

Distributed Systems (CS304)

Assignment - 8

U19CS012

Q.) Implement Lamport Logical Clock Algorithm.

"Happened Before" Relation / Casual Ordering

Lamport's Happened Before relationship:

For two events a and b , $a \rightarrow b$ if

- a and b are events in the same process and a occurred before b , or
- a is a send event of a message m and b is the corresponding receive event at the destination process, or
- $a \rightarrow c$ and $c \rightarrow b$ for some event c

Algorithm

Each process i keeps a clock C_i

- Each event a in i is time-stamped $C_i(a)$, the value of C_i when a occurred
- C_i is incremented by 1 for each event in i
- In addition, if a is a send of message m from process i to j , then on receive of m ,
$$C_j = \max (C_j, C_i(a)+1)$$

Code

```
// Implementation of Lamport Clock Algorithm [U19CS012]
#include <bits/stdc++.h>
using namespace std;

typedef vector<int> vi;
typedef pair<int, int> pi;

// Data Structure to Store all the Communication Lines {pid1,eid1} -> {pid2,eid2}
typedef pair<pi, pi> ppipi;

// Custom Comparator Function to Sort all the Communication Lines [Vector Clock]
bool my_sort(ppipi a, ppipi b)
{
    // Sort by Receiving Node {pid,eid} in ascending Order & Sending Node in descending Order
    return ((a.second.second < b.second.second) && (a.second.second < b.first.second) &&
(a.second.first < b.second.second) && (a.second.first < b.first.second) && (a.first.first >
b.first.first));
}

int main()
{
    // freopen("input1.txt", "r", stdin);
    // freopen("input2.txt", "r", stdin);
    // freopen("input3.txt", "r", stdin);

    // There are n Process namely P1, P2, P3,...PN
    int n;
    cout << "\nEnter the Number of Processes : ";
    cin >> n;
    cout << '\n';

    // Store the Number of Events in Each Process
    vi events(n, 0);
    int max_events = 0;

    for (int pid = 1; pid <= n; pid++)
    {
        int evnts;
        cout << "Enter the Number of Events in Process " << pid << " : ";
        cin >> evnts;

        // Update the Maximum Number of Events
        if (evnts > max_events)
            max_events = evnts;

        events[pid - 1] = evnts;
    }
    cout << '\n';
```

```

// Input the Communication Lines
int comm_lines;
cout << "Enter the Number of Communication Lines : ";
cin >> comm_lines;
cout << '\n';

// Data Structure to Store the Communication Lines
vector<ppipi> lines;

for (int c = 0; c < comm_lines; c++)
{
    cout << "Communication Line Number " << c + 1 << " : \n";

    // For Each Communication Line {pid1,i1} -> {pid2,i2}
    int pid1, eid1, pid2, eid2;

    cout << "Enter the Co-Ordinates {process_id,event_id} of Sending Node : ";
    cin >> pid1 >> eid1;
    // Valid Input Checks 1
    assert(pid1 >= 1 && pid1 <= n);
    assert(eid1 >= 1 && eid1 <= events[pid1 - 1]);

    cout << "Enter the Co-Ordinates {process_id,event_id} of Receiving Node : ";
    cin >> pid2 >> eid2;
    // Valid Input Checks 2
    assert(pid2 >= 1 && pid2 <= n);
    assert(eid2 >= 1 && eid2 <= events[pid2 - 1]);

    lines.push_back({{pid1, eid1}, {pid2, eid2}});
    cout << '\n';
}

sort(lines.begin(), lines.end(), my_sort);

// Sorted Communication Lines
cout << "Communication Lines after Custom Sorting : \n";
for (int i = 0; i < lines.size(); i++)
{
    cout << lines[i].first.first << " " << lines[i].first.second << " -> " <<
lines[i].second.first << " " << lines[i].second.second << "\n";
}
cout << '\n';

// Lamport Clocks
vector<vector<vi>> vec(n, vector<vi>(max_events, vi(1, 0)));

// ? Intialize all the Lamport Clock(s) with Rule 1
for (int i = 0; i < n; i++)
    for (int j = 0; j < events[i]; j++)

