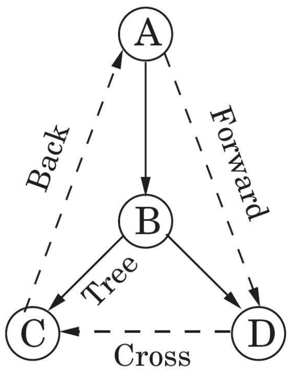
Topological sort

Strong connected components

Conclusion

- Types of edges

DFS tree



- To identify the type of an edge: pre/post ordering

Introduction

Explore graphs

▷ Topological sort

Directed acyclic graphs

DAG and Topological Sort

Topological Sort: Using DSF

Topological Sort: Using Source Removal

Example

Strong connected components

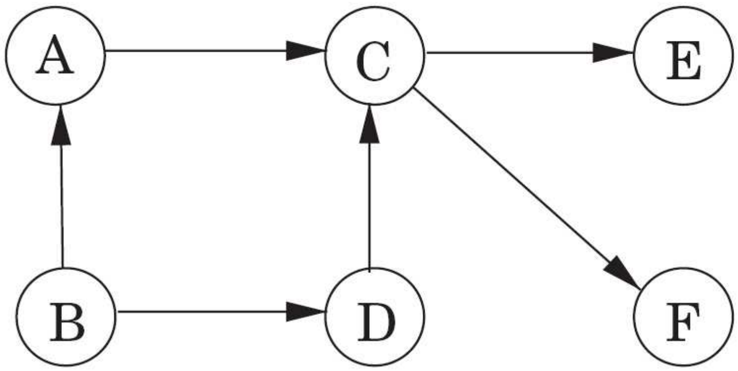
Conclusion

Topological sort

Directed acyclic graphs

Introduction
Explore graphs
Topological sort
▷ Directed acyclic graphs
DAG and Topological Sort
Topological Sort: Using DSF
Topological Sort: Using Source Removal
Example
Strong connected components
Conclusion

- A graph G without (directed) cycle is a *directed acyclic graphs* (DAG)
- DAG can be found in modeling many problems that involve prerequisite constraints (construction projects, document version control)

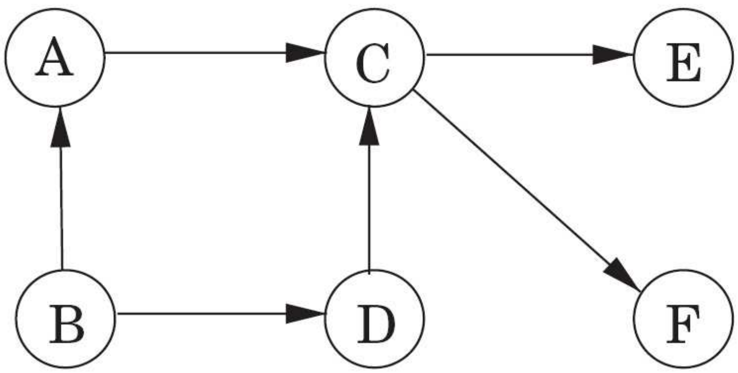


- Given a *directed* graph G, identify *cycles* in G
  - *proof*

DAG and Topological Sort

Introduction
Explore graphs
Topological sort
Directed acyclic graphs
▷ DAG and Topological Sort
Topological Sort: Using DSF
Topological Sort: Using Source Removal
Example
Strong connected components
Conclusion

- **Topological sorting** or **Linearization**: Vertices of a DAG can be linearly ordered so that:
  - Every edge its starting vertex is listed before its ending vertex
  - Being a DAG is also a necessary condition for topological sorting be possible
- Example:



Topological Sort: Using DSF

Introduction
Explore graphs
Topological sort
Directed acyclic graphs
DAG and Topological Sort
▷ Topological Sort: Using DSF
Topological Sort: Using Source Removal
Example
Strong connected components
Conclusion

- Compute DSF and reverse the visit order

Algorithm 0.3: $\mathrm{TS}(G = \{V, E\})$

- Why does it work?

- Time complexity?

Topological Sort: Using Source Removal

Introduction

Explore graphs

Topological sort

Directed acyclic graphs

DAG and Topological Sort

Topological Sort: Using DSF

▷ Topological Sort: Using Source Removal

Example

Strong connected components

Conclusion

- Identify and remove sources iteratively.
  - A source is a vertex without incoming edges.

Algorithm 0.4: $\mathrm{TS}(G = \{V, E\})$

- Why does it work?

- Time complexity?

