Distributed System

Assignment 7

SOURABH PATEL | U19CS082

Implement Lamport's clock synchronization algorithm and discuss its time complexity.

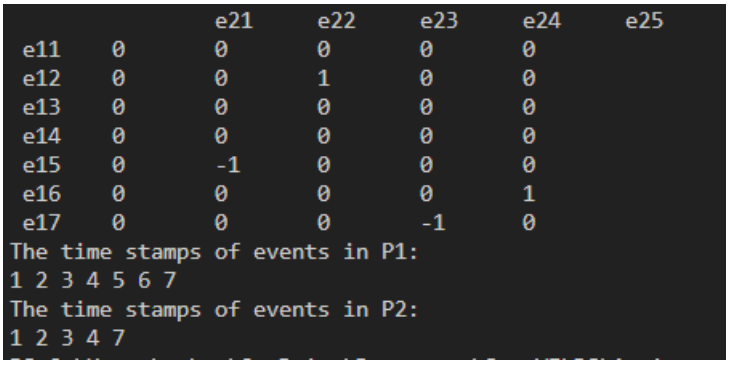
Code:

```cpp
// C++ program to illustrate the Lamport's
// Logical Clock
#include <bits/stdc++.h>
using namespace std;
// Function to display the logical timestamp
void display(int e1, int e2, int p1[5], int p2[3])
{
  int i;
  cout << "\nThe time stamps of events in P1:\n";
  for (i = 0; i < e1; i++)
  {
    cout << p1[i] << " ";
  }
  cout << "\nThe time stamps of events in P2:\n";
  // Print the array p2[]
  for (i = 0; i < e2; i++)
    cout << p2[i] << " ";
}
```

```
// Function to find the timestamp of events
void lamportLogicalClock(int e1, int e2, int m[7][5])
{
  int i, j, k, p1[e1], p2[e2];
  // Initialize p1[] and p2[]
  for (i = 0; i < e1; i++)
    p1[i] = i + 1;
  for (i = 0; i < e2; i++)
    p2[i] = i + 1;
  cout << "\t";
  for (i = 0; i < e2; i++)
    cout << "\te2" << i + 1;
  for (i = 0; i < e1; i++)
  {
    cout << "\n e1" << i + 1 << "\t";
    for (j = 0; j < e2; j++)
      cout << m[i][j] << "\t";
  }
  for (i = 0; i < e1; i++)
  {
    for (j = 0; j < e2; j++)
    {
      // Change the timestamp if the
      // message is sent
      if (m[i][j] == 1)
      {
        p2[j] = max(p2[j], p1[i] + 1);
        for (k = j + 1; k < e2; k++)
          p2[k] = p2[k - 1] + 1;
      }
      // Change the timestamp if the
```

```
      // message is received
      if (m[i][j] == -1)
      {
        p1[i] = max(p1[i], p2[j] + 1);
        for (k = i + 1; k < e1; k++)
          p1[k] = p1[k - 1] + 1;
      }
    }
  }
  // Function Call
  display(e1, e2, p1, p2);
}
// Driver Code
int main()
{
  int e1 = 7, e2 = 5, m[7][5] = {0};
  // message is sent and received
  // between two process
  /*dep[i][j] = 1, if message is sent
  from ei to ej
  dep[i][j] = -1, if message is received
  by ei from ej
  dep[i][j] = 0, otherwise*/
  m[1][2] = 1;
  m[5][4] = 1;
  m[4][1] = -1;
  m[6][3] = -1;
  // Function Call
  lamportLogicalClock(e1, e2, m);
  return 0;
}
```

Output:



Time Complexity:

O(e1 * e2 * (e1 + e2))