Detaillierte Rückmeldungen - Semesterarbeit S.Schellenberg, J.Popp

Zusammenfassung:

Die Konzeption der Unterrichtseinheit beinhaltet die wesentlichen Teile ist grundsätzlich gepflegt. Bei der Reflexion zum Vorwissen sollte noch nachgebessert werden. Es tauchen in der LPU Begriffe und Konzepte auf, die in der Planung (Concept Map) nicht vorhanden sind bzw. vorausgesetzt werden wie zum Beispiel die *Warteschlange*, die *Adjazenzliste* oder nicht ausführlich genug aktiviert werden (*Adjazenzmatrix und deren Implementierung mit Python*). Die *Adjazenzmatrix* ist zu diesem Zeitpunkt (2. Quartal des Ergänzungsfachs) wohl allgemein noch nicht bekannt. Aber wir können dies der Einfachheit halber einmal annehmen. Ansonsten reicht die Zeit nicht aus, um den Weg umzusetzen, den sie eingeschlagen haben. Die Wiederaktivierung nimmt dementsprechend auch viel Zeit in Anspruch (sehr technisch). Eine eingehende Thematisierung der *Warteschlange (Queue)* bzw. *Adjazenzliste* wäre sicher etwas einfacher sinnvoller.
Die Planung der Unterrichtssequenz könnte noch etwas mehr herausgearbeitet werden, allenfalls mit benötigten Zeiteinheiten. Die Erreichung der Lernziele wird in 90 Minuten mit vorliegendem Umfang eher knapp.
Der Start in die LPU bzw. die Repetition wäre mit einem *konkreten Beispiel* bzw. *einer spannenden Fragestellung* besser. Die Repetition zu den Graphen und Matrizen erfolgt so ohne ausreichende Motivation. Die SuS repetieren „natürlich" und „automatisch" mit einer gegebenen Problemstellung.
Gut gelungen und spannend aufgebaut ist die schrittweise Entwicklung des Algorithmus am Beispiel eines Sozialnetzwerks (auch wenn das kürzeste-Wege-Problem an anderen Beispiel sicher eine gute Alternative wäre).
Gewisse Übungen mit den Tramverbindungen sind zu wenig durchdacht, da ein viel zu grosser Graph entsteht.
Die Berechnung der Laufzeit etc. ist im 2. Quartal auch verfrüht (hier aber vorausgesetzt)

Fachliche Korrektheit:	5.00
Fachdidaktische Qualität:	4.50
Gesamtbewertung:	4.75

Zur Konzeption

Zu 1.1. bzw. 1.2.2

1) → Was genau meinen sie mit *Input* und *Output* als neue Konzepte?
→ *Adjazenzmatrizen* setzen sie als gefestigtes Vorwissen aus dem 1. Quartal voraus. In der Regel wird die Zeit im 1.Quartal dafür nicht reichen um eine gefestigten Umgang zu garantieren. Hier bedarf es, wenn überhaupt, einer ausführlicheren Aktivierung. Die Implementierung mit all den Operationen auf Matrizen ist anspruchsvoll. → Mein Vorschlag wäre: *Adjazenzmatrizen* optional als weiterführende Struktur zu behandeln und in der LPU wegzulassen.
→ Weiter setzen sie ein grundlegendes Verständnis der *Komplexitätsanalyse* voraus. Die Analyse von Programmen wie in Aufgabe 2.20 bzw. 2.21 werden die SuS aber wahrscheinlich überfordern.
→ Die *Warteschlange (Queues)* als Repräsentation von Knotenmengen und allenfalls das *kürzeste Wege Problem* tauchen als wichtige Konzepte hier (Concept Map) nicht auf...
→ *Adjazenzliste* ? (Aufg. 2.5)

2) → Voraussetzung *einfache* Programme in Tigerjython. Die gestellten Aufgaben mit Matrixoperationen (Listen in Listen) sind relativ komplex.

3) → Abschnitt 1.2.1 beinhaltet auch Vorwissen (und ist in 1.2.2 repetitiv) und Abschnitt 1.2.2 bereits ein Aufbau der Unterrichtssequenz, was warum gemacht wird. (evtl. Titel/Struktur anpassen?)
→ *Pseudocode* fehlt in Vorwissen.
→ *Adjazenzliste* Vorwissen? (Aufgabe 2.5)
→ *Wartschlange* Vorwissen? (Aufgabe 2.9)

Zu 1.4

4) → Dieser Abschnitt ist die eigentliche Sequenzierung der Unterrichtseinheit. (Titel?)
Zeitliche Angaben nicht zwingend, würden aber helfen, abzuschätzen, wie realistisch die LPU wirklich durchführbar ist.
→ Einstieg mit Repetition...warum? Motivation? Hier wäre eine konkrete Problemstellung, welche die Breitensuche aufs Tapet bringt, wünschenswert. Zum Beispiel mittels IU.

Zu den Unterrichtsunterlagen

→ Concept Map nicht abdrucken**.**

5) → *Konkrete Problemstellung* welche SuS „abholt" (Vorwissen), kognitiv aktivierend ist und zur Breitensuche führt!
Zum Beispiel mit einem **IU** . Die Bemerkung „Eine interessante Frage ist immer über wie viele Verbindungen zwei Objekte miteinander verknüpft sind..." warum? Was sind Objekte (zu abstrakt)
6) Wenn vorgängig eine Problemstellung vorhanden ist, lässt sich das Vorwissen daran aktivieren. Abschnitt würde ich weglassen bzw. kürzen. SuS überlesen vieles oder schalten ab.

