Name: ______________________________ UH ID: ______________

University of Houston

Homework 4 Solutions

COSC 3320
Algorithms and Data Structures

Due: Tuesday, April 30 2024
11:59 PM

Note

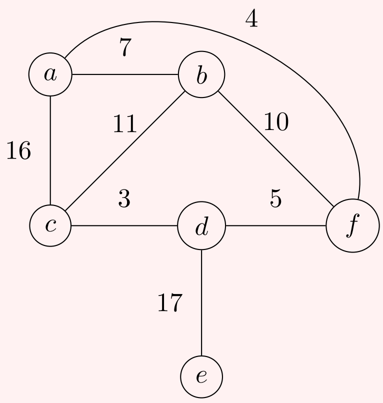
Read the Academic Honesty policy.

Name: ______________________________ UH ID: ______________

1 Exercises

Exercise 1: (MS|SP)T (20 Points)

Consider the graph G given below.



For each of the following, with vertex a as the root, draw the

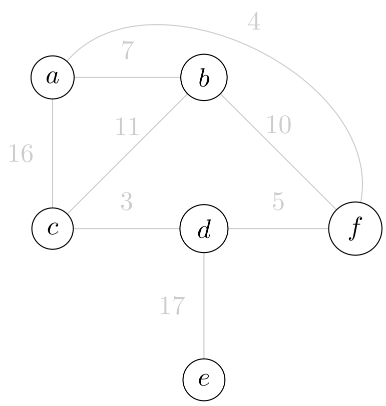
1. minimum spanning tree at each step of
   (a) Kruskal's Algorithm
   (b) Prim's Algorithm
2. shortest path tree at each step of
   (a) Dijkstra's Algorithm
   (b) the Bellman Ford Algorithm

Solution.

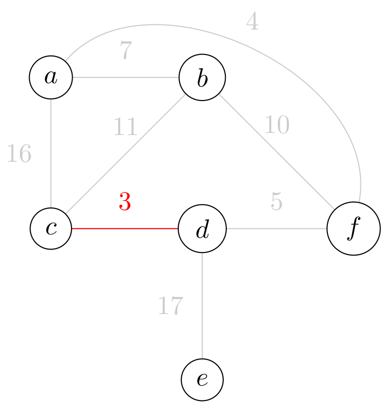
MST

Kruskal's

(1)

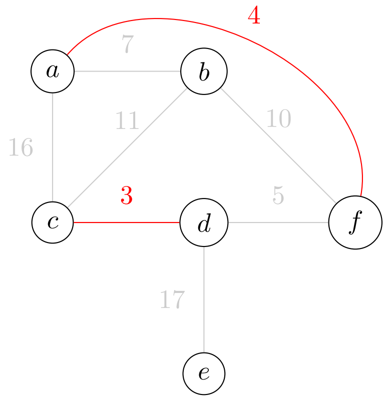


(2)

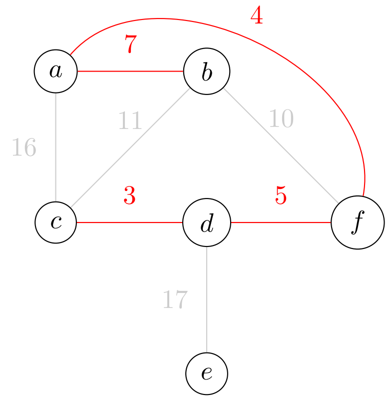


Name: ____________________ UH ID: __________

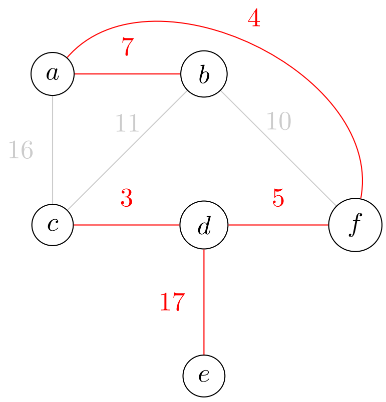
(3)



