CS301 Software Development

Instructor: Peter Kemper

Graph Algorithms

Graphs and Minimal Spanning Trees

- Graph G = (V,E)
  - V set of vertices, E set of edges
  - here: undirected, edges can carry a numerical weight or cost
- Graph concepts
  - reachability: Path from node v to w
  - minimal spanning tree:
    - subset of edges such that all vertices are connected, i.e. for all nodes v and w there exists a path from v to w and vice versa
    - minimal: sum of weights is minimal

Search Algorithm

Given an undirected graph G:

- How can you find a spanning tree?
  - Depth-First-Search:

  Starting at some node, explore the graph only via edges that lead to then new, unvisited nodes. Follow path as far as you can, backtrack if node does not have any new adjacent nodes.

  - Breadth-First-Search:

  Same as DFS but we keep a queue of current edges and explore edges to new nodes in the order we recognize them.

Search Algorithm

Given an undirected graph G:

- How can you find a minimal spanning tree?
  - Prim's algorithm
  - Kruskal's algorithm
  - …

Prim's Algorithm (also Jarnik or Prim-Jarnik Algorithm)

- Origin: Jarnik (1930), Prim (1957), Dijkstra (1959)
- Greedy algorithm for minimal spanning tree
- Idea:
  - grow a tree by adding cheapest edge to connect another node
  - has a frontline between known and unknown nodes
  - adds cheapest edge across frontline

- Figure: CC Purpy Pupple, http://en.wikipedia.org/wiki/Prim's_algorithm
- Source: http://en.wikipedia.org/wiki/Prim's_algorithm

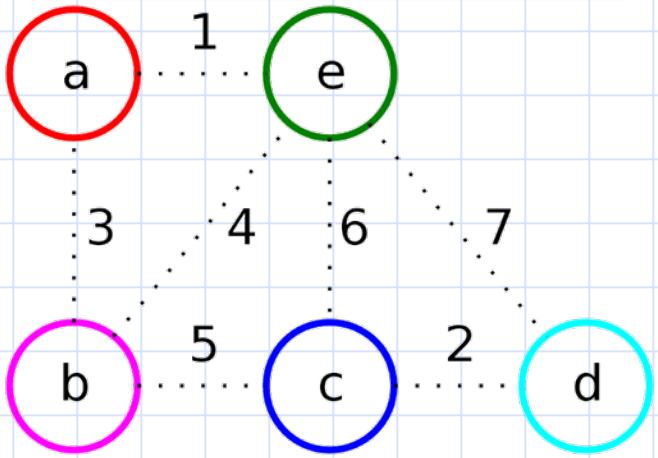
Prim's Algorithm (also Jarnik or Prim-Jarnik Algorithm)

- Input: Non-empty connected weighted graph G
- Initialize:
  - V′ = {x}, for some arbitrary starting point x in V, E′ = {}
- Repeat until V′ = V:
  - Choose an edge {u, v} with minimal weight such that u is in V′ and v is not (if there are multiple edges with the same weight, any of them may be picked)
  - Add v to V′, and {u, v} to E′
- Output: V′ and E′ describe a minimal spanning tree

Kruskal's Algorithm

- Origin: Kruskal (1956)
- Greedy Algorithm for minimal spanning tree
- Idea:
  - grow trees by adding cheapest edge that connects and thus merges two trees

Edge	ab	ae	bc	be	cd	ed	ec
Weight	3	1	5	4	2	7	6



- Figure: CC Schulllz from https://en.wikipedia.org/wiki/Kruskal%27s_algorithm
- Source: https://en.wikipedia.org/wiki/Kruskal%27s_algorithm

Kruskal's Algorithm

- Input: A non-empty connected weighted graph with vertices V and edges E.
- Initialization:
  - create a forest F (a set of trees), where each vertex in the graph is a separate tree
  - create a set S containing all the edges in the graph
- While S is nonempty and F is not yet spanning
  - select and remove an edge e with minimum weight from S
  - if e connects two different trees,
    then add it to the forest F, combining two trees into a single tree
- Output: F is a single spanning tree

Trees, Forests and Mazes

- How to generate a Maze?
  - Think of a set of possible positions V in a space
  - Need a path from any position x to any other position y
  - Do not want too many paths or maze gets boring …
- Metaphor:
  - Think of walls between all positions V
  - Bulldozer starts at some node and creates paths

Trees, Forests and Mazes

- Maze generation seen as a graph problem
  - Positions are the set of nodes V
  - Edges are possible connections (paths of length 1) between nodes
  - Spanning tree is a graph that
    - has a path from any position x to any other position y
    - does not have a single edge to spare or it falls apart
  - Minimal?
    - Concept of weights not necessary here => all weights set to 1

Randomized Algorithms

- Maze needs irregular construction
  - Aside: weights don't matter, all weights = 1
- Solution: randomize decisions
  - Use random number generator to "throw the dice"
- Decision:
  - Which edge to pick next

Randomized Algorithms

- In DFS/BFS algorithms
  - Current node v selected via DFS or BFS
  - For node v: randomly pick an edge to some new node
- In Prim's algorithm: which edge to pick next
  - if all weights equal, select random edge from frontier set
- In Kruskal's algorithm: which edge to pick next
  - if all weights equal, select random edge to connect trees

Summary

- Maze generation problem
- Seen as a spanning tree problem
  - No weights
  - Randomized decisions
- Candidates for algorithms
  - DFS/BFS
  - Prim
  - Kruskal
  - …