Homework #5

Arman Jasuja

May 27, 2023
Collaborators: Ritvik Gunturu

1. Let G be a weighted, undirected graph whose edges have distinct weights. Show that G has a unique minimum spanning tree.

   Proof (By Contradiction): In a finite graph, given a minimum spanning tree, T (which we generate by Kruskal's algorithm), we assume that $\exists\, T'$ s.t.

   $$\sum_{e \in E(T)} w(e) = \sum_{e \in E(T')} w(e)$$

   where w is the weight of each edge. Note that $E(T) \neq E(T')$. Let T' be the particular graph s.t. $|E(T) \cap E(T')| = \max$. Now, we consider a particular edge $e_i \in E(T)$, which is the first edge that is not shared between T and T' (i.e. they both share $(e_0, \ldots e_{i-1}) \in E(T)$), and consider a corresponding edge $e_* \in E(T')$. As T is a minimum spanning tree, we pick e_* s.t. $e_* \leq e_i$, and, from the fact that the edge weights are distinct, we get $e_* < e_i$. Now, we shall construct a new tree, called T^*, where $T^* = T' + e_i - e_*$. Then, we get

   $$\sum_{e \in E(T^*)} w(e) < \sum_{e \in E(T)} w(e)$$

   which contradicts our assumption that T is a minimum spanning tree. $\therefore$ the claim is proved. ■

2. Let G be a weighted directed graph on $\mathbb{Z}_5$. The weight function on its edge set is described by the following adjacency matrix:

   $$\begin{pmatrix} 0 & 10 & 20 & \infty & 17 \\ 7 & 0 & 5 & 22 & 33 \\ 14 & 13 & 0 & 15 & 27 \\ 30 & \infty & 17 & 0 & 10 \\ \infty & 15 & 12 & 8 & 0 \end{pmatrix}$$

   Describe in detail the transcript of your runs of Dijkstra's algorithm for obtaining the length function l and the corresponding r-branching when r is successively taken to be 0, 1, 2, 3 and 4.

   (a) $r = 0$ Initial step: $S = \{0\}$ and

   $$t(k) = \begin{cases} 0 & \text{if } k \in S \\ \infty & \text{if } k \notin S \end{cases} \tag{1}$$

   Algorithm: I find the vertex v that has the minimum $0 + w(0, v)$, which from the adjacency matrix is the vertext 1, and the corresponding edge to tack on is $0 + 10$. i.e. as $t(1) > t(0) + w(0, 1)$, we replace $t(1)$ with 10. Then, our current state is $S = \{0, 1\}$ and

   $$t(k) = \begin{cases} 0 & \text{if } k = 0 \\ 10 & \text{if } k = 1 \\ \infty & \text{if } k \notin S \end{cases} \tag{2}$$

Next, I find the vertex v that has the minimum $t(k) + w(k, v) \forall k \in S$ this is vertex 2 with the weight sum being $10 + 5$. This comes from first $t(2) > t(0) + w(0, 2) = 20$, which allows me to set $t(2) = 20$. Then $t(2) > t(1) + w(1, 2) = 10 + 5 = 15$, which allows me to set $t(2) = 15$. Then, the current state is $S = \{0, 1, 2\}$ and

$$t(k) = \begin{cases} 0 & \text{if } k = 0 \\ 10 & \text{if } k = 1 \\ 15 & \text{if } k = 2 \\ \infty & \text{if } k \notin S \end{cases} \tag{3}$$

The next state is $S = \{0, 1, 2, 4\}$ and

$$t(k) = \begin{cases} 0 & \text{if } k = 0 \\ 10 & \text{if } k = 1 \\ 15 & \text{if } k = 2 \\ 17 & \text{if } k = 4 \\ \infty & \text{if } k \notin S \end{cases} \tag{4}$$

The next state is $S = \{0, 1, 2, 4, 3\}$ and

$$t(k) = \begin{cases} 0 & \text{if } k = 0 \\ 10 & \text{if } k = 1 \\ 15 & \text{if } k = 2 \\ 17 & \text{if } k = 4 \\ 25 & \text{if } k = 3 \end{cases} \tag{5}$$

(b) $r = 1$ Initial step: $S = \{1\}$ and

$$t(k) = \begin{cases} 0 & \text{if } k \in S \\ \infty & \text{if } k \notin S \end{cases} \tag{6}$$

Algorithm: The next state is $S = \{1, 2\}$

$$t(k) = \begin{cases} 0 & \text{if } k = 1 \\ 5 & \text{if } k = 2 \\ \infty & \text{if } k \notin S \end{cases} \tag{7}$$