Example

Introduction

Explore graphs

Topological sort

Directed acyclic graphs

DAG and Topological Sort

Topological Sort: Using DSF

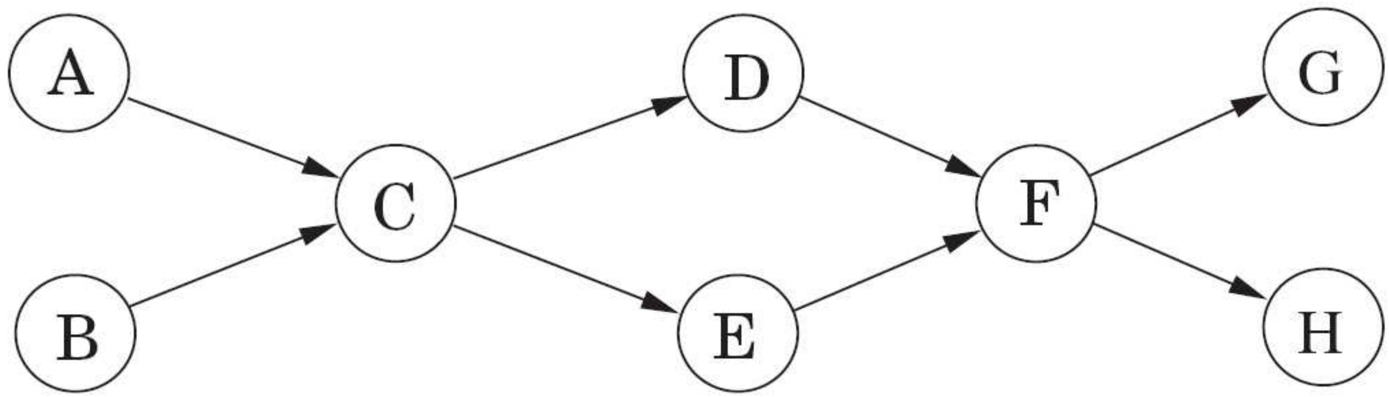
Topological Sort: Using Source Removal

▷ Example

Strong connected components

Conclusion

- Example:



Introduction
Explore graphs
Topological sort
▷ Strong connected components
Strongly connected components
Strongly connected components
Strongly connected components and DAG
Strongly connected components and DAG
Strongly connected components and DAG
Conclusion

Strong connected components

Strongly connected components

Introduction

Explore graphs

Topological sort

Strong connected components

▷ Strongly connected components

Strongly connected components

Strongly connected components and DAG

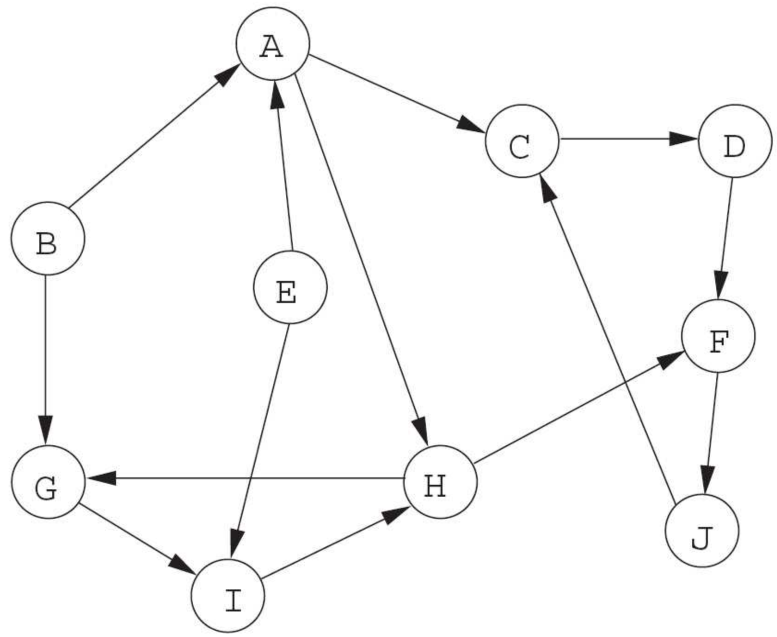
Strongly connected components and DAG

Strongly connected components and DAG

Conclusion

- **Definition**: Two nodes u and v are from the connected if and only if there is a path from u to v and a path from v to u.

- **Definition**: A set of vertices form a strongly connected component (SCC) iff any pairs of vertices are connected.



