T1: Find a schedule that causes a deadlock using the two-phase locking algorithm

```
begin
write C
read B
write C
commit
```

T2:

```
begin
write B
read C
read C
commit
```

Find a schedule that causes a deadlock

Assume all **exclusive locks**

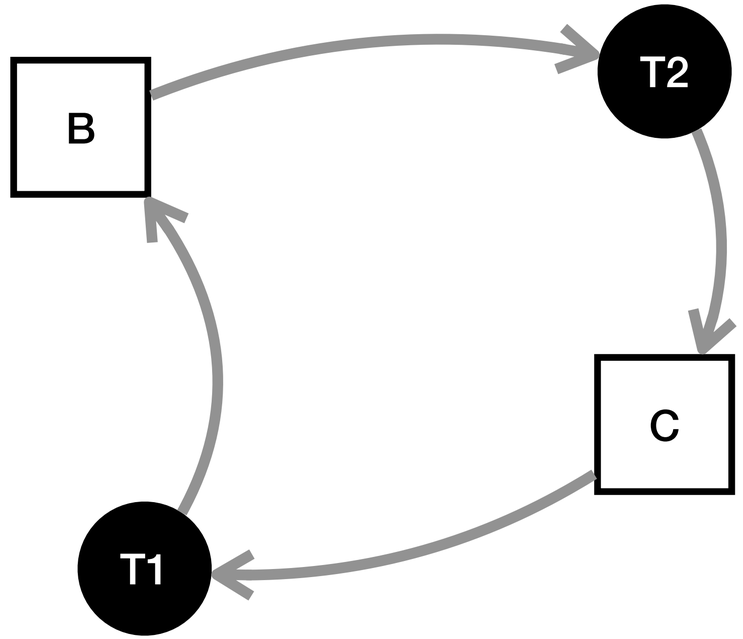
T1:

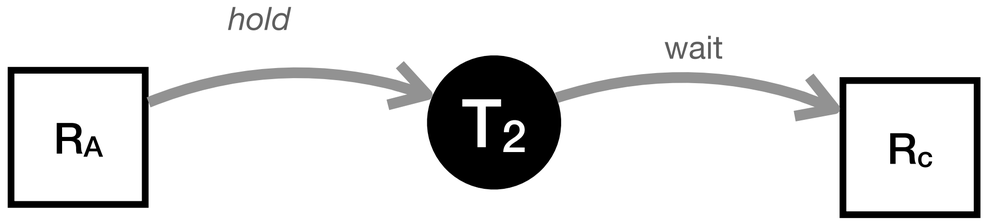
```
begin
write C
read B
write C
commit
```

T2:

```
begin
write B
read C
read C
commit
```

T1	T2
Write C	
	Write B
Read B (blocked)	
	Read C (blocked)
Write C	
	Read C





Find a schedule that causes a deadlock

Assume all **exclusive locks**

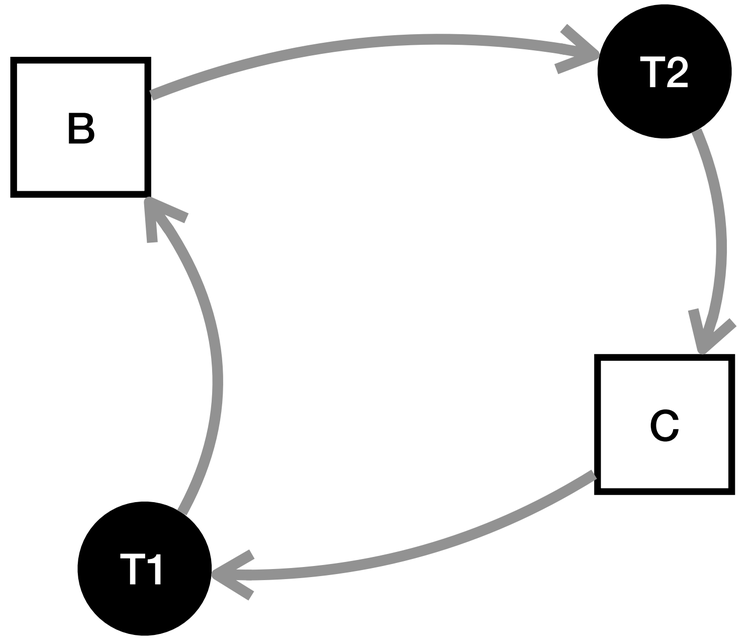
T1:

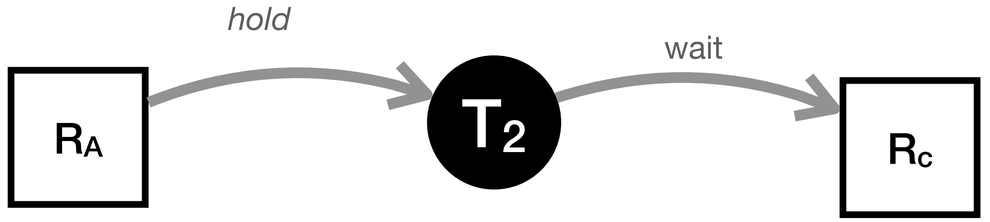
```
begin
write C
read B
write C
commit
```

T2:

```
begin
write B
read C
read C
commit
```

T1	T2
Get C	
Write C	
	Get B
	Write B
Get B - Blocked	
Read B	
Release B	
	Get C - Blocked
	Read C
	Read C
	Release C
Write C	
Release C	