```

```

        vec[i][j][0] = j + 1;

// ? Implement Lamport Clock Algorithm {Rule 2}
int p1, e1, t1, p2, e2, t2;
for (int x = 0; x < comm_lines; x++)
{
    // Since Zero Based Indexing
    p1 = lines[x].first.first - 1;
    e1 = lines[x].first.second - 1;
    p2 = lines[x].second.first - 1;
    e2 = lines[x].second.second - 1;

    vec[p2][e2][0] = max(vec[p2][e2][0], vec[p1][e1][0] + 1);

    // ! [IMP] Update the Following Lines after 'e2' Event, So it Reflects in Other
    // Process as well
    for (int k = e2 + 1; k < events[p2]; k++)
    {
        vec[p2][k][0] = vec[p2][k - 1][0] + 1;
    }
}

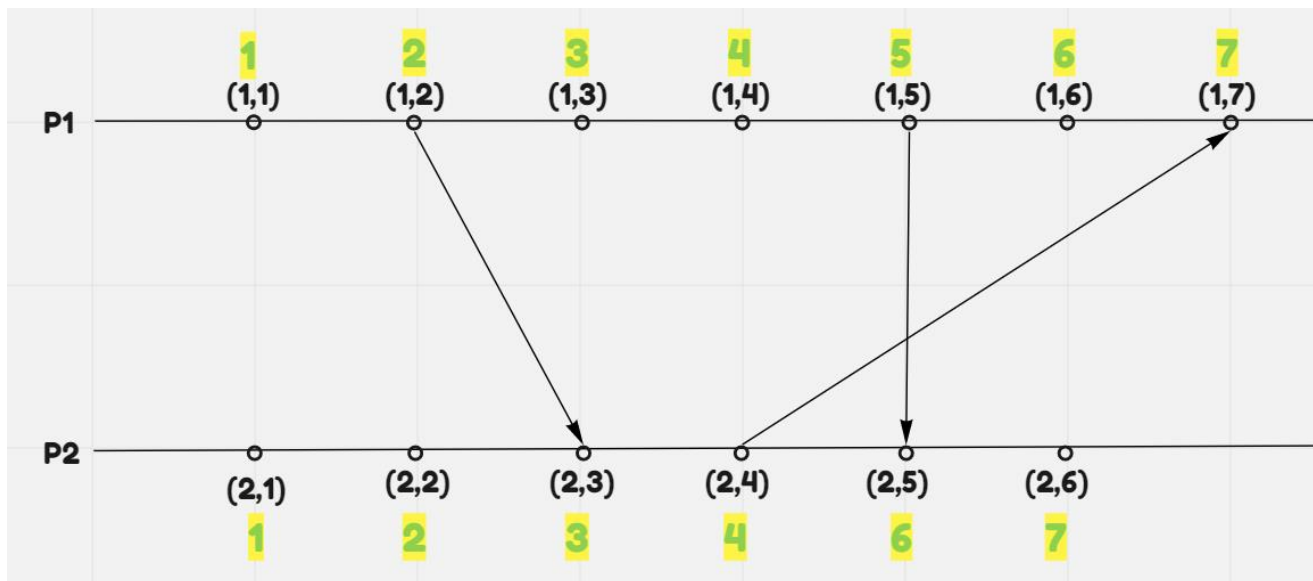
// Print the Time Stamps of ALL the Processes
for (int pid = 0; pid < n; pid++)
{
    cout << "Process " << pid + 1 << " : ";
    // Intial Vector Clock
    cout << "(" << 0 << ")" << " : ";

    // Remaining Vector Clocks of Process 'pid'
    for (int eid = 0; eid < events[pid]; eid++)
    {
        cout << "(" << vec[pid][eid][0] << ")" << " ";
    }
    cout << '\n';
}

return 0;
}

```

Input (Basic)



Output

```
PS C:\Users\Admin\Desktop\LAB8> cd "c:\Users\Admin\Desktop\LAB8\" ; if ($?)  
. \lampont_clock }
```

Enter the Number of Processes : 2

Enter the Number of Events in Process 1 : 7

Enter the Number of Events in Process 2 : 6

Enter the Number of Communication Lines : 3

Communication Line Number 1 :

Enter the Co-Ordinates {process_id,event_id} of Sending Node : 1 2

Enter the Co-Ordinates {process_id,event_id} of Receiving Node : 2 3

Communication Line Number 2 :

Enter the Co-Ordinates {process_id,event_id} of Sending Node : 2 4

Enter the Co-Ordinates {process_id,event_id} of Receiving Node : 1 7

Communication Line Number 3 :

Enter the Co-Ordinates {process_id,event_id} of Sending Node : 1 5

Enter the Co-Ordinates {process_id,event_id} of Receiving Node : 2 5

Communication Lines after Custom Sorting :

