1. Precedence

Match (c:Case)-[r1:Case\_To\_Event]->(a:Event{name:'B'})

where not exists((c)-[:Case\_To\_Event]->(:Event{name:'A'}))

return distinct c.name

union all

Match (a:Event{name:'A'})<-[r2:Case\_To\_Event]-(c:Case)-[r1:Case\_To\_Event]->(b:Event{name:'B'})

where toInteger(r2.position) < toInteger(r1.position)

and ((tofloat(r1.startTime) - tofloat(r2.startTime))/3600)\*60 > $xtime

return distinct c.name

1. Response

Match (c:Case)-[r1:Case\_To\_Event]->(a:Event{name:$'A'})

where not exists((c)-[:Case\_To\_Event]->(:Event{name:'B'}))

return distinct c.name

union all

Match (a:Event{name:'A'})<-[r2:Case\_To\_Event]-(c:Case)-[r1:Case\_To\_Event]->(b:Event{name:'B'})

where toInteger(r1.position) > toInteger(r2.position)

and ((tofloat(r1.startTime) - tofloat(r2.startTime))/3600)\*60 > $xtime

return distinct c.name

1. Response with exclude

Match

(:Event{name:'A'})<-[r1:Case\_To\_Event]-(c:Case)-[r2:Case\_To\_Event]->(:Event{name:'B'}),

(:Event{name:'C'})<-[r3:Case\_To\_Event]-(c:Case)

where tointeger(r1.position) < tointeger(r3.position) and tointeger(r3.position) < tointeger(r2.position)

Return distinct c.name

1. co-Existence

Match (c:Case)-[r1:Case\_To\_Event]->(a:Event{name:'A'})

where not exists((c:Case)-[:Case\_To\_Event]->(:Event{name:'B'}))

return distinct c.name as casename

union all

Match (c:Case)-[r1:Case\_To\_Event]->(b:Event{name:'B'})

where not exists((c:Case)-[:Case\_To\_Event]->(:Event{name:'A'}))

return distinct c.name as casename

1. Existence

Match (c:Case)-[r1:Case\_To\_Event]->(a:Event{name:'A'})

with count(a) as count, c.name as casename

where count < $nTime return distinct casename

1. Alternate Precedence

Match (c:Case)-[r1:Case\_To\_Event]->(a:Event)

where (a.name='A' or a.name= 'B')

with c.name as casename,[a.name,r1.position] as data order by data[1] with casename,collect(data) as activ

unwind range(0,size(activ)-1) as i

with casename,activ[i] as acta, activ[i+1] as actb ,activ[i+2] as actb1

where (acta[0] <> actb[0] and acta[1]< actb[1] and acta[0]='A' and actb[0]='B' and actb[0]=actb1[0] )

return distinct casename

1. Alternate Response

Match (c:Case)-[r1:Case\_To\_Event]->(a:Event)

where (a.name='A' or a.name= 'B')

with c.name as casename,[a.name,r1.position] as data order by data[1]

with casename,collect(data) as activ

unwind range(0,size(activ)-1) as i with casename,activ[i] as acta, activ[i+1] as acta1 ,activ[i+2] as actb

where (acta[0] <> acta1[0] and acta[0] <> actb[0] and acta[0] ='A')

return distinct casename