Aufgabe 1. Implementiert eine Datenstruktur für Graphen als Adjazenzliste. Schreibt eine Funktion, um Graphen mit Kantengewichten aus einer Datei nach dem in der Vorlesung besprochenen Format einzulesen. In weiser Voraussicht könnte diese Einleseroutine den Index des Knotens `s` zurückliefern, auch wenn dieser für die unmittelbar folgenden Aufgaben noch nicht relevant ist.

Aufgabe 2. Schreibe eine Funktion, die irgendeinen gerichteten Kreis in einem Graphen in linearer Zeit findet und diesen ausgibt. Implementiere dazu **D**epth **F**irst **S**earch (DFS) mit einem Callback-Argument:

```
typedef int (*DFS_CALLBACK)(
        GRAPH *G,
        int v,      /* Knotenindex */
        void *data  /* Zusatzdaten (optional) */
);

/* Führt DFS auf dem Graphen G durch. Bei jedem Knoten
   wird (*cb) aufgerufen, wobei der Knotenindex und
   der data-Pointer übergeben werden. Wenn die
   Rückgabe dieses Aufrufs 0 ist, so soll die Tiefen-
   suche abgebrochen werden. */
void graph_dfs(GRAPH* G, DFS_CALLBACK cb, void *data);
```

Aufgabe 3. Verwendet eure Heaps von gestern, um den Dijkstra-Algorithmus zu implementieren. Auf der Homepage findet ihr eine Graphendatei im besprochenen Format zum Testen.

Bonus! Wir werden morgen zufällig Knotenindizes abfragen, und wer zuerst den Abstand von `s` zu diesem Knoten nennen kann, bekommt eine Coke. **Bonus-Coke!!!**

Aufgabe 4. Kauft euch Firefly und seht es euch an.

Aufgabe 5 (∗)**.** Implementiere auch Breitensuche auf deinem Graphen mit Callback. Dazu musst du deine Listenimplementierung als Queue verwenden.