Timur Garifullin
Due: 11:59pm Thursday, Feb 15

Homework 4

Spring 2024, Bhakta
CS 315, Algorithms

This assignment covers the Bellman-Ford algorithm for shortest paths and Minimum Spanning Trees.

For these questions, you may need to submit graph drawings. You should either

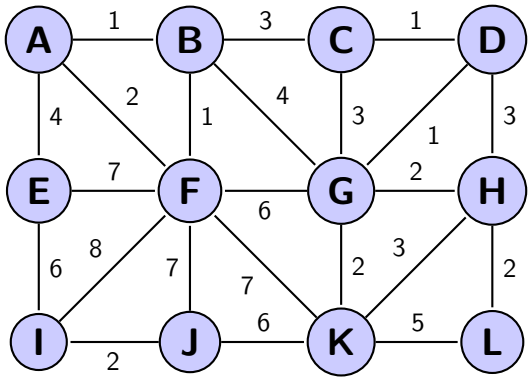
- Learn how to use the TikZ markup to create graph drawings in LaTeX. This isn't too hard (there are examples in this homework), but is harder than the other options. (1 bonus point)
- Draw a graph using other drawing software, and use the includegraphics command to add it to your pdf file. You could also draw a graph on paper, take a picture, and use includegraphics to add the picture to the final pdf document.

0. (0 points) (NOT OPTIONAL) Who did you work with on this homework? What sources did you consult? How long did this assignment take?

Solution: Consulting: Artur Idrissov, Caroline Guza
Source: Class Notes, Office Hours
Time Spent: 14 hours

Homework 4

1. (10 points) Run Prim's algorithm on the following graph, starting at node A. When you have a choice of new vertex to add, choose the one that is alphabetically first, and attach it to its alphabetically first neighbor. List the edges of the final MST in the order that they are added by Prim's Algorithm. What is the weight of the final MST?

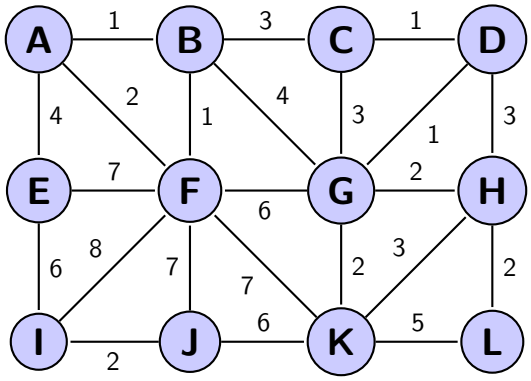


Solution:

Edges of the final MST: [{A,B}, {B,F}, {B,C}, {C,D}, {D,G}, {G,H}, {G,K}, {H,L}, {A,E}, {E,I}, {I,J}]

Weight of the final MST: 1 + 3 + 1 + 1 + 2 + 2 + 2 + 1 + 4 + 6 + 2 = 25

2. (10 points) Run Kruskal's Algorithm on the same graph, also listing the edges in the order that they are added. Confirm that the weight of the final MST is the same.



Solution:

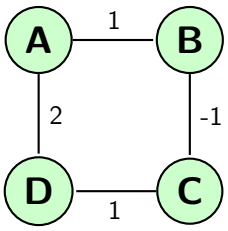
Edges of the final MST: [{A, B}, {C, D}, {B, F}, {D, G}, {G, H}, {I, J}, {G, K}, {H, L}, {B, C}, {A, E}, {E, I}]

Weight of the final MST: 1 + 3 + 1 + 1 + 2 + 2 + 2 + 1 + 4 + 6 + 2 = 25

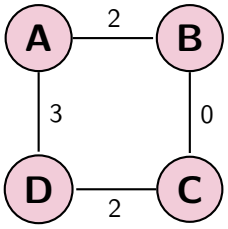
3. (20 points) Rather than deal with the expense of the Bellman-Ford algorithm, Dr. Bhakta wants to find the shortest path tree in graphs by adding a large enough constant c to all edges so that all edges now have positive weight. Dr. Bhakta will then run Dijkstra's algorithm to find the shortest path tree on the graph with these new edge weights.

   Show that his idea won't always work by finding a counterexample.

Solution:
The shortest path from A to D: [(A,B), (B,C), (C,D)]
The total weight of the shortest path: 1 + (-1) + 1 = 1
(Starting node is A and end node is D)



Now we add a constant $c = 1$ to make all the edge weights positive.
We get the following graph:



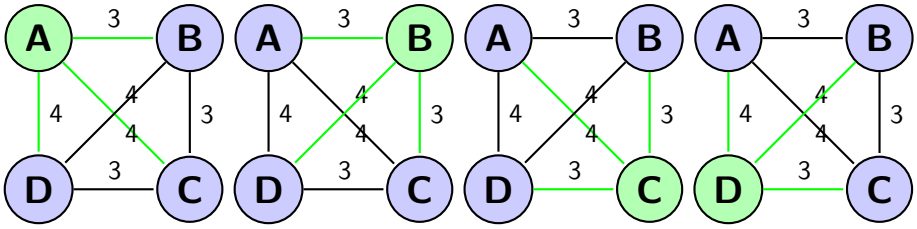
The shortest path from A to D: [(A,D)]
The total weight of the shortest path is now 3.
As can be seen, the path that the algorithm has taken is not the same.

