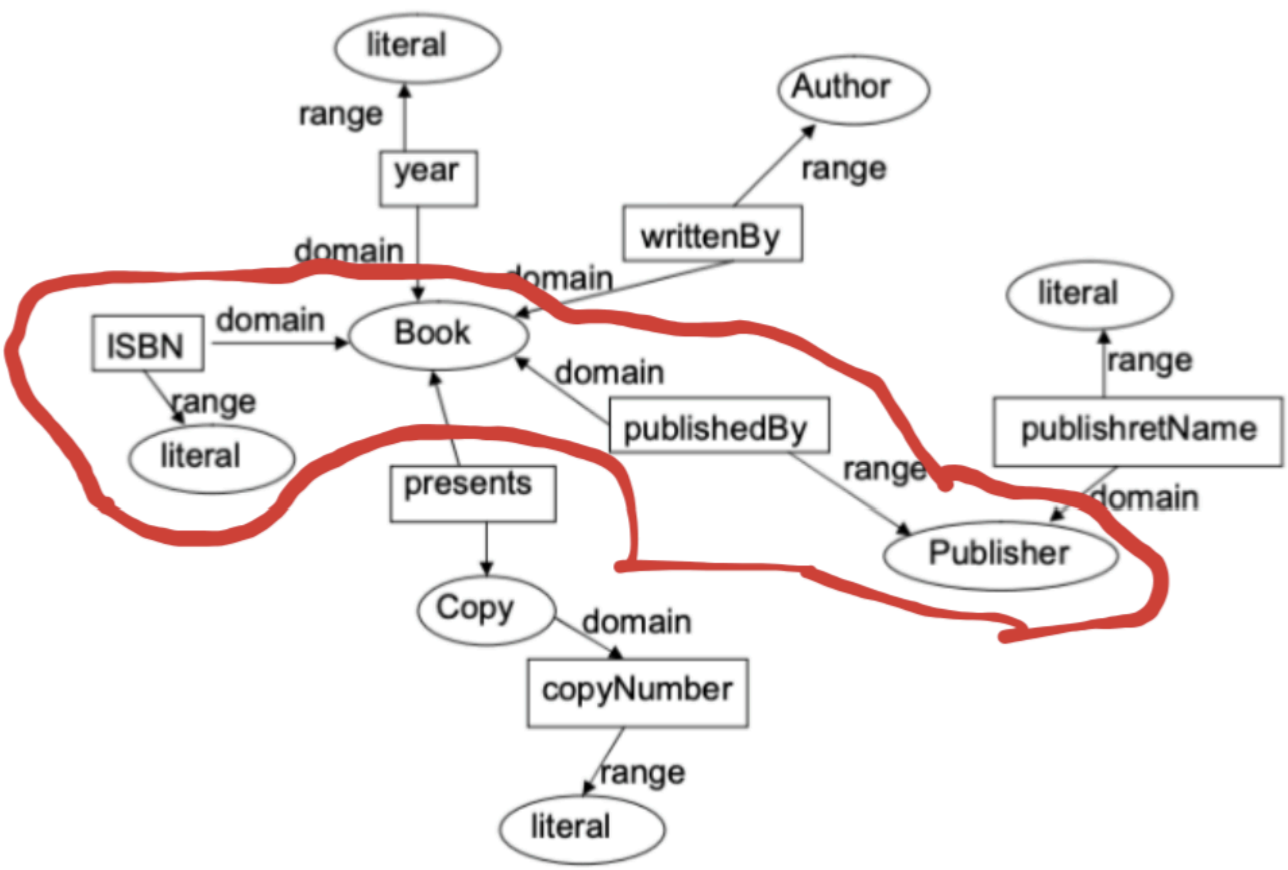
# 1

Consider the following RDF graph, a graph representation of an RDF Schema for a library. A fraction of XML/RDF statements are provided below to model parts of the above RDF schema.

## a

Find and mark all the nodes that are classes and properties in the given RDF statements (you may directly mark them in the figure).