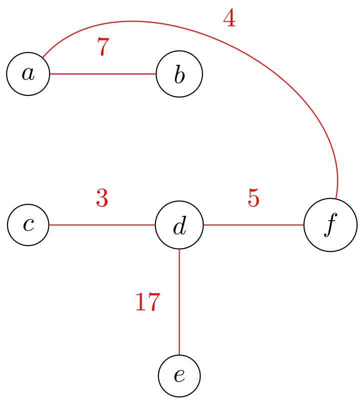
(4)



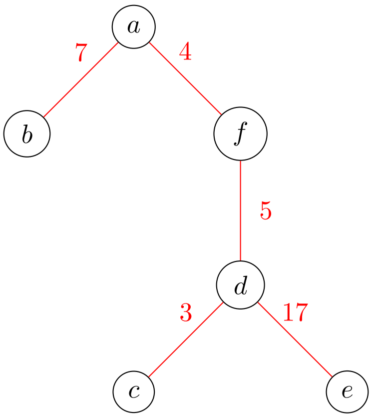
(5)



(6)



(7)

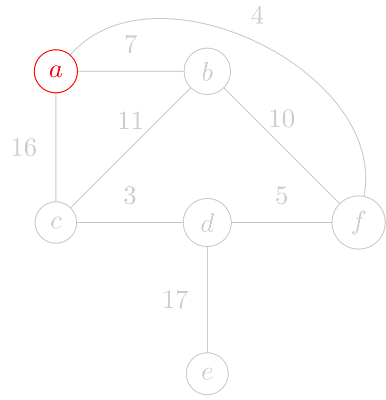


Prim's

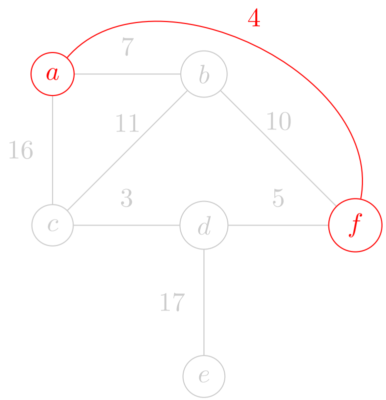
Name: ____________________ UH ID: ____________

(1)

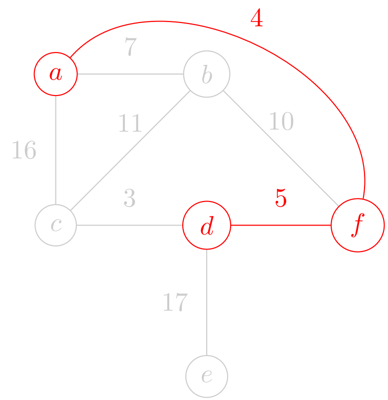
(2)



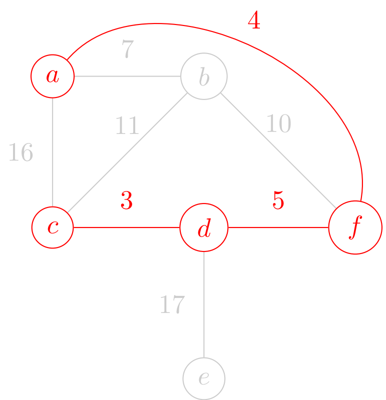
(3)



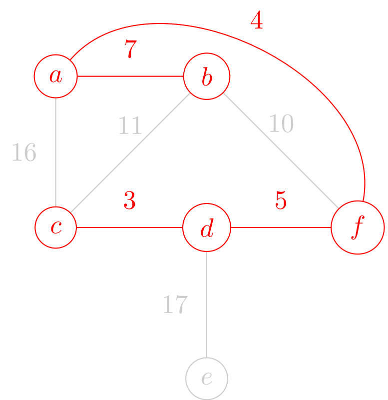
(4)



