CMPSCI 311: Introduction to Algorithms

Review for First Exam

Dan Sheldon

University of Massachusetts

Last Compiled: March 1, 2017

MST

What to know:

- Definitions: spanning tree, MST, cut
- Cut property: lightest edge across any cut belongs to every MST
- Prim's algorithm: maintain a set S of explored nodes. Add cheapest edge from S to $V - S$. Repeat.
- Kruskal's algorithm: consider edges in order of cost. Add edge if it does not create a cycle.

Greedy Algorithms

- Greedy algorithms are "short sighted" algorithms that take each step based on what looks good in the short term.
  - **Example:** Kruskal's Algorithm adds lightest edge that doesn't complete a cycle when building an MST.
  - **Example:** When maximizing the number of non-overlapping TV shows we always added the show that finished earliest out of the remaining shows.

Greedy Algorithms

- Things to note:
  - If a greedy algorithm requires first sorting the input, remember to include the running time of sorting in your overall analysis.
  - It's usually easy to show that greedy algorithms run in polynomial time. . .
  - . . . but extra work may be required to get the most efficient implementation (e.g., priority queue for Dijkstra/Prim; union-find data structure for Kruskal).
  - Focus on correctness proofs: "greedy stays ahead", "exchange argument", induction, contradiction
- What to know:
  - Apply/adapt proof techniques for scheduling problems; solve similar problems
  - Working knowledge of MST algorithms, Dijkstra: apply to concrete examples, understand principles and proof techniques

Graph Algorithms: BFS and DFS Trees

- BFS from node s:
  - Partitions nodes into layers $L_0 = \{s\}, L_1, L_2, L_3 \ldots$
  - L_i defined as neighbors of nodes in L_{i-1} that aren't already in $L_0 \cup L_1 \cup \ldots \cup L_{i-1}$.
  - L_i is set of nodes at distance exactly i from s
  - Returns tree T: for any edge (u, v) in graph, u and v are in same layer or adjacent layer
  - Can be used to test whether G is bipartite, find shortest path from s to t
- DFS from node s
  - Returns DFS tree T rooted at s
  - For any edge (u, v), u is an ancestor of v in the tree or vice versa.
- Both run in time $O(m + n)$
- Both can be used to find connected components of graph, test whether there is a path from s to t

Related "Traversal" Algorithms

Algorithms that grow a set S of explored nodes from starting node s

- BFS (traversal): add all nodes v that are neighbors of some node $u \in S$. Repeat.
- Dijkstra (shortest paths): add node v with smallest value of $d(u) + \ell(u, v)$ for some node u in S, where $d(u)$ is distance from s to u. Repeat.
- Prim (MST): add node v with smallest value of $c(u, v)$ where $u \in S$. Repeat.

Bipartite, Directed Graphs

- An undirected graph G is bipartite if its nodes can be colored red and blue such that no edge has two endpoints of the same color
  - G is bipartite if and only if it does not contain an odd cycle
  - G is bipartite if and only if, after running BFS from any node, there is no edge between two nodes in the same layer
- A directed graph is acyclic (a DAG) if there is no directed cycle
  - There is no directed cycle if and only if there is a topological ordering.
  - Can find a topological order using the fact that a DAG has a node with no incoming edges.

Asymptotic Analysis

Given two positive functions $f(n)$ and $g(n)$:

- $f(n)$ is $O(g(n))$
  - if any only if $\exists c \geq 0, n_0 \geq 0$ s.t. $f(n) \leq cg(n)$ for all $n \geq n_0$
- $f(n)$ is $\Omega(g(n))$
  - if and only if $\exists c \geq 0, n_0 \geq 0$ s.t. $f(n) \geq cg(n)$ for all $n \geq n_0$
  - if and only if $g(n)$ is $O(f(n))$
- $f(n)$ is $\Theta(g(n))$
  - if and only if $f(n)$ is $O(g(n))$ and $f(n)$ is $\Omega(g(n))$
- Know how to apply definitions, compare functions, use to analyze running time of algorithms

Stable Matching

- Colleges, students, preference lists, instability
- Have working knowledge of definitions and algorithm