Parameters:

**Dist**: an array whose length is |V|, dist[j] represents the distance from source vertex to the vertex j at the moment.

**Matrix**: a |V| \* |V| matrix, matrix[i][j] represents the weight of edge between vertex i and vertex j.

**Src\_idx**: the index of source vertex.

**Flag**: a int variable which is either 0 or 1. Flag will be changed to 1 if the distance of any vertex is updated. If flag is not changed in an iteration, then all vertices have found their shortest distance to source vertex. As a result, the loop can end early.

**Mask/ Mask1:** an array of 0 and 1.

If a vertex is updated, then we need to relax the edges from this vertex to its neighbours.

CUDA\_INITIALIZATION (dist, src\_idx, mask, mask1)

1. idx = getThreadIDx
2. dist[idx] = INFINITY
3. mask[idx] = 0
4. mask1[idx] = 0
5. if idx == src\_idx then
6. dist[idx] = 0
7. mask[idx] = 1 # mark the source vertex as updated

CUDA\_RELAXATION (dist, matrix, mask, mask1, flag)

1. i = getThreadIDx
2. j = getThreadIDy
3. temp = dist[j]
4. if mask[i] == 1 then # only relax edges (u, v) where u is updated in last iteration
5. atomicMin(dist[j], dist[i] + matrix[i][j])
6. end if
7. if dist[j] < temp then # if vertex j is updated, then mark it as updated in **mask1**
8. mask1[j] = 1
9. flag = 1
10. end if

CUDA\_SWAP (mask, mask1, flag)

1. i = getThreadIDx
2. temp[i] = mask1[i] # Store the value in mask1 to mask
3. syncthreads ()
4. mask[i] = temp[i]
5. mask1[i] = 0 # Set the mask1’ values to be 0
6. flag = 0

MAIN\_BELLMAN\_FORD (dist, src\_idx, matrix)

1. flag = 0
2. mask = [0, …, 0]
3. mask1 = [0, …, 0]
4. CUDA\_INITIALIZATION (dist, src\_idx, mask, mask1)
5. for i = 1 to (|V| - 1) do
6. CUDA\_RELAXATION (dist, matrix, mask, mask1, flag)
7. if flag == 0 do
8. break
9. CUDA\_SWAP (mask, mask1, flag)