COMP 182: Algorithmic Thinking
Homework 5: Description for Problems 2 and 3
Luay Nakhleh

1 Weighted Graphs and Their Minimum Spanning Trees

We have learned about *minimum spanning trees* (MST) of weighted, undirected graphs. In this homework, we will learn about an analogous structure, but for weighted, directed graphs. We begin with a definition.

Definition 1 *Let $g = (V, E)$ be a directed graph, and let $r \in V$ be a distinguished node that we call the root. A rooted, directed, spanning tree (RDST) of g is a subgraph $T = (V, E')$ of g (that is, $E' \subseteq E$) such that two conditions hold:*

1. *If we ignore the directions of the edges in E', then T is a spanning tree of g; and,*
2. *If we take the directions of the edges in E' into account, then there is a (directed) path from the root r to every node $v \in V \setminus \{r\}$.*

The RDST of a directed graph is not necessarily unique. Figure 1(a) shows a directed graph and the designated root r, along with two RDST's of it in panels (b) and (c).

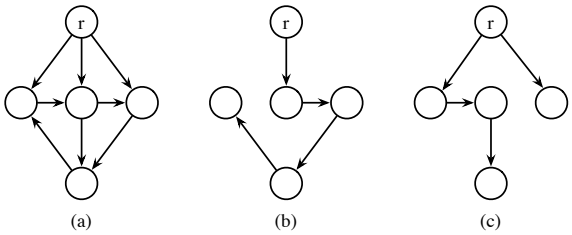

Figure 1: A directed graph (a) and two RDSTs in (b) and (c).

In the case of undirected graphs, a graph g has an MST if and only if g is connected. Analogously, a directed graph g has an RDST rooted at r if and only if there is a directed path from r to every other node in g. Thus, finding an RDST can be solved using, for example, **DFS**. However, the problem becomes more challenging when the graph is weighted and the goal is to find an RDST of smallest weight. The weight of an RDST in a weighted graph is the sum of the weights of the edges in the RDST.

Definition 2 *Let $g = (V, E, w)$ be a weighted, directed graph with weight function $w : E \to \mathbb{R}^+$ (here $\mathbb{R}^+$ is the set of all non-negative reals) and root node r. A rooted, directed minimum spanning tree, or RDMST, is an RDST of g that has the smallest weight among all possible RDST's of g.*

Figure 2(a) shows a weighted, directed graph and the designated root r, along with the RDMST of the graph (in this case, the RDMST is unique, but this does not have to be the case in general).

We are interested in solving the ROOTED, DIRECTED MINIMUM SPANNING TREE PROBLEM, or the RDMST PROBLEM for short, which is defined as follows.

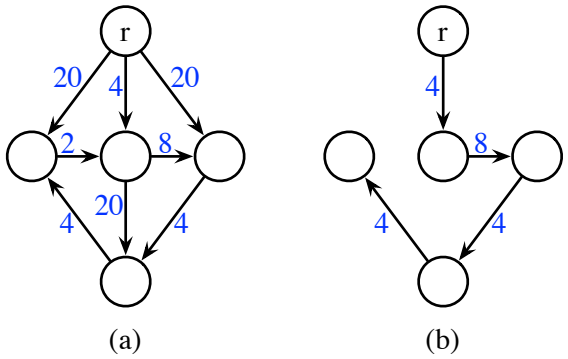

Figure 2: A weighted, directed graph (a) and its RDMST in (b).

Definition 3 *The* RDMST PROBLEM

Input: *A weighted, directed graph $g = (V, E, w)$, with $w : E \to \mathbb{R}^+$, and a root node $r \in V$.*

Output: *An RDMST $T = (V, E')$ of g rooted at node r.*

Notice that the greedy approach of Prim's and Kruskal's algorithms does not work here, since the edge with the smallest weight is not necessarily part of the RDMST (e.g., see Figure 2). Indeed, while the algorithm for solving the RDMST PROBLEM that we develop here has elements of greedy algorithms, it is more involved.