(5)

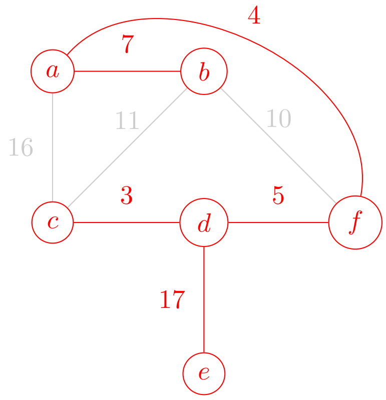


(6)

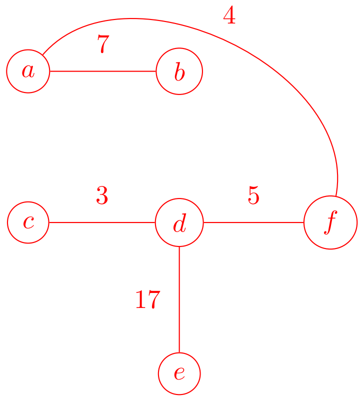


Name: ______________________________ UH ID: ____________

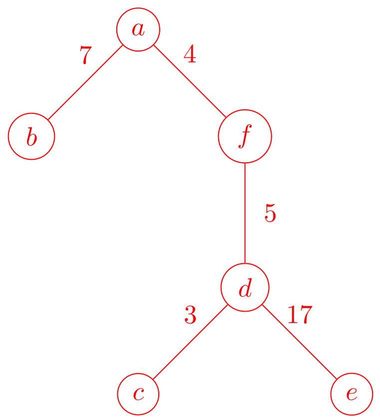
(7)



(8)



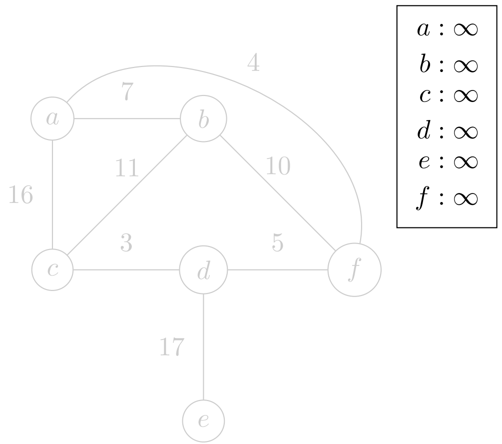
(9)



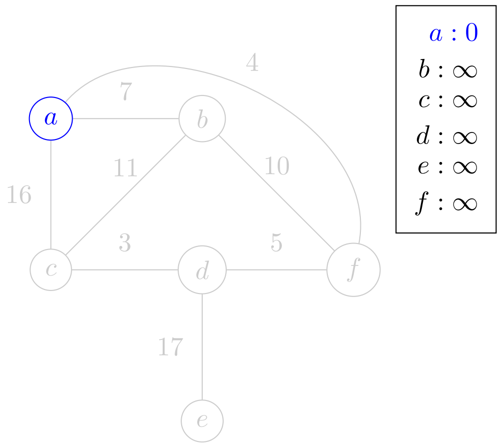
SPT

Dijkstra's

(1)

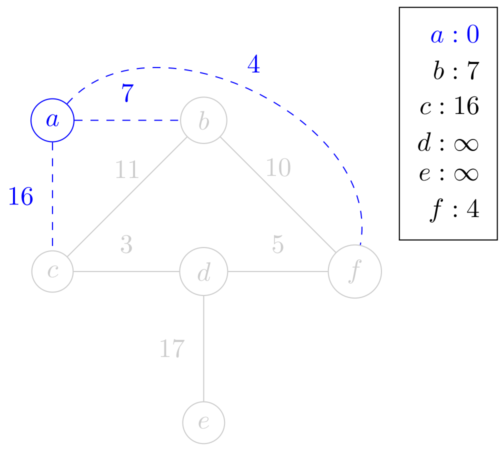


(2)

