Data race
At least two accesses to the same location in memory that are "simultaneous" unless >= 1 of the following holds:
- all the accesses are reads
- all the accesses are using atomic operations
- there is a "happens before" relationship between the accesses

```c
// thread 2
if (!ready) {
    // ready is false
    while (true) {
    }
    process(graph);
} else { // ready is true
    process(graph);
}
```

```c
// global
Graph* graph;
bool ready;

// thread 1
tmp = big_computation();
graph = tmp; // relaxed
ready = true; // release

// thread 2
while (!ready) { // acquire
    // wait
}
process(graph); // relaxed
```

```c
bool x = false;
bool y = false;
bool z = false;

// thread 3
while (!x) { // seq cst
    // wait
}
// x is true
if (!y) { // seq cst
    printf("thread 1 happened first\n");
} else {
    z = true; // release
}
```

```c
// after all threads are joined
assert(z); // crash -> acquire
// it is possible for both prints to execute

bool x = false;
bool y = false;
bool z = false;

// thread 4
while (!y) { // seq cst
    // wait
}
// y is true
if (!x) { // seq cst
    printf("thread 2 happened first\n");
} else {
    z = true; // release
}
```

Sequential consistency
- guarantees that there is a single global modification order among "sequentially consistent" operations

Acquire (applies to loads)
- no memory operations may be moved up past the load
Release (applies to stores)
- no memory operations may be moved down past the store