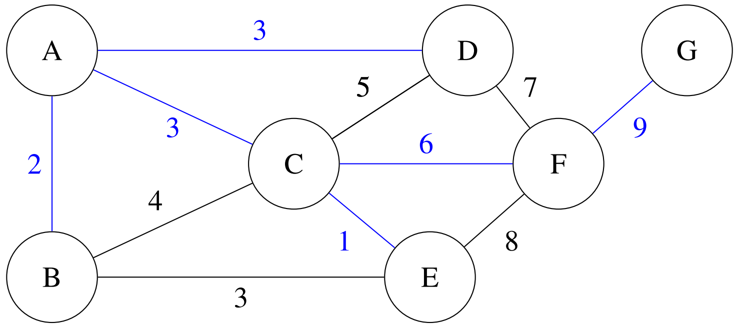
CS 61B
Summer 2020

Small Group Tutoring
Section 11: MSTs

Worksheet 11

1 Minimum Spanning Tree

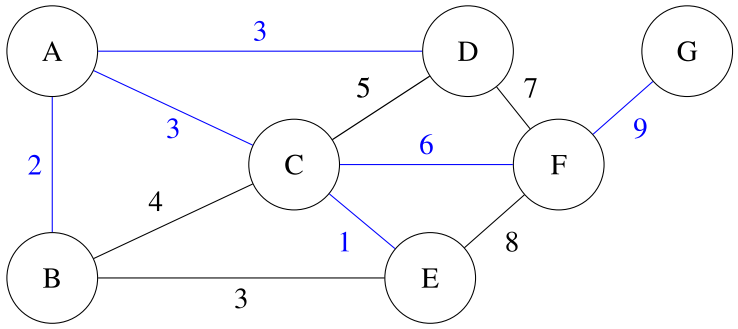


(a) Given the graph above, run Kruskal's and Prim's algorithm to determine the minimum spanning tree of this graph. For Prim's algorithm, assume we start at node A and fill in the following chart including the value cost(v) for all vertices v for that iteration as well as which node was popped off of the fringe for that iteration. (Note: Ties are broken in alphabetical order)

Solution: To find the minimum spanning tree, both Kruskal's and Prim's algorithm apply the cut property, which states that **the shortest edge between two disjoint sets of nodes must be in the minimum spanning tree**.

Prim's algorithm starts at a node, and we'll use A for this example. Now, we identify the smallest edge out from A, which is the one to B, and add it to the minimum spanning tree because the two nodes are not already connected. We perform the same operation on our new set of nodes, and the smallest edge out is from A to C. After that, our set of connected nodes includes A, B, and C, and the smallest edge out from that is the one between C and E, and so on and so forth until we end up with the resulting image. We can accordingly fill in the chart below:

v	init	Pop a	Pop b	Pop c	Pop e	Pop d	Pop f	Pop g
cost(a)	**0**	0	0	0	0	0	0	0
cost(b)	∞	**2**	2	2	2	2	2	2
cost(c)	∞	3	**3**	3	3	3	3	3
cost(d)	∞	3	3	3	**3**	3	3	3
cost(e)	∞	∞	3	**1**	1	1	1	1
cost(f)	∞	∞	∞	6	6	**6**	6	6
cost(g)	∞	∞	∞	∞	∞	∞	**9**	9



(b) Run Kruskal's algorithm on the same graph.

Solution: To employ Kruskal's algorithm on this graph, we'll start with all the nodes disconnected. From there, we locate that smallest edge, which is the one between C and E. Since those two nodes are not already connected, we know that this edge must be in the minimum spanning tree. Using the same logic, we add the edge between A and B. In the next iteration, we have three edges with weight 3. We can see that adding the edge between A and D connects disjoint sets of nodes. Now, we add the edge between A and C. Afterwards, the next smallest edge is between B and E, but there is already a path from B to E, so that edge is not in the minimum spanning tree. Anyways, the final edges to the graph are illustrated using the same logic as before.

(c) Does Kruskal's algorithm for finding the minimum spanning tree work on graphs with negative edge weights? Does Prim's?

