CSCI 323 Lecture 11 Note Monday, 03/11/2022

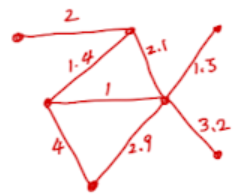
Announcements:
- The midterm will be on Wednesday, March 23, 2022.
- The Lecture Recordings will be available on the following YouTube Playlists Link: https://youtube.com/playlist?list=PLZaTmV9UMKlgYpo2cAiMaEWxqyvbiXDFd

Greedy Algorithm

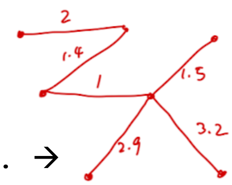
References:
Algorithm Design - Chapter 4.5

Minimum Spanning Tree [4.5]

- Input: a weighted (undirected) graph $G = (V, E, w_e > 0)$,
  w_e is the weight on edge e.
- Assume All w_e are distinct.
  - o vertices – cities
  - o edges – cost of laying a cable between the endpoint cities.



The cheapest way to connect all cities. →



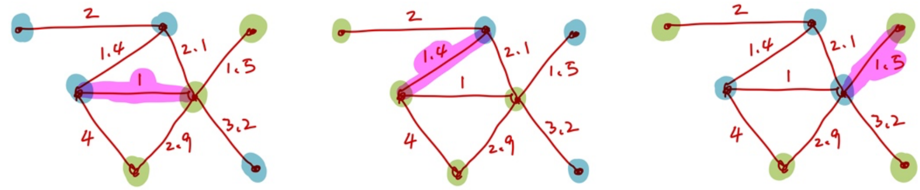
- Question: Given $G = (V, E, w_e > 0)$ as input, find the minimum total cost. (spanning tree)
  - o Spanning tree of a graph is a connected subgraph with no cycle that have all the vertices.

- There are three algorithms to find MST.
- I will state the three algorithms and there are two properties.
- From the two properties, we will prove the optimality of the three algorithms.
- <u>Algorithm 1 – Kruskal's Algorithm</u>
  1) Sort edges in increasing order of weight
  2) Starting with the empty graph, add edges one-by-one in this increasing order, as long as addition of an edge does not create a cycle.
  - o Example, in the above graph it will add the edges in following order: $1,\ 1.4,\ 1.5,\ 2,\ 2.9,\ 3.2$.

- <u>Algorithm 2 – Prim's Algorithm</u>
  1) Start with any vertex s, add the cheapest edge out of s.
  2) At each time, attach the vertex to our partial tree that has least connection cost.
  - o Example, start with s, it will add the edges in following order: $3.2,\ 1,\ 1.4,\ 1.5,\ 2,\ 2.9$.
  - o Cheapest attachment cost to any vertex in the tree so far.
- Kruskal's Algorithm and Prim's Algorithm gives the same outcome.
  - o Kruskal's Algorithm adds the edges all over the places.
  - o Prim's Algorithm build the tree from the source.

- <u>Algorithm 3 – Reverse Delete "Reverse Kruskal"</u>
  - o Start with the graph, delete edges in decreasing order of cost, as long as deletion does not disconnect the graph.
  - o Example, in the above graph it will delete the edges in following order: $4,\ 2.1$.

- Property 1 – Cut Property
  - A *cut* is a partition of the vertices into two disjoint subsets.
  - Assume that all edge costs are distinct. Let S be any subset of nodes that is neither empty or equal to V and let e be the cheapest edge with one edge in S and the other in $V-S$, the complement of S. (e is the cheapest edge that cross the cut.)
  - The cut property states that, then e must be in every minimum spanning tree.
    - For examples:



In the 1st example e = 1, in the 2nd example e = 1.4, in the 3rd example e = 1.5. So, the edges 1, 1.4, 1.5 need to be in any MST.

- Cut property justifies when an edge can be included in the MST.
- Assume the cut properties, we can prove optimality of Prim's and Kruskal's Algorithms.
  - Prim's: If S is the set of nodes that we have already connected, then Prim's algorithm indeed adds the cheapest edge e with one endpoint in S and another in $V-S$. Therefore, all edges added by Prim are justified by the cut property.
  - Kruskal's: Consider an edge $e=(u,v)$ just added by Kruskal. What is the cut $(S,V-S)$ that justifies addition of e?
    $S=\{$set of vertices to which u has a path to/from in the tree so far$\}$
    Clearly $u\in S$. But $v\notin S$ (otherwise adding e will create a cycle)
    → $v\in V-S$ (Kruskal added a cut edge). But e must be the cheapest edge connecting S and $V-S$ because if there was a cheaper such edge, Kruskal's must have encountered it before e and added it since it would not have created a cycle.
- Proves OPTIMALITY of Prim's and Kruskal's algorithm.

- Property 2 – Cycle Property (will prove optimality of reverse-delete)
  - Consider any cycle C in the graph. Let e be the most expensive edge in C. Then e cannot be in any minimum spanning tree.
  - Assuming the cycle property, the reverse-delete algorithm is optimal. Because every edge it deletes is justified by the cycle property.
    If RD deletes e then e must be part of a cycle $C=P\cup\{e\}$. P exits because deletion of e does not disconnect u from v.
    e must be the most expensive edge in C because RD considers edges in decreasing cost.

What to expect or prepare for the next class:

- Minimum Spanning Tree, prove of the two properties

Reading Assignment

Algorithm Design: 4.5