4. (20 points) Find a graph (with non-negative weights) such that the shortest path tree of G generated by Dijkstra's algorithm, no matter which vertex v in G that you start from, is not an MST of G.

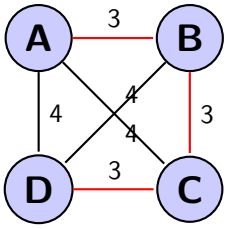
   Hint: Design a small symmetric graph to reduce the number of cases that you have to consider.

Solution: We have a graph with four nodes and edges with weights of three and four.

No matter which vertex v in G we start from, we will NOT be able to get the MST of the G after running Dijkstra's algorithm (path of the algorithm is shown in green) [starting vertex v is shown in green]:



On the other hand, the actual MST of G is shown below (in red):



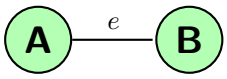
As can be seen, the path of the Dijkstra algorithm is not the MST no matter the vertex Dijkstra started from.

5. (20 points) True/False: This question concerns variants of the MST cut property. Determine if each statement is true or not and give a proof or counter example as appropriate. Please read the language of each statement carefully.

   (a) If e is the unique heaviest edge among the edges that cross some cut of G, then e can't be part of any MST of G.

   Solution:
   This statement is false.

   Here is a counterexample, which shows that the edge e can be part of the following MST of G:

   The edge e is the only edge in this graph, thus it is the heaviest edge among the edges that cross some cut of G. Edge e is part of MST.

   (b) If e is the unique heaviest edge among the edges of some cycle of G, then e can't be part of any MST of G.

   Solution:
   This statement is true.

   We chose to do a proof by contradiction:

   Let e be the unique heaviest edge in the cycle C in a graph G, then e can't be part of any MST of G.

   Assume that e is in the MST T of G.

   There is a cut that crosses e and separates T into two trees $T1$ and $T2$.

   Since there is a cut through e and C is a cycle, there is at least one other edge z that crosses the cut and has a smaller weight than e.

   The MST created using e is greater than the MST created using z: $T1 + T2 + e > T1 + T2 + z$

   Therefore there is a contradiction because the MST with e in it is not the actual MST.

6. (20 points) This question concerns the union-find data structure.

   Recall that the union find data structure maintains a set of rooted trees, called a forest. Here, every node keeps a pointer to its parent, except the roots which have null/none/etc. as their parent pointers.

   The height of one of these trees is the number of edges in the longest path between any node in that tree and the root. The height of a lone root node with no children is 0.

   Everytime two trees are merged together, the root of the tree with the smaller height has its parent set to the root of the tree with larger height. In the event of ties, we can point either root at the other.

   Claim: A tree in a union-find data structure with height n contains at least 2^n nodes.

   Prove this claim by induction on n.

   Hint: Think about how a tree can even first get to height n in the first place.

Solution: A tree in a union-find data structure with height n contains at least 2^n nodes.

We are using proof by induction:
Induction on n, the height of a union-find tree.

Base case:
$n = 0$: There is one node with height 0, so the node has no children. Since the height is 0, this tree contains at least 2^n nodes, $20 = 1$, and is true.

Inductive hypothesis:
Assume true for $n = k$: Assuming that a tree with height k correctly contains at least $2k$ nodes.

Inductive step:
Show true for $n = k{+}1$: We want to prove a tree with $k{+}1$ nodes contains at least $2k{+}1$ nodes.

Let T be a tree with height $k{+}1$. In order to construct T, there are 2 trees both of height k. We know that both of these trees have at least 2* nodes through the inductive hypothesis. Thus, T has at least $2k$ * 2 nodes. Simplifying this expression gives us $2k{+}1$ following Laws of Exponents: When multiplying like bases, keep the base the same and add the exponents.

Conclusion:
Therefore, since we proved a tree with height $k{+}1$ has at least $2k{+}1$ nodes, it must be true that a tree with height k has at least 2* nodes.

7. (Bonus) [10]

   The **Traveling Salesperson Problem (TSP)** is a famous problem in Graph Theory. The problem asks: given weighted, undirected graph $G = (V, E)$, with weights $w(e) > 0$ on each edge e, find a cycle C that visits every vertex of G exactly once and has minimum total weight among all such cycles.

   For now, we'll deal only with **Metric TSP**, where the weight function w satisfies the *triangle inequality*. This means that for all vertices a, b, c, $w(a, c) \leq w(a, b) + w(b, c)$. As a consequence, there must a direct edge between every two vertices (think about why).

   There are no known polynomial time algorithms to solve general or even the metric version of TSP, but we will give an *approximation algorithm*, that finds a cycle of cost at most twice the weight of the optimum cycle C. Remember that even though we don't know what the weight of the optimum cycle is, we can prove that the cycle we return will be less than twice that weight.

   (a) Prove that the MST of the graph has weight strictly less than the optimal TSP cycle C.

   Solution:

   (b) Perform a DFS on the edges of the MST, listing each vertex as you first encounter it. Argue that the cycle formed by taking these vertices in order must have total weight less than $2C$.

   Hint: Imagine "tracing around the boundary" of the MST in the order that the DFS visits the vertices. In this case, list each vertex both as you first encounter it and also when returning from a child's DFS call. In this list, some vertices may appear multiple times.

   You can think of this traversal as a sort of "improper" TSP cycle because it has repeat vertices. Relate the weight of this improper cycle to both the weight of the MST, and the weight of the "proper" TSP cycle that you would be returning.

   Solution: