Our first problem in this lab relates to connecting a number of buildings (i.e., a campus) with fiber optic cable. Each building will have a router and wireless network equipment which will serve a local area network within the building. Routers can be configured to route network traffic from any building to any other building provided there is some fiber optic cable path from the source to the destination. It is Ok if the network traffic has to go through several routers to get to its destination. That’s good news: we do not have to directly connect every building to every other building. But, we have to provide enough connections so there is a pathway from every building to every other. The cost of connecting different pairs of buildings varies, depending on the distance between the buildings, and depending on what lies between the buildings. For example, walkways that are cut-through by a trench digger must be rebuilt, adding to the cost. Cutting through a parking lot, and making subsequent repairs is even more expensive.

It may be impractical to directly connect certain pairs of buildings. For these buildings, every plausible trench plan either 1) goes through another building which is in the way, or 2) crosses natural gas pipelines which is not allowed due to safety regulations, or 3) goes through property on which we have no legal right-of-way. For these pairs of buildings, we will designate the cost of connecting these buildings as +\infty to indicate that they cannot be directly connected. You have sent your field engineers out to survey the site, and obtain contract bids on the cost of connecting each pair of buildings. We label our eight buildings: B_1, B_2, B_3, B_4, B_5, B_6, B_7, and B_8. The report of the field engineers indicates the following costs (in thousands of dollars):

<table>
<thead>
<tr>
<th></th>
<th>B_1</th>
<th>B_2</th>
<th>B_3</th>
<th>B_4</th>
<th>B_5</th>
<th>B_6</th>
<th>B_7</th>
<th>B_8</th>
</tr>
</thead>
<tbody>
<tr>
<td>B_1</td>
<td>0</td>
<td>12</td>
<td>+\infty</td>
<td>+\infty</td>
<td>15</td>
<td>+\infty</td>
<td>+\infty</td>
<td>+\infty</td>
</tr>
<tr>
<td>B_2</td>
<td>12</td>
<td>0</td>
<td>10</td>
<td>4</td>
<td>2</td>
<td>7</td>
<td>+\infty</td>
<td>+\infty</td>
</tr>
<tr>
<td>B_3</td>
<td>+\infty</td>
<td>10</td>
<td>0</td>
<td>6</td>
<td>+\infty</td>
<td>8</td>
<td>+\infty</td>
<td>+\infty</td>
</tr>
<tr>
<td>B_4</td>
<td>+\infty</td>
<td>4</td>
<td>6</td>
<td>0</td>
<td>+\infty</td>
<td>+\infty</td>
<td>+\infty</td>
<td>5</td>
</tr>
<tr>
<td>B_5</td>
<td>15</td>
<td>2</td>
<td>+\infty</td>
<td>+\infty</td>
<td>0</td>
<td>11</td>
<td>9</td>
<td>17</td>
</tr>
<tr>
<td>B_6</td>
<td>+\infty</td>
<td>7</td>
<td>8</td>
<td>+\infty</td>
<td>11</td>
<td>0</td>
<td>3</td>
<td>13</td>
</tr>
<tr>
<td>B_7</td>
<td>+\infty</td>
<td>+\infty</td>
<td>+\infty</td>
<td>+\infty</td>
<td>9</td>
<td>3</td>
<td>0</td>
<td>+\infty</td>
</tr>
<tr>
<td>B_8</td>
<td>+\infty</td>
<td>+\infty</td>
<td>+\infty</td>
<td>5</td>
<td>17</td>
<td>13</td>
<td>+\infty</td>
<td>0</td>
</tr>
</tbody>
</table>

Finally, it is your job to ensure that the project is completed at the lowest possible cost. You must come up with an infrastructure plan (a list of pairs of buildings which will be directly connected) which is lowest cost of all possible infrastructure plans. You must also convince the Comptroller’s office and the Financial V.P. that your plan is optimal.

The business problem described above is accurately modeled by a weighted graph. A weighted graph is defined as follows:
**Definition:** A weighted graph is a tuple $G = (V, E)$ where $V$ is a set of $n$ vertices

$$V = \{v_1, v_2, v_3, ..., v_n\}$$

and a set of $m$ edges

$$E = \{(v_{i_1}, v_{j_1}), (v_{i_2}, v_{j_2}), (v_{i_3}, v_{j_3}), ..., (v_{i_m}, v_{j_m})\}$$

together with a cost function

$$f : V \times V \to (\mathbb{R}^+ \cup +\infty)$$

A sub-graph (with no cycles) which connects all of the vertices in the graph is called a spanning tree.

