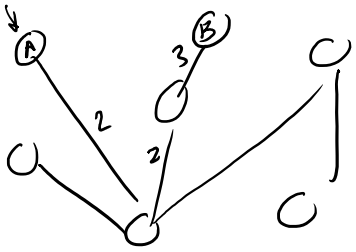
2020-04-14 Minimum Spanning Trees

Tuesday, April 14, 2020 6:59 AM

Student Questions

- Confusion over definition of MST
  - Given a set of vertices, how might we connect them such that every vertex in the graph is connected using the least possible edge weight



- What are practical applications?
  - Running wire (electrical, network, or anything else that costs money)

- Is there a reason why one would choose Kruskal's or Prim's algorithm?
  - What are the runtime efficiencies of each?
  - What are the space efficiencies of each?
- Some students find Kruskal's MST (disjoint sets) easier vs Prim's (similar to Dijkstra's)

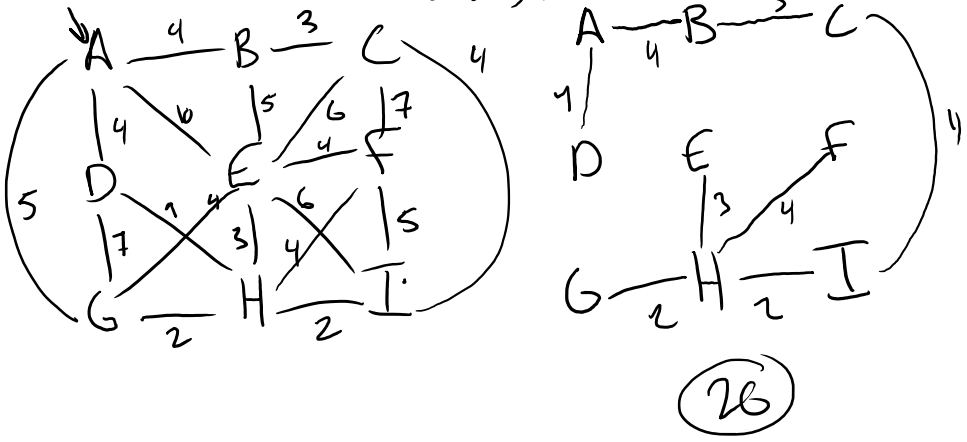
Review of Prim's MST algorithm

1. Pick some arbitrary starting vertex. From the selected vertex, we push all outgoing edges into a PQ (very similar to Dijkstra's algorithm)
2. While the graph is not fully connected:
   a. Pop off top edge. If vertex is not seen before, accept vertex. Push all new edges from this accepted vertex to the PQ

$E \leq V^2$

$O(V)$ or $O(E)$

$\log(cV) \leq \log(E)$



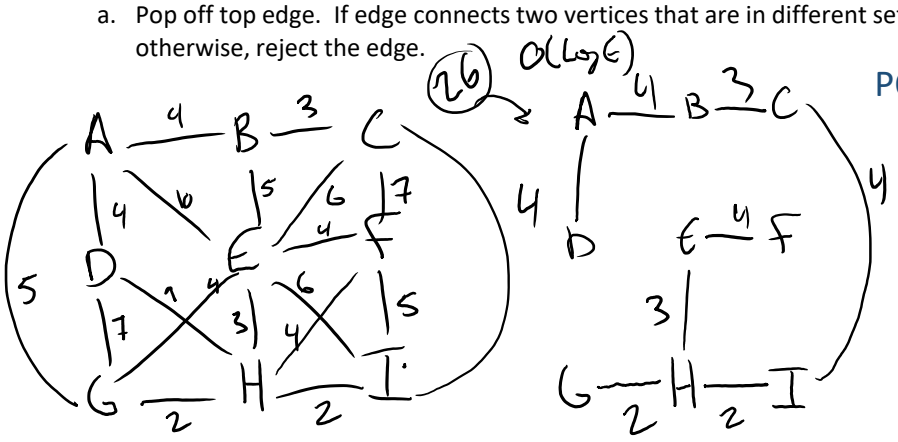
Priority Queue

- ~~AB->4~~
- ~~AD->4~~
- AG->5
- AE->6
- ~~BC->3~~
- BE->5
- ~~CI->4~~
- CF->7
- CE->6
- DG->7
- DH->9
- ~~IH->2~~
- IE->6
- IF->5
- ~~HG->2~~
- HD->9
- ~~HE->3~~
- ~~HF->4~~
- ~~GE->4~~
- EF->4

Kruskal's MST

O(E)

1. Pushes all edges into a min PQ
2. While all vertices not connected: O(V) O(E)
   a. Pop off top edge. If edge connects two vertices that are in different sets, accept the edge. otherwise, reject the edge.

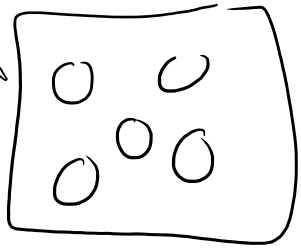


PQ

- ~~GH->2~~
- ~~HI->2~~
- ~~BC->3~~
- ~~HE->3~~
- ~~AD->4~~
- ~~AB->4~~
- ~~EF->4~~
- CI->4
- HF->4
- BE->5
- AG->4
- FI->5

…(all other edges follow)

disjoint



not disjoint