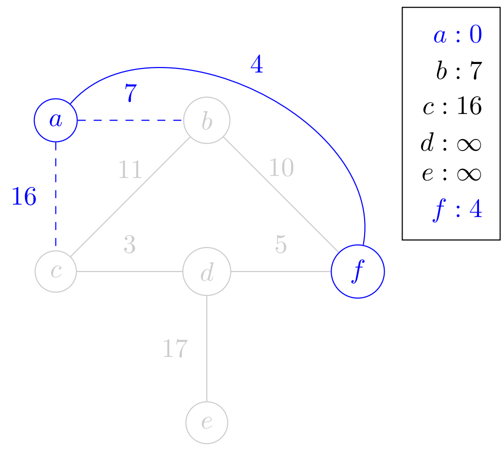


Name: ______________________ UH ID: __________

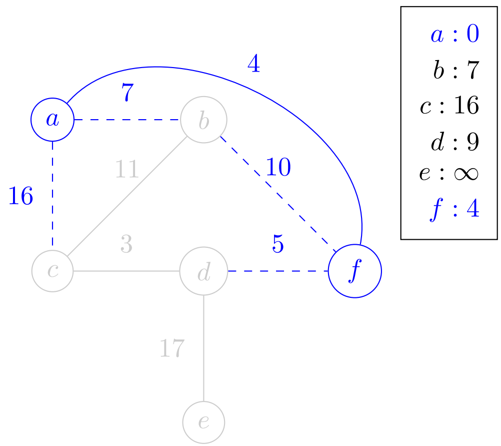
(3)



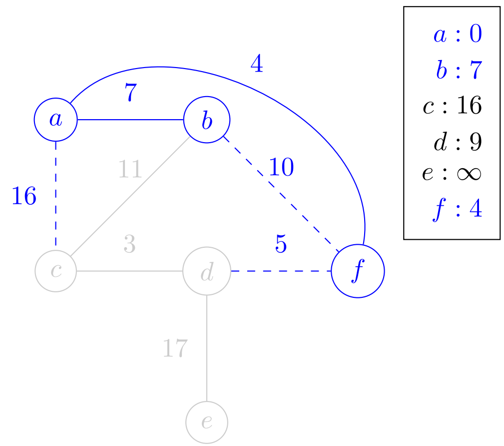
(4)



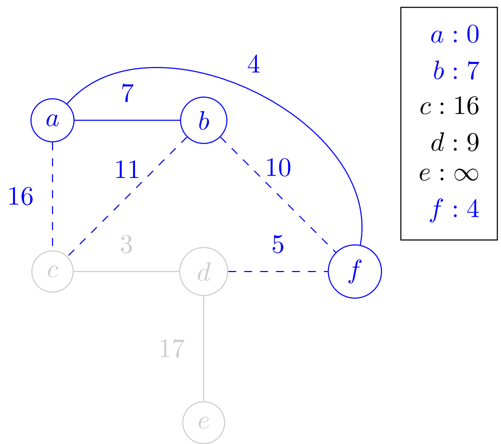
(5)



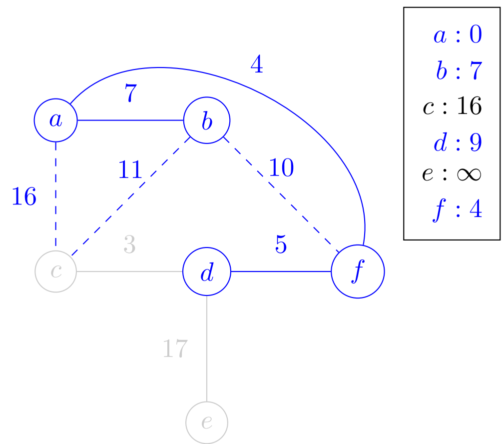
(6)



