CompSci 201, L25: Minimum Spanning Tree (MST) and Disjoint Sets
Logistics, Coming up

• Optional APT 11: Not required, will give APT makeup credit if you do it
  • Makeup credit: Will grade it, add that to apt grade if missing points there.
  • “Due” (for makeup credit) Wednesday 11/30 with grace/late

• Midterm Exam 3 next Monday 12/5

• Project 6: Due next Wednesday 12/7
Midterm Exam 3

• Logistics:
  • 60 minutes, in-person, short answer
  • Can bring 1 reference/notes page

• Major Topics:
  • Trees:
    • binary search trees,
    • binary heaps,
    • recursion
    • Red-black trees: Properties yes, rebalance algorithm, no.
  • Graphs:
    • Recursive & Iterative Stack DFS
    • Iterative Queue BFS
    • Weighted graphs, Dijkstra’s algorithm
Minimum Spanning Tree (MST) and Greedy Algorithms
Minimum Spanning Tree (MST) Problem

• Given N nodes and M edges, each with a weight/cost...
• Find a set of edges that connect *all* the nodes with minimum total cost. (will be a tree)

Weighted undirected graph with:
• Edges labeled with weights/costs
• Minimum spanning tree highlighted
Motivating/Applying Minimum Spanning Tree

• You want to create a connected cable/data network with the least cable/cost/energy possible.

• City planning: Connect several metro stops with least tunneling

• Image Segmentation
You are given an array `points` representing integer coordinates of some points on a 2D-plane, where `points[i] = [x_i, y_i]`.

The cost of connecting two points `[x_i, y_i]` and `[x_j, y_j]` is the **manhattan distance** between them: `|x_i - x_j| + |y_i - y_j|`, where `|val|` denotes the absolute value of `val`.

Return *the minimum cost to make all points connected*. All points are connected if there is exactly one simple path between any two points.
Intuitive Inductive Reasoning

• Suppose we have the MST on N-1 vertices.
• We consider the next vertex to get the MST on N vertices.
  • Must use the cost 2 or the cost 5 edge *regardless* of the rest of the MST
  • Might as well use the cheaper cost 2 edge
Greedy Optimization: Prim’s Algorithm

- Initialize?
  - Choose an arbitrary vertex

- Partial solution?
  - MST connecting \textit{subset} of the vertices.

- Greedy step?
  - Choose the cheapest / least weight edge that connects a new vertex to the partial solution.
Visualizing Prim’s Algorithm

In the visualization:

• Edges between all pairs of vertices
• Weights are implicit by distances
• Algorithm greedily grows by choosing closest unconnected vertex

By Shiyu Ji - Own work, CC BY-SA 4.0,
https://commons.wikimedia.org/w/index.php?curid=54420894
More Intuitive Inductive Reasoning

- Suppose we have chosen some spanning trees so far.
- Must connect all of them, might as well choose the *cheapest* edge connecting two trees.

![Graph](https://commons.wikimedia.org/w/index.php?curid=644030)

Bold green edges = chosen
Greedy Optimization Again: Kruskal’s Algorithm

• Initialize?
  • All nodes in *disjoint sets*

• Partial solution?
  • Forest of spanning trees in disjoint sets

• Greedy step?
  • Choose the cheapest / least weight edge that connects two disjoint sets / trees, connect them.
Visualizing Kruskal’s Algorithm

In the visualization:

• Edges between all pairs of vertices
• Weights are implicit by distances
• Algorithm greedily grows by cheapest edge that connects disjoint sets/trees.

By Shiyu Ji - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=54420894
WOTO
Go to duke.is/yewr2

Not graded for correctness, just participation.

Try to answer *without* looking back at slides and notes.

But do talk to your neighbors!
Kruskal’s Algorithm in Pseudocode

Input: N node, M edges, M edge weights

• Let MST to an empty set
• Let S be a collection of N disjoint sets, one per node
• While S has more than 1 set:
  • Let (u, v) be the minimum cost remaining edge
  • Find which sets u and v are in. If not equal:
    • Union the sets
    • Add (u, v) to MST
• Return MST
Kruskal’s Algorithm Runtime?

Input: N node, M edges, M edge weights

• Let MST to an empty set
• Let S be a collection of N disjoint sets, one per node
• While S has more than 1 set:
  • Let (u, v) be the minimum cost remaining edge
  • Find which sets u and v are in. If not equal:
    • Union the sets
    • Add (u, v) to MST
• Return MST

Looping over (worst case) all M edges
Remove from binary heap, O(log(M))
O(M(log(M)+C)) where C is time for Union/Find
Solving Example MST Problem

leetcode.com/problems/min-cost-to-connect-all-points

Live Coding