Tutorial 10, Nov 24, 2023

Prim's MST Algorithm

- Intuition: build up the MST from one node, adding one edge to the tree at a time by selecting the smallest edge out of the tree
- Build a tree T^* and keep an array $closest[u]$ so that at the end of each iteration, for $u \notin T^*$, $closest[u]$ is a node $v \in T^*$ that has the edge of minimum cost between u and v
- Initially, T^* is just a single node v, so initialize $closest[u]$ to v for every other node
- For each iteration:
  - Iterate over all nodes to find the minimum weight edge going out of T^*, connecting to node v
    - For ever node w, check the weight of the edge between w and $closest[w]$
  - Add the edge and node v to the current tree
  - For every node u that's not in the current tree, update $closest[u] \leftarrow \min(closest[u], c[u, v])$ where $c[u, v]$ is the weight of the edge connecting u, v (infinite if no such edge exists)
- Note that we use an adjacency matrix to look up edge weights in $O(1)$
- In this formulation, the complexity is $O(n^2)$

Uniqueness of MSTs

- Theorem: if G has only distinct edge weights, then its MST is unique
- Proof:
  - Suppose that G has 2 distinct MSTs: A and B
  - Take $diff(A, B)$ to be all edges that are in A but not B, or in B and not A
  - Let e_1 be the (unique) edge of minimum weight in $diff(A, B)$; WLOG assume it is in A and not in B
  - Now consider $B' = B + e_1$; B' now has a unique cycle C containing it
  - A cannot contain C, so C has an edge $e_2 \notin A$, but $e_2 \in B$, so $e_2 \in diff(A, B)$, which means e_2 has greater weight than e_1
  - Let $B^* = B + e_1 - e_2$; after removing an edge from the cycle, we are left with a tree, so B^* is a spanning tree
  - However the weight of B^* is less than the weight of B since the weight of e_2 is greater than the weight of e_1
  - This leads to a contradiction because B is an MST, so B^* cannot have a smaller weight