2 Designing the Algorithm for the RDMST Problem

In this part, we assume that the root is a node in the digraph, and that the digraph has a path from the root to every other node in it. This applies to all lemmas, the theorem, and the algorithm.

We will solve the RDMST PROBLEM by establishing a set of lemmas and putting them together to yield a correct and efficient algorithm for the problem. Hereafter, we will assume that the weights of edges are non-negative reals. Further, for a given weighted, directed graph $g = (V, E, w)$, we will make use of two notations

- For a node $u \in V$, we denote by $m(u)$ the minimum weight of an edge whose head is node u.
- For a node $u \in V$, we denote by $me(u)$ an edge whose head is node u and whose weight is $m(u)$. If $me(u)$ is not unique, we assume that the edge is chosen arbitrarily.

Lemma 1 *Let $g = (V, E, w)$ be a weighted, directed graph with designated root $r \in V$. Let*

$$E' = \{me(u) : u \in (V \setminus \{r\})\}.$$

Then, either $T = (V, E')$ is an RDMST of g rooted at r or T contains a cycle.

Proof: Complete it as part of Homework 3. □

What Lemma 1 states is that if we take a set that consists of a lightest edge per node (except for the root r), then that set is either an RDMST or, if it isn't, then it must contain a cycle. Put differently, if the set E' forms an RDST, then that RDST is also an RDMST. If the set E' is not an RDST, then it contains a cycle. From an algorithmic perspective, we use Lemma 1 as follows: The algorithm computes the set E' and if it forms an RDST, the algorithm returns it as a solution to the RDMST PROBLEM. If E' is not an RDST, then it makes use of the next set of results. The first result is that we modify the weights of g's edges in a specific way, then T is an RDMST of g if and only if it is an RDMST of g with the modified edge weights. How we make use of this result in designing the algorithm becomes clear later.

Lemma 2 *Let $g = (V, E, w)$ be a weighted, directed graph with designated root $r \in V$. Consider the weight function $w' : E \to \mathbb{R}^+$ defined as follows for each edge $e = (u, v)$:*

$$w'(e) = w(e) - m(v).$$

Then, $T = (V, E')$ is an RDMST of $g = (V, E, w)$ rooted at r if and only if T is an RDMST of $g = (V, E, w')$ rooted at r.

Proof: Complete it as part of Homework 3. □

This lemma states that if for all edges incoming into a node v we subtract the minimum weight of an edge incoming into v, then we obtain a new weighted graph whose RDMST is necessarily an RDMST of the original graph. Figure 3(b) shows the result of applying the edge weight modification based on Lemma 2 to the graph in Figure 3(a). You can verify for yourself that an RDMST of the weighted, directed graph in Figure 3(a) is also an RDMST of the weight, directed graph in Figure 3(b).