Strongly connected components

Introduction

Explore graphs

Topological sort

Strong connected components

Strongly connected components

▷ Strongly connected components

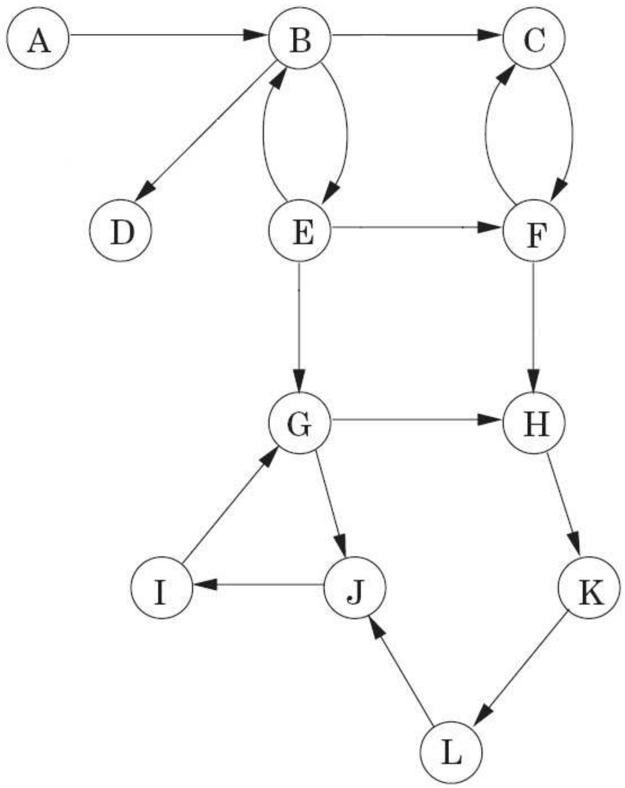
Strongly connected components and DAG

Strongly connected components and DAG

Strongly connected components and DAG

Conclusion

- Connected components in directed graph is less intuitive than that of undirected graph.
  - How many connected components are there in the graph below?



- How to compute SCCs from a directed graph?

Strongly connected components and DAG

Introduction
Explore graphs
Topological sort
Strong connected components
Strongly connected components
Strongly connected components
▷ Strongly connected components and DAG
Strongly connected components and DAG
Strongly connected components and DAG
Conclusion

- **Observation 1**:
- **Observation 2**:

- Our strategy to find all SCC:
  –
  –
  –

Strongly connected components and DAG

Introduction

Explore graphs

Topological sort

Strong connected components

Strongly connected components

Strongly connected components

Strongly connected components and DAG

▷ Strongly connected components and DAG

Strongly connected components and DAG

Conclusion

- **Task**: Find a vertex u in DAG sink node
- **Fact**: It is easier to find a source node.
  -
  -
    - *proof*:
- How to find a vertex u in DAG sink node?
  -
  -

Strongly connected components and DAG

Introduction

Explore graphs

Topological sort

Strong connected components

Strongly connected components

Strongly connected components

Strongly connected components and DAG

Strongly connected components and DAG

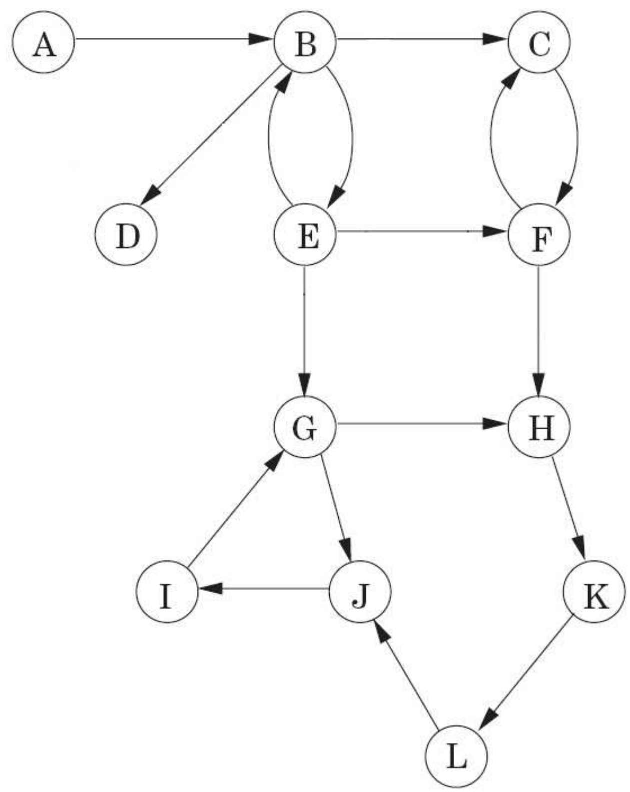
▷ Strongly connected components and DAG

Conclusion

- **Task**: Remove all nodes from the previous SCC and identify a new sink node

  –

- Example:



Introduction
Explore graphs
Topological sort
Strong connected components
▷ Conclusion
Summary
More graph algorithms

Conclusion

Introduction
Explore graphs
Topological sort
Strong connected components
Conclusion
▷ Summary
More graph algorithms

Summary

- Graphs can be very useful for many problems.
- DFS can be used for
  - Explore the graph
  - Reveal relationship between the graph nodes and types of edges
  - Linearization for DAG
  - Identify cycles, connected components, strongly connected components
- Assignment
  -

More graph algorithms

Introduction
Explore graphs
Topological sort
Strong connected components
Conclusion
Summary
▷ More graph algorithms

- Next week we will discuss problems related to paths in graph
  - shortest path in undirected and directed graphs
  - shortest path in weighted graphs
  - shortest path in graphs with negative edges
  - shortest path in DAG