The next state is $S = \{1, 2, 0\}$ and

$$t(k) = \begin{cases} 0 & \text{if } k = 1 \\ 5 & \text{if } k = 2 \\ 7 & \text{if } k = 0 \\ \infty & \text{if } k \notin S \end{cases} \tag{8}$$

The next state is $S = \{1, 2, 0, 3\}$ and

$$t(k) = \begin{cases} 0 & \text{if } k = 1 \\ 5 & \text{if } k = 2 \\ 7 & \text{if } k = 0 \\ 20 & \text{if } k = 3 \\ \infty & \text{if } k \notin S \end{cases} \tag{9}$$

The next state is $S = \{1, 2, 0, 3, 4\}$ and

$$t(k) = \begin{cases} 0 & \text{if } k = 1 \\ 5 & \text{if } k = 2 \\ 7 & \text{if } k = 0 \\ 20 & \text{if } k = 3 \\ 24 & \text{if } k = 4 \end{cases} \tag{10}$$

(c) $r = 2$ Initial step: $S = \{2\}$ and

$$t(k) = \begin{cases} 0 & \text{if } k \in S \\ \infty & \text{if } k \notin S \end{cases} \tag{11}$$

Algorithm: The next state is $S = \{2, 1\}$

$$t(k) = \begin{cases} 0 & \text{if } k = 2 \\ 13 & \text{if } k = 1 \\ \infty & \text{if } k \notin S \end{cases} \tag{12}$$

The next state is $S = \{2, 1, 0\}$ and

$$t(k) = \begin{cases} 0 & \text{if } k = 2 \\ 13 & \text{if } k = 1 \\ 14 & \text{if } k = 0 \\ \infty & \text{if } k \notin S \end{cases} \tag{13}$$

The next state is $S = \{2, 1, 0, 3\}$ and

$$t(k) = \begin{cases} 0 & \text{if } k = 2 \\ 13 & \text{if } k = 1 \\ 14 & \text{if } k = 0 \\ 15 & \text{if } k = 3 \\ \infty & \text{if } k \notin S \end{cases} \tag{14}$$

The next state is $S = \{2, 1, 0, 3, 4\}$ and

$$t(k) = \begin{cases} 0 & \text{if } k = 2 \\ 13 & \text{if } k = 1 \\ 14 & \text{if } k = 0 \\ 15 & \text{if } k = 3 \\ 25 & \text{if } k = 4 \end{cases} \tag{15}$$

(d) $r = 3$ Initial step: $S = \{3\}$ and

$$t(k) = \begin{cases} 0 & \text{if } k \in S \\ \infty & \text{if } k \notin S \end{cases} \tag{16}$$

Algorithm: The next state is $S = \{3, 4\}$

$$t(k) = \begin{cases} 0 & \text{if } k = 3 \\ 10 & \text{if } k = 4 \\ \infty & \text{if } k \notin S \end{cases} \tag{17}$$

The next state is $S = \{3, 4, 2\}$ and

$$t(k) = \begin{cases} 0 & \text{if } k = 3 \\ 10 & \text{if } k = 4 \\ 17 & \text{if } k = 2 \\ \infty & \text{if } k \notin S \end{cases} \tag{18}$$

The next state is $S = \{3, 4, 2, 1\}$ and

$$t(k) = \begin{cases} 0 & \text{if } k = 3 \\ 10 & \text{if } k = 4 \\ 17 & \text{if } k = 2 \\ 25 & \text{if } k = 1 \\ \infty & \text{if } k \notin S \end{cases} \tag{19}$$

The next state is $S = \{3, 4, 2, 1, 0\}$ and

$$t(k) = \begin{cases} 0 & \text{if } k = 3 \\ 10 & \text{if } k = 4 \\ 17 & \text{if } k = 2 \\ 25 & \text{if } k = 1 \\ 30 & \text{if } k = 0 \end{cases} \tag{20}$$

(e) $r = 4$ Initial step: $S = \{4\}$ and

$$t(k) = \begin{cases} 0 & \text{if } k \in S \\ \infty & \text{if } k \notin S \end{cases} \tag{21}$$

Algorithm: The next state is $S = \{4, 3\}$

$$t(k) = \begin{cases} 0 & \text{if } k = 4 \\ 8 & \text{if } k = 3 \\ \infty & \text{if } k \notin S \end{cases} \tag{22}$$

The next state is $S = \{4, 3, 2\}$ and

$$t(k) = \begin{cases} 0 & \text{if } k = 4 \\ 8 & \text{if } k = 3 \\ 12 & \text{if } k = 2 \\ \infty & \text{if } k \notin S \end{cases} \tag{23}$$

The next state is $S = \{4, 3, 2, 1\}$ and

$$t(k) = \begin{cases} 0 & \text{if } k = 4 \\ 8 & \text{if } k = 3 \\ 12 & \text{if } k = 2 \\ 15 & \text{if } k = 1 \\ \infty & \text{if } k \notin S \end{cases} \tag{24}$$

