Algorithm 1 dijkstra GPU APSP

Input: $G(V, E)$;
Output: $dist(u)(v)$, $(v, u \in V)$, 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 dijkstraCudaFunc(G(V,E), dist, predist)   ▷ G(V,E), the initially distance array dist, a
    temporary distance array predist;
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:
18:     s ← s0;                                         ▷ set the source vertex in a block;
19:     while s<|V| do
20:         while true do
21:             if flag = 0 then
22:                 break;
23:             end if
24:             flag ← 0;
25:
26:             u ← u0;
27:             while u<|V| do
28:                 if dist(s)(u) is not modified then
29:                     continue;                       ▷ this IF can speedup;
30:                 end if
31:                 for each (u, v, w) ∈ E do
32:                     atomicMin(&predist(s)(v), dist(s)(u) + w);   ▷ use the atomic opt to exclusive
    mutually;
33:                 end for
34:                 u ← (u + offset);                   ▷ reuse the thread;
35:             end while
36:
37:             __syncthreads();                        ▷ synchronize all threads in the same block;
38:
39:             u ← u0;
40:             while u<|V| do
41:                 if predist(s)(u)<dist(s)(u) then
42:                     dist(s)(u) ← predist(s)(u);
43:                     flag ← 1;                       ▷ some vexter is updated;
44:                 end if
```

```
45:                 u ← (u + offset);
46:             end while
47:
48:             __syncthreads();                    ▷ synchronize all threads in the same block;
49:
50:         end while
51:         s ← (s + blockNum);                     ▷ reuse block;
52:     end while
53: end function
54:
55: initial(V);
56:
57: host_to_device(dist), host_to_device(G(V, E));  ▷ copy the dist and
    G(V, E) from main memory to GPU memory;
58:
59: dijkstraCudaFunc();                             ▷ call the CUDA kernal;
60: device_to_host(dist);                           ▷ copy the dist back;
61:
62: return result
```