Graphs

CMSC 451, Summer 2009
Sorelle Friedler

Homework 1 Assigned

• Exercises 2.2, 2.5, 3.1, 3.5, 4.5, 4.8, 4.18 from the textbook.
• Remember that these are practice for the exams
• There are solved exercises in the book that may also be helpful
• I will post write-up guidelines later today
Review from Last Class

Graph
Nodes (set of size n)
Edges (set of size m)
Cycle

Graph Representation:
Adjacency Matrix

- $A_{uv}=1$ if $(u,v)$ is an edge
- Space $O(n^2)$
- Checking edge existence takes time $\Theta(1)$
- Two representations of each edge
Graph Representation: Adjacency List

- Node indexed array of lists
- Checking edge existence takes time $O(\text{deg}(u))$
- Two representations of each edge

Breadth-first Search

- Write-up the algorithm
- How fast does it run?
- What graph representation do you use?
Depth-first Search

• What is the order in which we find the nodes?

![Diagram of a tree structure with nodes labeled 1 to 7.]

• Algorithm? Analysis?

Bipartite Graphs

• Undirected graph
• Can be partitioned into two sets such that no edges exist between nodes in the same set
• Models assignments from resources to requests

![Diagram of a bipartite graph with blue and red nodes.]

Directed Acyclic Graphs (DAGs)

- A directed graph with no cycles
- Models precedence constraints
- **Topological order**: An ordering of a directed graph’s nodes $v_1, v_2, \ldots, v_n$, so that for every edge $(v_i, v_j)$, $i < j$

Topological Order and DAGs

- Theorem: G has a topological order if and only if G is a DAG
- Topological order implies DAG:
  - Proof by contradiction: Choose $i$ to be smallest in cycle
  - Choice of $i$ means that $i < j$, but since $(v_i, v_j)$ is an edge, $j < i$
Topological Order and DAGs

• DAG implies topological order:
  – The graph G has some node with no incoming edges
    • Proof by contradiction
    • Keep following some path backwards from a node, eventually you’ll see the same node twice

• DAG implies topological order
  – By induction on n for a graph G of size n
  – Base case: true for n = 1
  – Induction hypothesis: All DAGs on less than n vertices have topological orderings
  – Induction case: let v be a node with no incoming edges
    • G-{v} is a DAG, so it has a topological ordering
    • Put v at the beginning of the topological ordering, append the nodes of G-{v} in topological order
Topological Ordering Algorithm

To compute a topological ordering of G:
Find a node \( v \) with no incoming edges and order it first
Delete \( v \) from \( G \)
Recursively compute a topological ordering of \( G-\{v\} \)
and append this order after \( v \)

- Initialization: For each node, determine the number of incoming edges. Create a set of edges for which this is 0. One scan through the graph \( \mathcal{O}(n+m) \).
- When deleting \( v \), update the counts for all adjacent nodes and add to the set if the count is 0 \( \mathcal{O}(\deg(v)) \)
- Total time: \( \mathcal{O}(n+m) \)

Topological Ordering Algorithm: Example

Topological order:
Topological Ordering Algorithm:

Example

Topological order: $v_1$

Topological Ordering Algorithm:

Example

Topological order: $v_1, v_2$
Topological Ordering Algorithm: Example

Topological order: $v_1, v_2, v_3$

Topological Ordering Algorithm: Example

Topological order: $v_1, v_2, v_3, v_4$
Topological Ordering Algorithm: Example

Topological order: \(v_1, v_2, v_3, v_4, v_5\)

Topological Order: \(v_1, v_2, v_3, v_4, v_5\)
Topological Ordering Algorithm: Example

Topological order: $v_1, v_2, v_3, v_4, v_5, v_6, v_7$. 