Markup

> Classes: Book, Publisher
>
> Properties: ISBN, publishedBy

## b

Complete the rest of the RDF statements so it covers all the classes, property and domain/range relations in the RDF graph.

```
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">

    <rdfs:Class rdf:ID="author">
        <rdfs:comment>The class of authors</rdfs:comment>
    </rdfs:Class>

    <rdfs:Class rdf:ID="Copy">
        <rdfs:comment>The class of copies</rdfs:comment>
    </rdfs:Class>

    <rdfs:Property rdf:ID="writtenBy">
        <rdfs:comment>Relates books to authors</rdfs:comment>
        <rdfs:domain rdf:resource="#book"/>
        <rdfs:range rdf:resource="#author"/>
    </rdfs:Property>

    <rdfs:Property rdf:ID="presents">
        <rdfs:comment>Relates books to copies</rdfs:comment>
        <rdfs:domain rdf:resource="#book"/>
        <rdfs:range rdf:resource="#copy"/>
    </rdfs:Property>

    <rdfs:Property rdf:ID="year">
        <rdfs:comment>It is a property of books and takes literals as values</rdfs:comment>
        <rdfs:domain rdf:resource="#book"/>
        <rdfs:range rdf:resource="&rdf;Literal"/>
    </rdfs:Property>

    <rdfs:Property rdf:ID="publishretName">
        <rdfs:comment>It is a property of publishers and takes literals as values</rdfs:comment>
        <rdfs:domain rdf:resource="#publisher"/>
        <rdfs:range rdf:resource="&rdf;Literal"/>
    </rdfs:Property>

    <rdfs:Property rdf:ID="copyNumber">
        <rdfs:comment>It is a property of copiess and takes literals as values</rdfs:comment>
        <rdfs:domain rdf:resource="#copy"/>
        <rdfs:range rdf:resource="&rdf;Literal"/>
    </rdfs:Property>

</rdf:RDF>
```

## 2

Consider the following biomedicine knowledge graph G, which describes proteins and the relevant diseases that may be, for example, treated, caused, or expressed by them.

## a

Write SPARQL queries to express the following questions and give their answers. You may define variable names and RDF tags as you want, as long as the queries correctly express the questions.

- “Find all the adverse events that are caused by Arimidex.”

```
SELECT ?v
WHERE {
  ?v rdf:type :AdverseEvent .
  :Arimidex :Causes ?v .
}
```

- “Find the proteins that are associated with the adverse events caused by ’Fulvestrant”.

```
SELECT ?p
WHERE{
  ?p rdf:type :Protein .
  :Fulvestrant :Causes ?e .
  ?e :Assoc ?p .
}
```

- “Find the drugs that cause both Headache and Kidney Infection “

```
SELECT ?drug
WHERE {
  ?drug rdf:type :Drug .
  ?drug :Causes :Headache .
  ?drug :Causes :KidneyInfection .
}
```

- “What are the proteins associated with all the diseases that can be treated by Arimidex”

```
SELECT ?p
WHERE {
  ?p rdf:type :Protein .
  :Arimidex :Treat ?disease .
  ?disease :Assoc ?p .
}
```

- “What are the proteins that can interact with those associated with diseases treated by Fulverstrant”.

```
SELECT ?p
WHERE {
  :Fulverstrant :Treat ?disease .
  ?disease :Assoc ?ap .
```

```
  ?p :Interact ?ap .
}
```

## b

A “knowledge hacker” tries to fool medical AI systems/ChatGPT which consults the above knowledge graph, by injecting fake triples or hiding true facts. He “attacks” G with the following modifications

- Insert a triple <Arimidex, causes, Kidney Infection>

> Query 1 now also accepts Kidney Infection -> Output: KidneyInfection, Headache
>
> Query 3 now accepts both Arimidex and Fulverstrant -> Output: Arimidex, Fulverstrant

- Remove a triple <Fulvestrant, causes, Kidney Infection>

> Query 2 no longer accepts PIM1 since Fulverstrant no longer causes KidneyInfection -> Output: CASP8
>
> Query 3 no longer accepts Fulverstrant -> Output: Arimidex
>
> *Note this is if 1st attack is also in place, otherwise output is blank

- Remove the edges between BRCA1 and ESR2.

> Query 5 no longer accepts BRCA1 -> Output: ESR2, ESR1

How your answers to any of the above queries change, under such an attack? Give the changed answers.