For a weighted graph, we can define the cost of a spanning tree to be the sum the costs of all of the edges in the graph. Various spanning trees are possible, and each will have a (possibly) different cost. A minimal spanning tree is a spanning tree which is as low cost as any other spanning tree. An important theorem relating to minimal spanning trees is: *If all of the edge costs are distinct, then the minimal spanning tree is unique.*

The cost table shown above can be illustrated by the following weighted graph:

![Graph Illustration]

Fortunately, there are several straightforward algorithms to solve the minimal spanning tree problem. One such algorithm is known as Kruskal’s algorithm. 2

There are several approaches to algorithm design. Kruskal’s algorithm takes an approach known as a greedy algorithm. A greedy algorithm makes a “greedy” local choice without any global knowledge; further, a sequence of greedy local choices leads to a globally correct solution. In the case of Kruskal’s algorithm, the local decision is to choose the next lowest-cost edge. I.e., if you want a lowest-cost spanning tree, build it using the lowest-cost edges. At the point in time where the algorithm makes the

---

1 The curious reader is referred to the graph theory literature for a mathematical proof of this theorem.

“greedy” choice, it has not examined the other edges (i.e., it has no “global” knowledge of the problem details). Not all problems can be solved by a greedy approach.\footnote{For example, a greedy algorithm approach fails miserably at solving the well known “Knapsack Problem”. E.g., \( \{3, 5, 7, 15, 19, 22, 30\} \) with \( K = 26 \).} Mathematical proof is required to show that the sequence of greedy local choices leads to a globally correct solution.

A pseudo-code statement of Kruskal’s algorithm is given below:

\[\begin{align*}
\text{Input:} & \quad \text{A weighted undirected graph} \; G = (V, E) \; \text{with weight function} \; f : V \times V \rightarrow \mathbb{R}^+ \cup +\infty \\
\text{Output:} & \quad \text{A minimal spanning tree} \; T \; \text{for} \; G.
\end{align*}\]

Method:

\[\begin{align*}
T &= \emptyset \; ; \quad /* \; \text{Empty set.} */ \\
S &= \emptyset \; ; \quad /* \; \text{Empty set.} */ \\
\text{Construct a priority queue} \; Q \; \text{of all edges in} \; E \; ; \\
\text{for each vertex} \; v \in V \; \text{do} \; \text{add} \; \{v\} \; \text{to} \; S. \\
\text{while} \; ||S|| > 1 \; \text{do} \\
\quad \text{choose edge} \; (v, w) \; \text{from} \; Q \; \text{of lowest cost} \; ; \\
\quad \text{delete} \; (v, w) \; \text{from} \; Q \; ; \\
\quad \text{if} \; ( \; v \; \text{and} \; w \; \text{are in different sets} \; W_1 \; \text{and} \; W_2 \; \text{in} \; S \; ) \; \text{then} \\
\quad \quad \text{Replace} \; W_1 \; \text{and} \; W_2 \; \text{in} \; S \; \text{by} \; W_1 \cup W_2 \; ; \\
\quad \quad \text{Add} \; (v, w) \; \text{to} \; T \; ; \\
\quad \text{end if} \\
\text{end while}
\end{align*}\]

\textbf{Your Task:} Use Kruskal’s algorithm to find a solution to the network planning problem posed above. Draw a separate diagram to illustrate the steps in the algorithm as each edge is added to the minimal spanning tree. As you consider the edges in order of increasing cost, indicate which edges are not added to the minimal spanning tree. What is the total cost of your fiber optic infrastructure plan? Are you sure there does not exist a lower cost plan (why or why not)?

\textbf{Your Next Task:} Use Dijkstra’s algorithm to find the shortest path from \( B_1 \) to all the other vertices. On each step, indicate the vertex that you choose, and the current contents of the array \( D \).

\[\begin{align*}
\text{Input:} & \quad \text{A weighted undirected graph} \; G = (V, E) \; \text{with weight function} \; f : V \times V \rightarrow \mathbb{R}^+ \cup +\infty \\
& \quad \text{A source vertex} \; v_1.
\end{align*}\]

\[\begin{align*}
\text{Output:} & \quad \text{A table} \; D \; \text{indicating cost of the shortest path from} \; v_1 \; \text{to every other vertex in} \; V.
\end{align*}\]
Method:

\[ S = \{ v_1 \} \quad /* \text{Set of vertices for which shortest path is known.} */ \]

\textbf{for each} vertex \( v \) \textbf{do}
\[ D[v] = f(v_1, v) \quad /* \text{Cost of the edge from } v_1 \text{ to } v. */ \]

\textbf{while} \( S \neq V \) \textbf{do}
\begin{itemize}
  \item choose vertex \( w \) not in \( S \) such that \( D[w] \) is smallest
  \item add \( w \) to \( S \)
\end{itemize}
\textbf{for each} vertex \( v \) in \( V \) but not in \( S \) \textbf{do}
\[ D[v] = \min(D[v], D[w] + f(w, v)) \]
\textbf{end for}
\textbf{end while}