```
374     def matrix_multiplication(self):
375         # Variable to store the number of vertices
376         n = len(list(self.parse_vertices()))
377         # Weight matrix
378         w = {}
379         # Previous matrix
380         prev = {}
381         # Initialize the matrices
382         for vertex_x in sorted(self.parse_vertices()):
383             for vertex_y in sorted(self.parse_vertices()):
384                 # If the vertices are the same, the cost is 0 and the previous vertex is None
385                 if vertex_x == vertex_y:
386                     w[vertex_x, vertex_y] = 0
387                     prev[vertex_x, vertex_y] = None
388                 # If the edge exists, the cost is the cost of the edge and the previous vertex is the first vertex
389                 elif self.is_edge(vertex_x, vertex_y):
390                     w[vertex_x, vertex_y] = self.get_edge_cost(vertex_x, vertex_y)
391                     prev[vertex_x, vertex_y] = vertex_x
392                 # If the edge does not exist, the cost is infinity
393                 else:
394                     w[vertex_x, vertex_y] = math.inf
395         # Exponentiation by squaring
396         k = 1
397         # Initialize the intermediate matrices
398         inter_w = [w]
399         while k < n:
400             next_w = {}
401             next_p = {}
402             for vertex_x in sorted(self.parse_vertices()):
```

```
            for vertex_y in sorted(self.parse_vertices()):
                next_w[vertex_x, vertex_y] = math.inf
                for z in self.parse_vertices():
                    # If the cost is smaller, update the cost and the previous vertex
                    if next_w[vertex_x, vertex_y] > w[vertex_x, z] + w[z, vertex_y]:
                        next_w[vertex_x, vertex_y] = w[vertex_x, z] + w[z, vertex_y]
                        # If both weights are different, the previous vertex is the previous vertex of z,y
                        if next_w[vertex_x, vertex_y] != w[vertex_x, vertex_y]:
                            next_p[vertex_x, vertex_y] = prev[z, vertex_y]
                        # If both weights are the same, the previous vertex is the previous vertex of x,y
                        else:
                            next_p[vertex_x, vertex_y] = prev[vertex_x, vertex_y]
                # If the cost is infinity and the vertices are the same, we have negative cost cycle, return None
                if next_w[vertex_x, vertex_y] < 0 and vertex_x == vertex_y:
                    return None
        # Append the intermediate matrix
        inter_w.append(next_w)
        # Squaring the exponent
        k = k * 2
        # Update the matrices
        w = next_w
        prev = next_p
    # Return the weight matrix, the previous matrix and the intermediate matrix
    return w, prev, inter_w
```

```
280     def min_cost_path_matrix_multiplication(self):
281         vertex_x = input("Please enter the first vertex: ")
282         if not vertex_x.isnumeric():
283             raise ValueError(f"First vertex must be an integer")
284         vertex_x = int(vertex_x)
285
286         vertex_y = input("Please enter the second vertex: ")
287         if not vertex_y.isnumeric():
288             raise ValueError(f"Second vertex must be an integer")
289         vertex_y = int(vertex_y)
290
291         source = vertex_x
292         target = vertex_y
293
294         result = self.__graph.matrix_multiplication()
295
296         # If there is a negative cost cycle
297         if result is None:
298             print("There exist negative cost cycles")
299         # If there is no negative cost cycle
300         else:
301             # Extract the result
302             w, prev, inter = result
303             # Reconstruction of the path
304             path = []
305             while target != source:
306                 path.append(target)
307                 target = prev[source, target]
308             path.append(source)
309             path.reverse()
310             print("The minimum cost path is: ", path)
```

```
311             print("The cost of the minimum cost path is: ", w[vertex_x, vertex_y])
312             print("\nThe intermediate matrices:")
313             # Print the intermediate matrices
314             for inter_w in inter:
315                 for vertex_x in sorted(self.__graph.parse_vertices()):
316                     for vertex_y in sorted(self.__graph.parse_vertices()):
317                         print(inter_w[vertex_x, vertex_y], end=" ")
318                     print()
319                 print()
```