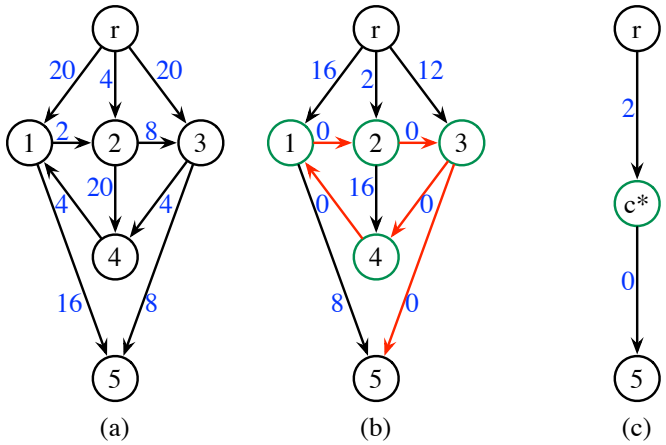

Figure 3: (a) A weighted, directed graph $g = (V, E, w)$ with root r. (b) The graph $g = (V, E, w')$ obtained by modifying the edge weights according to Lemma 2. The red edges form the set E' that is computed based on Lemma 1, and the green circles denote the nodes on the cycle C in the subgraph (V, E'). (c) The graph that results from contracting the cycle C in the graph of panel (b). While the three edges $(r, 1)$, $(r, 2)$ and $(r, 3)$ all enter the cycle C, the edge (r, c^*) is based on edge $(r, 2)$ because it has the smallest weight of all edges entering the cycle. Similarly, while two edges exit the cycle C and enter node 5, the edge $(c^*, 5)$ is based on the edge $(3, 5)$ because it has the smallest weight of all edges exiting the cycle *and* entering node 5. Notice that we do not allow self-loops, so the edge (c^*, c^*) must not be added when the cycle is contracted.

It is important here to notice that if we make use of Lemma 2 (that is, modify the weights of the edges of g as described by the lemma) and then consider the set E' that is built by Lemma 1, then the weight of every edge in E' is 0 (the red edges in Figure 3(b)). In particular, if the set E' contains a cycle, then the weight of the cycle is 0 (the cycle is formed by the green nodes in Figure 3(b)). This means that we can use as many edges from the cycle (contained in E') as we want, since including those edges does not affect the total weight (each contributes a weight of 0).

The final result that we need to design the algorithm is the following.

Lemma 3 *Let $g = (V, E, w)$ be a weighted, directed graph whose edge weights have been modified based on Lemma 2 and that has an RDST rooted a designate node $r \in V$. Let C be a cycle in g that consists of edges whose weights are 0 and $r \notin C$. Then, there exists an RDMST of g rooted at r that has exactly one edge entering C.*

Proof: Let T be an RDMST of digraph g rooted at node r. By definition of an RDMST, there are paths in T from the root r to every node $u \in V \setminus \{r\}$. In particular, since the cycle consists of nodes in $V \setminus \{r\}$, the RDMST T must have at least one edge that enters the cycle C (here, we say that an edge (u, v) enters the

cycle C if $v \in C$, and it exits the cycle if $u \in C$). If there is exactly one edge in T the enters the cycle, then the result is established and we are done.

Now, assume that at least two edges in T enter cycle C. We show how to modify T to obtain an RDST T' that (1) enters C exactly once, and (2) has a total weight no greater than the total weight of T. Let $e = (u, v)$ be the lightest edge that satisfies: u is reachable from the root r and $v \in C$ (if more than one such edge exists, choose one arbitrarily). We delete from T all edge of T that enter C, except for the edge e, and add in all the edges of C (that is, edges whose both endpoints are in C) except for the edge whose head is v. Let T' be the resulting subgraph of g.

First, notice that the total weight of T' is no greater than the total weight of T, since the only edges that belong to T' but not to T have weight 0 (they are on the cycle). We only need to show that T' is an RDST. Observe that for every node in g, except for r, there is exactly one edge in T' that enters that node (by construction of T'). Therefore, all that is left for us to show is that there is path from r to every other node in the digraph. Let $v \in V \setminus \{r\}$ be an arbitrary node in digraph g. If $v \in C$, then there is a path from r to v in T', since there was a path from r to v in T and the construction of T' maintains a path from r to all the nodes on the cycle. Now, consider a node $v \in V \setminus \{r\} \setminus C$, and let p be the path from r to v in T. If no node on p is in C, then p is also a path in T'. However, if p intersects with C (that is, some nodes on p are also in C), let y be the last (closest to v) node on the path p that is also a node in C. By construction of T', a path from y to v exists in T'. Since we've already argued that all nodes on the cycle are reachable from r, then y is reachable from r. Therefore, v is reachable from r (through the node y). This completes the proof. □

For example, in Figure 3(b), the RDMST has exactly one edge that enters the cycle C, which is edge $(r, 2)$. Since we can use as many of a cycle's edges as we want (discussed above), we can now put all these results together to produce the general idea of Algorithm **ComputeRDMST** which takes as input a digraph $g = (V, E, w)$ and root r (and g is assumed to have an RDST rooted at r; the algorithm does not need to check this) and returns an RDMST of g.

