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. Each cell with positive value will create independent columns in the incidence matrix
    a. We can parallelize traversal of the adjacency matrix by distributing n/p rows among each processor
    b. While assigning columns to each edge we can again parallelize it by distributing it into p processors
2. Since the memory is stores matrix in the row-major form we can store rows for each edge and then transpose it, this can help us optimize the cache usage.
    a. If the memory was column-major then we could traverse in adjacency matrix column-wise and distribute columns into processors instead of rows.

Pseudo Code for the Parallel Algorithm

```
// Gloabal Region
N ← #Nodes in graph
P ← #Processors
Local_size ← N/P

//Adj_mat and Inc_mat are shared between all the processors

    // Parallel Region
    // Parallelise this loop into processors with N/P rows to each one
    LOOP (0,N) :
        LOOP (0,N) :
            // Parallelise this loop
            While(Adj_mat[i][j]>0) :
                if(i==j)
                    Adj_mat[i][j] ← Adj_mat[i][j] / 2
                else
                    Adj_mat[i][j] ← Adj_mat[i][j] - 1
                Temp ← Vector (N,0)
                Temp[i] ← Temp[i] + 1
                Temp[j] ← Temp[j] + 1
                Inc_mat.append(Temp)

// Global Region
Inc_mat ← transpose(Inc_mat)
```