The next state is $S = \{4, 3, 2, 1, 0\}$ and

$$t(k) = \begin{cases} 0 & \text{if } k = 4 \\ 8 & \text{if } k = 1 \\ 12 & \text{if } k = 2 \\ 15 & \text{if } k = 3 \\ 22 & \text{if } k = 0 \end{cases} \tag{25}$$

3. Let T denote a minimum-weight spanning tree in G, and let T' be another spanning tree in G. Prove that T' can be transformed into T with a list of steps that exchange one edge of T' for one edge of T such that the edge set is always a spanning tree and the total weight never increases.

   Proof: I will do this proof by construction. Assume we are given a minimum spanning tree T and another spanning tree T'. T and T' may share edges, so we treat the shared edges to be $(e_1, ...e_{i-1}) \in E(T) \cap E(T')$ where $1 \leq i \leq n + 1$. Now from the fact that T is MST, it must be the case that

$$\sum_{e \in E(T)} w(e) \leq \sum_{e \in E(T')} w(e)$$

   and additionally, from our shared edges that

$$\sum_{\substack{1 \leq x \leq i-1 \\ e_x \in E(T)}} w(e_x) = \sum_{\substack{1 \leq x \leq i-1 \\ e_x \in E(T')}} w(e_x)$$

   This then means that the non-shared edge weight sum is:

$$\sum_{\substack{i \leq x \leq n \\ e_x \in E(T)}} w(e_x) \leq \sum_{\substack{i \leq x \leq n \\ e_x \in E(T')}} w(e_x)$$

As both graphs are spanning trees, for any edge I pick in either graph, it will be a single edge, connecting two acyclic connected components. Now, from the non-shared edge weight sum above, it must the be the case that $\exists\ e^* \in E(T')$ s.t. $w(e^*) \geq w(e)$ for some $e \in E(T)$ and connects the same connected components as e. (We can check this by contradiction i.e., if every edge e^* in the non shared edges between the two spanning trees that connected the components had a weight s.t. $w(e^*) < w(e)$ then the non-shared edge weight sum would be violated and T would not be a minimum spanning tree). Then, we generate our new spanning tree $T^* = T' + e - e^*$ which does not increase the weight sum. Now, our new tree is a spanning tree which shares more edges with T, and we can apply the same reasoning recursively, generating a new spanning tree which shares more and more edges with T, until all the edges are the same. ■

4. Give a detail proof of correctness for the Dijkstra's algorithm.

   Proof(By Induction): Let $s(u)$ be the shortest path (weight) from some starting vertex v to a vertex u. WTS that $t(u) = s(u)$

   Base case: Consider the starting state of Djikstra's when $S = \{v\}$. We let $u = v$ in this case. Then, $t(v) = 0$ which is clearly equal to $s(v)$

   Induction step: WTS that for any new u added to S, given that $t(k) = s(k) \forall k \in S$ for $|S| \geq 1$, that $t(u) = s(u)$. Assume by contradiction that $t(u) \neq s(u)$

   Case 1: $t(u) < s(u)$

   This case is trivially false, as $s(u)$ is the shortest path between two vertices, and as $t(u)$ is a path between v and u, it can never be less than the shortest path.

   Case 2: $t(u) > s(u)$

   As $s(u)$ may be some completely different path, Let us consider $w(k, z)$ to be the first weight edge that extends out of S for some $k \in S$ that is in $s(u)$. Then $s(k) + w(k, z) \leq s(u)$ as u is a vertex outside S. Then, from our induction hypothesis we get that $t(k) + w(k, z) \leq s(u)$ as $t(k) = s(k) \forall k \in S$. Now, as z is adjacent to k, it must have been updated with Djikstra's, which means that $t(z) \leq t(k) + w(k, z)$. However, as u was picked by the algorithm, it must be the case that $t(u) \leq t(z)$. This leads us to the contradiction that $t(u) \leq s(u) < t(u)$. Hence, the claim is proved. ■

5. Let G be a weighted connected graph (with positive edge weights). Describe an algorithm for finding a spanning tree which is such that the product of its edge weights is minimum.

   Algorithm: We can use Kruskal's algorithm.

   Initial step: Construct a graph G' with $|E(G')| = 0$ and $V(G') = V(G)$

   Algorithmic step: At each stage of the algorithm pick the minimum weight edge that does not add a cycle to G' in $G - G'$ and add it to G'. Repeat this process $n - 1$ times.

   Reasoning: If you are minimizing the sum on the same number of elements from an arbitrary selecition, then you are also minimizing the product.

   ■