Zu 2.1

7) → allenfalls aufbauend auf Problemstellung (siehe 6)) können die Begriffe konkret erklärt werden. Sie könnten dies dann mit Beispiel 2.2. verknüpfen.
→ eigene, einfachere Formulierung eines Graphen (diese hier ist 1 zu 1 aus Wikipedia übernommen)

8) → Die Repetition der Adjazenzmatrix $A=a_{ij}$ ist zu knapp und formal. Die SuS wissen wahrscheinlich sicher nicht mehr (oder noch nicht), was die Indizes bedeuten. Machen sie ein anhand ihrer Beispiels 2.3 noch eine Ergänzung zu einigen Kanten und z.B. a_{01} =1 weil es eine Kante von 0 → 1 gibt etc.
Zudem müssen die SuS auf den Beginn bei i=0 und j=0 sensibilisiert werden. a_{00} !
Zudem wird ja {0,2} und {2,0} in der Matrix abgebildet....was die SuS immer verwirrt.
→ Der Fehler in Beispiel 2.4 wird die SuS stark verunsichern.

→ Wie wird eine Matrix in Tigerjython gespeichert? [[1,2] , [3,4], ..] unbedingt exemplarisch

9) → gehört diese Aufgabe zu den Lernzielen?
→ Kennen die Schüler kennen den Begriff *Adjazenzliste*? Steht jedenfalls nicht im Vorwissen!
→ Aufgabe ausser Kontext und für Fortgeschrittene. Wie wird dann ein Graph in einer Textdatei gespeichert? An dieser Stelle Arbeiten sie direkt mit Listen im Programm anstatt Einlesen eines Graphen. Oder geben sie den SuS diese Programm.

10) → Benutzen sie konsequent die gleichen Begriffe. Schreiben sie Knoten (anstelle von Vertices)

11) → Letzter Satz ausarbeiten: welcher Knoten ist nun Nachbar in einem gerichteten Graphen?

12) Im Beispiel haben sich zwei **Fehler** eingeschlichen. Sie verweisen hier auf die zweite Zeile der Matrizen, obwohl es die dritte ist. Sie haben hier fälschlicherweise die Indizes gemeint. Benutzen sie(wenn schon) auch die mathematische Schreibweise a_{20}=1 etc. oder direkt die Listenschreibweise
M[2][0]

Zu 2.2 Breitensuche

→ Dieses Kapitel baut auf abstrakten Beispielen auf. Sie müssen das nicht ändern, aber auch hier wäre ein konkretes Beispiel (z.B. ein Labyrinth zielführender. Zum Beispiel was den Begriff Traversieren oder Erreichbarkeit betrifft. Das würde sich den SuS in einem Labyrinth sofort erschliessen.

13) Was ist ein Graphenalgorithmus? Wie wendet man ihn auf einen Graphen an? Markierter Satz höchst unverständlich. Vorschlag: lassen sie diesen Abschnitt weg.

14) Das Ziel ist es, einen Algorithmus zu entwickeln, der entscheidet, ob ein Knoten von einem Startknoten erreichbar ist. Im folgenden Abschnitt schwenken sie dann gleichzeitig zum kürzesten-Wege-Problem. Bearbeiten sie doch zuerst die Erreichbarkeit und kommen dann über den existierenden Weg zum kürzesten. Oder sie setzen die Erreichbarkeit voraus und behandeln das kürzeste-Wege-Problem.

15) Bei Aufgabe 2.8 stellen sie eine kognitiv aktivierende Aufgabe. Leider „verraten" sie doch etwas viel beim anschliessendes Tipp (eigentlich die Lösung)

16) → Hier taucht die *Warteschlange* auf, die sie nicht weiter herausarbeite. Im Pseudocode verwenden sie die Operationen pop() und push(). Machen sie hier unbedingt ein konkretes Beispiel einer Warteschlange mit konkreten Knoten. Was heisst entfernen und einfügen in eine Warteschlange FIFO etc.
→Bemerken sie noch, was die genaue Aufgabe (bzw. Ausgabe) des Algorithmus 1) sein soll.

17) → Nummerierung der Namen? Hier gäbe es sonst unzählige unterschidliche Adjazenzmatrizen.
Geben sie hier die Nummerierung vor: Tom:0, John :1 oder so ähnlich.
→Eingabe Liste anstatt „einlesen"

18) Erklärung ist gut, aber auch hier wäre ein konkretes Beispiel mit zwei Zwei Listen Knoten und Entfernungen und einem Graphen eingängiger.

19) → Diese Aufgabenstellung bedarf einer konkreteren Hilfestellung als es der Hinweis bietet. Ein konkretes Beispiel. Formulieren sie die Aufgabenstellung so, dass eine Funktion weg_zum_ziel (vorgaenger, ziel) zu erstellen und diese dann in der breitensuche zu verwenden ist.

zu 2.2.4 Beispiele
→ Idee grundsätzlich gut, aber zu umfangreich für Modellierung mit Graph. Einfacheres Beispiel!
20) → Die Lösung zur Aufgabe 2.16 fehlt. Das verwundert nicht, denn wieder soll der Graph mittels 31 x 31 Adjazenzmatrix modelliert werden!
In der Aufgabe 2.17 müssen die SuS lediglich beantworten, ob Enge erreichbar ist! Wozu der Aufwand mit dem Programm, wenn man dies auf einen Blick beantworten kann? (nicht kognitiv aktivierend)

21) Anstelle von Antwortsätzen wie in 2.16 den Graphen und 2.17 bitte den Weg über die Stationen.