Find a schedule that causes a deadlock

Assume all **exclusive locks**

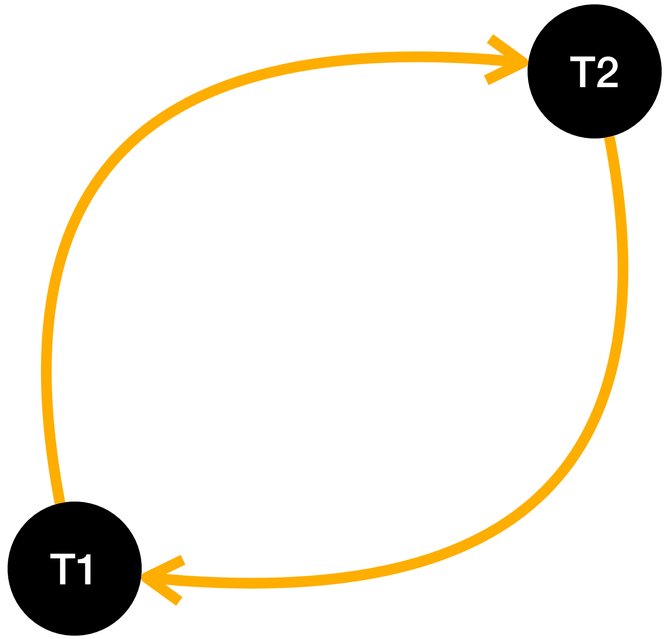
T1:

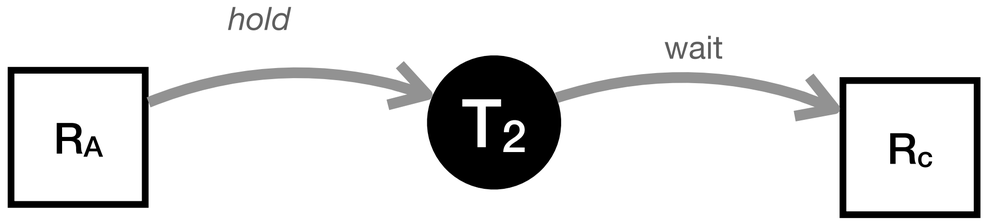
```
begin
write C
read B
write C
commit
```

T2:

```
begin
write B
read C
read C
commit
```

T1	T2
Write C	
	Write B
Read B (blocked)	
	Read C (blocked)
Write C	
	Read C





Find a schedule that causes a deadlock: this is also 2PL compliant

Assume all **exclusive locks**

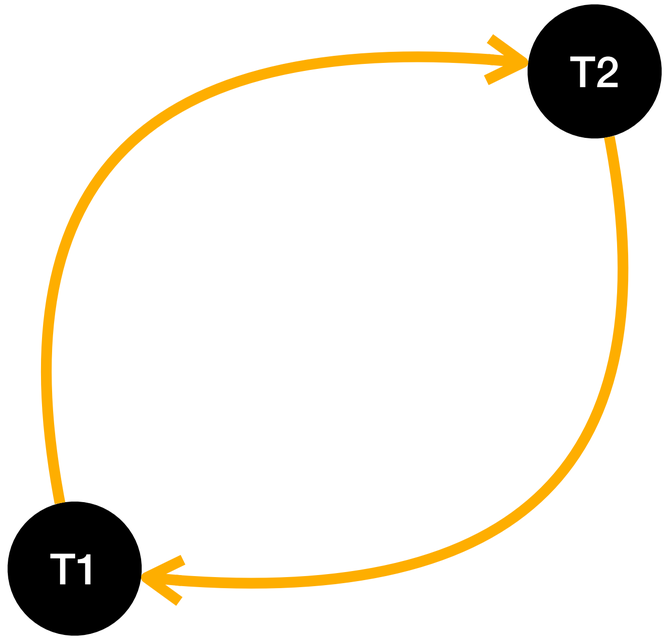
T1:

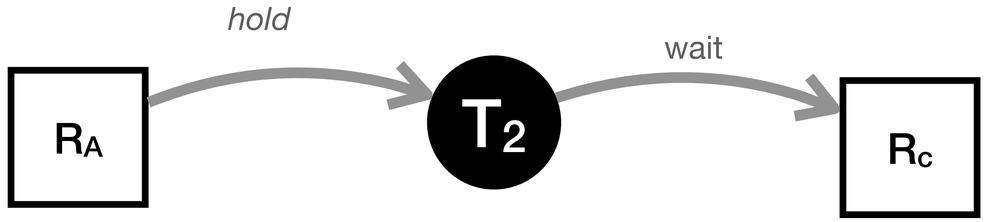
```
begin
write C
read B
write C
commit
```

T2:

```
begin
write B
read C
read C
commit
```

T1	T2
Write C	
	Write B
Read B (blocked)	
	Read C (blocked)
Write C	
	Read C





T1:

```
begin
write C
read B
write C
commit
```

T2:

```
begin
write B
read C
read C
commit
```

This is not a 2PL compliant schedule because T1 releases C at t 3 and then gets it back at t 13

T1	T2
Get C	
Write C	
Release C	
	Get B
	Write B
	Release B
Get B	
Read B	
Release B	
	Get C
	Read C
	Read C
	Release C
Get C	
Write C	
Release C	

Assume all **exclusive locks**

Find a schedule with **non-repeatable reads**

(assuming no concurrency control)

T1	T2
	Write B
	Read C
Write C	
Read B	
Write C	
	Read C

T1	T2
Write C	
Read B	
	Write B
	Read C
Write C	
	Read C