1 2 -> 2 3

2 4 -> 1 7

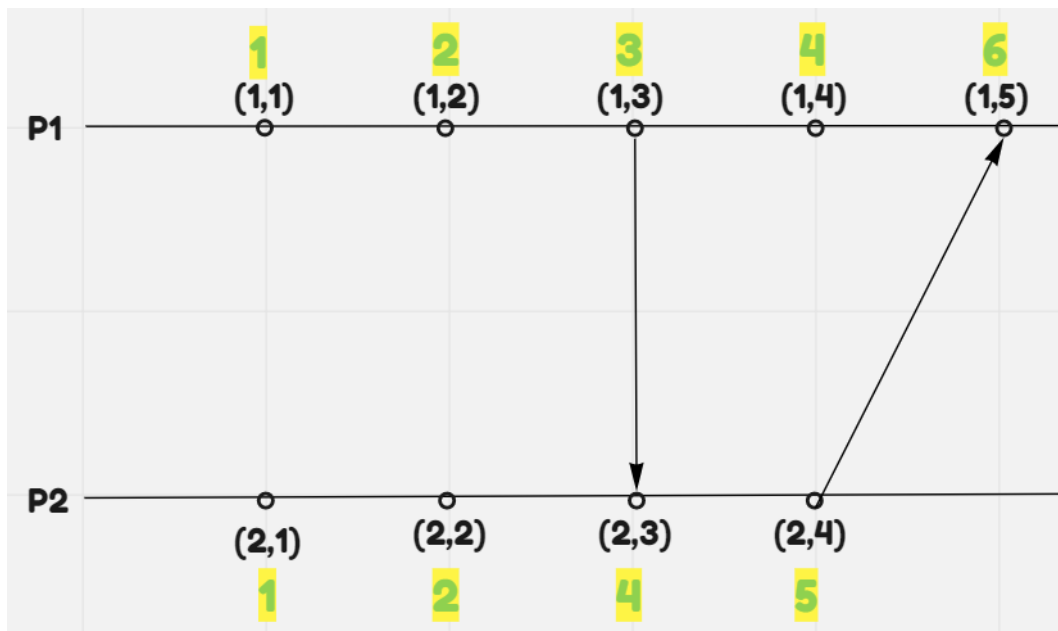
1 5 -> 2 5

Process 1 : (0) : (1) (2) (3) (4) (5) (6) (7)

Process 2 : (0) : (1) (2) (3) (4) (6) (7)

PS C:\Users\Admin\Desktop\LAB8>

Input (Medium)



Output

```
.\lamport_clock }
```

```
Enter the Number of Processes : 2
```

```
Enter the Number of Events in Process 1 : 5
```

```
Enter the Number of Events in Process 2 : 4
```

```
Enter the Number of Communication Lines : 2
```

```
Communication Line Number 1 :
```

```
Enter the Co-Ordinates {process_id,event_id} of Sending Node : 1 3
```

```
Enter the Co-Ordinates {process_id,event_id} of Receiving Node : 2 3
```

```
Communication Line Number 2 :
```

```
Enter the Co-Ordinates {process_id,event_id} of Sending Node : 2 4
```

```
Enter the Co-Ordinates {process_id,event_id} of Receiving Node : 1 5
```

```
Communication Lines after Custom Sorting :
```

```
1 3 -> 2 3
```

