173. An automotive company has three assembly lines (Line 1, Line 2, Line 3) to produce different car models. Each line has a series of stations, and each station takes a certain amount of time to complete its task. Additionally, there are transfer times between lines, and certain dependencies must be respected due to the sequential nature of some tasks. Your goal is to minimize the total production time by determining the optimal scheduling of tasks across these lines, considering the transfer times and dependencies.

PROGRAM:

```
class Task:
    def __init__(self, line, station, start_time):
        self.line = line
        self.station = station
        self.start_time = start_time

def calculate_schedule(num_lines, num_stations, station_times, transfer_times, dependencies):
    # Initialize start times with -1 (meaning not yet scheduled)
    start_times = [[-1 for _ in range(num_stations)] for _ in range(num_lines)]

    # Helper function to find the earliest time a task can start
    def find_earliest_start(line, station):
        if start_times[line][station] != -1:
            return start_times[line][station]

        # Earliest start time based on the previous station in the same line
        if station > 0:
            prev_station_time = find_earliest_start(line, station - 1) + station_times[line][station - 1]
        else:
            prev_station_time = 0

        # Earliest start time based on transfer from the previous line
        if line > 0:
            prev_line_time = find_earliest_start(line - 1, num_stations - 1) + station_times[line - 1][num_stations - 1] + transfer_times[line - 1][line]
        else:
            prev_line_time = 0

        # Earliest start time based on dependencies
        dep_time = 0
        for (from_station, to_station) in dependencies:
```

```
            if to_station == station:
                dep_time = max(dep_time, find_earliest_start(line, from_station) + station_times[line][from_station])

        # The task can start only after all preceding tasks are done
        start_time = max(prev_station_time, prev_line_time, dep_time)
        start_times[line][station] = start_time
        return start_time

    # Calculate the start times for all tasks
    for line in range(num_lines):
        for station in range(num_stations):
            find_earliest_start(line, station)

    # Print the results
    for line in range(num_lines):
        for station in range(num_stations):
            print(f"Line {line + 1}, Station {station + 1}: Start time {start_times[line][station]}")

# Example usage
num_lines = 3
num_stations = 3
station_times = [[5, 9, 3], [6, 8, 4], [7, 6, 5]]
transfer_times = [[0, 2, 3], [2, 0, 4], [3, 4, 0]]
dependencies = [(0, 1), (1, 2)]

calculate_schedule(num_lines, num_stations, station_times, transfer_times, dependencies)
```
OUPUT:

```
Line 1, Station 1: Start time 0
Line 1, Station 2: Start time 5
Line 1, Station 3: Start time 14
Line 2, Station 1: Start time 19
Line 2, Station 2: Start time 25
Line 2, Station 3: Start time 33
Line 3, Station 1: Start time 41
Line 3, Station 2: Start time 48
Line 3, Station 3: Start time 54

=== Code Execution Successful ===
```

TIME COMPLEXITY: O(num_lines * num_stations)