Algorithm ComputeRDMST

Input: Weighted digraph $g = (V, E, w)$ and node $r \in V$.

Output: An RDMST T of g rooted at r.

1. Modify the weights of all edges in g (except for those entering r) according to Lemma 2, resulting in $g = (V, E, w')$.
2. Compute the digraph $T = (V, E')$ where E' is computed by Lemma 1.
3. If T forms an RDST, then return it.
4. Else, T contains a directed cycle C (C is the set of nodes on the cycle) that does not involve r. In this case,
   (a) Construct a graph $g^* = (V^*, E^*, w^*)$ as follows:
      i. $V^* \leftarrow V \setminus C \cup \{c^*\}$. That is, remove all nodes on the cycle and replace them by a single node, c^*.
      ii. For every edge $(u, v) \in E$:
         - If $u \notin C$ and $v \notin C$: add (u, v) with weight $w^*(u, v) = w'(u, v)$ to E^*.
         - If $u \notin C$ and $v \in C$: add (u, c^*) with weight $w^*(u, c^*) = w'(u, v)$ to E^*.
         - If $u \in C$ and $v \notin C$: add (c^*, v) with weight $w^*(c^*, v) = w'(u, v)$ to E^*.
      iii. If there are parallel edges from a node u to a node v in E^*, remove all of them except for one with the smallest weight among those parallel edges.
   (b) Compute (recursively) an RDMST T' of g^*.
   (c) Expand T' back to an RDMST T of the original graph g: (1) replace c^* by the set of nodes in C; (2) replace the edge (u, c^*) in T' by the single edge (u, v^*) in E that was the "origin" of edge (u, c^*); (3) replace every edge (c^*, v) by the single edge in E that was the origin of edge (c^*, v); and (4) add every edge whose two endpoints are on the cycle C, except for the edge incoming into node v^*. The correctness of this step follows from Lemma 3.
   (d) Return T.

Figure 3(c) shows the graph $g^* = (V^*, E^*, w^*)$ that results from contracting the cycle denoted by the green circles (nodes) of the graph in Figure 3(b). When **ComputeRDMST** is called recursively on this graph g^*, it returns T' whose nodes are $\{r, c^*, 5\}$ and whose edges are $\{(r, c^*), (c^*, 5)\}$. Now this tree, T', is expanded back so that it includes the original nodes ($\{1, 2, 3, 4\}$) and excludes c^*. Based on Step 4(c) of the algorithm, this expansion is done as follows:

- The node c^* is replaced by the set $\{1, 2, 3, 4\}$ of nodes; thus, the new set of nodes in T' is $\{r, 1, 2, 3, 4, 5\}$.
- The origin of edge (r, c^*) is the edge $(r, 2)$; thus, edge (r, c^*) is replaced by edge $(r, 2)$ in T'.
- The origin of edge $(c^*, 5)$ is the edge $(3, 5)$; thus, edge $(c^*, 5)$ is replaced by edge $(3, 5)$ in T'.
- The four edges on the cycle, except for the one incoming into node 2 (which is node v^* in Step 4(c) of the algorithm), are added: $(2, 3)$, $(3, 4)$, and $(4, 1)$.

Thus, the final RDMST of the graph has the following set of edges: $\{(r, 2), (2, 3), (3, 4), (3, 5), (4, 1)\}$.

Using three three lemmas we established above, we can prove that the algorithm is correct.