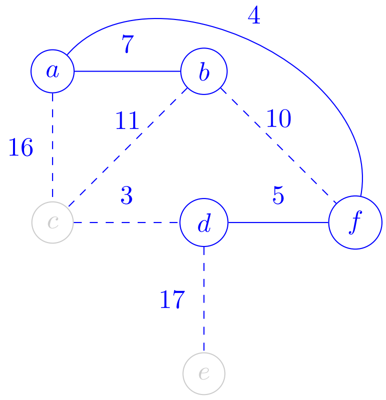
(7)



(8)

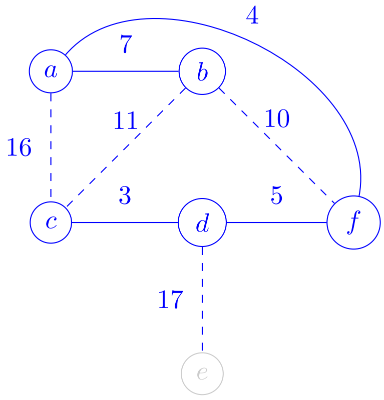


(9)



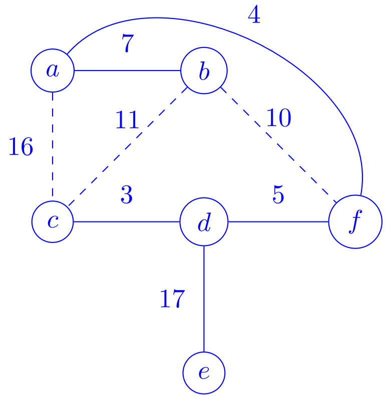
$a : 0$
$b : 7$
$c : 12$
$d : 9$
$e : 26$
$f : 4$

(10)



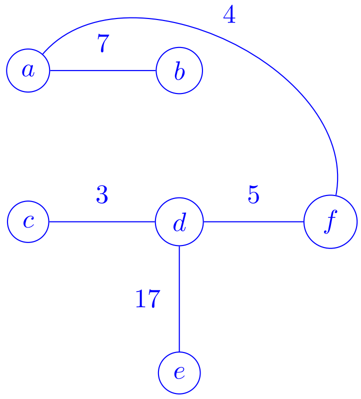
$a : 0$
$b : 7$
$c : 12$
$d : 9$
$e : 26$
$f : 4$

(11)



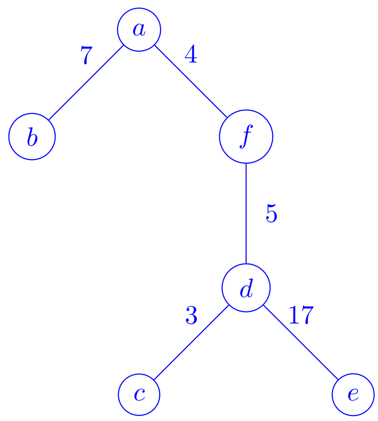
$a : 0$
$b : 7$
$c : 12$
$d : 9$
$e : 26$
$f : 4$

(12)



$a : 0$
$b : 7$
$c : 12$
$d : 9$
$e : 26$
$f : 4$

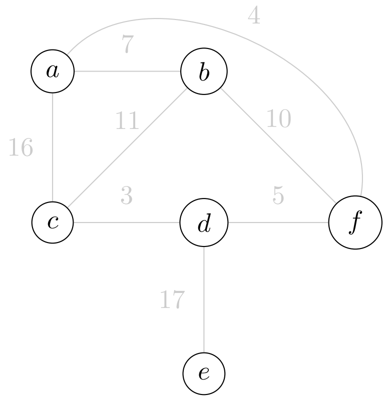
(13)