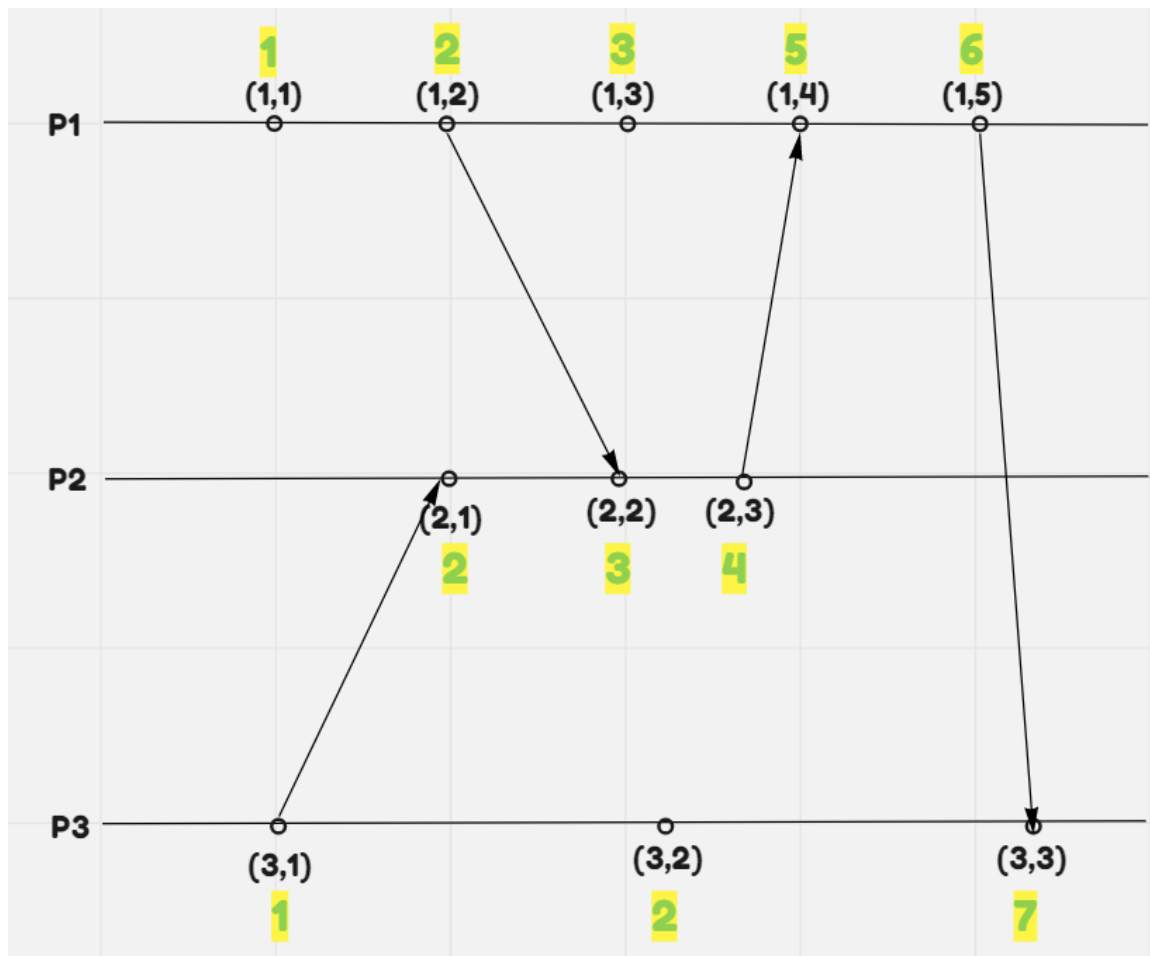
```
2 4 -> 1 5
```

```
Process 1 : (0) : (1) (2) (3) (4) (6)
```

```
Process 2 : (0) : (1) (2) (4) (5)
```

```
PS C:\Users\Admin\Desktop\LAB8>
```

Input (Hard)



Output

```
PS C:\Users\Admin\Desktop\LAB8> cd "c:\Users\Admin\Desktop\LAB8\" ; if ($?)  
. \lambport_clock }
```

Enter the Number of Processes : 3

Enter the Number of Events in Process 1 : 5

Enter the Number of Events in Process 2 : 3

Enter the Number of Events in Process 3 : 3

Enter the Number of Communication Lines : 4

Communication Line Number 1 :

Enter the Co-Ordinates {process_id,event_id} of Sending Node : 3 1

Enter the Co-Ordinates {process_id,event_id} of Receiving Node : 2 1

Communication Line Number 2 :

Enter the Co-Ordinates {process_id,event_id} of Sending Node : 1 2

Enter the Co-Ordinates {process_id,event_id} of Receiving Node : 2 2

Communication Line Number 3 :

Enter the Co-Ordinates {process_id,event_id} of Sending Node : 2 3

Enter the Co-Ordinates {process_id,event_id} of Receiving Node : 1 4

Communication Line Number 4 :

Enter the Co-Ordinates {process_id,event_id} of Sending Node : 1 5

Enter the Co-Ordinates {process_id,event_id} of Receiving Node : 3 3

Communication Lines after Custom Sorting :

3 1 -> 2 1

1 2 -> 2 2

2 3 -> 1 4

1 5 -> 3 3

Process 1 : (0) : (1) (2) (3) (5) (6)

Process 2 : (0) : (2) (3) (4)

Process 3 : (0) : (1) (2) (7)

PS C:\Users\Admin\Desktop\LAB8>

SUBMITTED BY: U19CS012

BHAGYA VINOD RANA