Solution: Yes, both algorithms work with negative edge weights because the cut property still applies.

(d) True or False: A graph with unique edge weights has a unique minimum spanning tree.

Solution: True. This can be proved using the cut property. Unique edge weights implies that for every cut, there exactly one minimum-weighted edge.

s

2 Fiat Lux

After graduating from Berkeley with solid understanding of CS61B topics, Josh became a billionaire and wants to build power stations across Berkeley campus to help students survive from PG&E power outages. Josh want to minimize his cost, but due to the numerous power outages when he took CS61B, he did not learn anything about Prim's or Kruskal's algorithm and he is asking for your help! We must meet the following constrains to power the whole campus:

- There are V locations where Josh can build power stations, and it costs v_i dollars to build a power station at the i^{th} position.
- There are E positions we can build wires and it cost e_{ij} to build a wire between location i and j.
- All locations must have a power station itself or be connected to another position with power station.
- $e_{ij} << v_i, \forall i, j$

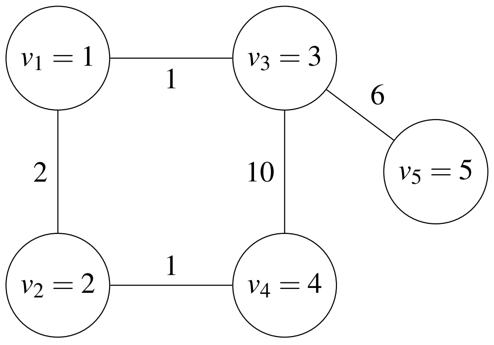
Use the Prim's or Kruskal's algorithm taught in class to find a strategy that will minimize the cost while still fulfilling the constrains above.

Solution:

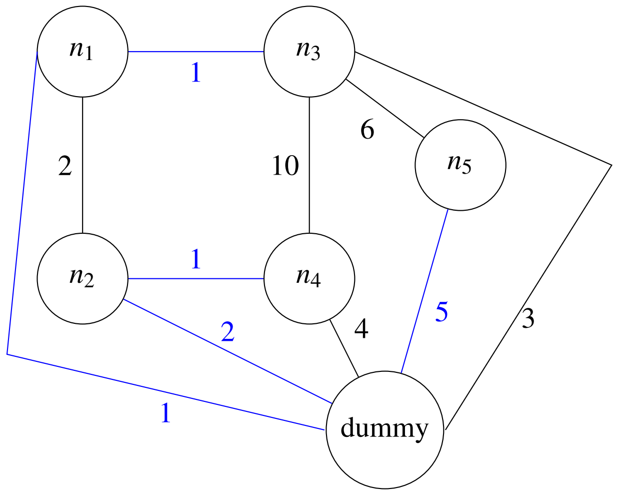
As the question suggests, this problem can be reduced to a graph problem where we want to have nodes either be marked themselves or connected to a marked one. To do so, we first create a graph with one vertex per location with edges between them corresponding to the cost of building a wire between them.

To also account for the cost to build the power stations (i.e. the value of each node), we will add on dummy node and connect it to every node in our graph. The weight of the edge from the dummy node to node n_i is v_i, which is the cost to build the power station at location i. Now we can simply run Kruskal's or Prim's algorithm to find the MST in this graph. For all edges in the MST we find, if the edge connects n_i to the dummy node, we will build the station at position i; if the edge connects two nodes n_i, n_j in the original graph, we will build a wire between those location i and j.

For example, if originally we have five locations, and we are given the value v_i and e_{ij}, we can first build the graph on the left hand side. Thereafter, we can add a dummy node as mentioned above and reconstruct the graph to obtain the graph on the right hand side. Then, we can run Kruskal's or Prim's algorithm on the new graph and obtain a MST (drawn in blue). In this case, the best strategy is to build power station at n_1, n_2, n_5 and build wire to connect n_1 with n_3, n_2 with n_4;



$v_1 = 1$
$v_3 = 3$
$v_5 = 5$
$v_2 = 2$
$v_4 = 4$
1
6
2
10
1



n_1
n_3
n_5
n_2
n_4
dummy
1
6
2
10
1
2
4
5
3
1