$a : 0$
$b : 7$
$c : 12$
$d : 9$
$e : 26$
$f : 4$

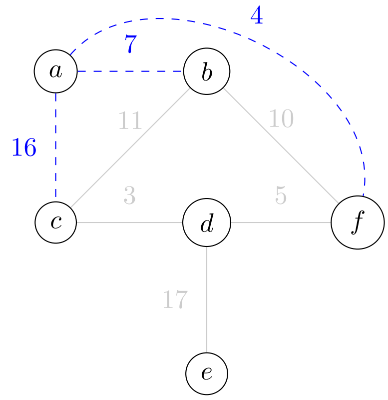
Bellman Ford

(1)



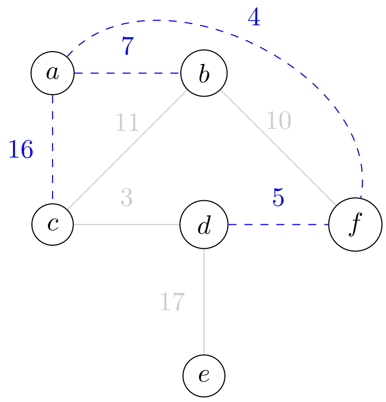
$a : 0$
$b : \infty$
$c : \infty$
$d : \infty$
$e : \infty$
$f : \infty$

(2)



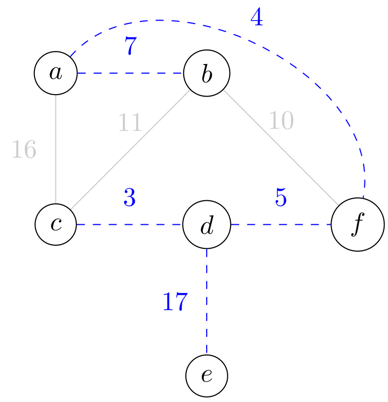
$a : 0$
$b : 7$
$c : 16$
$d : \infty$
$e : \infty$
$f : 4$

(3)



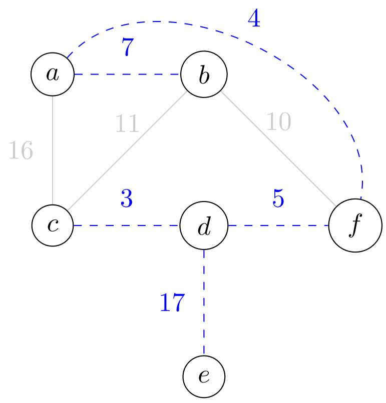
$a : 0$
$b : 7$
$c : 16$
$d : 9$
$e : \infty$
$f : 4$

(4)



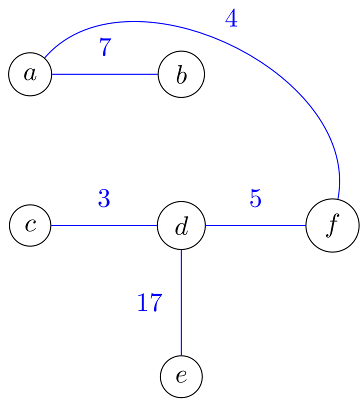
$a : 0$
$b : 7$
$c : 12$
$d : 9$
$e : 26$
$f : 4$

(5)



$a : 0$
$b : 7$
$c : 12$
$d : 9$
$e : 26$
$f : 4$

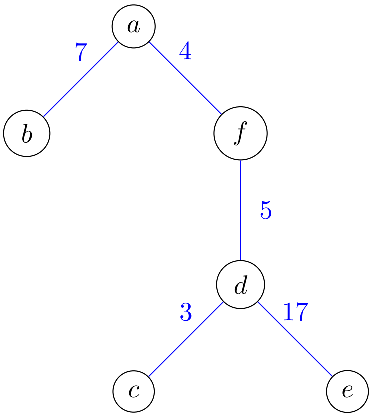
(6)



$a : 0$
$b : 7$
$c : 12$
$d : 9$
$e : 26$
$f : 4$

Name: ______________________________ UH ID: ______________

(7)



$a : 0$
$b : 7$
$c : 12$
$d : 9$
$e : 26$
$f : 4$

$\square$

Name: ______________________ UH ID: ____________

Exercise 2: Cycle Property (20 Points)

The cycle property states that, in any weighted graph, the largest weight edge of any cycle belongs to *no* MST[1].

