Depth-First-Durchlauf eines Graphen

Schreib eine rekursive Funktion, die die Knoten eines gerichteten Graphen *in Tiefensuch-Reihenfolge durchnummeriert*. Weiters soll in jedem Knoten die *Anzahl der eingehenden Kanten* gespeichert werden.
Der Graph ist intern mit *Adjazenzlisten der ausgehenden Kanten* gespeichert:

- Jeder *Knoten* ist eine *Struktur* **`node`** mit 3 Membern:
  Dem Zeiger auf seine *Adjazenzliste* (**`edges`**),
  einem **`int`**-Member **`dfs_nr`** für die zu berechnende *DFS-Reihenfolge-Nummer* und
  einem **`int`**-Member **`in_cnt`** für die zu berechnende *Anzahl eingehender Kanten*.
- Die *Kanten-Strukturen* **`edge`** in den Adjazenzlisten bestehen aus
  einem Zeiger **`neighbor`** auf den *Zielknoten* der Kante
  und der *Listenverkettung* **`next`** der Adjazenzliste.

Die Funktion **`dfs`** wird mit einem *Pointer auf den aktuellen Knoten* des BFS-Durchlaufes und der *nächsten zu vergebenden DFS-Reihenfolge-Nummer* aufgerufen
(das *Hauptprogramm* macht den ersten Aufruf mit dem *Startknoten* und der Nummer **`0`**).
Sie soll alle vom aktuellen Knoten ausgehend erreichbaren Knoten des Graphen nach dem im Skriptum angegebenen *rekursiven DFS-Verfahren besuchen* und dabei zwei Dinge erledigen:

- Die *nächste zu vergebende DFS-Nummer* verwalten und im Member **`dfs_nr`** die *Knoten nummerieren*: Der *aktuelle Knoten* bekommt die *nächste Nummer*, und alle *rekursiven Aufrufe* geschehen mit einer entsprechend *erhöhten Nummer*.

  Als *Returnwert* gibt jeder Aufruf die *erste von ihm nicht benutzte Nummer* zurück.

  Die Funktion darf sich darauf verlassen, dass vor dem ersten Aufruf der Funktion in allen Knoten das Member **`in_cnt`** auf **`0`** und das Member **`dfs_nr`** auf **`-1`** *initialisiert* wurden. Das Member **`dfs_nr`** kann und soll daher gleichzeitig als *"visited"-Flag des Algorithmus* verwendet werden.
- Das *Feld* **`in_cnt`** der besuchten Knoten befüllen: Bei jedem Knoten, zu dem eine Kante vom aktuellen Knoten führt, wird es um **`1`** *erhöht*.
- Wird **`NULL`** als Startknoten übergeben, soll die Funktion sofort die nächste Nummer unverändert zurückgeben.

Du kannst entweder mein **Hauptprogramm** verwenden oder es selbst versuchen:

- Das Programm wird mit *zwei Zahlen* auf der Befehlszeile aufgerufen:
  Der *Anzahl der Knoten* im Graph und der *Nummer des Startknotens*
  (die Knoten sind von **`0`** bis **`n-1`** durchnummeriert).
- Die *Knoten* werden intern in einem *Array* gespeichert,
  das Array wird *dynamisch* in der richtigen Größe angelegt.
  Die **`dfs_nr`** aller Knoten wird auf **`-1`** initialisiert, **`in_cnt`** auf **`0`**.
- Die *Kanten* des Graphen werden *vom Terminal eingelesen* und in *Adjazenzlisten* (mit einzeln dynamisch angelegten Listenelementen) gespeichert:
  Die Eingabe der Kanten erfolgt in der Reihenfolge der Ausgangsknoten
  (zuerst die Kanten ausgehend von Knoten 0, dann jene von Knoten 1, usw.).

Eine Kante wird durch die *Knotennummer des Zielknotens* eingegeben,
die Liste der Kanten jedes Knotens wird durch die Eingabe von **`-1`** abgeschlossen.

Die *Reihenfolge* der Nachbarn eines Knotens in der Eingabe muss erhalten bleiben
und beim Depth-First-Durchlauf berücksichtigt werden
(der zuerst eingegebene Nachbar wird zuerst besucht).

- Dann wird die *DFS-Funktion* mit dem Startknoten und **`0`** aufgerufen
  und ihr Ergebnis (erste unbenutzte Nummer) ausgegeben.
- Am *Ende* des Programmes werden die Knoten *in Einlese- bzw. Array-Reihenfolge*
  ausgegeben, für jeden Knoten wird seine fortlaufende *Nummer*,
  seine *DFS-Reihenfolge-Nummer* und seine *Anzahl eingehender Kanten* angezeigt.