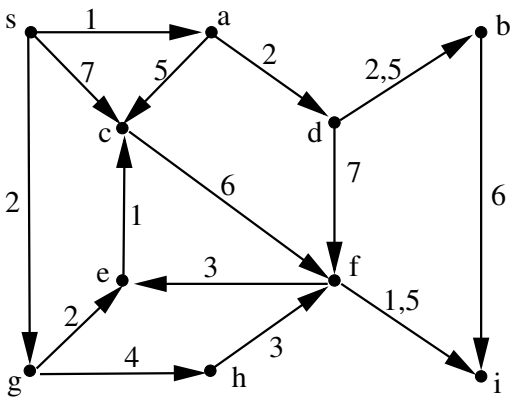
12. Übung **Algorithmen und Datenstrukturen** **WS 2016/17**

Klaus Kriegel **Abgabe: 30.01.2016, 12:00 Uhr**

Aufgabe 1: **Dijkstra-Algorithmus** (5 Punkte)



Führen Sie auf dem abgebildeten gerichteten Graphen den Dijkstra-Algorithmus aus. Erfassen Sie in einer Tabelle die Reihenfolge der Knoten u, die aus Q entfernt werden, sowie die Werte $d[u]$ und $\Pi[u]$ zu diesem Zeitpunkt.

Aufgabe 2: **A^*-Algorithmus** (4 + 2 + 2 Punkte)

Mit dieser Aufgabe wollen wir an einem Beispiel die Vorteile des A^*-Algorithmus im Vergleich mit dem Dijkstra-Algorithmus deutlich machen. Dabei ist unser Beispielgraph so konstruiert, dass man zur Berechnung der kürzesten Wege keinen dieser Algorithmen brauchen würde, denn die Lösung ergibt sich für beliebige Knotenpaare auf triviale Weise. Trotzdem wollen wir das Verhalten der beiden Algorithmen vergleichen, um die Unterschiede deutlich zu machen.

Wie betrachen den Gittergraphen $G = (V, E)$ mit $V = [-2n, 2n] \times [-2n, 2n]$ wobei hier $[-2n, 2n]$ alle ganzen Zahlen zwischen $-2n$ und $2n$ bezeichnen soll und zwei Knoten genau dann durch eine Kante mit Gewicht 1 verbunden sind, wenn sie eine gemeinsame Koordinate haben und sich die zweiten genau um 1 unterscheiden (also das normale quadratische Gitter). Sei $s = (0, 0)$ als Startknoten und $t = (n, n)$ als Zielknoten gegeben. Wir wollen für die beiden Algorithmen, die Anzahlen der Knoten, die (garantiert) vor t aus der Prioritätswarteschlange Q gelöscht werden, vergleichen.

Für den Dijkstra-Algorithmus ist das sehr einfach, denn für $v = (a, b)$ haben wir den Manhattan-Abstand zu s, also $d[v] = |a| + |b|$. Für dem A^*-Algorithmus verwenden wir den Euklidischen Abstand $dist(v, t)$ zum Zielknoten als Heuristik und kommen so zum Schlüsselwert $f[v] = d[v] + dist(t, v)$. Sei also $M_D = \{v \in V \mid d[v] < d[t]\}$ und $M_A = \{v \in V \mid f[v] < f[t]\}$.

a) Implementieren Sie ein Programm (Java oder Python), das für ein gegebenes n die Größen der beiden Mengen bestimmt.

b) Untersuchen Sie experimentell, wie sich der Quotient aus beiden Größen für wachsende n entwickelt.

c) Ihr Algorithmus hat wahrscheinlich eine quadratische Laufzeit. Skizzieren Sie verbal eine Idee, wie man das auf lineare Laufzeit verbessern könnte (ein Beweis wird hier nicht verlangt).

Aufgabe 3: **Graph-Konstruktionen** (2+3+3 Punkte)

In der 9. Übung wurde das Konzept des sogenannten Line-Graphen $L(G)$ eines ungerichteten Graphen $G = (V, E)$ eingeführt. Sei nun konkret $V = \{1, 2, \ldots, n\}$ und die Kantenmenge durch (aufsteigend geordnete) Adjazenzlisten $Adj[v]$ für alle $v \in V$ gegeben ($u \in Adj[v] \Leftrightarrow v \in Adj[u]$, denn G ist ungerichtet). Wir wollen nun die Kanten von G, also die Knoten von $L(G)$ wie folgt nummerieren. Man beginnt mit allen zu 1 inzidenten Kanten aufsteigend geordnet nach dem zweiten inzidenten Knoten, danach alle noch nicht nummerierten Kanten, die zum Knoten 2 inzident sind usw.

a) Zeichnen Sie den Graphen $G = (\{1, 2, \ldots, 6\}, E)$ in dem zwei Knoten genau dann adjazent sind, wenn ihr ggT gleich 1 ist und ordnen Sie dazu die Knoten auf einem regelmäßigen Sechseck an. Nummerieren Sie die Kanten nach der gegeben Vorschrift (die Nummerierung beginnt mit 1).

b) Beschreiben Sie in Pseudocode ein Programm, dass bei gegebenen Graphen G und einer Kante $e = \{u, v\} \in E$ die Nummer von e berechnet.

c) Schreiben Sie ein Programm, dass die Adjazenzlistendarstellung von $L(G)$ bezüglich der berechneten Nummerierung erzeugt, genauer gesagt soll bei Eingabe $k \leq |E|$ die Adjazenzliste $Adj_{L(G)}[k]$ als Liste der Nummern der zur k-ten Kante benachbarten Kanten erzeugt werden.