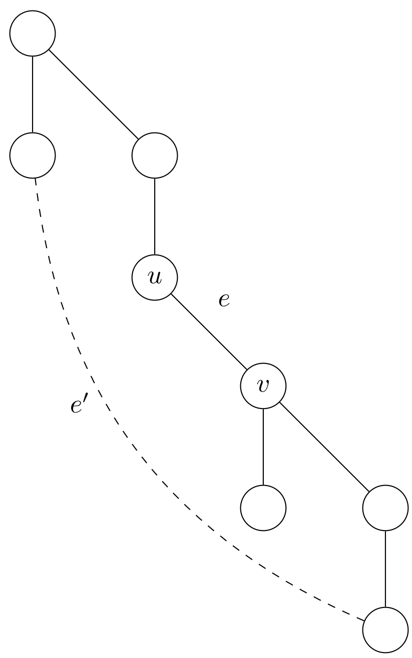
1. Prove the cycle property for an undirected graph.
2. Devise an algorithm to output the MST of a graph using the cycle property.
3. Implement your algorithm in pseudocode and analyze the runtime. You may assume you have a function IS-PART-OF-CYCLE(e), based on DFS, which outputs `True` if e belongs to a cycle and `False` otherwise in $\mathcal{O}(m+n)$ time[2].

[1] Strictly speaking, this is only true if the largest weight is *strictly* greater than the remaining edges in the cycle. More generally, if e is the largest weight edge in any cycle, then there exists at least one MST that does not contain e.

[2] Though you are welcome to change the function signature as desired.

Solution.

1. Consider some spanning tree T containing the maximum weight edge $e = (u, v)$ on some cycle C — if we remove e from T, we now have two trees, T_1, and T_2, with $u \in T_1$ and $v \in T_2$. Now, since T is a tree, it cannot contain all edges in C. In particular, since there are two paths from u to v (one directly through edge (u, v), and one via the remaining edges in C), there must be at least one edge e' in C that crosses the cut (T_1, T_2) and has strictly smaller weight than e.



   Taking $T' = T_1 \cup e' \cup T_2$ creates a new spanning tree with strictly lower weight T. Thus, T cannot be an MST.

2. Simply iterate through edges in decreasing order of weight and check if an edge belongs to a cycle (or, equivalently, if the edge's removal fails to disconnect the graph). If so, remove the edge, since it belongs to no MST. Repeat until there are $n - 1$ edges remaining

3. We iterate through m edges and each time check connectivity, for a total runtime of

$$\mathcal{O}(m(m+n)) = \mathcal{O}(m^2 + mn) = \mathcal{O}(m^2)$$

```
for each edge e in decreasing order of weight:
    if IS-PART-OF-CYCLE(e):
        delete edge e
```

```
return remaining edges
```

□

Exercise 3: Counting Shortest Paths (20 Points)

Let G be an undirected, weighted, connected graph, with no negative edge weights. Given two nodes s and t in the graph, count the number of *distinct* shortest paths between them. Give an efficient algorithm for the problem, argue its correctness, and analyze its run time.

(Hint: Modify Dijsktra's algorithm to count the number of shortest paths from s to t. When u is popped from the heap, consider the case where the shortest path to a neighbor of u is updated, and the case where it is not.)

