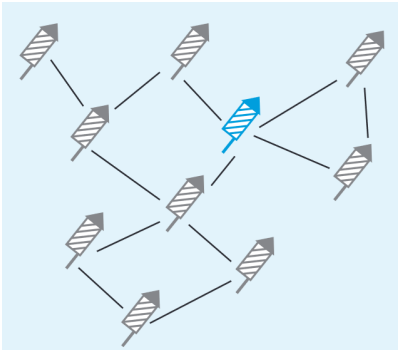
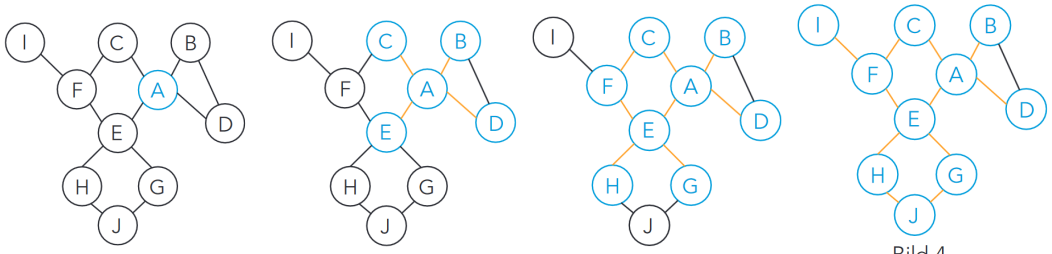
3-3 Breitensuche auf Graphen



An Sylvester möchte die Stadt ein großes Feuerwerk veranstalten und hat hierzu mehrere Batterien mit Zündschnüren zusammengeschalten:

A0) Bestimme welche Batterien zuletzt zünden

A1) Beschreibe den Ablauf der Breitensuche ausgehend von Knoten A.



- Von Knoten A aus gelangt man über die von A _____ zu den **Nachbarknoten** _____
- B und D haben keine weiteren _____ Nachbarknoten
- C und E zünden weitere Batterien: _____
- Zuletzt zünden dem Prinzip folgend: _____

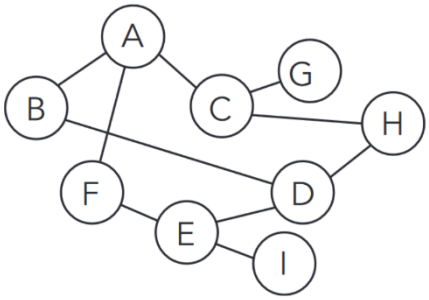
Die **Breitensuche** ist ein Verfahren zur **Traversierung** eines (vorrangig ungerichteten) Graphen. Beginnend bei einem Startknoten werden zunächst alle Nachbarn in eine Warteschlange gelegt und anschließend besucht. Sie werden markiert und anschließend all deren Nachbarn, die noch nicht besucht waren zur Warteschlange hinzugefügt, um im nächsten Schritt besucht zu werden. Somit wird die Pfadlänge mit jeder **Iteration** _____.

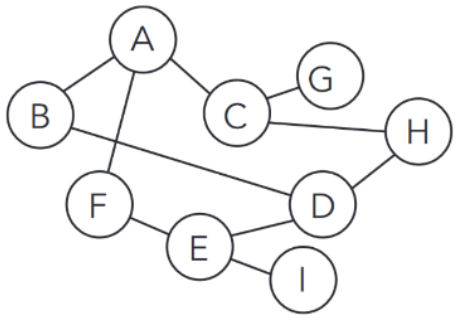
A2)

a) Führe die Breitensuche ausgehend von Knoten A und E aus. Gib jeweils für jeden **Iterationsschritt** die Knoten an.

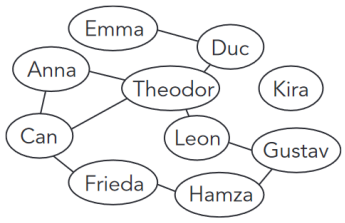
b) Entwickle ausgehend vom Startknoten A und E (Rückseite) einen gerichteten Graphen, der die Reihenfolge des Graphendurchlaufs veranschaulicht.

c) Anton behauptet: „Unabhängig vom Graphen werden bei der Breitensuche alle Knoten besucht.“ Diskutiere die Aussage mit deinem Nachbarn und nimm zu dieser Stellung.





A3) Neben der Breitensuche gibt es auch noch die **Tiefensuche**. Recherchiere im Internet den Ablauf und führe diese beginnend mit Theodor aus. Gib eine mögliche Reihenfolge, in der die Knoten besucht werden, an.



A4) Öffne das Projekt 11-3-3 in der online-ide und untersuche welche Bestandteile des Pseudocodes bereits vorhanden sind. Bearbeite anschließend die TODOs

Pseudocode-Algorithmus für die Breitensuche

```
breitensuche(int start, int ziel)
Liste für besuchteKnoten
Liste für warteschlange
Zählvariable für aktuellerKnoten
aktuellerKnoten in die Warteschlange abspeichern

## solange aktuellerKnoten nicht zielKnoten ist
   -- Durchlaufe alle Verbindungen (von aktuellerKnoten aus) in adja
        ** Prüfe ob Nachbarknoten des aktuellerKnoten zur warteschlange
        hinzugefügt werden soll (-> Verbindung in in ajda & nicht in
        besucht & nicht in warteschlange)

           Verbindung zur warteschlange hinzufügen

        ** ende wenn
   -- Ende Wiederholung mit fester Anzahl

   Füge aktuellerKnoten zu besucht hinzu
   Setze aktuellerKnoten auf nächstes element in warteschlange

## Ende bedingte Wiederholung
```

Weitere Veranschaulichung: https://www.geeksforgeeks.org/breadth-first-search-or-bfs-for-a-graph/