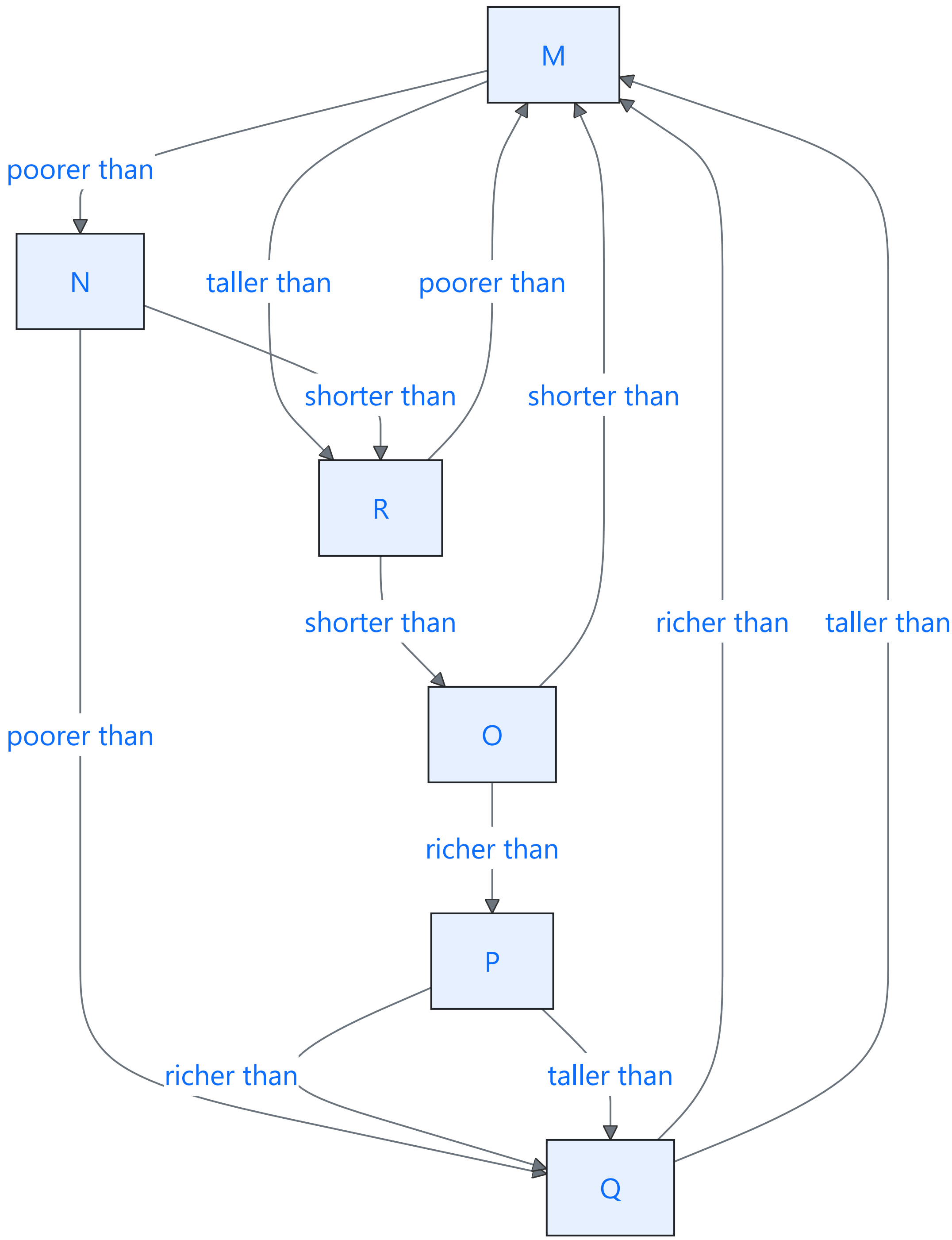
# 3

Among the advantages of representing facts with RDF (knowledge) graphs is its capability of supporting (logic) reasoning. Consider 6 people M, N, O, P, Q and R have the following facts.

- M is poorer than N, and taller than R.
- N is poorer than Q and shorter than R.
- O is richer than P and shorter than M.
- P is both richer and taller than Q.
- Q is richer as well as taller than M.
- R is poorer than M and shorter than O.

## a

Draw a directed graph that can encode the above facts. For convenience, you may consider “reverting” the relation ‘poorer’ to ‘richer’ or vice versa as needed and changing the edge directions.




## b

An (s, r, o) triple in an RDF statement is also simply denoted as r(s,o). A relation r is transitive if r(A,B) and r(B,C) implies r(A,C). Clearly, both “richer” and “taller” relations are transitive. Describe how you can use the graph and above transitivity rule to find the answer to the following questions (which is an example of “symbolic reasoning”):

- Who is the richest person?

> $O$ is the richest.

- Who is the tallest person?

> $P$ is the tallest.

- Find all people who are both taller and richer than M.

> $P$ and $Q$ are both taller than and richer than $M$

- How many people are poorer than Q?

> $M, N, R$ are all poorer than $Q$ so $3$ people are poorer than $Q$.

#### C

Assume we turn the graph G into a "edge weighted" graph, where an edge r(A,B) with a relation r and a weight w suggests A is taller than B by w inches (if r is "taller") or by w dollars (if r is "richer"). Describe what algorithms you can use to find out the answers for the following questions.

- Find out those who are twice as rich as N.

> Use Djikstra's greedy algorithm to always take edges with the greatest increase in wealth from $N$. Unfortunately it is impossible to determine if an edge is twice as rich as $N$ unless we have a base that $N$ (or some other node) has $x$ dollars, otherwise we can only determine that a node is $w_1 + w_2 + \ldots$ richer than $N$.

- Find out those who are at least 0.5 inches shorter than O

> Apply a breadth first search from $O$ and only take the edges for height with negative weighting to find people that are shorter than 0. We can calculate each person's height as we traverse and see if their height is below -0.5 from $O$.