Solution. Let $d(u)$ denote the shortest distance from s to u in G and say v is a shortest-path vertex of u, denoted $v \in \textsc{spv}(u)$, if v is the last node visited before u on any shortest path from s to v. Let $\textsc{Num-Paths}(u)$ denote the number of shortest paths from s to u. Clearly,

$$\textsc{Num-Paths}(u) = \sum_{v \in \textsc{spv}(u)} \textsc{Num-Paths}(v)$$

and $v \in \textsc{spv}(u)$ if and only if

$$d(u) = d(v) + w(u, v)$$

From this, we have a straightforward modification of Dijkstra's algorithm: let $\texttt{num-paths}[u]$ denote the number of shortest paths to v. Initially, $\texttt{num-paths}[u] = 0$ for all $u \neq s$ and $\texttt{num-paths}[s] = 1$. When popping a node u from the heap, for each neighbor $\texttt{nbr}$ of u, if we update $\texttt{dist}[\texttt{nbr}]$ i.e., if

$$\texttt{dist}[\texttt{nbr}] > \texttt{dist}[u] + w(u, \texttt{nbr})$$

then we set

$$\texttt{num-paths}[\texttt{nbr}] = \texttt{num-paths}[u]$$

If the distance to $\texttt{nbr}$ through u is *the same* as the current distance to $\texttt{nbr}$, i.e., if

$$\texttt{dist}[\texttt{nbr}] == \texttt{dist}[u] + w(u, \texttt{nbr})$$

then we have found a new shortest path for each shortest path to u, and we set

$$\texttt{num-paths}[\texttt{nbr}] \mathrel{+}= \texttt{num-paths}[u]$$

Finally, if the distance to $\texttt{nbr}$ through u is greater, then we do nothing, as this is not a shortest path.

It is clear that this modification does not change the asymptotic complexity of Dijsktra's algorithm, and thus has runtime $\mathcal{O}(m \log n)$. To see that it is correct, we maintain similar invariants to the proof of correctness of Dijkstra's algorithm — at each "phase":

- if v is in the tree, then
  - $\texttt{dist}[v]$ is the weight of the shortest path from s to v in G
  - $\texttt{num-paths}[v]$ is the number of shortest paths from s to v in G
- if v is not in the tree, then
  - $\texttt{dist}[v]$ is the weight of the shortest path from s to v using *only* tree-edges

Note that the invariants for $\texttt{dist}$ have already been shown in the proof of correctness for Dijkstra's algorithm.

The proof for the $\texttt{num-paths}$ invariant is by induction: We will show that the invariant holds at the end of every phase. Clearly, it holds at the start of the algorithm. Assume it holds for the first $i-1$ phases. Then, for the i-th phase, consider the node u popped from the heap. Now, note that, by construction:

$$\texttt{num-paths}[u] = \sum_{v \in T \cap \textsc{spv}(u)} \texttt{num-paths}[v]$$

From here, it suffices to show that $\textsc{spv}(u) \subseteq T$, i.e., that every shortest-path vertex is already in the tree. In fact, this must be the case: if a vertex v satisfies

$$\texttt{dist}[u] = \texttt{dist}[v] + w(u, v)$$

then $\texttt{dist}[v] \leq \texttt{dist}[u]$, for otherwise v would have been popped instead of u. Thus, by our induction hypothesis, $\texttt{num-paths}[v]$ is the number of shortest paths to v for every $v \neq u$ in T. From this, we conclude that

$$\begin{aligned}
\texttt{num-paths}[u] &= \sum_{v \in T \cap \textsc{spv}(u)} \texttt{num-paths}[v] \\
&= \sum_{v \in \textsc{spv}(u)} \texttt{num-paths}[v] \text{ since } \textsc{spv}(u) \subseteq T \\
&= \sum_{v \in \textsc{spv}(u)} \textsc{Num-Paths}(v) \text{ by our induction hypothesis} \\
&= \textsc{Num-Paths}(u)
\end{aligned}$$

is the number of shortest paths to u. $\square$

Exercise 4: Minimum Spanning Tree (20 Points)

Solve the problem Min Cost to Connect All Points at LeetCode using:

1. Kruskal's Algorithm
2. Prim's Algorithm

Submit links to the submission results for each.