Problem 1

Let $X$ be a set of elements. A function $f : X \times X \to \mathbb{R}$ is called a metric if it satisfies:

- $d(x, x) = 0$.
- $d(x, y) \geq 0$.
- $d(x, y) = d(y, x)$.
- $d(x, y) \leq d(x, z) + d(z, y)$.

Let $G = (V, E)$ be an undirected graph. Let $w$ be a non-negative weight function on the edges of $G$. Is $w$ always a metric? If yes prove it, if no explain why not and give a counter example.

Let $d(a, b)$ be the weight of a shortest path from $a$ to $b$. Prove that $d(a, b)$ is a metric.

Now consider the case of a directed graph. Does $d(a, b)$ define a metric? If yes prove it, if no explain why not and give a counter example.

Problem 2

Let $G = (V, E)$ be a directed graph. Let $w : V \times V \times V \to \mathbb{R}$ be a non-negative weight function on sequences of 3 vertices.
In analogy to the usual length of a path in a weighted graph we can define the value of a path \( P = (v_1, \ldots, v_k) \) by a sum

\[
v(P) = \sum_{i=1}^{k-2} w(v_i, v_{i+1}, v_{i+2}).
\]

Give an efficient algorithm for computing a minimum value path between a pair of vertices. You should give the fastest algorithm you can. Prove the correctness of the algorithm and a running time bound.

Hint: You could use a standard shortest path algorithm on a “bigger graph”. You can also modify Dijkstra or Bellman-Ford algorithm to solve the problem directly.

Can you give a possible application for this problem?

**Problem 3**

Let \( G = (V, E) \) be a directed graph. Let \( c : E \rightarrow \mathbb{R} \) define a capacity for an edge. The capacity of a path \( P \) is the minimum capacity among edges in \( P \). For example, if the capacity represents a bound on the number of cars per second that a road supports, the bottle-neck of \( P \) gives the maximum number of cars per second that can go along \( P \). In another example the edges might represent circuits, and \( c(e) \) is the maximum clock-rate for \( e \) to operate reliably. The capacity of \( P \) gives the maximum clock-rate for the pipeline defined by \( P \) to operate reliably.

Show how to modify Dijkstra’s algorithm to find the highest capacity path from the source \( s \) to every vertex in the graph. Prove the correctness of your algorithm.