Algorithm 1 edge GPU APSp

Input: $G(V, E)$;
Output: $dist(u)(v)$, $(u, v \in V)$, the weight of the shortest path from u to v;

```
1:
2: function initial(V)
3:     for each u ∈ V do
4:         for each v ∈ V do
5:             dist(u)(v) ← +∞;                          ▷ initialize dist to positive infinity;
6:         end for
7:         dist(u)(u) ← 0;                               ▷ set the source distence to 0;
8:     end for
9: end function
10:
11: function edgeCudaFunc(G(V, E), dist)                 ▷ G(V, E), the initially distance array dist;
12:     u0 ← threadId;                                    ▷ get the thread id;
13:     offset ← blockDim;                                ▷ get the number of threads in a block;
14:     s0 ← blockId;                                     ▷ get the block id;
15:     blockNum ← gridDim;                               ▷ get the number of blocks in all grids;
16:     flag ← (__shared__ memory) 1;                     ▷ whether the dist has changed;
17:     old ← −1;
18:     s ← s0;
19:     while s<|V| do
20:         while true do
21:             if flag = 0 then
22:                 break;
23:             end if
24:             flag ← 0;
25:             for each (u, v, w) ∈ |E| do
26:                 old ← atomicMin(&dist(s)(v), dist(s)(u) + w);   ▷ use the atomic opt to exclusive
    mutually;
27:                 if old >dist(v) then
28:                     flag ← 1;
29:                 end if
30:                 old ← atomicMin(&dist(s)(u), dist(s)(v) + w);   ▷ use the atomic opt to exclusive
    mutually;
31:                 if old >dist(u) then
32:                     flag ← 1;
33:                 end if
34:             end for
35:
36:             __syncthreads();                          ▷ synchronize all threads in the same block;
37:
38:             if flag == 0 then
39:                 break;
40:             end if
41:         end while
42:         s ← (s + blockNum);
43:     end while
44: end function
```

45:

46: $initial(s, V)$;

47:

48: $host\_to\_device(dist), host\_to\_device(G(V, E))$; ▷ copy the $dist$ and $G(V, E) from main memory to GPU memory$;

49:

50: $edgeCudaFunc()$; ▷ call the CUDA kernal;

51: $device\_to\_host(dist)$; ▷ copy the $dist$ back;

52:

53: **return** $result$