Theorem 1 *Let $g = (V, E, w)$ be a weighted, directed graph that has an RDST rooted at node $r \in V$. Then, Algorithm* **ComputeRDMST** *computes an RDMST of g rooted at r.*

Proof: Let $n = |V|$. We prove the result by (strong) mathematical induction on n. For the base case of $n = 1$, the graph has one node (assume it is the root) and the algorithm returns the node itself as the RDST, which is also the RDMST (of weight 0).

For the inductive step, assume that the algorithm correctly computes an RDMST of graphs with $|V| < n$ nodes and consider a graph g with n nodes. If the RDST computed in Step 2 of the algorithm is a tree, then it is an RDMST, by Lemma 1. Otherwise, we consider the graph with the modified weights, which is equivalent to g by Lemma 2. Let g^* be the digraph that results from contracting cycle C in g. The number of nodes in g^* is $< n$. Then, by the inductive hypothesis, **ComputeRDMST** computes the RDMST of g^*. Then, by Lemma 3, it follows that an RDMST of g can be obtained from that of g^*. □

Designing the algorithm above in a great example of how modular thinking allows us to break down the problem into smaller steps, each of which can be solved independently and then these solutions are combined to yield the main algorithm. Indeed, each line in Algorithm **ComputeRDMST** translate into a single function call in the Python implementation of function `compute_rdmst` shown below. Implementing each of the individual functions that are called within `compute_rdmst` is the subject of Problem 2 on the homework.

Weighted Digraph Representation in Python. The standard representation of a weighted digraph is a dictionary where the value for each key i is a dictionary whose keys are the nodes to which there are edges from i and whose values are the weights of the edges. For example, the standard representation of the graph g in Figure 3(a) is (node r is numbered 0 here):

```
g = {0: {1: 20, 2: 4, 3: 20}, 1: {2: 2, 5: 16}, 2: {3: 8, 4: 20}, 3: {4: 4, 5: 8},
       4: {1: 4}, 5: {}}
```

For some of the functions that implement parts of Algorithm **ComputeRDMST**, it is easier to work with a reversed representation where the value for each key i is a dictionary whose keys are the nodes from which there are edges to i and whose values are the weights of the edges. The reversed representation of the graph g in Figure 3(a) is:

```
g = {0: {}, 1: {0: 20, 4: 4}, 2: {0: 4, 1: 2}, 3: {0: 20, 2: 8}, 4: {2: 20, 3: 4},
       5: {1: 16, 3: 8}}
```

```
def compute_rdmst(graph,root):
    """
        Computes the RDMST of a weighted digraph rooted at node root.
        It is assumed that:
            (1) root is a node in graph, and
            (2) every other node in graph is reachable from root.

        Arguments:
        graph -- a weighted digraph in standard dictionary representation.
        root -- a node in graph.

        Returns:
        An RDMST of graph rooted at root. The edge weights do not have to be
        the original weights.
    """

    # reverse the representation of graph
    rgraph = reverse_digraph_representation(graph)

    # Step 1 of the algorithm
    modify_edge_weights(rgraph, root)

    # Step 2 of the algorithm
    rdst_candidate = compute_rdst_candidate(rgraph, root)

    # compute a cycle in rdst_candidate
    cycle = compute_cycle(rdst_candidate)

    # Step 3 of the algorithm
    if not cycle:
        return reverse_digraph_representation(rdst_candidate)
    else:
        # Step 4 of the algorithm

        g_copy = copy.deepcopy(rgraph)
        g_copy = reverse_digraph_representation(g_copy)

        # Step 4(a) of the algorithm
        (contracted_g, cstar) = contract_cycle(g_copy, cycle)
        #cstar = max(contracted_g.keys())

        # Step 4(b) of the algorithm
        new_rdst_candidate = compute_rdmst(contracted_g, root)

        # Step 4(c) of the algorithm
        rdmst = expand_graph(reverse_digraph_representation(rgraph),
                             new_rdst_candidate, cycle, cstar)

        return rdmst
```

3 Elucidating the Transmission of a Bacterial Infection

The development of antibiotics is one of medicine's biggest success stories in the 20th century. These compounds have helped control bacterial infection and save millions of lives. However, an unfortunate outcome of the use and abuse of antibiotics is the emergence of antibiotic-resistant bacteria (often referred to as "superbugs"). In fact, this emergence is an example of "evolution in action," where bacteria mutate and

new forms that are resistant to antibiotics become more fit. And, unfortunately, hospitals tend to contribute (unintentionally, of course) to this problem.

