1) Graphen

Gegeben seien folgende Datenstrukturen:

a) Materialfluss in einem Betrieb
b) Stromnetz
c) Freundschaftsbeziehungen innerhalt einer Gruppe von Leuten
d) Geschwister innerhalb der Gesellschaft

Stellen diese Datenstrukturen (im Modell) Graphen dar? Wenn nein, warum nicht?

Wenn ja, kategorisieren Sie diese Graphen nach gerichteter/ungerichteter, gewichteter/ungewichteter, vollständiger/nicht vollständiger, zyklischer/azyklischer und zusammenhängender/unzusammenhängender Graph. Begründen Sie jeweils Ihre Entscheidungen.

Wählen Sie eine geeignete Implementierung für den Graphen aus (Adjazenzmatrix, Adjazenzliste, direkte Objektreferenzen) und erklären warum Ihnen diese Implementierung Ihnen am geeignetesten erscheint.

Hinweis: Es ist nicht so wichtig, wie Sie die einzelnen Datenstrukturen kategorisieren (nicht immer eindeutig, manchmal von der Sichtweise und der Detailtreue des Modells abhängig). Viel wichtiger ist die genaue Begründung für Ihre Kategorisierung.

Lösung:
Es muss begründet werden, warum ein Modell gerichtet/ungerichtet, gewichtet/ungewichtet, etc. ist. Ein "gleiches" Modell kann problemlos unterschiedlich zugeordnet werden, sofern die jeweiligen Begründungen (bzw. Einschränkungen im Modell) korrekt sind.

Zusammenfassung:

Nr	gerichtet	gewichtet	vollständig	zyklisch	zusammenhängend
a	ja	ja	nein	nein	ja
b	nein	nein	nein	ja	ja
c	ja	ja	nein	ja	nein
d	nein	nein	ja	ja	nein

zu a) Materialfluss in einem Betrieb

- Es gibt Arbeitsstationen (Knoten) und Förderbänder für den Materialtransport (Kanten) => Graph
- Die Rohstoffe, Zwischenprodukte und Produkte fließen nur in **eine bestimmte Richtung** auf den Förderbändern => gerichtet
- Die Förderbänder können jeweils nur eine **bestimmte (unterschiedliche) Menge** an Materialen aufnehmen (gleiches gilt für die **unterschiedliche Transportdauer** der Materialien) => gewichtet
- **Nicht jede** Arbeitsstation ist **mit jeder anderen verbunden** => nicht vollständig
- Materialien werden in jeder Arbeitsstation bearbeitet und dadurch "veredelt", im Materialfluss gibt es daher **keine Zyklen** => azyklisch
- Es wird nur ein Produkt hergestellt, es gibt daher nur genau **eine "Senke"** zu der alle Materialien fließen (Ausnahme: Abfall, dieser mündet in der Senke Müllcontainer),

alle Materialflüsse können daher **nicht unabhängig** voneinander fließen => zusammenhängend

Mögliche Implementierung:
Direkte Verpointerung, falls der Graph für die Simulation des Materialflusses eingesetzt werden soll. Jeder Knoten ist durch ein Objekt dargestellt, dass eine bestimmte Arbeitsstation simuliert.

zu b) Stromnetz

- Es gibt Stromverteiler, Umspannwerke, ... (Knoten) und Leitungen (Kanten) => Graph
- Die Stromleitungen haben **keine Richtung**, da der Strom in beide Richtungen gleich gut geleitet wird => ungerichtet
- Der Graph ist nicht gewichtet, da es in diesem Modell **nur um die Verbindung** von Produzenten untereinander und zu den Verbrauchern dargestellt werden soll und nicht z.B. die Verluste und Spannungsabfälle auf den einzelnen Leitungen => nicht gewichtet
- Die Verbraucher bekommen **keine eigene Leitung** zum Kraftwerk => nicht vollständig
- Es gibt **Kreise** im (ungerichteten) Stromnetz (z.B. parallel ausgeführte Leitungen) => zyklisch
- Das Stromnetz **hängt** in diesem Modell **zusammen** => zusammenhängend

Mögliche Implementierung:
Netzplan dargestellt als Adjazenzliste, da in diesem Fall nur die Verbindungen der einzelnen Knoten interessant sind und von einem Knoten üblicherweise nicht allzu viele Kanten (geringerer Speicherbedarf bei Adjazenzliste als bei Adjazenzmatrix) ausgehen.

