Operation	**CPU / GPU**	**Inputs**	**Operation**	**Return**

CAS	yes / yes	x,y	mem.oldval=mem.val
			if (mem.val==x) then mem.val = y;
				mem.oldval

ADD	yes / yes	y	mem.oldval=mem.val; mem.val = mem.val+ y;	mem.oldval
MIN	yes / yes	y	mem.oldval=mem.val; if (mem.val >y) mem.val=y;	mem.oldval

Table 0.1 Atomic Operations

Algorithm 0.1: Parallel Breadth-First Search-With atomics, path stored

```
BFS(Point src, Graph G) {
    foreach( Point p In G ) in parallel {
        $p.dist = \infty$;
        $p.pred$ = NULL;
    }
    $src.dist = 0$;
    while( 1 ){
        changed = False;
        foreach( Point p In G ) in parallel {
            foreach( Point t In p.outnbrs ){
                atomic if ( $t.dist > (p.dist + 1)$ ){
                    $t.dist = p.dist + 1$;
                    $t.pred$ = p;
                    changed = True;
                }
            }
        }
        if(changed == False) break;
    }
}
```

Algorithm 0.2: Parallel Breadth-First Search Level based with no atomics, path not stored

```
BFS(Point src, Graph G) {
    foreach( Point p In G ) in parallel {
        $p.dist = \infty$;
    }
    $src.dist = 0$;
    level = 0;
    while( 1 ){
        changed = False;
        foreach( Point p In G ) in parallel {
            foreach( Point t In p.outnbrs ){
                if (  $t.dist >$(level+ 1) ){
                    $t.dist =$ level+1;
                    changed = True;
                }
            }
        }
        if(changed == False) break;
        level=level+1;
    }
}
```

Algorithm 0.3: Parallel SSSP

```
SSSP(Point src, Graph G) {
    foreach( Point p In G ) in parallel {
        $p.dist = \infty$;
        $p.pred =$ NULL;
    }
    $src.dist = 0$;
    while( 1 ){
        changed = False;
        foreach( Point p In G ) in parallel {
            foreach( Point t In p.outnbrs ){
                atomic if ( t.dist >(p.dist + G.getEdgeWeight(p,t)) ){
                    $t.dist = p.dist +$ G.getEdgeWeight(p,t);
                    $t.pred =$ p;
                    changed = True;
                }
            }
        }
        if(changed == False) break;
    }
}
```