Algorithm 1 edge GPU SSSP

Input: $G(V, E)$, source vertex s;

Output: $dist(v)$, $(v \in V)$, the weight of the shortest path from s to v;

```

function $initial(s, V)$
    for each $v \in V$ do
        $dist(v) \leftarrow +\infty$;                                   ▷ initialize $dist$ to positive infinity;
    end for
    $dist(s) \leftarrow 0$;                                             ▷ set the source distence to 0;
end function

function $edgeCudaFunc(G(V, E), dist)$                              ▷ $G(V, E)$, the initially distance array $dist$;
    $u0 \leftarrow threadId$;                                           ▷ get the thread id;
    $offset \leftarrow blockDim$;                                       ▷ get the number of threads in a block;
    $flag \leftarrow$ ($\_\_shared\_\_\ memory$) 1;                     ▷ whether the $dist$ has changed;
    $old \leftarrow -1$;
    while $true$ do
        if $flag = 0$ then
            $break$;
        end if
        $flag \leftarrow 0$;
        for each $(u, v, w) \in |E|$ do
            $old \leftarrow atomicMin(\&dist(v), dist(u) + w)$;         ▷ use the atomic opt to exclusive mutually;
            if old >dist(v) then
                $flag \leftarrow 1$;
            end if
            $old \leftarrow atomicMin(\&dist(u), dist(v) + w)$;         ▷ use the atomic opt to exclusive mutually;
            if old >dist(u) then
                $flag \leftarrow 1$;
            end if
        end for

        $\_\_syncthreads()$;                                            ▷ synchronize all threads in the same block;

        if $flag == 0$ then
            $break$;
        end if
    end while
end function

$initial(s, V)$;

$host\_to\_device(dist), host\_to\_device(G(V, E))$;                   ▷ copy the $dist$ and
    $G(V, E) frommainmemorytoGPUmemory$;

$edgeCudaFunc()$;                                                       ▷ call the CUDA kernal;
$device\_to\_host(dist)$;                                               ▷ copy the $dist$ back;

return $result$
```