In 2011, an outbreak of an antibiotic-resistant bacteria, *Klebsiella pneumoniae*, infected 18 patients at a hospital and caused the death of six of them, all within about six months. When the first patient showed up at the hospital, it was known that she was infected with the bacteria. Therefore, strict isolation and control procedures were followed by the hospital staff to make sure the infection did not spread to other patients. Indeed, no other infections with the bacteria were observed at the hospital for the month that the patient was there. However, a few weeks later, a new patient was found to be infected, followed by a third patient, and, eventually, a total of 18 patients were infected, six of whom died. The question of interest became: How did the infection spread among the patients? The infection transmission was especially perplexing because, for example, the first patient did not come into contact with any of the other patients. Further, the bed locations of the patients did not seem to provide any clues on how the infection was transmitted.

To answer this question, researchers followed the algorithmic thinking process that you learn in this course. The researchers obtained bacterial isolates from all 18 patients and sequenced the genomes of these isolates; this is what we refer to here as the *genetic data*. Further, they obtained data on the location and overlap of the 18 patients during the outbreak. In particular, they obtained a timeline of first positive cultures of the bacterial strain for the 18 patients, as well as other data points; this is what we refer to here as the *epidemiological data*.

Based on the genetic data, the researchers computed pairwise genetic distances, where the genetic distance, G_{AB}, between two genomes A and B (you can think of it as the genetic distance between the patients from whom the bacterial genomes A and B were obtained) is the number of positions at which the two genomes differ. This is known as the Hamming distance. For example, if the genome A is '00101' and genome B is '10100', then the genetic, or Hamming, distance between these genomes is 2 since they differ only at their first and last positions (it is required that the genomes are of the same length). Further, the researchers computed pairwise epidemiological distances, where the epidemiological distance between patient A and patient B is

$$E_{AB} = \max \begin{cases} \max(E) & \text{No transmission opportunity} \\ D(A) + R(B) & \text{otherwise} \end{cases}$$

where D is the minimum number of days of silent colonization in the donor (here, patient A is the donor) and R is the minimum number of days of silent colonization in the recipient required for transmission event to have occurred. Here, $\max(E)$ denotes the maximum finite distance in the matrix E.

Finally, given that genetic and epidemiological distances between every pair of patients could be computed, the researchers built a complete weighted graph, where the weight of the edge (A, B), where A and B are patients is

$$D_{AB} = G_{AB} + \left(\frac{999 \cdot \left(\frac{E_{AB}}{\max(E)} \right)}{10^5} \right). \tag{1}$$

Using this weighted digraph, the researchers decided that the most probable transmission map of the infection outbreak is a tree whose root is the first patient, that spans all patients, and that has the smallest total combined genetic-epidemiological weight (that is, weight based on D). In other words, the researchers sought the RDMST of the weighted graph rooted at the node that corresponds to the first patient.

In Problem 3, you will implement this process and analyze the actual data that was obtained from patients. Notice that the sequences in the genetic data are all encoded as 0's and 1's (rather than nucleotides). You are provided with

- a function that reads the genetic data from an input file;
- a function that reads the epidemiological data from an input file;
- a function that computes the pairwise epidemiological distances; and,
- a function that computes the pairwise genetic distances, but you need to implement a function that computes the Hamming distance between two strings.

Your task will be to write the remaining functions, infer the transmission map on the actual data, and discuss your findings.