Assumptions

1. There are **P** processors in CPU architecture
2. The order in which edges are entering the incidence matrix does not matter
3. Storing of the matrix in memory is in row-major format

Observations

1. First, we will take the transpose of the Incidence Matrix so that we can traverse raw wise to optimize cache performance
    a. Each processor will get M/P rows where M = #Edges
2. Each row will now update one cell of adj_mat which will be shared between all the processors
    a. If there is an overlap while updating the cell critical section will handle it

Pseudo Code for the Parallel Algorithm

```
// Gloabal Region
M ← #Edges in graph
P ← #Processors
Local_size ← M/P
Adj_mat ← Zeroes(N,N)
Inc_mat ← transpose(Inc_mat)

//Adj_mat and Inc_mat are shared between all the processors

    // Parallel Region
    // Parallelise this loop into processors with M/P rows to each one
    LOOP (i,0,M) :
        X = -1
        Y = -1
        LOOP (j,0,N) :
            if(Inc_mat[i][j]>0)
                if(X<0)
                    X ← j
                else
                    Y ← j
        // Critical Section
        if(Y<0)
            Adj_mat[X][X] ← Adj_mat[X][X] + 2
        else
            Adj_mat[X][Y] ← Adj_mat[X][Y] + 1
```

Drawbacks

1. While updating one cell of Adj_mat there can be overlaps such that two processors are accessing the cell simultaneously, in this case, the performance of algorithm will be affected