zu c) Modell der Freundschaftsbeziehungen innerhalb einer Gruppe von Leuten

- Menschen (Knoten), Freundschaftsbeziehungen (Kanten) => Graph
- Die Freundschaftsbeziehungen sind **nicht** immer **symmetrisch** => gerichtet
- Freundschaftsbeziehungen sind **stärker und weniger stark ausgeprägt** => gewichtet
- **Nicht alle** sind miteinander befreundet => nicht vollständig
- "Freundes**kreise**" => zyklisch
- Einige Cliquen sind möglicherweise **nach außen abgeschottet**, manche Menschen leben als **Ein**siedler, ... => nicht zusammenhängend

Mögliche Implementierung:
Freundschaftsbeziehung als Adjazenzmatrix. Gut geeignet, wenn alle im Schnitt viele Freundschaften (wenn auch viele mit geringem Gewicht z.B. bei "entfernten Bekannten" – Somit wäre die Modellierung einer kleineren Gemeinde ein sehr dicht besetzter Graph) haben. Die Elemente der Matrix sind gewichtet (d.h. es sind z.B. vom Typ Integer). Eventuell eine Kompression der Matrix sinnvoll.

zu d) Modell der Geschwister in einer Gesellschaft

- Person (Knoten), ist Schwester/Bruder von (Kanten) => Graph
- Alle Geschwister sind **gegenseitig in gleicher Weise** verwandt (Annahme: Halbgeschwister seien dabei nicht berücksichtigt) => ungerichtet
- Geschwister sind untereinander gleich verwandt unterliegt den **gleichen Bedingungen** => nicht gewichtet
- Jeder Teilgraph besteht aus 1 oder mehr Knoten (die **alle miteinander direkt verbunden** sind) => vollständig

- Eine Schwester/Bruder einer/s Schwester/Bruder einer bestimmten Person ist auch Schwester/Bruder dieser Person (**jeder ist mit jedem gleich** verwandt, ab 3 Geschwistern gilt das dann im Kreis) => zyklisch
- Es gibt mehrere Familien und somit **mehrere Geschwisterbeziehungen mit unterschiedlichen Eltern** => nicht zusammenhängend

Mögliche Implementierung:
Geschwister-Graph als Adjazenzliste. Der Graph ist innerhalb von Familien zwar vollständig und benötigt viele Kanten, bei einem großen Graph (vielen Menschen) ist das relativ gesehen aber nicht sehr viel.

2) Eigenschaften von Graphen

	Ja	Nein
DFS und BFS taugen beide nicht zur Suche nach dem kürzesten Weg von A nach B in einem gewichteten Graphen	X	
Liegt in einem Graphen jeder Knoten auf einem Kreis, so ist die Anzahl der Kanten größer als die Anzahl der Knoten		X
Fügt man zu einem beliebigen spannenden Baum eine Kante hinzu, so erhält man immer genau einen Kreis	X	

3) BFS/DFS

Es soll in einem beliebigen ungewichteten, ungerichteten Graphen zwischen 2 Knoten der kürzeste Pfad ermittelt werden.
Vergleichen Sie DFS und BFS. Welche Methode ist für diese Aufgabe besser geeignet? Welche Nachteile gibt es jeweils? Gibt es Möglichkeiten diese Nachteile zu vermeiden? Gibt es für solche Aufgabenstellungen u.U. zusätzliche Optimierungen?

Lösung:
BFS, da gleichmäßig und gleichzeitig in alle Richtungen gesucht wird (Begrenzt von Saumknoten). Sobald der gesuchte Knoten gefunden wurde ist der Rückweg auch schon der kürzeste.
Nachteil: Man muss sich viele Saumknoten merken.

DFS nicht geeignet, da u.U. sehr viel mehr Knoten durchsucht werden müssen (Vor allem wenn man zuerst in die falsche Richtung sucht, hat man schon einen Weg gefunden, so kann man die Tiefensuche auf die aktuelle Tiefe beschränken).
Abhilfe: "Iterative Deepening" – es wird zuerst mit max. Tiefe 1 gesucht, wenn nicht gefunden mit max. Tiefe 2, dann Tiefe 3, etc. D.h. es wird ein BFS mittels modifiziertem DFS simuliert. Vorteil: Man muss sich nicht so viele Knoten merken. Nachteil: Wege müssen mehrfach durchlaufen werden (oft aber kein wirkliches Problem).

Optimierungen:
BFS: Man sucht von beiden Knoten gleichzeitig weg und trifft sich in der Mitte (nicht immer möglich, "Rückwärtssuche" muss möglich sein).

Kennt man ungefähr die Richtung in die man Suchen muss, so beschleunigt sich die Sache mitunter sehr stark (=> Priority first search).